



Université du Québec
à Rimouski

**UN NOUVEAU MODÈLE DE CONTRÔLE D'ACCÈS APPLIQUÉ AUX
ENVIRONNEMENTS INFORMATIQUES; ÉTUDE DE CAS ET APPLICATION EN
INDUSTRIE 4.0**

Thèse présentée

dans le cadre du programme de doctorat en ingénierie de l'UQAC,
offert par extension à l'UQAR,
en vue de l'obtention du grade de Philosophiae Doctor

PAR

©NADINE KASHMAR

Juin 2022

Jury members:

Pr. Adrian Ilinca, jury president, Université du Québec à Rimouski (UQAR)

Pr. Mehdi Adda, research director, Université du Québec à Rimouski (UQAR)

Dr. Hussein Ibrahim, research co-director, Centre de recherche et d'innovation en intelligence énergétique (CR2ie)

Pr. Hamid Mcheick, external examiner, Université du Québec à Chicoutimi (UQAC)

Dr. Sasan Sattarpanah Karganroudi, internal examiner, Université du Québec à Trois Rivières (UQTR)

Initial deposit on 25-05-2022

Final deposit on 28-06-2022

UNIVERSITÉ DU QUÉBEC À RIMOUSKI
Service de la bibliothèque

Avertissement

La diffusion de ce mémoire ou de cette thèse se fait dans le respect des droits de son auteur, qui a signé le formulaire “*Autorisation de reproduire et de diffuser un rapport, un mémoire ou une thèse*”. En signant ce formulaire, l’auteur concède à l’Université du Québec à Rimouski une licence non exclusive d’utilisation et de publication de la totalité ou d’une partie importante de son travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, l’auteur autorise l’Université du Québec à Rimouski à reproduire, diffuser, prêter, distribuer ou vendre des copies de son travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris Internet. Cette licence et cette autorisation n’entraînent pas une renonciation de la part de l’auteur à ses droits moraux ni à ses droits de propriété intellectuelle. Sauf entente contraire, l’auteur conserve la liberté de diffuser et de commercialiser ou non ce travail dont il possède un exemplaire.

*To mom and dad, who taught me about the
dreams and how to catch them...*

*To all beloved ones who were behind this
work...*

ACKNOWLEDGMENT

This thesis is the fruit of a constant effort made through many years, and also of many beneficial exchanges and collaborations. This work would not have been accomplished without the help and the support that I have received from valuable and helpful people who shared the same passion for scientific research. It is with great pleasure that I thank today all those who supported me during these years of hard work and enabled me to succeed in this thesis.

I would like to express my sincere gratitude to my supervisors Pr. Mehdi Adda and Dr. Hussein Ibrahim for providing their invaluable guidance, comments, and suggestions but mostly for their continuous presence and support throughout this journey, which helped me grow and improve the quality of my work during these years. I thank them for their availability, and patience and for motivating me and guiding me towards the exploration of the interesting world of research.

Sincere thanks to Pr. Adrian Ilinca - Université du Québec à Rimouski, who honored me by chairing the jury. Also, I would like to thank Pr. Hamid Mcheick from Université du Québec à Chicoutimi, and Dr. Sasan Karganroudi from Université du Québec à Trois Rivières, for accepting to be jury members of this thesis, for the time taken reviewing, and for examining my thesis, and for their valuable comments that helped me to improve the quality of this document.

My deepest gratitude and thanks to dear friends Mirna Mekdad, Farida Aliane, Ali Raad, Ghassan Tarhini, and Hazem Abdel Ghaffar for all the support they made available throughout this journey, even though some of them live abroad, I am happy that our life paths crossed before and during this journey. I would also extend my thanks to my colleagues with whom I have shared good times at the office, the library, the lab, coffee breaks, and many others.

Finally, I would also express my very profound gratitude to my parents, sisters, and brother for their encouragement throughout the years of my study, this accomplishment would not have been possible without them.

RÉSUMÉ

L'émergence de la nouvelle génération d'environnements de mise en réseau avec la transformation numérique, tels que l'internet des objets (IdO) et l'industrie 4.0 avec leurs différentes applications, fait ressortir de nouvelles tendances, concepts et défis pour intégrer des systèmes plus intelligents et avancés dans des systèmes critiques et structures hétérogènes. Ce fait, en plus de la pandémie de COVID-19, a suscité un besoin plus important que jamais de contrôle d'accès (CA) en raison de la généralisation du télétravail et de la nécessité d'accéder aux ressources et aux données liées à des domaines critiques tels que le gouvernement, les soins de santé, l'industrie et les autres. Tout cela ouvre de nouvelles perspectives aux systèmes d'information traditionnels et aux méthodes CA en fusionnant de nouvelles technologies et services pour un accès transparent aux sources d'information à tout moment et n'importe où, en particulier avec la présence de cybercriminels et de cyberattaques. Dans cette réalité, toute cyberattaque ou attaque physique réussie peut perturber les opérations ou même réduire les services essentiels à la société. Pour assurer la sécurité et la confidentialité, plusieurs mécanismes de sécurité ont été utilisés et CA est l'une des exigences de sécurité essentielles dans ce domaine. Ce qui rend cette réalité également difficile, c'est la diversité et l'hétérogénéité des modèles CA qui sont mis en œuvre et intégrés à d'innombrables systèmes d'information. L'importance des exigences de sécurité, de protection des données et de confidentialité augmente avec la présence massive de nouveaux paradigmes et technologies, le déploiement de solutions numériques et intelligentes basées sur le concept de l'industrie 4.0, ainsi que la généralisation du télétravail. Pour empêcher l'accès non autorisé aux actifs logiques ou physiques, plusieurs méthodes CA sont mises en œuvre pour contrôler à quoi les utilisateurs peuvent accéder, quand et comment en appliquant les politiques organisationnelles définies.

Parallèlement à la progression technologique, divers travaux de recherche ont été menés en se concentrant sur le développement et l'amélioration des méthodes CA en cinq étapes principales (1) modèles CA communs, (2) modèles hybrides, (3) modèles étendus et (4) modèles abstraits, atteignant le niveau actuel. étape de développement de (5) métamodèles CA. Les modèles courants mis en œuvre dans différents environnements informatiques sont le contrôle d'accès discrétionnaire (DAC), le contrôle d'accès obligatoire (MAC), le contrôle d'accès basé sur les rôles (RBAC) et le contrôle d'accès basé sur les attributs (ABAC). Pour trouver des fonctionnalités CA plus avancées et définir un ensemble plus large de règles CA, divers modèles hybrides avec des fonctionnalités combinées de deux modèles ou plus sont proposés (par exemple, le modèle hybride RBAC/ABAC). De plus, différents modèles sont étendus en ajoutant de nouveaux composants en plus de ceux existants, plusieurs modèles CA sont également abstraits et de nouveaux composants sont ajoutés pour améliorer leurs fonctionnalités. La réalité actuelle des environnements informatiques impose la nécessité de se concentrer sur le développement de méthodes CA plus robustes et avancées, d'autant plus que les modèles CA communs, hybrides, étendus et abstraits ont atteint leurs limites et sont actuellement insuffisants pour répondre aux exigences CA nécessaires. Ce qui rend ce fait

également difficile, c'est l'hétérogénéité de tout - réseaux, applications, appareils, etc. - en plus de l'hétérogénéité des modèles CA. Par conséquent, les métamodèles CA sont proposés dans la littérature pour servir de cadres unificateurs pour inclure la plupart des fonctionnalités et des composants des modèles CA afin de permettre l'instanciation de divers modèles et la définition et l'application d'un ensemble plus large de politiques statiques et dynamiques.

Malheureusement, les métamodèles proposés ont des limites communes puisqu'ils ne sont (1) pas assez génériques et n'incluent pas toutes les fonctionnalités des modèles CA, (2) pas assez dynamiques pour suivre les mises à jour technologiques et (3) pas extensibles. En outre, ils (4) ne prennent pas en charge la fonctionnalité de hiérarchie pour tous les composants, (5) n'expliquent pas comment la collaboration et l'interopérabilité entre les modèles CA peuvent être atteints et (6) n'abordent pas la question de la migration d'un modèle à un autre. . Pour aborder les limitations existantes des métamodèles CA, dans ce projet de recherche, nous abordons les limitations génériques, extensibles, dynamiques et hiérarchiques. Nous proposons un métamodèle CA hiérarchique, extensible, avancé et dynamique (HEAD) pour les structures dynamiques et hétérogènes qui est capable d'englober l'hétérogénéité des modèles CA où divers modèles CA peuvent être dérivés (modèles existants et non existants). Pour l'implémentation, nous utilisons Eclipse (xtext) pour définir le langage spécifique au domaine (DSL) du métamodèle HEAD. Nous illustrons notre approche avec plusieurs instanciations réussies de divers modèles pour montrer comment elle prend en charge des fonctionnalités avancées par rapport à d'autres métamodèles. Pour l'évaluation et la validation, le métamodèle HEAD est utilisé pour spécifier les politiques CA nécessaires pour deux études de cas inspirées de l'environnement informatique de l'Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada ; le premier est destiné à l'environnement local (non IdO) d'ITMI et le second à l'environnement IdO d'ITMI. Pour chaque étude de cas, le modèle CA nécessaire est dérivé à l'aide du métamodèle DSL du HEAD, puis la notation xtend (un dialecte expressif de Java) est utilisée pour générer le code Java nécessaire qui représente l'instance concrète du modèle dérivé. Au niveau du système, pour obtenir les règles CA nécessaires, des requêtes Cypher sont générées puis injectées dans la base de données Neo4j pour représenter la politique de contrôle d'accès de nouvelle génération (NGAC) sous forme de graphique. Le cadre NGAC est utilisé comme point d'application pour les règles générées de chaque étude de cas. Les résultats montrent que le métamodèle HEAD peut être adapté et intégré à divers environnements locaux et distribués, capable de servir de cadre unificateur, de répondre aux exigences CA actuelles et de suivre les mises à niveau de politique nécessaires. De plus, nous implémentons un panneau d'administration pour le métamodèle HEAD, comme exemple supplémentaire, en utilisant VB.NET et SQL pour montrer que le métamodèle peut être implémenté pour générer des règles CA à l'aide d'autres plates-formes.

Mots clés: contrôle d'accès; maquette; métamodèle ; Industrie 4.0 ; IdO ; sécurité et confidentialité ; hétérogène; dynamique; hiérarchie; politique; mise en vigueur; transformation numérique ; COVID-19; DSL, Neo4j, NGAC;

ABSTRACT

The emergence of the new generation of networking environments with the digital transformation, such as the internet of things (IoT) and industry 4.0 with their different applications, brings out new trends, concepts, and challenges to integrate more intelligent and advanced systems into critical and heterogeneous structures. This fact, in addition to COVID-19 pandemic has prompted a greater need than ever for access control (AC) due to the widespread of telework and the need to access resources and data related to critical domains such as government, healthcare, industry, and others. All this releases new prospects to traditional information systems and AC methods by merging new technologies and services for seamless access to information sources at anytime and anywhere, especially with the presence of cyber-criminals and cyber-attacks. With this reality, any successful cyber or physical attack can disrupt operations or even decline critical services to society. To ensure security and privacy, several security mechanisms have been employed and AC is one of the essential security requirements in this domain. What makes this reality also challenging is the diversity and the heterogeneity of AC models that are implemented and integrated with countless information systems. The importance of security, data protection, and privacy requirements increases with the massive presence of new paradigms and technologies, the deployment of digital and intelligent solutions based on the industry 4.0 concept, also with the widespread of telework. To prevent unauthorized access to logical or physical assets, several AC methods are implemented to control what users can access, when, and how by enforcing the defined organizational policies.

Along with technology progression, various research works were conducted focusing on developing and enhancing AC methods in five main stages (1) common AC models, (2) hybrid models, (3) extended models, and (4) abstracted models, reaching the current stage of developing (5) AC metamodels. The common models that are implemented in different computing environments are Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC). To find more enhanced AC features and define a larger set of AC rules, various hybrid models with combined features from two or more models are proposed (e.g., hybrid RBAC/ABAC model). Furthermore, different models are extended by adding new components in addition to the existing ones, also several AC models are abstracted and new components are added to enhance their features. The current reality of computing environments imposes the need to focus on developing more robust and advanced AC methods, especially since the common, hybrid, extended, and abstracted AC models have reached their limits and are currently insufficient to meet the needed AC requirements. What makes this fact also challenging is the heterogeneity of everything—networks, applications, devices, etc.—in addition to the heterogeneity of AC models. Hence, AC metamodels are proposed in the literature to serve as unifying frameworks to include most features and components of AC models to allow instantiating various models and defining and enforcing a larger set of static and dynamic policies.

Unfortunately, the proposed metamodels have common limitations since they are (1) not generic enough and do not include all features of AC models, (2) not dynamic enough to follow technology upgrades, and (3) not extensible. Also, they (4) do not support the feature of hierarchy for all components, (5) do not explain how collaboration and interoperability between AC models can be achieved, and (6) do not address the issue of migration from one model to another. To address the existing limitations of AC metamodels, in this research project we address the generic, extensible, dynamic, and hierarchical limitations. We propose a Hierarchical, Extensible, Advanced, and Dynamic (HEAD) AC metamodel for dynamic and heterogeneous structures that is able to encompass the heterogeneity of AC models where various AC models can be derived (existing and the non-existing models). For the implementation, we use Eclipse (xtext) to define the domain-specific language (DSL) of HEAD metamodel. We illustrate our approach with several successful instantiations of various models to show how it supports advanced features compared to other metamodels. For the evaluation and validation, HEAD metamodel is employed to specify the needed AC policies for two case studies inspired by the computing environment of Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada; the first is for ITMI's local (non-IoT) environment and the second for ITMI's IoT environment. For each case study, the needed AC model is derived using the DSL of HEAD metamodel, then xtend notation (an expressive dialect of Java) is used to generate the needed java code which represents the concrete instance of the derived model. At the system level, to get the needed AC rules, Cypher queries are generated and then injected into Neo4j database to represent the Next Generation Access Control (NGAC) policy as a graph. NGAC framework is used as an enforcement point for the generated rules of each case study. The results show that HEAD metamodel can be adapted and integrated with various local and distributed environments, able to serve as a unifying framework, answer the current AC requirements and follow the needed policy upgrades. Moreover, we implement an administrative panel for HEAD metamodel, as an additional example, using VB.NET and SQL to show that the metamodel can be implemented to generate AC rules using other platforms.

Keywords: access control; model; metamodel; Industry 4.0; IoT; security and privacy; heterogeneous; dynamic; hierarchy; policy; enforcement; digital transformation; COVID-19; DSL, Neo4j, NGAC;

TABLE OF CONTENTS

ACKNOWLEDGMENT	ix
RÉSUMÉ	xi
ABSTRACT	xiii
TABLE OF CONTENTS	xv
LIST OF TABLES	xix
LIST OF FIGURES	xxi
LIST OF ABBREVIATIONS	xxvii
CHAPTER 1 GENERAL INTRODUCTION	1
1.1 Research Context and Motivation	1
1.2 Access Control	4
1.3 Access Control Challenges within Dynamic and Heterogeneous Structures	9
1.4 Problematic	11
1.5 Objectives	17
1.6 Research Methodology	18
1.7 Thesis Originality and Contributions	28
1.8 The Structure of the thesis	30
CHAPTER 2 ACCESS CONTROL METAMODELS: REVIEW, CRITICAL ANALYSIS, AND RESEARCH ISSUES	31
2.1 Introduction	32
2.2 Access Control Models	33
2.3 Access Control Metamodels	34
2.4 Discussion and Critical Analysis	36
2.5 Research issues and open questions	38
2.6 Conclusion and Future Perspectives	40
CHAPTER 3 HEAD ACCESS CONTROL METAMODEL FOR DYNAMIC AND HETEROGENEOUS STRUCTURES	43
3.1 Introduction	44
3.2 Related Works	46
3.3 Formalization of Access Control Policies	49
3.4 Defining the Grammar of HEAD Metamodel	53
3.5 Deriving Access Control Models	55
3.6 Generating Policies: Examples and Illustrations	66

3.7	Conclusions and Future Perspectives	69
CHAPTER 4	INSTANTIATION AND IMPLEMENTATION OF HEAD META- MODEL IN INDUSTRIAL ENVIRONMENT: NON-IOT AND IOT CASE STUDIES	73
4.1	Introduction	74
4.2	Related Works	76
4.3	HEAD Metamodel	78
4.4	The Subject of Study: Technological Institute for Industrial Maintenance (ITMI)	80
4.5	Case Study 1—ITMI: non-IoT	83
4.6	Case Study 2—ITMI: IoT	96
4.7	HEAD administrative panel	108
4.8	Evaluation and Validation of HEAD metamodel	112
4.9	Limitations of HEAD Metamodel	114
4.10	Conclusions and Future Perspectives	115
CHAPTER 5	HEAD ACCESS CONTROL METAMODEL: DISTINCT DESIGN, ADVANCED FEATURES, AND NEW OPPORTUNITIES	117
5.1	Introduction	118
5.2	Access Control Challenges within Dynamic and Heterogeneous Environments	120
5.3	Access Control Models: the development stages	121
5.4	Issues and Limitations of the Existing AC Metamodels	124
5.5	HEAD Metamodel: Development Approach to Access Control in Dynamic and Heterogeneous Environments	128
5.6	Open Issues and New Opportunities	136
5.7	Conclusions	137
CHAPTER 6	GENERAL CONCLUSION	141
6.1	Achieved objectives	142
6.2	Comparison between HEAD Metamodel and other AC metamodels	143
6.3	Future Perspectives	143
APPENDIX I	FROM ACCESS CONTROL MODELS TO ACCESS CONTROL METAMODELS: A SURVEY	145
APPENDIX II	A REVIEW OF ACCESS CONTROL METAMODELS	167
APPENDIX III	A NEW DYNAMIC SMART-AC MODEL METHODOLOGY TO ENFORCE ACCESS CONTROL POLICY IN IOT LAYERS	177
APPENDIX IV	SMART-AC: A NEW FRAMEWORK CONCEPT FOR MODEL- ING ACCESS CONTROL POLICY	183

APPENDIX V	ACCESS CONTROL IN CYBERSECURITY AND SOCIAL MEDIA	193
APPENDIX VI	DERIVING ACCESS CONTROL MODELS BASED ON GENERIC AND DYNAMIC METAMODEL ARCHITECTURE: INDUSTRIAL USE CASE . . .	227
APPENDIX VII	ACCESS CONTROL METAMODEL FOR POLICY SPECIFICATION AND ENFORCEMENT: FROM CONCEPTION TO FORMALIZATION	237
Bibliography	245

LIST OF TABLES

Table 1 AC Metamodels: the state-of-the-art (Kashmar, Adda, Atieh, and Ibrahim, 2021c) (Kashmar, Adda, and Ibrahim, 2021a)	9
Table 2 The Research Methodology	19
Table 3 The timeline	30
Table 4 Some Limitations of the Common AC Models	34
Table 5 Summary of the Proposed Access Control Metamodels	37
Table 6 Objective(s) and Limitation(s) of The Proposed Access Control Metamodels	39
Table 7 Metamodeling layers and details	50
Table 8 Comparison between HEAD Metamodel and other AC metamodels. . .	143

LIST OF FIGURES

Figure 1	The dynamic and heterogeneous structures (Kashmar, Adda, and Ibrahim, 2021b)	2
Figure 2	The Steps of Access Request	4
Figure 3	The common AC models	5
Figure 4	Hybrid AC models.	6
Figure 5	Illustration for AC model extension concept.	7
Figure 6	Illustration for model abstraction concept	7
Figure 7	Illustration for AC metamodel concept	8
Figure 8	The concept of generic AC metamodel.	12
Figure 9	AC features in the core structure for each of the proposed metamodels.	12
Figure 10	The concept of dynamic AC metamodel.	13
Figure 11	The concept of extensible AC metamodel.	14
Figure 12	Examples for hierarchy of (a) roles, (b) actions, (c) objects, and (d) contexts Kashmar, Adda, and Ibrahim, 2021b.	15
Figure 13	The concept of collaboration and interoperability of AC models. . .	16
Figure 14	The concept of migration from one AC model to another.	17
Figure 15	The initial step towards HEAD metamodel (Kashmar, Adda, Atieh, and Ibrahim, 2019a)	21
Figure 16	The main layers of our AC framework concept (Kashmar, Adda, Atieh, and Ibrahim, 2019b)	21
Figure 17	The Logical Architecture (Kashmar et al., 2020)	23
Figure 18	Towards the Formal representation of metamodel: A general instance (Kashmar, Adda, Atieh, and Ibrahim, 2021b)	25
Figure 19	The HEAD metamodel (Kashmar, Adda, and Ibrahim, 2021b) . . .	26
Figure 20	HEAD Metamodel: The DSL grammar (Kashmar, Adda, and Ibrahim, 2021b)	27

Figure 21 The aim of AC metamodels 33

Figure 22 Historical Evolution of common AC Models 34

Figure 23 Classification for the proposed AC Metamodels 35

Figure 24 The Era of Access Control Metamodels 36

Figure 25 Illustration for the concept generic metamodel 37

Figure 26 Illustration for the concept of metamodel extension 38

Figure 27 The common limitations in the existing AC metamodels 38

Figure 28 The dynamic and heterogeneous structures 46

Figure 29 Unifying heterogeneous concepts of AC models 50

Figure 30 HEAD metamodel: the kernel elements 51

Figure 31 Examples for hierarchy of (a) roles; (b) actions; (c) objects; and (d) contexts 52

Figure 32 HEAD Metamodel: The DSL Grammar 54

Figure 33 DAC model instance 56

Figure 34 DAC Policy Definition 56

Figure 35 MAC model instance 57

Figure 36 MAC Policy Definition 57

Figure 37 RBAC model instance 58

Figure 38 RBAC Policy Definition 58

Figure 39 ABAC model instance 59

Figure 40 ABAC Policy Definition 59

Figure 41 Hybrid MAC/RBAC model instance 60

Figure 42 Hybrid MAC/RBAC Policy Definition 60

Figure 43 Hybrid RBAC/ABAC model instance 61

Figure 44 Hybrid RBAC/ABAC Policy Definition 61

Figure 45 Dynamic AC metamodel: Scenario 1 62

Figure 46 Dynamic AC metamodel: Scenario 2	64
Figure 47 Extensibility: RBAC example	65
Figure 48 RBAC: (a) definition of role/object hierarchy; (b) hierarchy of role/object entities	66
Figure 49 A model instance based on RBAC	67
Figure 50 Example 1: generating RBAC policy	67
Figure 51 A model instance based on hybrid MAC/RBAC	68
Figure 52 Example 2: Generating MAC/RBAC policy	69
Figure 53 Example: RBAC instance	80
Figure 54 ITMI-Role hierarchy	81
Figure 55 The Implementation Phases	83
Figure 56 The system architecture of ITMI: non-IoT environment	83
Figure 57 A graph model representing the information flow of ITMI's non-IoT environment	84
Figure 58 Case Study 1: HEAD metamodel instance: (a) A hybrid model based on user-groups RBAC and ABAC entities/attributes; (b) rule expressions	87
Figure 59 Case Study 1: A sample of the Xtend notation to generate the java code for the Explicit entities and their hierarchies	89
Figure 60 Case Study 1: A sample of the generated java code for the subject and object entities of group-based; RBAC and ABAC model	90
Figure 61 Case Study 1: NGAC Policy Configuration	92
Figure 62 Case Study 1: A sample java code output	93
Figure 63 Case Study 1: NGAC graph	94
Figure 64 Case Study 1: Users' Permissions based on their roles (and role hierarchy)	95
Figure 65 Case Study 1: Examples of association relationship properties for role permissions	95
Figure 66 Case Study 1: Examples of NGAC authorization responses to Cypher statements	96

Figure 67	The system architecture of ITMI: non-IoT environment	97
Figure 68	A graph model representing the information flow of ITMI's IoT environment	97
Figure 69	Case Study 2: HEAD metamodel instance: (a) A hybrid model with two PCs (hybrid RBAC/ABAC and ABAC); (b) rule expressions	100
Figure 70	Case Study 2: A sample of the Xtend notation to generate the java code for the (a) PC; (b) AU (and AU hierarchy) entities; (c) the assignment of Ex -AUs entities	102
Figure 71	Case Study 2: A sample of the generated java code for (a) AU root entities and their child entities; and (b) a sample of the assignment of user to root role nodes (Us to UAs)	103
Figure 72	Case Study 2: A sample of java output (a) to configure PC1 and PC2; (b) user-role assignment	104
Figure 73	Case study 2: A sample of Cypher code output (a) PCs Us and Os nodes; (b) UAs of roles and Workers with U-UA assignment; and (c) a sample U/O-UA/OA assignment of PC2	104
Figure 74	Case study 2: NGAC Policy Configuration	105
Figure 75	Case Study 2: NGAC graph	106
Figure 76	Case Study 2: Examples of Users' access rights	107
Figure 77	Case Study 2: Example of association relationship properties for Workers permission	107
Figure 78	Case Study 2: Examples of NGAC authorization responses to Cypher statements	108
Figure 79	HEAD metamodel: Administrative Panel example	109
Figure 80	HEAD Administrative Panel: Instantiation of AC model	109
Figure 81	HEAD Administrative Panel: AC policy configuration	110
Figure 82	HEAD Administrative Panel: AC policy configuration steps	110
Figure 83	HEAD Administrative Panel: Adding Attributes to Model Components	111
Figure 84	HEAD Administrative Panel: Formulation of AC Rules	111

Figure 85 The components of the proposed AC metamodels 129

Figure 86 The development approach 130

Figure 87 Heterogeneous models with different policy expressions 130

Figure 88 An example of hybrid policy 131

Figure 89 A sample of DSL of HEAD metamodel 132

Figure 90 A flow chart: Instantiation of AC models using the DSL of HEAD
metamodel 133

Figure 91 Exmples of policy expressions using the meta-policy of HEAD
metamodel 134

Figure 92 A sample of Eclipse Xtend notation 135

LIST OF ABBREVIATIONS

AC Access Control

ABAC Attribute-based Access Control

AI Artificial Intelligence

AU Authorization Unit

BLP Bell–LaPadula model

CBAC Category-based Access Control

CRABC Context-Sensitive Role-based Access Control

CSPM Cloud Security and Privacy Metamodel

CW Chinese Wall model

DAC Discretionary Access Control

ECCAPAC Enhanced Context-aware Capability based Access Control

E_x Explicit

HEAD Hierarchical, Extensible, Advanced, and Dynamic

HGABAC Hierarchical Group and Attribute-based Access Control

HoBAC Higher-order Attribute-based Access Control

IACS Industrial Automation and Control Systems

IIoT Industrial Internet of Things

I_m Implicit

IoT Internet of Things

IT Information Technology

ITMI Institut technologique de maintenance industrielle

IS Information System

MAC Mandatory Access Control

NGAC Next Generation Access Control

O Object

OA Object Attribute

PC Policy Class

PU Procedural Unit

RBAC Role-based Access Control

S_t Setting

U User

UA User Attribute

WCMS Web Content Management System

CHAPTER 1

GENERAL INTRODUCTION

1.1 RESEARCH CONTEXT AND MOTIVATION

The substantial advancements in information technologies, with the emergence of ubiquitous computing and digital transformation such as the internet of things (IoT), cloud computing, etc., have brought unprecedented concepts and challenges to provide solutions and integrate advanced and self-ruling systems in critical and heterogeneous structures. This evolution releases new prospects to traditional information systems by merging new technologies and services for seamless access to information sources that are distributed everywhere and need to be accessed from anywhere at anytime (Jaidi et al., 2018) (Kashmar, Adda, and Ibrahim, 2021a). This fact, in addition to the COVID-19 pandemic has prompted a greater need than ever for access control (AC) due to the widespread of telework and the need to access resources and data related to critical domains such as government, healthcare, industry, and others (Krehling et al., 2021) (Antunes et al., 2021) (Kashmar et al., 2022b). However, security, data protection, and privacy have always been of utmost importance in any computing environment. The importance of these requirements increases with the massive presence of new paradigms and technologies (e.g., IoT), also with the deployment of digital and intelligent solutions based on the industry 4.0 or smart industry concept. Despite the significant developments in this sector over the past ten years, security requirements constitute a major challenge for developers, computer scientists, researchers, companies, and organizations that collect, store and use data in their operations or for research and innovation purposes.

However, as technologies grow, the way how people interact with devices and applications changes. Correspondingly, the needed security methods must be upgraded since any successful cyber or physical attack can disrupt operations or even decline critical services to society. To ensure security and privacy, several techniques have been employed and AC is one of the essential security requirements in this domain (Cruz-Piris et al., 2018) (Kashmar, Adda, and Ibrahim, 2021b) (Kashmar, Adda, and Ibrahim, 2021a). In the literature, AC models are developed and implemented to define and enforce AC policies in order to specify users' access rights to resources and verify that they can only access resources they are al-

lowed to in a given context. AC policies are among the most significant security mechanisms that are essential to increase the privacy and confidence of any information system (Jaidi et al., 2018) (Kashmar, Adda, and Ibrahim, 2021b). Figure 1 illustrates the notion of heterogeneous structures which include heterogeneous systems, platforms, networks, and devices, in addition to the heterogeneity of the implemented AC models to define and enforce different AC policies of organizations and industry sectors (Kashmar, Adda, and Ibrahim, 2021b).

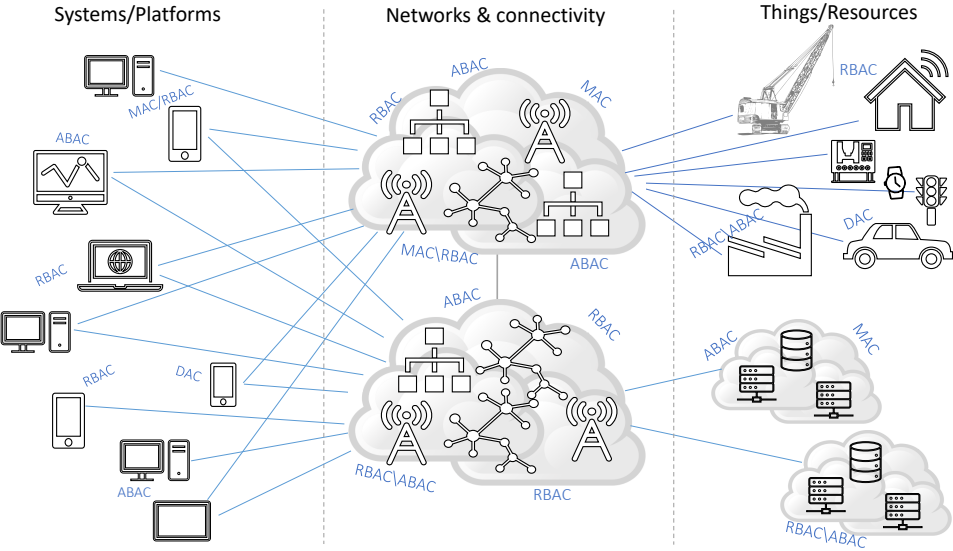


Figure 1: The dynamic and heterogeneous structures (Kashmar, Adda, and Ibrahim, 2021b)

The dynamic and heterogeneous structures of the current computing environments—networks, applications, devices, etc.—in addition to the heterogeneity of AC models that are implemented in different centralized and distributed computing environments, make the process of controlling access even more complicated. Up to the present time, AC research and real-world AC implementations to define and enforce AC policies broadly fall under one of the five stages:

1. Traditional AC models discretionary access control (DAC), mandatory access control (MAC), role-based access control (RBAC), attribute-based access control (ABAC) (Kashmar, Adda, Atieh, and Ibrahim, 2021a) (Kashmar, Adda, and Atieh, 2019);
2. Hybrid models, by means of combining features of two or more AC models, for example, the hybrid RBAC/ABAC model (Rajpoot et al., 2015a);

3. Extended AC models, by means of adding new component(s) to a model to enhance its features, for example, the extended model (Servos et al., 2014);
4. Abstract AC models, by means of abstracting a model and adding new components to it, then deriving different instances of it, for example, a higher-order attribute-based AC model (Aliane et al., 2019);
5. AC metamodels, by means of including all of the above, for example, the metamodel (Kashmar, Adda, and Ibrahim, 2021b).

However, technology is an essential aspect of information security, especially in recent years, it encompasses various trends and concerns. In the following we highlight the trends that increase the evolution of security threats:

- The growth in usage of mobile devices such as tablets, e-readers, smartphones, etc.;
- The approbation of social networks in different domains, for example, media sharing, online shopping, etc.;
- The widespread of the IoT concept, and the variety of IoT applications and devices (e.g. wearable devices);
- The adoption of the internet into daily activities, e.g. online learning, and
- The widespread acceptance of dynamic computing environments such as IoT, cloud computing, etc.

With the evolution of technology trends, it is realized that controlling users' access and the operations they perform on information cannot be overlooked when developing approaches related to information security. Security solutions must be manageable and adaptable to track the evolution of security threats that accompany technology upgrades. Unfortunately, the existing AC models—common models, hybrid models, extended AC models, and abstract AC models— no longer meet the increasing demand for privacy and security standards. In other words, they have reached their limits and are currently insufficient to answer the increasing demand for security and privacy standards (Kashmar, Adda, and Atieh, 2019) (Al Kukhun, 2012). Likewise, the existing AC metamodels have several shortcomings and are not able to follow the continuous technology progression (Kashmar, Adda, Atieh, and Ibrahim, 2021b) (Kashmar, Adda, Atieh, and Ibrahim, 2021c) (Kashmar, Adda, and Ibrahim, 2021a). To address this issue, in this research we have developed a Hierarchical, Extensible, Advanced,

and Dynamic (HEAD) AC metamodel, with unconventional features, that are able to include the heterogeneity of the existing AC models, derive non-existing AC models, and follow the needed AC requirements with the current evolution of technology.

1.2 ACCESS CONTROL

Access control is the process of restricting access to logical or physical resources. An AC policy is the definition of rules that must be regulated in an organization, they are usually defined by managers and systems administrators based on a set of guidelines for a system, to control and authorize access to resources. (Kashmar, Adda, Atieh, and Ibrahim, 2021c) (Kashmar, Adda, Atieh, and Ibrahim, 2021a). Figure 2 summarizes the AC request steps:

1. User authentication, where an unknown user (or subject) after verifying his identity becomes a known user;
2. The authenticated user now can make a request to access resources (or objects);
3. Authorize request is the most essential phase in this process, authorization is to allow an authenticated subject to access a resource after checking the defined AC policy;
4. A subject is allowed or denied to perform some action on a resource.

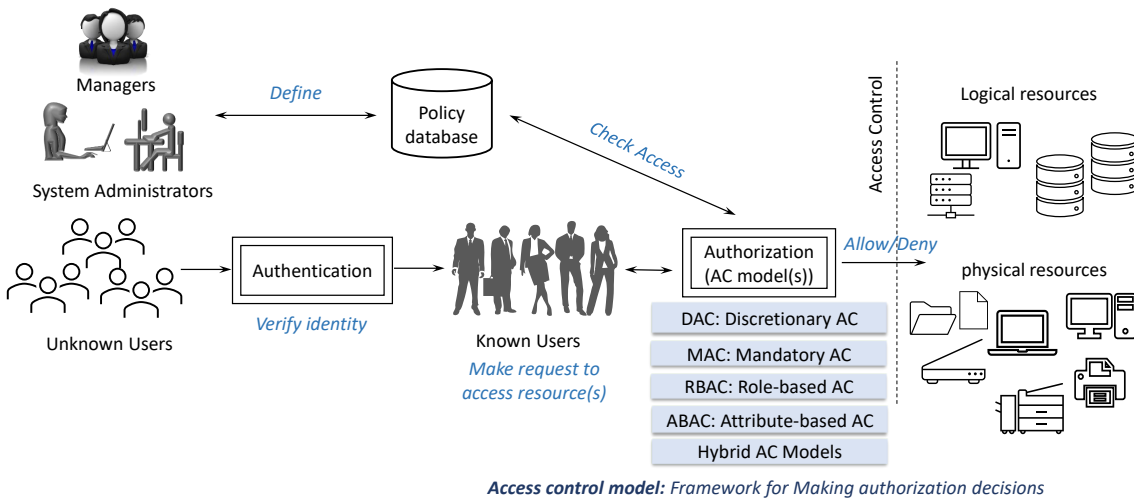


Figure 2: The Steps of Access Request

Authentication is insufficient to protect resources, and authorization includes the following phases (Jaidi et al., 2018) (Kashmar, Adda, and Ibrahim, 2021b):

1. defining a security policy (set of AC rules);
2. selecting an AC model that matches the defined policy;
3. implementing the model and enforcing the defined AC rules.

1.2.1 Access Control Models: the development stages

Over the decades, various information technologies (IT) have been developed, and this impose the need to implement various AC methods to find secure communication environments. In the following sections, we summarize the proposed AC models in the literature and their development stages in order to enhance AC features.

1.2.1.1 Common AC Models

The well know AC models, or common models, that are proposed in the literature to define and enforce AC policy to prevent any illegal access to resources are DAC, MAC, RBAC, and ABAC models. Figure 3 summarizes their features, and their major components which are used to formulate and enforce organizational AC policies (Kashmar, Adda, and Atieh, 2019) (Kashmar, Adda, Atieh, and Ibrahim, 2021a).

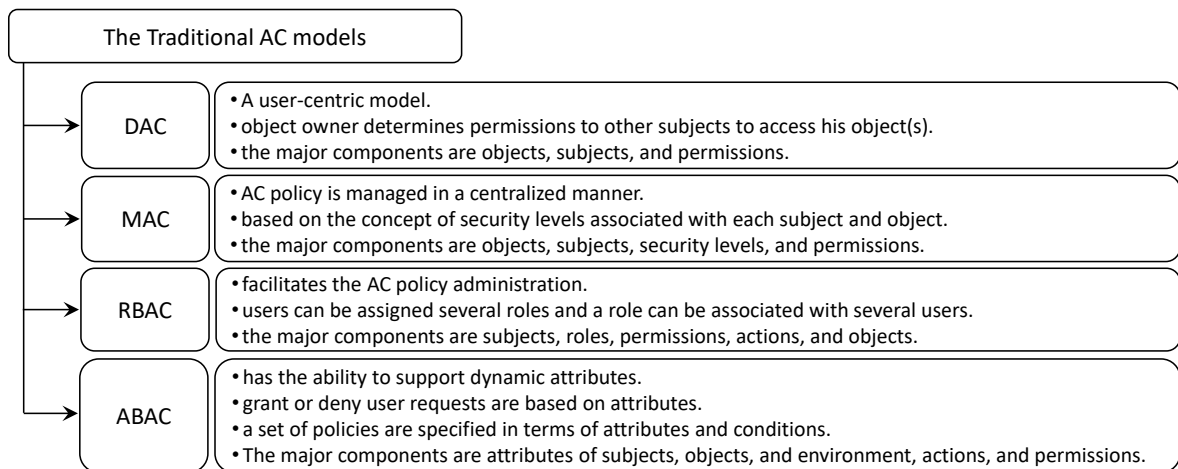


Figure 3: The common AC models

1.2.1.2 Combining AC Models

The continuous technology upgrade, with the distributed computing environments, increases the presence of security threats. This imposes the need to find hybrid AC models, by combining features of two or more AC models, to allow defining larger sets of AC rules in order to enhance the process of access management (Kashmar, Adda, Atieh, and Ibrahim, 2021c). Several hybrid AC models are proposed in the literature that combine features of RBAC and ABAC, for example (Hasiba et al., 2017), (Rajpoot et al., 2015b), and (Kaiwen et al., 2014); DAC, MAC and RBAC, for example (Oh, 2007); MAC and RBAC, for example (Kim et al., 2014); and many other hybrid models. Figure 4 illustrates the idea of hybrid models.

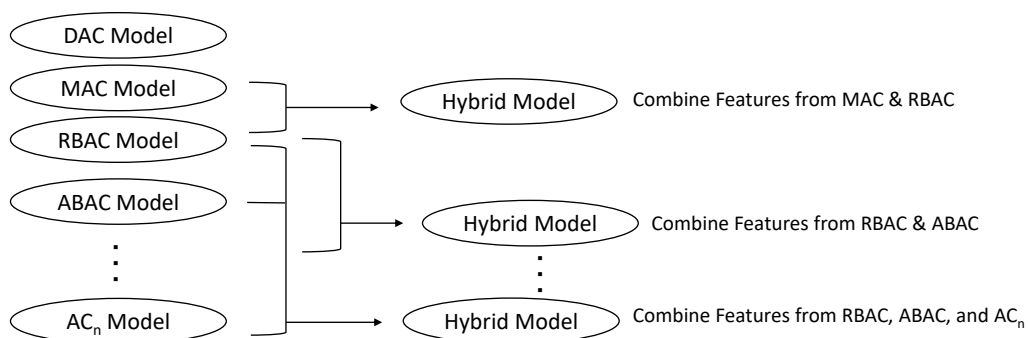


Figure 4: Hybrid AC models.

1.2.1.3 Extending AC Models

Other research works extend AC models by adding new components to them such as new types of roles, permissions, and relationships. For example, the core or flat RBAC model with major components such as subjects, objects, permissions, actions, and roles is extended to hierarchical RBAC where a new component is added to support role hierarchy. Moreover, it is extended to constrained RBAC where a new component is added to enforce the separation of duties. Then, symmetric RBAC which includes hierarchical RBAC and constrained RBAC (Ennahbaoui et al., 2013). Aliane et al., 2019 propose a higher-order attribute-based access control (HoBAC) model as an extension for the ABAC model, which extends the basic concepts of ABAC with aggregation operations that yield hierarchies. Another ABAC model extension is presented by Servos et al., 2014, called the hierarchical group and attribute-based access control (HGABAC), where groups and hierarchies of subjects and objects are added.

Moreover, several other AC model extensions are proposed in this domain, for example an extended model proposed by Layouni et al., 2009. Figure 5 illustrates the concept of AC model extension.

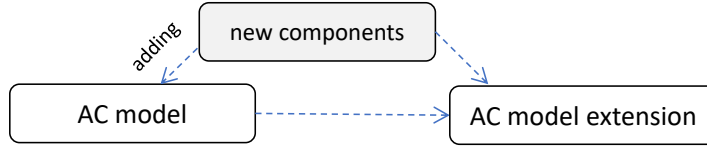


Figure 5: Illustration for AC model extension concept.

1.2.1.4 Abstracting AC Models

Some AC models are abstracted, then new components are added to enhance their features and allow expressing a larger set of AC policies. Subsequently, the derived AC model is an extended model with the old and new AC features. For example, Nguyen et al., 2013 add a delegation component to the abstracted RBAC model (RBAC metamodel) to have an RBAC-based delegation model. In addition to defining RBAC permission rules, their approach would allow defining delegation rules to specify which actions are accessible to users by delegation. Moreover, Klarl et al., 2009 propose a business and system role-based access control (B&S-RBAC) metamodel where business and system roles are defined and mapped to overcome the weakness of business role definitions and RBAC models. Hence, the RBAC model is abstracted, a system and business role component is added, then it is extended for business usage. Moreover, Adda et al., 2020 propose a generalization for the ABAC model where the core concepts of HoBAC (Aliane et al., 2019) are first revisited and refined, then present new concepts to complete and reinforce its theoretical foundations. Figure 6 illustrates the concept of AC model abstraction.

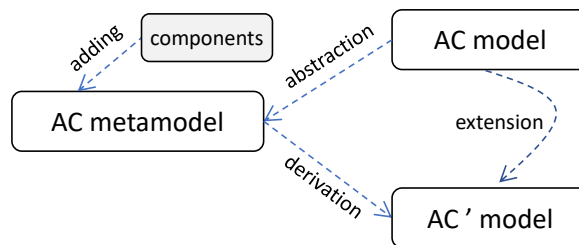


Figure 6: Illustration for model abstraction concept

1.2.1.5 Access Control Metamodels

Access control metamodels are proposed to serve as frameworks that unify and include most or all features of AC components to derive various instances of AC models (Kashmar, Adda, and Atieh, 2019) (Kashmar, Adda, Atieh, and Ibrahim, 2021c). Figure 7 illustrates the concept of the AC metamodel. However, AC metamodels should handle all of the above concepts (combining AC models, extending AC models, and abstracting AC models). In other words, having an abstract model (metamodel) that is able to include all AC models components, or have the ability to define the needed ones, would allow combining and extending different AC models.

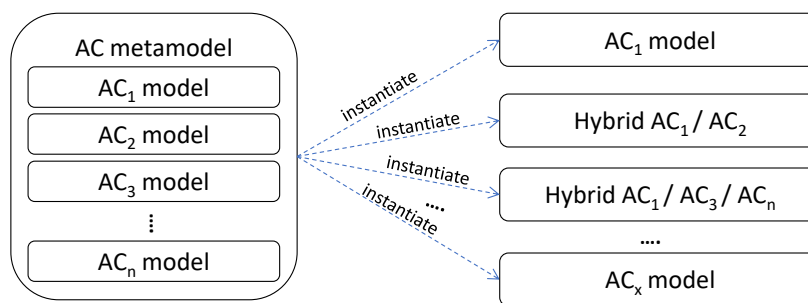


Figure 7: Illustration for AC metamodel concept

Controlling access to resources with the current generation of networking environments is a challenging task and needs advanced AC frameworks. Within the decade, a limited number of AC metamodels are proposed in the literature, Table 1, to find more advanced AC features and answer the increasing demand of security and privacy. They can be summarized as follows (Kashmar, Adda, Atieh, and Ibrahim, 2021c) (Kashmar, Adda, and Ibrahim, 2021a):

- Some AC metamodels are proposed as generic metamodels such as (Barker, 2009), (Bertolissi et al., 2014), (Khamadja et al., 2013), and (Trninić et al., 2013).
- Some others are proposed as hybrid metamodels to provide a generic base metamodel concept such as (Slimani et al., 2011), and (Abd-Ali et al., 2015).
- Others are proposed as metamodel extensions for some of the existing metamodels and some software development frameworks such as (Alves et al., 2014), and (Korman et al., 2016).

Table 1
AC Metamodels: the state-of-the-art (Kashmar, Adda, Atieh, and Ibrahim, 2021c)
(Kashmar, Adda, and Ibrahim, 2021a)

ref.	Year	Proposed for	Metamodel	Based on	Instances	Modeling lang.
<i>AC metamodels are proposed as generic metamodels</i>						
Barker, 2009	2009	Enterprise	Barker's Meta-model	DAC, MAC, RBAC	RBAC, MAC	Rule/Logic Language
Bertolissi et al., 2014	2014	Distributed system of several sites	Distributed Metamodel	DAC, MAC, RBAC	hybrid models of DAC, MAC, RBAC	rewrite-based operational semantics
Khamadja et al., 2013	2013	Cloud Computing	CatBAC meta-model	DAC, MAC, RBAC	hybrid models of DAC, MAC, RBAC	First-order logic
Trninić et al., 2013	2013	Set of systems	PolicyDSL	RBAC models	RBAC and hybrid models	Textual DSL
<i>AC metamodels are proposed as hybrid AC metamodels</i>						
Slimani et al., 2011	2011	Enterprise	UACML Meta-model	DAC, MAC, RBAC	Group based, MAC, RBAC, hybrid model	Object constraint language (OCL)
Abd-Ali et al., 2015	2015	Enterprise	Integration metamodel	CW, BLP, BIBA, RBAC	Hybrid models	First-order logic
<i>AC metamodels are proposed as metamodel extension for some of the existing metamodels</i>						
Alves et al., 2014	2014	Enterprise	Obligations in CBAC Meta-model	DAC, MAC, RBAC	hybrid models of DAC, MAC, RBAC	rewrite-based operational semantics
Korman et al., 2016	2016	Enterprise Architecture framework	Unified Meta-model	DAC, BLP, Biba, CW, RBAC, ABAC	DAC, BLP, CW, RBAC, ABAC	ArchiMate

The continuous technology progression impose developing different stages of AC methods, AC metamodel is a recent development stage in this domain. Unfortunately, the proposed metamodels lack some key features and are insufficient to answer the AC requirements of the current networking generation (Kashmar, Adda, Atieh, and Ibrahim, 2021c) (Kashmar, Adda, and Ibrahim, 2021a). The issues and limitations of the existing AC metamodels are explained in Section 1.4.

1.3 ACCESS CONTROL CHALLENGES WITHIN DYNAMIC AND HETEROGENEOUS STRUCTURES

The new generation of networking environments is dynamic and ever-evolving environments where resources are distributed and accessed from everywhere. This evolution of pervasive systems and ubiquitous computing, especially the concept of Industry 4.0 and IoT applications, imposes the need to enhance AC methods due to the increasing demand for pri-

vacy and security standards. Controlling access in these environments is a challenging and complicated task due to the presence of cyber-criminals and cyber-attacks. In the following, we summarize the existing challenges in this domain (Kashmar et al., 2022b) (Kashmar, Adda, and Ibrahim, 2021b) (Al Kukhun, 2012) (Kashmar et al., 2020):

- Access control is considered as a protective shield for the existing resources in organizations, industries, homes, etc. against cybersecurity risks that accompany transparency, shareability, and interoperability while answering users' needs in accessing resources, especially with distributed systems and IoT environments where data is dynamic and generated in a real-time basis.
- Access control is highly essential to ensure secure communications in open, dynamic, and heterogeneous environments, especially with the existence of several contexts (network type, time, location, machine characteristics ...). For example, to provide an access decision, several or multi-level, contexts must be considered. The context itself is a challenge since it could be defined, expressed, and interpreted in different ways according to the application domain, the needed objectives, and the existing techniques.
- Access control method should be flexible and controllable enough to deal with various situations that confront users while requesting access to different types of resources, for example, rapid progression, unexpected events, system failures, and highly dynamic environments that need a very quick and controlled response, various hierarchical organizational structures, the different contexts, dynamism of IoT devices, etc.
- With the large adoption of telework, especially with the COVID-19 pandemic, employees can work anytime, anywhere, and sometimes using their own personal devices. With this fact, security and AC issues have been raised, especially that AC mechanisms should be adapted to users' context, their profiles, their devices, the existing conditions, and many other parameters to improve system usability. With this fact, AC needs to be dynamically managed to minimize human intervention.

Besides, the following are the key AC enforcement challenges (Kashmar et al., 2022b) (Bertino et al., 2018) (Soltani et al., 2017):

- The heterogeneity of the implemented AC models in heterogeneous environments.
- Deciding upon the most relevant AC model(s) to implement in an organization that conforms to its AC rules based on the type and sensitivity of resources they have.

- The necessity to enforce persistent AC policies with the current generation of dynamic and heterogeneous structures where information flows from the clouds and/or servers to everywhere (companies, hotels, homes, cars, ...) without traditional borders.
- The possible need for several AC solutions within the same organization (or industry sector) where various technologies (e.g., local network, IoT, and cloud) may need to work in concert to fulfill the needed requirements of AC.
- The importance of upgrading AC policies and enforcing them in accordance with technology upgrades and due to dynamically changing conditions.

Due to the above challenges, and the limitations of traditional models, hybrid models, extended AC models, and abstracted AC models in answering the needed AC requirements of the current computing environments, and since they are unable to follow up the continuous technology progression (Kashmar, Adda, Atieh, and Ibrahim, 2021c) (Kashmar et al., 2020), (Kashmar, Adda, and Atieh, 2019), the research interest for developing AC metamodels have gained the attention in the last decade (Kashmar, Adda, and Ibrahim, 2021a).

1.4 PROBLEMATIC

Although the proposed AC metamodels in the literature provide some advancement for some scenarios and use cases, they have several limitations and shortcomings which can be explained and illustrated as follows:

1.4.1 Generality

In the literature, the proposed metamodels as generic, are not generic enough and they do not include all features of AC models. They are hybrid templates to derive some AC models that are employed in their core structure rather than metamodels (Kashmar, Adda, and Ibrahim, 2021a). A generic AC metamodel should include all features and components of common AC models and other models. In Figure 8, we illustrate the concept of generic metamodel where all AC components are included, with the relationships between them. Figure 9 summarizes AC model features which are employed in the core structure for each of the proposed metamodels. As shown in Table 1, the metamodels by Barker, 2009, Bertolissi et al., 2014, Khamadja et al., 2013, and Trninić et al., 2013 are proposed as generic by including DAC, MAC, and RBAC features and components, and by Slimani et al., 2011

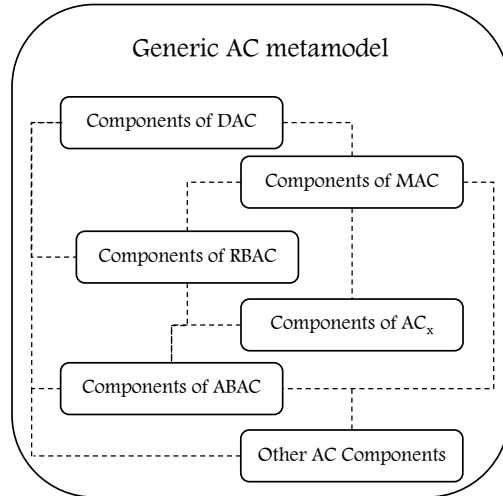


Figure 8: The concept of generic AC metamodel.

and Abd-Ali et al., 2015 as hybrid by combining DAC, MAC, and RBAC AC features and components. Korman et al., 2016 extend the ArchiMate development framework to support features and components of DAC, MAC, RBAC, and ABAC models. Alves et al., 2014 extend the category-based AC (CBAC) metamodel which includes features of DAC, MAC, and RBAC to support obligations. Hence, the proposed approaches are not generic enough and do not include or combine all AC features and components. As illustrated in Figure 9, the generic metamodel should include the features and components of common AC models and other models.

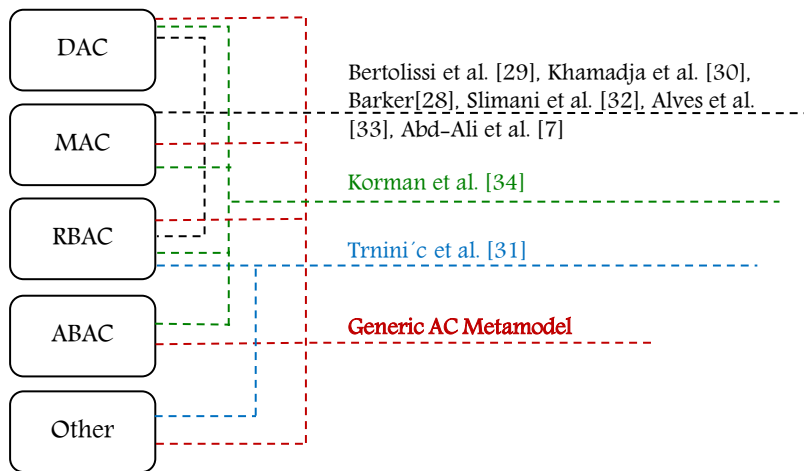


Figure 9: AC features in the core structure for each of the proposed metamodels.

1.4.2 Dynamism

Security solutions must be manageable and adaptable to track the evolution of security threats that come along with technology upgrades. In this context, an AC metamodel must be upgradable due to changing conditions or rules. The structure of a metamodel should be dynamic and describe how its properties can be modified over time, for example, due to changing conditions (e.g., system, environment, etc.). The metamodel is dynamic if it allows defining new types of attributes, for example, contextual attributes, and components with the relationships between them in order to update and formulate different models; hence, it allows defining a larger set of policies for static and dynamic enforcement. In reference to Table 1, the concept of a dynamic metamodel is not considered since none of them allows defining new components/attributes. Consequently, dynamism is an additional feature that can be implemented for a generic metamodel. Figure 10 illustrates the concept of a dynamic metamodel.

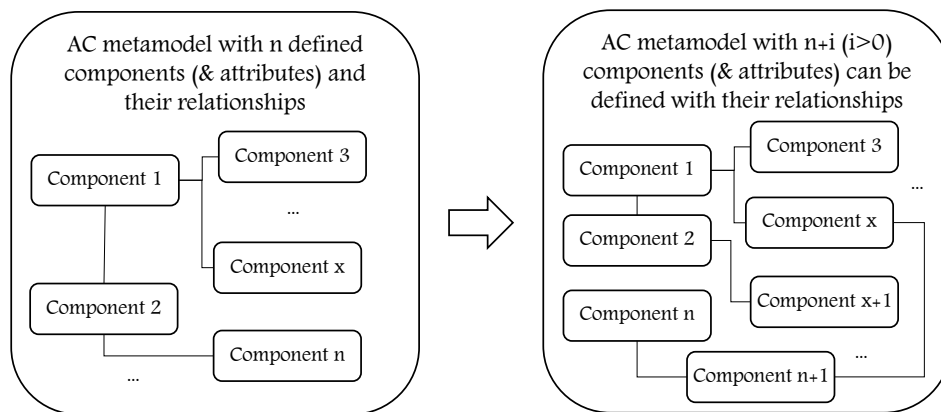


Figure 10: The concept of dynamic AC metamodel.

1.4.3 Extensibility

Besides designing a generic and dynamic AC metamodel, it is important to consider the feature of extensibility. Extensible metamodel means that new components could be defined and integrated with already existing models and frameworks to support new AC features in addition to the previous ones. However, the proposed metamodel extensions, in Table 1, by Alves et al., 2014 and Korman et al., 2016 extend some of the existing AC metamodels and software development frameworks to support some AC features of some AC models, but they

do not explain how their approaches could be extended beyond the proposed limit. In other words, they are extended but not extensible. In Figure 11, we illustrate the concept of the extensible AC metamodel.

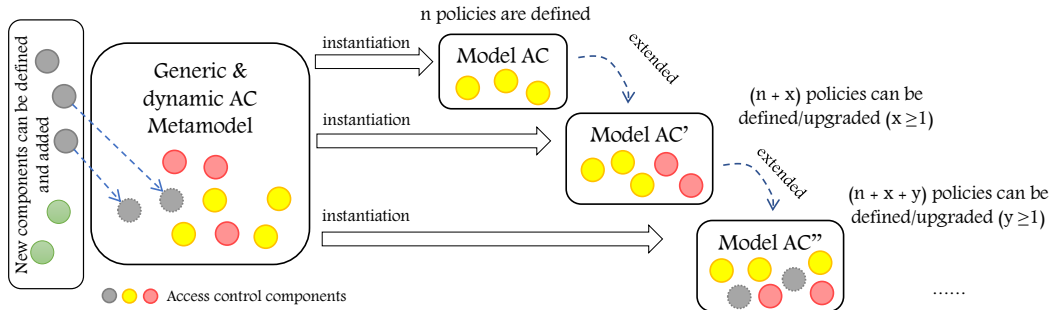


Figure 11: The concept of extensible AC metamodel.

1.4.4 Hierarchy of Components

The concept of hierarchy is important to define multi-level components (e.g., roles, actions, objects, etc.). It reflects the structure of an organization and, for example, the respective responsibilities/priorities of the hierarchical components. A component hierarchy, e.g., role, defines roles that have unique attributes and may contain other roles; one role may implicitly include the actions that are associated with another role. Within this structure, access rights are determined by an entity's place in the hierarchy, for example, in complex scenarios (e.g., IoT), administrators can start by creating a number of entities and then add their hierarchy. This would help in controlling access to data with fewer maintenance costs compared to creating a large number of non-hierarchical entities. Hence, hierarchical authorization is the authorization determined based on the hierarchy. Figure 12 represents examples of hierarchy in an organization or industry sector (e.g., role hierarchy, Figure 12a; action hierarchy, Figure 12b; object hierarchy, Figure 12c; context hierarchy, Figure 12d). For example, in an academic and research situation, a role called Dean could contain the roles of Director and Team Leader, Figure 12a. This means that subjects (users) of the role Dean are implicitly connected with the actions associated with their roles as Director and Team Leader without the administrator having to explicitly list the Director and Team Leader actions. In the literature, several models and metamodels are extended to support the feature of hierarchy for some components, but in complex scenarios and highly dynamic environments, it is important to consider this feature for all components, for example, none of the proposed metamodels con-

sider context hierarchy. The metamodels proposed by Khamadja et al., 2013 and Bertolissi et al., 2014 support hierarchy of category (e.g., role, groups, etc.); by Slimani et al., 2011 supports hierarchy of category, action, object; and by Abd-Ali et al., 2015, Korman et al., 2016, and Trninić et al., 2013 support hierarchy of roles.

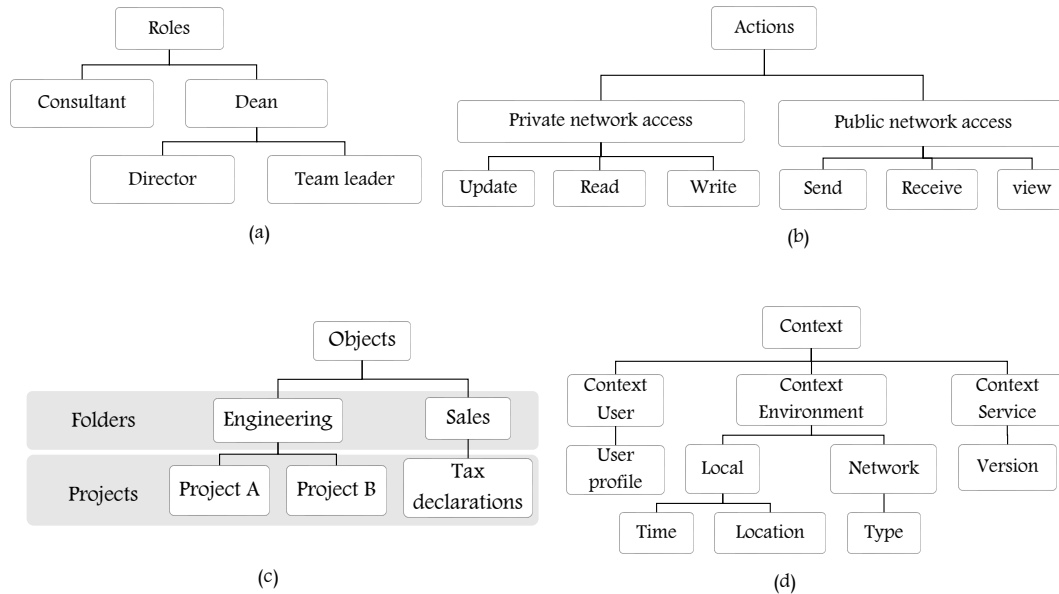


Figure 12: Examples for hierarchy of (a) roles, (b) actions, (c) objects, and (d) contexts Kashmar, Adda, and Ibrahim, 2021b.

1.4.5 Collaboration and Interoperability

In pervasive computing environments, the software enables the connection between various heterogeneous devices and users within a dynamic and heterogeneous environment; it carries out the needed mappings between each task and the required services that users need. In the field of AC, and with the current technologies, AC policies are employed to administer decisions in systems. They are increasingly used for implementing flexible and adaptive systems to control access to resources in today's internet services, networks, security systems, and others. An AC metamodel should deal with the dynamicity of devices and objects used, events, situations encountered, users accessing systems and the environments from which they are connected, and others. It should be highly adaptable and flexible, and it should be able to integrate many devices and information systems to provide the needed services for users (and organizations) and ensure homogeneity and collaboration between its compo-

nents. Moreover, it should provide the administrative required tasks and enable interoperable interactions between several derived AC models. Figure 13 illustrates the collaboration and interoperability concept an advanced AC metamodel should provide.

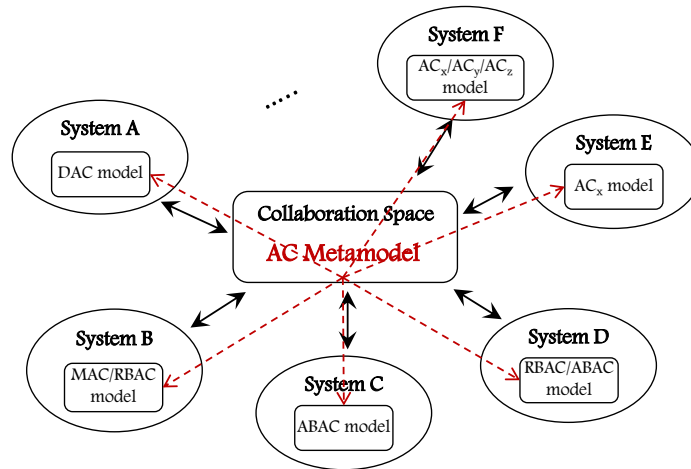


Figure 13: The concept of collaboration and interoperability of AC models.

1.4.6 Migration

Due to continuous technology upgrades, it is essential to consider the concept of migration where different software components are transferred from one computing environment to another. This concept must also be considered for AC models where the AC migration is a kind of modernization for AC policies (set of rules that are generated using different components) from one model to another covered by a generic, dynamic, and extensible AC metamodel. Though, none of the proposed approaches tackle the concept of migrating AC policies. In Figure 14, we illustrate the concept of policy migration from one model to another.

To sum up, the proposed metamodels are not enough to answer the current AC requirements, and they have the following limitations:

- 1- Each metamodel is itself a case and does not encompass a general base concept to derive various instances for all AC models. In other words, they are planned for dedicated scenarios or case studies based on some features of AC models;

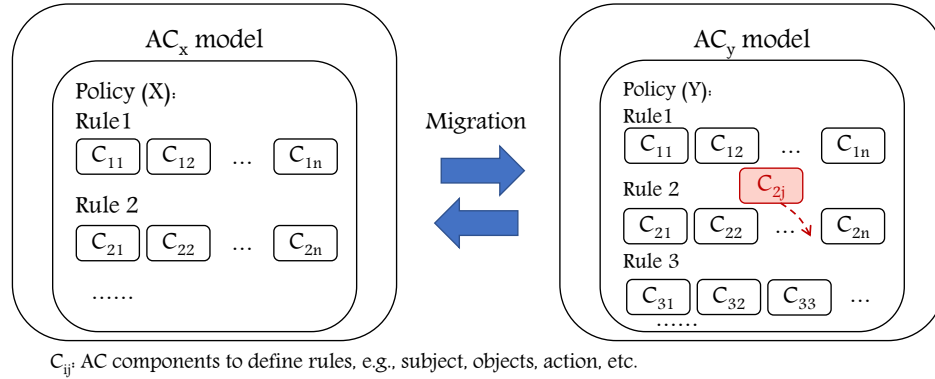


Figure 14: The concept of migration from one AC model to another.

- 2- They do not support the ability to define various types of attributes. So, they are not dynamic enough to follow the continuous technology upgrades.
- 3- Neither the generic nor extended proposed metamodels are enough to address the needed target of enforcing AC policy, especially with the current technologies and continuous upgrades;
- 4- They do not support the feature of hierarchy for all components which is essential to conform hierarchical organizational structure;
- 5- No provided explanations about how the derived models could collaborate within the same computing architecture e.g., IoT;
- 6- An essential aspect is not considered in all of the presented AC metamodels which is the migration of AC policy from one AC model to another. Having a metamodel should make it possible to translate an existing AC policy between the different AC models covered by the metamodel.

1.5 OBJECTIVES

The main objective of this research is to develop a new AC metamodel adaptable to different computing environments that overcomes the limitations of the existing metamodels. To achieve this, we first review the proposed AC metamodels in the literature to study their objectives, weaknesses, and their limitations in controlling access to various resources. As well, we identify information security needs and AC requirements with the recent technologies. In this context, we find that although the proposed metamodels have some enhanced features, they have common limitations (section 1.4) and are insufficient to meet the needed

AC requirements of the current generation of networking environments, especially with the emergence of industry 4.0 and IoT applications. To achieve the main objective of this research, we set the following specific objectives:

- 1- Design and develop generic, dynamic, extensible, and hierarchical AC metamodel.
- 2- Providing a simple and flexible DSL that overcomes the complication of existing language expressions.
- 3- Apply the metamodel to industry 4.0 environment.

1.6 RESEARCH METHODOLOGY

In this research project we have proposed a hierarchical, extensible, advanced, and dynamic AC metamodel, named HEAD metamodel, with several advantages compared to the existing metamodels in terms of flexibility and the ability to provide effective and dynamic security models for dynamic systems. In this section, we explain the research methodology which is conducted in four phases over a period of four years. The first phase, is the contextualization of research topic where we review and analyze the existing metamodels to find out the effectiveness of their features in the presence of ubiquitous computing and pervasive information systems, and if they are upgradable and able to follow continuous technology progression. During this phase, the problem is framed then the possible solution is proposed. The second phase is the design of the formal metamodel where the applied steps are explained, starting from scratch and reaching the needed design. The third phase is the implementation phase where the tools for metamodel instantiation are designed, and the DSL for policy instantiation is developed. The fourth phase, is the evaluation and validation of HEAD metamodel where it is employed to define and enforce the needed AC policies for two case studies inspired by the computing environment of Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada; the first is for ITMI's local (non-IoT) environment and the second for ITMI's IoT environment. In Table 2, we summarize the research plan to achieve the needed objectives.

Table 2
The Research Methodology

Problem	An AC metamodel with advanced features that is able to include the heterogeneity of AC models and answer the needed AC requirements and challenges of the current networking generation is not yet achieved.	
Research objectives	1	Develop an AC metamodel that is able to instantiate different AC models (the common AC models, hybrid models, and others).
	2	Design dynamic AC metamodel, flexible, and upgradable due to continuously changing requirements and policies.
	3	Design extensible metamodel that considers any future extensions or transformations and allow describing a larger set of possible rules for given AC model(s).
	4	Develop the feature of finding hierarchical relationships between the defined policy components/entities, for example, the hierarchy of roles.
	5	Applying the metamodel to industrial IoT (industry 4.0) - case study.
Research Plan	Research objective 1	<ul style="list-style-type: none"> • Review the literature (AC models and AC metamodels). • Analyze the existing metamodels and study their objectives and the limitations. • Frame the problem, set the goals, and develop the possible scope of solution.
	Research objectives 2 and 3	<ul style="list-style-type: none"> • Build the concept of the new generic and dynamic metamodel with preliminary examples. • Explain the logical architecture of the new metamodel with examples • Represent a General Metamodel Instance with examples • Apply the metamodel to a case study case
	Research objective 4	<ul style="list-style-type: none"> • Formal representation for the AC metamodel • Implementation of the metamodel <ul style="list-style-type: none"> - Designing tools for metamodel instantiation - Developing DSL for policy instantiation - Illustrations and examples
	Research objective 5	<ul style="list-style-type: none"> • Case study (ITMI) – non-IoT and IoT scenario • Instantiate the AC model(s) for the case studies based on the metamodel • Generate AC policies using xtend notations and use NGAC as an enforcement framework. • Final Validation

1.6.1 Phase 1: Review and Critical Analysis of AC metamodels

1.6.1.1 Literature Review

In this step, the state-of-the-art for the proposed AC models and metamodel in various computing environments is reviewed (Kashmar, Adda, and Atieh, 2019) (Kashmar, Adda,

Atieh, and Ibrahim, 2021a) (Kashmar, Adda, Atieh, and Ibrahim, 2021c). We focus on the recent research issue in this domain which is the AC metamodels, we find that the proposed metamodels have limited AC features in particular and lack essential features in general. Hence, they are insufficient to follow up on technology progression and answer the increasing demand of AC requirements. The common limitations, section 1.4, of the existing AC metamodels are generality, dynamism, extensibility, hierarchy, collaboration and interoperability, and migration of AC policies. In this research, we address the generality, dynamism, extensibility, and hierarchy issues.

1.6.1.2 The Scope of Solution

To find the needed solution, our idea is designed by considering the aforementioned limitations (section 1.3) that accompany the recent technologies and their continuous progression in general, and the enforcement challenges in organizations in particular. At the initial stages of our work, we have realized that:

The only thing we can expect with confidence about the future of security frameworks is that things we cannot expect will happen.

Hereafter, we set our idea based on this insight. Figure 15 illustrates the initial step towards constructing our metamodel. Our idea is based on the concept of including everything and anything essential for the AC decision and included in the AC policy, whether they are subjects, objects, permissions, roles, categories, attributes, etc., this step is the definition of components. Then, the defined components should be oriented to find one or many AC models (e.g., $AC_1=DAC$, ..., $AC_x=$ hybrid RBAC/ABAC, ..., $AC_n=RBAC$), this step is the model formulation (Kashmar, Adda, Atieh, and Ibrahim, 2019a).

1.6.1.3 The Framework Concept

In this step, a deeper look for our idea is proposed by providing a new framework concept for modeling AC policy (Kashmar, Adda, Atieh, and Ibrahim, 2019b). Figure 16 illustrates the general AC framework layers where all components of common AC models, in addition to new ones, can be defined and formulated which makes it flexible for any new extensions or transformations. It is composed of three main levels:

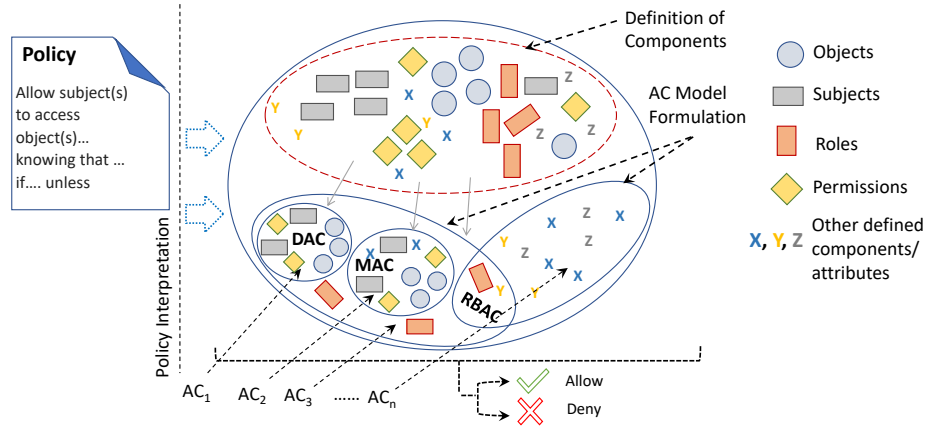


Figure 15: The initial step towards HEAD metamodel (Kashmar, Adda, Atieh, and Ibrahim, 2019a)

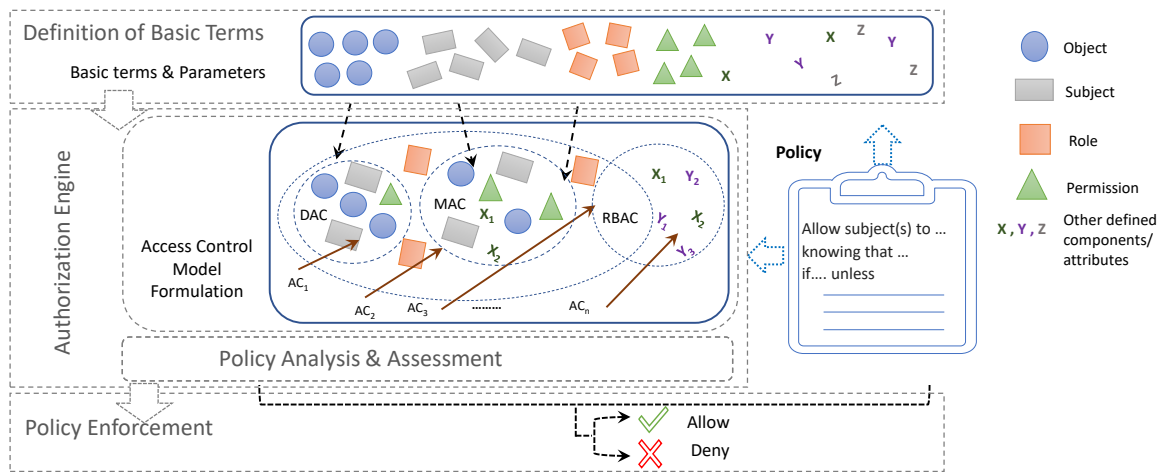


Figure 16: The main layers of our AC framework concept (Kashmar, Adda, Atieh, and Ibrahim, 2019b)

- **The definition of all components:** where all AC components/attributes are defined (e.g., subjects, objects, roles, permissions, security levels, attributes. . .) with the ability to define new ones (e.g., X_i = actions, Y_j =views . . .).
- **The authorization engine:** is mainly composed of two sublayers, the first sublayer is where the formulation of AC method to model the policies of authorization takes place. The second sublayer is the policy analysis and assessment to check the consistency, relevance, minimality, completeness, etc. of AC rules and assess their correctness, enforceability, etc. before enforcing them, in other words, checking if there are any needs to modify, update, or re-design the policy. Note that, the policy analysis and assessment

sublayer is not addressed in this research.

- **The policy enforcement:** where AC decisions are enforced after determining whether a subject is allowed or denied accessing a certain object(s).

During the first phase, the following papers are published:

- **Four conference papers:** the first paper is a preliminary review for common AC models and the existing AC metamodels with some research questions (see Appendix 1); the second is a review with critical analysis for the proposed AC metamodels, and their objectives and limitations (see Appendix 2); the third paper explains our proposed idea to overcome the existing limitations and the scope of the solution to control access in heterogeneous structures, e.g., IoT (section 1.6.1.2) (see Appendix 3); and the fourth paper presents the framework concept (section 1.6.1.3) (see Appendix 4).
- **Book chapter:** during this phase, our aim was also to review more of the implemented AC methods in various domains, for this purpose we have reviewed some of the used AC methods in social media networks. The chapter title is "Access Control in Cybersecurity and Social Media" (see Appendix 5).
- **Journal paper:** deeper and detailed review with critical analysis for metamodels, their limitations, in addition to various research issues and open questions (Chapter 2).

1.6.2 Phase 2: The Design of HEAD Metamodel

1.6.2.1 The Logical Architecture

In this step, the classes and layers of dynamic and multi-layer metamodel are defined (Kashmar et al., 2020), and illustrated in Figure 17. It is mainly composed of 6 levels to formulate the needed AC model.

- Level 1: all attributes (a_1, \dots, a_n ; where $n \geq 0$) of subjects, objects, etc. are defined.
- Level 2: the defined attributes, in level 1, are aggregated to construct the needed classes (key components for an AC model), for example, subjects, objects, etc.
- Level 3: class instances are created for explicit components, for example, subjects (S_1, \dots, S_n ; $n \geq 0$) and objects (O_1, \dots, O_n ; $n \geq 0$). An explicit component refers to something that is real and exist in an organization.

- Level 4: class instances are created for implicit components, for example, roles (R_1, \dots, R_n), actions (A_1, \dots, A_n), permissions (P_1, \dots, P_n), security levels (L_1, \dots, L_n), Categories (C_1, \dots, C_n), etc. An implicit component refers to something described or explained in the guidelines or rules of a security policy.
- Level 5: aggregation of class instances for explicit and implicit components, created in in levels 3 and 4, for the needed assignments for example: subjects (S_1, S_2, S_7) \in role (R_1), or object (O_n) \in security level (L_n), etc.
- Level 6: the needed aggregations are performed to set the permissions, actions, etc. which can/not be performed by subjects over objects. This could be occurred by selecting and aggregating the needed instances and assignments of levels 3, 4 and 5 for the AC decision (allow/deny access) and checking constraints.

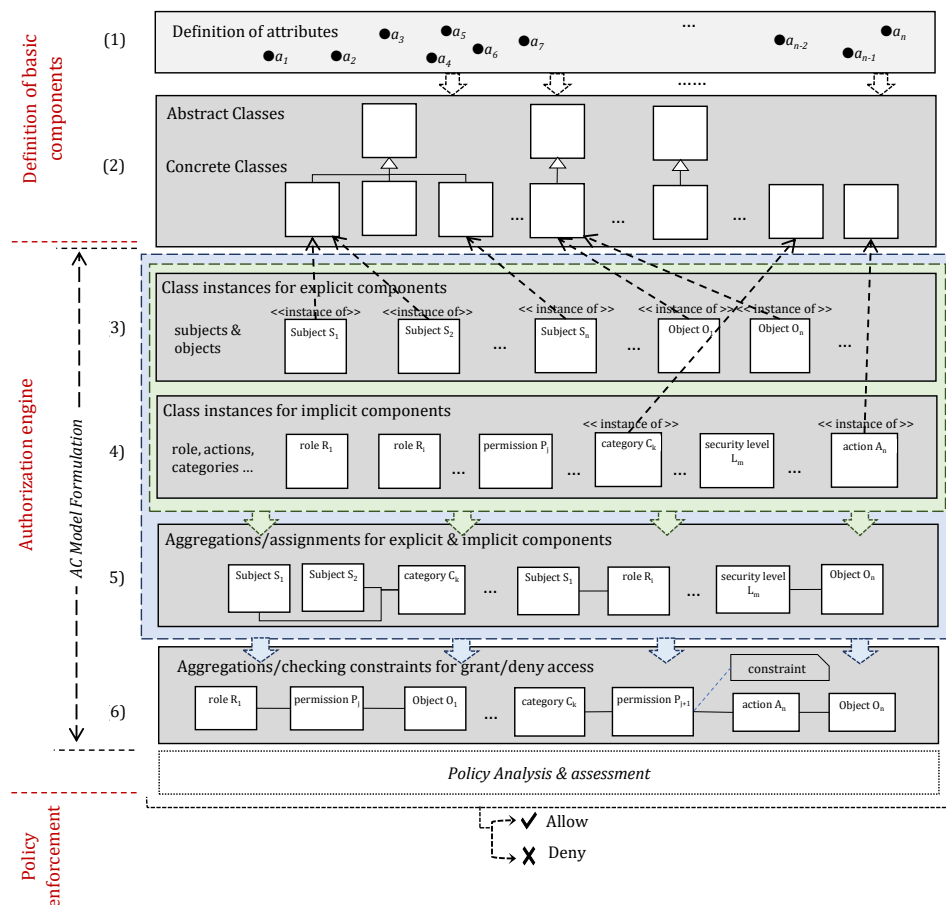


Figure 17: The Logical Architecture (Kashmar et al., 2020)

Moreover, an industrial use case is presented to provide a detailed description of how components/attributes can be aggregated to express policies. As mentioned earlier, the policy analysis and assessment phase is not addressed in this research. The main purpose of this step is to plan for the formal AC metamodel and explain the relationships between its components before designing and implementing it.

1.6.2.2 Visualization for a General Metamodel Instance

Based on the above logical architecture, a closer image for a general instance of the metamodel is presented, also the main concepts that any AC policy might include in order to identify the main components/classes of the metamodel are interpreted. In general, any AC policy includes a set of concepts (and attributes): (1) to describe subjects and objects; (2) describe the authorized subjects; (3) explain the different access rights; (4) set various constraints and conditions; (5) describe the context (environment) to access objects. Moreover, the components of AC models that are implemented to generate AC policies are heterogeneous, to unify them and make them adaptable to all AC models we classify them into explicit, implicit (authorization units and procedural units), and setting concepts as illustrated in Figure 18. *Explicit* concepts refer to something that is real and exists (e.g., subjects and objects). *Implicit* concepts refer to something described or explained in the guidelines or rules. Implicit concepts include *Authorization Units* (e.g., roles, security levels ...) and *Procedural Units* (actions, permissions ...). *Setting* refers to concepts that are included to have more accurate and regulated access to resources (e.g., context, constraints ...) (Kashmar, Adda, Atieh, and Ibrahim, 2021b). Unifying AC concepts of heterogeneous security policies was an essential step towards our formal AC metamodel.

1.6.2.3 The Formal Metamodel

After unifying the heterogeneous concepts of AC policies, in this step, the formal metamodel, its meta-components, and the relationships between them are illustrated (Figure 19). The relationship between explicit (E_x) and Authorization Unit (AU) is to assign, for example, zero or many (0..*) subjects to roles, groups, categories, or other AUs. The relationships between AU and Procedural Unit (PU), and PU and E_x are to represent which AUs are able to perform zero or many PUs (e.g., actions, permissions ...) and access some, for example, objects or services. Note that Implicit I_m components (AUs and PUs) might have zero or many

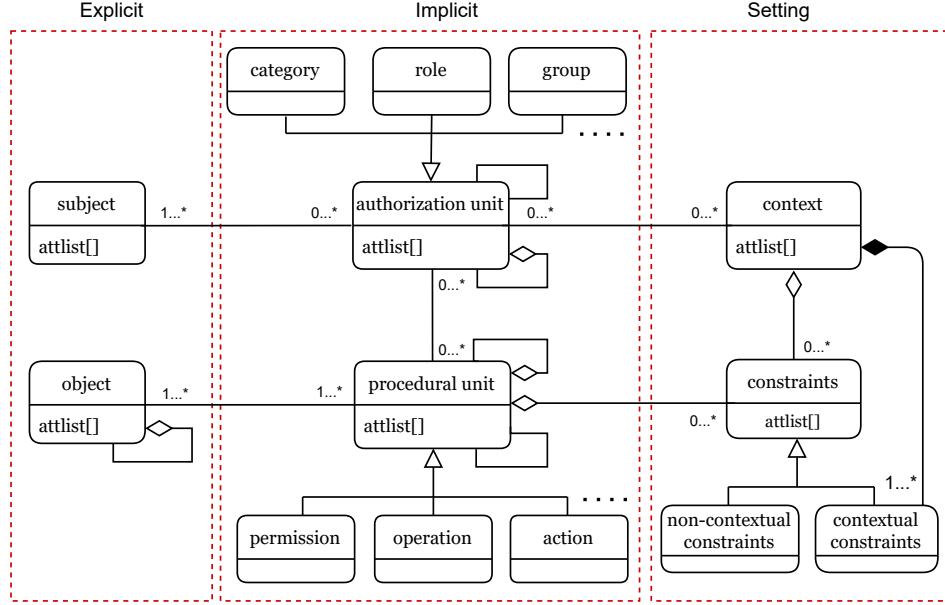


Figure 18: Towards the Formal representation of metamodel: A general instance (Kashmar, Adda, Atieh, and Ibrahim, 2021b)

Setting (S_t) (e.g., contextual and/or non-contextual constraints) before accessing/performing tasks on E_x components. Moreover, the metamodel provides support for formulating AC models and hybrid models for different policies by allowing AUs to be associated with other AUs, PUs to be associated with other PUs, E_x components to be associated with other E_x components, and S_t components to be associated with other S_t components. As shown in Figure 19 a self-association edge exists in each of the classes. Note that in some models, we might have an empty set of AU, or S_t , for example, in the DAC model, AU is an empty set since explicit components are not assigned to AUs. The formal metamodel is Hierarchical, Extensible, Advanced, and Dynamic (HEAD) metamodel, in phase 2 we explain the implementation details of HEAD metamodel.

During this phase, two conference papers are published, the first paper (see Appendix 6) explains the logical architecture of section 1.6.2.1; and the second paper (see Appendix 7) represents a visualization for a general metamodel instance of section 1.6.2.2. The tasks of section 1.6.2.3 are explained in detail in chapter 3.

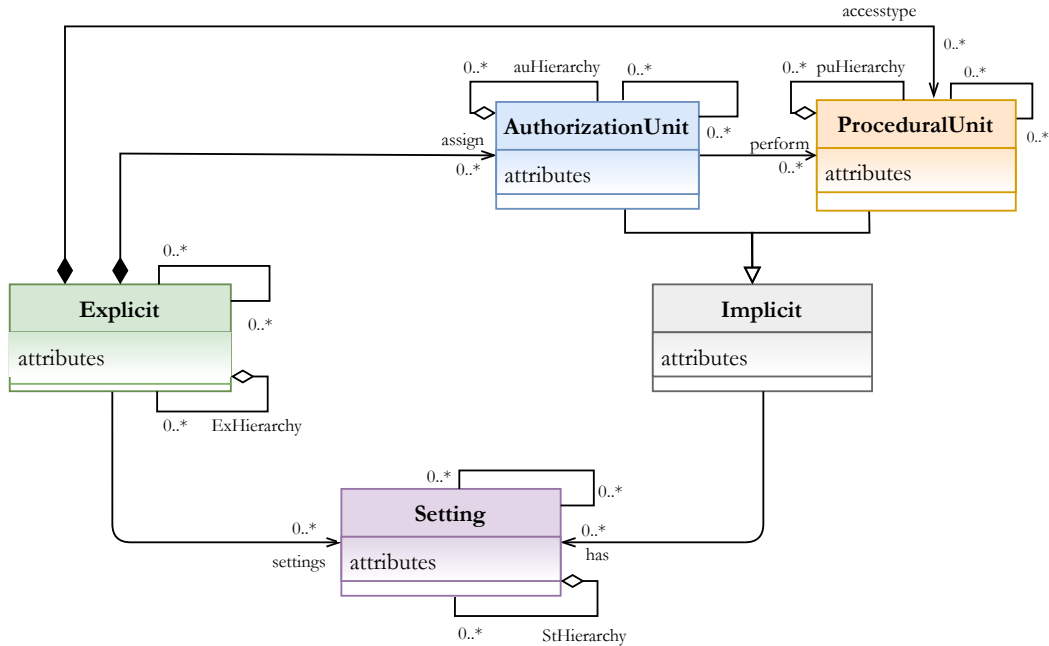


Figure 19: The HEAD metamodel (Kashmar, Adda, and Ibrahim, 2021b)

1.6.3 Phase 3: Implementation of HEAD Metamodel

In this phase, we have developed the DSL language of the HEAD metamodel using Eclipse-xttext to allow instantiating different AC models using this DSL. In Figure 20, we show the DSL of HEAD metamodel. Lines 1 to 39, to instantiate the needed components of AC models, the hierarchies, and the attributes. Lines 40 to 63, to express a set of AC rules using the derived components/attributes of E_x , AU, PU, and S_t with the access request decision. Using the DSL of HEAD metamodel we have derived various instances of common models and hybrid models. Hereafter, the derived models are encoded using Eclipse Xtend notation, an expressive dialect of Java, to represent the concrete instance of an AC policy and generate the needed java code. Chapter 3 explains the implementation details of this section (a manuscript is published in Sensors journal).

1.6.4 Phase 4: Evaluation and Validation for HEAD Metamodel

For the evaluation and validation, HEAD metamodel is employed to specify the needed AC policies for two case studies inspired by the computing environment of Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada; the first is for ITMI's

```

1- Metamodel:
2-   {Metamodel}
3-   'policy' (policy+=Policy)+ 'end'
4-   'rule:'
5-     decision=Decision
6- ;
7- Attribute:
8-   name=ID (array ?='[' (length=INT)? '']? ":" type=AttType
9- ;
10- Policy:
11-   name=ID (('attributes+=Attribute+')?)
12-   'explicit' (explicit+=Explicit)+ 'end'
13-     (implicit+=Implicit)+
14-   ('setting' (setting+=Setting)* 'end')?
15- ;
16- AttType:
17-   'String' | 'int' | 'boolean' | 'char' | 'float'
18- ;
19- Explicit:
20-   name=ID ((' attributes+=Attribute+')?)
21-     (['heirarchy+=Explicit+'])?
22- ;
23- Implicit:
24-   {Implicit}
25-   ('authorization' authunit+=AuthorizationUnit* 'end')?
26-   ('procedural' procunit+=ProceduralUnit* 'end'
27- ;
28- AuthorizationUnit:
29-   name=ID ((' attributes+=Attribute+')?)
30-     (['heirarchy+=AuthorizationUnit+'])?
31- ;
32- ProceduralUnit:
33-   name=ID ((' attributes+=Attribute+')?)
34-     (['heirarchy+=ProceduralUnit+'])?
35- ;
36- Setting:
37-   name=ID ((' attributes+=Attribute+')?)
38-     (['heirarchy+=Setting+'])?
39- ;
40- Decision:
41- ((' attributes+=Attribute*'))?
42-   '{' (explicit+=[Explicit|QualifiedName]
43-   (' (wth+=[Attribute|QualifiedName]* ')?')?)
44-   (' [' (authunit+=[AuthorizationUnit|QualifiedName] ('(wth+=[Attribute|QualifiedName]*'))?) * ' '])?
45-   (' {' procunit+=[ProceduralUnit|QualifiedName] ('(wth+=[Attribute|QualifiedName]*'))?)?
46-   => '{'
47-     (explicit+=[Explicit|QualifiedName]
48-     ('(wth+=[Attribute|QualifiedName]* ')?')?)
49-     (' [' (authunit+=[AuthorizationUnit|QualifiedName] ('(wth+=[Attribute|QualifiedName]*'))?) * ' '])?
50-     '{'
51-     (
52-       procunit+=[ProceduralUnit|QualifiedName] ('(wth+=[Attribute|QualifiedName]*'))?
53-       (' {' (setting+=[Setting|QualifiedName] ('(wth+=[Attribute|QualifiedName]*'))?) * ' '})?
54-     )+
55-     '}'
56-   )+
57-   (' ')?
58-   '}'
59- '}' '--> id+=ID)+
60- ;
61- QualifiedName:
62-   ID('. ' ID)*
63- ;

```

Figure 20: HEAD Metamodel: The DSL grammar (Kashmar, Adda, and Ibrahim, 2021b)

local (non-IoT) environment and the second for ITMI's IoT environment. For each case study, the needed AC model is derived using the DSL of HEAD metamodel, then xtend notation (an expressive dialect of Java) is used to generate the needed java code which represents the concrete instance of the derived model. At the system level and to get the needed AC rules, Cypher statements are generated (after configuring the AC policies) and then injected into Neo4j database to represent the Next Generation Access Control (NGAC) policy as a graph. NGAC framework is used as an enforcement point for the generated rules of each case

study. The results show that HEAD metamodel can be adapted and integrated with various local and distributed environments, able to serve as a unifying framework, answer the current AC requirements and follow the needed policy upgrades. Moreover, we have implemented an administrative panel for HEAD metamodel, as an additional example, using VB.NET and SQL to show that the metamodel can be implemented to generate AC rules using other platforms. Chapter 4 explains the instantiation and implementation details for two industrial case studies (a manuscript is submitted to Sensors journal - Special Issue: Cybersecurity in the Internet of Things).

In addition to all above, a manuscript is published in the Journal of Cybersecurity and Privacy (JCP) with the title “HEAD Access Control Metamodel: Distinct Design, Advanced Features, and New Opportunities”, explaining the distinct design of the HEAD metamodel, starting from the metamodel development phase and reaching to the policy enforcement phase. Moreover, we describe the remaining steps and how they can be employed to develop more advanced features and open new opportunities and answer the various challenges of technology progression in the domain.

1.7 THESIS ORIGINALITY AND CONTRIBUTIONS

The novelty of the thesis lies in the idea proposed to design and implement a hierarchical, extensible, advanced, and dynamic AC metamodel that serves as a unifying framework to derive (existing and non-existing) AC models. Having a new and advanced AC metamodel in the domain with advanced features that conforms to organizational AC security policies and adapts the decision making according to technology progression to meet organizational and users’ needs, represents a very innovative concept. HEAD metamodel features can be summarized as follows:

- It unifies the heterogeneous components of AC models.
- It is generic enough to include the common AC models and other models.
- It is dynamic and includes the feature of defining components (and attributes for all components).
- It is extensible since it allows extending the already derived AC models.
- It supports the hierarchy for any type of components.

- it allows instantiating non-existing AC models.
- it allows creating hybrid models with several policy classes to express combined and uncombined AC rules.

This research project is of major importance for organizations and industry sectors in general and for Quebec in particular, especially after the unveiling of the government's intention to create a ministry for digital and cybersecurity in October 2021. In other words, after taking serious steps to ensure its digital transformation and quickly penetrate the 4.0 era through the deployment of solutions and technologies using the principle of the IoT. This research project has several innovative aspects and they can be enumerated as follows:

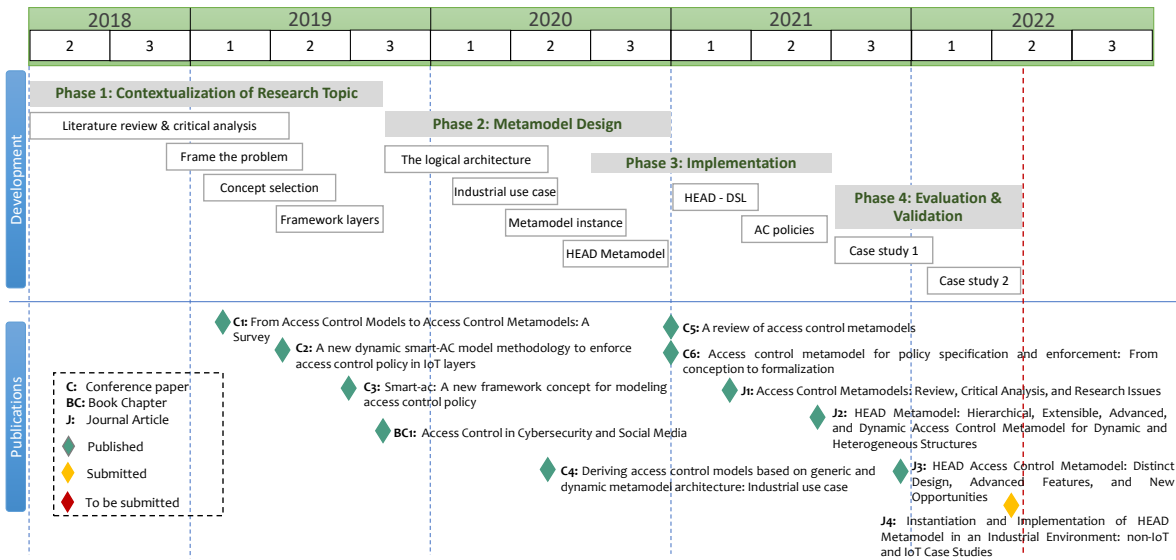
- **Development in the field:** due to limited number of the proposed works in the domain (the last metamodel proposed as generic was in 2014 and as hybrid was in 2015 (Kashmar et al., 2022a))
- **Novel Concept:** since there is no similar concept proposed in the domain.
- **Innovative Design:** it provides a novel and distinct design for an AC metamodel in the domain that address the current limitations of AC metamodels to improve the AC methods.
- **Contributes to cybersecurity:** against cyber-attacks compared to the existing AC metamodels due to its features.
- **Necessary Materials and Tools:** it provides the necessary examples and tools for the researchers in the domain.
- **Adaptability:** can be implemented in centralized and distributed computing environments.
- **Paradigm shift:** Contributes to the development of knowledge in the field and can be considered as a paradigm shift towards reshaping the existing models, especially in the field of industry.

The limited number of the proposed works in this domain opens various research directions to address different issues related to AC methods with the current generation of networking environments. Accordingly, based on HEAD metamodel various new research opportunities can be developed, for example, development of intelligent algorithms for data collection, prediction, and decision-making; development and emergence of new dynamic,

interoperable, and intelligent AC models; and many other directions based on a new and distinct concept. Some of the new opportunities and research directions are explained in chapter 5. In summary, there are no similar project that tackle this level of originality in this domain, and this project would be a first step in the industry’s shift towards IT security "4.0". As well, our proposed metamodel is very influential and this is reflected in the dissemination and transfer of knowledge via journal articles and participation in conferences and popular science activities.

In Table 3, we summarize the time frames for each phase during our work, with the various scientific papers that are published in international conferences and journals.

Table 3
The timeline



1.8 THE STRUCTURE OF THE THESIS

The remainder of this thesis is organized into five chapters. Each of them deals with a key element and is presented in the form of a journal article. In the following, we briefly present the content of each chapter.

In Chapter 2, we review the proposed AC metamodels and their implementation scenarios, we analyze them, their objectives, their limitations, and present current research issues and open questions that still need to be addressed.

In Chapter 3, we propose the HEAD metamodel and explain its characteristics. We use Eclipse (xtext) to define the DSL of the metamodel, then show how it is possible to use this DSL to derive various instances of AC models. We illustrate our approach with several successful instantiations for various models and hybrid models. Additionally, we explain how different static and dynamic AC policies can be expressed using its components, and we provide some examples to show how some of the derived models can be implemented to generate AC policies.

In Chapter 4, We present two case studies inspired by the computing environment of Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada; the first is for ITMI's local (non-IoT) environment, and the second for ITMI's IoT environment. The main objective is to show how HEAD metamodel can be adapted to specify and enforce the needed AC policies in different computing environments.

In Chapter 5, we explain the distinct design of the HEAD metamodel, then describe the remaining steps and how they can be employed to develop more advanced features in order to open new opportunities and answer the various challenges of technology progression and the impact of the COVID-19 pandemic in the domain. Subsequently, we present a novel approach in five main phases to help design secure systems that comply with the organizational security policies that are related to AC based on HEAD metamodel.

Finally, in Chapter 6, we present the general conclusion with the different contributions of this research work and the existing limitations of the HEAD metamodel. Moreover, we present the future perspectives we intend to undertake to address these limitations, in addition to the new opportunities and the various research directions which would enhance our HEAD AC metamodel to meet the expected AC requirements with the presence of various technology trends.

CHAPTER 2

ACCESS CONTROL METAMODELS: REVIEW, CRITICAL ANALYSIS, AND RESEARCH ISSUES

Published in Journal of Ubiquitous Systems & Pervasive Networks, 2021

Volume-16 , Issue 2, pp. 93 - 102; doi: 10.5383/JUSPN.16.02.006

Abstract: Several techniques have been employed to ensure security and privacy in various computing environments, and access control (AC) is one of the essential security requirements in this domain, especially for recent networking environments such as the internet of things (IoT), cloud computing, etc. The new networking environments are emerging and releasing new prospects to traditional information systems by merging new technologies and services for seamless access to information sources at anytime and anywhere. This emergence opens new threats to information security and new challenges to controlling access to resources. Moreover, the continuous technology upgrades and the diversity of AC models force the need to find AC metamodels with a higher level of abstraction to serve as unifying frameworks for specifying and enforcing AC policies. AC metamodels are proposed to encompass AC features and are used to derive various instances of AC models and methods. In this chapter, we review the proposed AC metamodels and their implementation scenarios, their objectives, and their limitations, then we analyze them to find their effectiveness in the light of new technologies. Consequently, the limited works of the proposed AC metamodels in this domain and the urgent need to develop advanced AC methods open various research issues and raise several questions in this domain that have not been addressed yet.

Résumé: Plusieurs techniques ont été employées pour assurer la sécurité et la confidentialité dans divers environnements informatiques, et le contrôle d'accès (CA) est l'une des exigences de sécurité essentielles dans ce domaine, en particulier pour les environnements de réseau récents tels que l'internet des objets (IdO), le cloud computing, etc. Les nouveaux environnements réseau émergent et offrent de nouvelles perspectives aux systèmes d'information traditionnels en fusionnant de nouvelles technologies et de nouveaux services pour un accès transparent aux sources d'information à tout moment et en tout lieu. Cette émergence ouvre de nouvelles menaces pour la sécurité de l'information et de nouveaux défis pour contrôler l'accès aux ressources. De plus, les mises à niveau technologiques continues et la diversité des modèles CA obligent à trouver des métamodèles CA avec un niveau d'abstraction plus élevé pour servir de cadres unificateurs pour spécifier et appliquer les politiques CA. Les métamodèles CA sont proposés pour englober les caractéristiques CA et sont utilisés pour dériver diverses instances de modèles et de méthodes CA. Dans ce chapitre, nous passons en revue les métamodèles CA proposés et leurs scénarios de mise en œuvre, leurs objectifs, leurs limites, puis nous les analysons pour trouver leur efficacité à la lumière des nouvelles technologies. Par conséquent, les travaux limités des métamodèles CA proposés dans ce domaine et le besoin urgent de développer des méthodes CA avancées ouvrent divers problèmes de recherche et soulèvent plusieurs questions dans ce domaine qui n'ont pas encore été abordées.

Access Control Metamodels: Review, Critical Analysis, and Research Issues

Nadine Kashmar^{a*}, Mehdi Adda^a, Hussein Ibrahim^b

^aUniversité du Québec à Rimouski, Rimouski, Canada, QC G5L 3A1

^bInstitut Technologique de Maintenance Industrielle, Sept-Îles, Canada, QC G4R 5B7

Abstract

The new generation of networking environments such as the internet of things (IoT), cloud computing, etc. is emerging and releases new prospects to traditional information systems by merging new technologies and services for seamless access to information sources at anytime and anywhere. Concurrently, this emergence opens new threats to information security and new challenges to controlling access to resources. To ensure security, several techniques have been employed, and access control (AC) is one of the essential security requirements especially for recent networking environments. Various authentication and AC methods are proposed to enforce AC policy and to prevent any unauthorized access to logical/physical assets. The continuous technology upgrades and the diversity of AC models force the need to find AC metamodels with a higher level of abstraction that serves as a unifying framework for specifying any AC policy. AC metamodels are proposed to encompass AC features and are used to derive various instances of AC models and methods. In this paper we review the proposed AC metamodels and their implementation scenarios, we analyze them, their objectives, their limitations, and present current research issues and open questions that still need to be addressed.

Keywords: Access control, metamodels, IoT, Industry 4.0, security and privacy, security policy

1. Introduction

The importance of security, data protection, and privacy requirements increases with the massive presence and integration of new paradigms and technologies, such as cloud computing and the Internet of Things (IoT), also with the deployment of digital and intelligent solutions based on the industry 4.0 concept [1, 2]. To contain and mitigate the impact of cyberattacks, several techniques have been employed, and access control (AC) is one of the essential solutions for privacy settings to measure and optimize IT security [3] in IoT [4], cloud computing [5], social networks [6] and other fields. Access control methods are implemented to control what users can access, when, and how by enforcing AC policy to prevent any unauthorized access for logical or physical assets. In any organization (or industry sector) there might be different types of policies such as: password policy, network access policy, remote access policy, etc., they are defined by managers

and system administrators based on the rules and the guidelines of the organization.

To enforce organizational policies, various AC models are developed such as are Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Organization Based Access Control (OrBAC), and Attribute-Based Access Control (ABAC) [7–9]. To enhance AC methods, various hybrid models are implemented by combining features of two or more AC models. Despite the advantages of the common AC models in controlling access in various computing environments, they also have various limitations. Moreover, with the emergence of highly dynamic environments, especially with the concept of industry 4.0 and IoT applications, it is realized that AC models (also hybrid models) have reached their limits. They no longer meet the increasing demand for privacy and security standards with the widespread of devices and resources [9]. This reality urges the need to find more advanced AC methods and develop AC metamodels with advanced features for specifying and enforcing different AC policies [10–12]. AC metamodels

*Corresponding author. Tel.: +14188338800

Fax: +141883311; E-mail: nadine.kachmar@gmail.com

© 2021 International Association for Sharing Knowledge and Sustainability.

DOI: 10.5383/JUSPN.03.01.000

are used to derive various instances for the common AC models, hybrid models, and other AC methods. Note that, in [8] we present a preliminary survey for the commonly used AC models with some proposed AC metamodels, then raise some questions in this domain. Fig.1 summarizes the aim of AC metamodels.

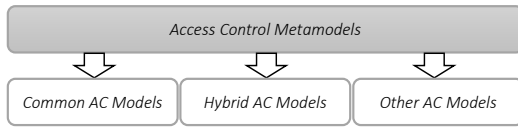


Fig. 1. The aim of AC metamodels

The objective of this paper is to present a literature review and investigate the state-of-the-art of AC metamodels, find out their limitations in the presence of new technologies, and determine the various research issues in this domain, then raise some essential research questions. This review can be considered as a step towards developing a new generic and dynamic AC metamodel with advanced features for IoT and non-IoT systems. In this paper, we provide a detailed literature review for the existing AC metamodels with discussion and critical analysis. The contribution of this paper can be summarized as follows:

- Reviewing recent studies of AC metamodels by providing a summary of the objectives of each study.
- Analyzing and criticizing the proposed AC metamodels.
- Explaining their limitations and why they are not effective in the presence of new technologies and for future upgrades.
- Determining different research issues in this domain and raise some essential research questions.

The remainder of this paper is organized as follows: Section 2 summarizes the existing AC models. Section 3 presents the state-of-the-art of the proposed metamodels, their objectives, and their limitations. Discussion and critical analysis and common limitations for the proposed metamodels are presented in section 4. Current research issues and open questions are proposed in section 5. Section 6 concludes this paper with future perspectives.

2. Access Control Models

In any computing environment subjects request permission (read, write...) to access some objects (file, class...). For this purpose, the defined AC policy that is formally represented by an AC model is enforced to control what objects a subject (user) can access when and how. A subject is allowed to perform some operation(s) on an object or denied accessing this object based on the defined access rights that are granted to him. An access right or a privilege definition might have the form (u, ar, o), which means a subject (u) has an access right (ar) to an object (o), another defined form is (ar, o), a capability of u or referred to as permission of u [7, 13]. AC policies might have the following form:

*Allow/Deny doctors, nurses, etc. to... and...
if... and/or... Except...when...*

2.1. The Common Access Control Models

Access control is the process of restricting access to a place or resource based on a defined set of security policies. Security

policies are the definition of rules that must be regulated in an organization, and they are usually defined by managers and system administrators. An AC model is a framework for making authorization decisions based on the defined AC policies, and an AC mechanism is the process of enforcing AC policy and translating user's access request [7, 8]. Despite the presence of several papers reviewing the state-of-the-art of the common AC models [8, 14], in this paper, we summarize them since their features are used in building different AC metamodels.

2.1.1. Discretionary Access Control (DAC)

DAC model was first introduced in the 1960s. The system protection notion includes three major components: objects, subjects, and permission. DAC is defined as a user-centric model where a file owner controls permissions that are given to other users requiring access to that file. The AC rights of subject(s) over object(s) are specified by Access Control Matrix (ACM). Other ACM variations include Capability Lists (CLs) and Access Control Lists (ACLs). Lampson and Harrison Ruzzo Ullman (HRU) are two variants of DAC model. It is very flexible to assign access rights between subjects and objects, and it is provided with operating systems to authenticate system administrators and users using some procedures, for example, passwords [7, 8].

2.1.2. Mandatory Access Control (MAC)

MAC model was presented in the 1970s. In MAC, users cannot define AC rights by themselves, AC policy is managed in a centralized manner. It is based on the concept of security levels associated with each subject and object where permissions and actions are derived. These levels have hierarchical and nonhierarchical components. Hierarchical components include unclassified, confidential, secret, and top-secret types. Nonhierarchical components represent a set of categories where labels are used to indicate security levels for objects classification and subjects clearance. Its key components are a set of objects, a set of subjects, permissions, and security levels. Bell and LaPadula (BLP) and BIBA (Kenneth J. Biba) are two MAC variants [7–9].

2.1.3. Role-Based Access Control (RBAC)

RBAC was proposed in 1992 as an alternative approach to MAC and DAC. It is based on several entities: users, roles, permissions, actions, operations, and objects. A role is a group of permissions to use object(s) and perform some action(s), it can be associated with several users. Also, users can be assigned to several roles (e.g., doctor). The aim of RBAC is to facilitate the administration of AC policy, it controls user's access to information through roles for which a user is authorized to perform [7–9]. RBAC example can be represented in the hospital system where there exists a variety of relations between doctors, nurses, etc. Only the system administrator has the right to control system security and assign roles to users [15]. Flat RBAC (RBAC0), Hierarchical RBAC (RBAC1), Constrained RBAC (RBAC2), and Symmetric RBAC (RBAC3) are RBAC variants [9].

2.1.4. Organization-Based Access Control (OrBAC)

OrBAC model was first presented in 2003 to solve some problems in DAC, MAC, and RBAC, by finding a more abstract control

policy. Each organization is comprised of a structured group of subjects having certain roles or entities. OrBAC exceeds the concept of only granting permissions to subjects, it also addresses the concept of prohibitions, obligations, and recommendations. A role may have a permission, prohibition, or obligation to do some activity on some view given an associated context. OrBAC is composed of seven entities that are distributed in two levels: the role, activity, and view are found in the abstract level, and the subject, action, and object entities in the concrete level; the context lies between the two levels to express dynamic rules [8, 13].

2.1.5. Attribute-Based Access Control (ABAC)

This is the latest AC model development, its concepts have paralleled that of RBAC. It has the ability to support dynamic attributes and its benefits in managing authorizations. It has three types of attributes: object, subject, and environmental (e.g., the current time, day of the week, etc.) attributes. It allows or denies user requests based on some attributes for users, objects, and environment, and a set of policies that are specified in terms of those attributes and conditions. It is dynamic since it uses attributes to determine access decisions, and subjects are enabled to access a wider range of objects without specifying individual relationships between each subject and each object. AC permissions are evaluated at the time of the actual user’s request which offers a larger set of possible combinations of variables to reflect a larger set of possible rules to express policies. Two standards that widely address the ABAC framework are: The Extensible AC Markup Language (XACML) and Next Generation AC (NGAC) with AC facility for applications and other important features [7, 8].

Fig. 2 summarizes the historical evolution of common AC models. Also, various models extensions are proposed in the literature to enhance their features along with the technology progressions, for example, Integrity-OrBAC (I-OrBAC) [16] and Multi-Organization Environments called Trust-OrBAC [17] are two OrBAC extensions, a Higher-order Attribute-Based Access Control Model (HoBAC) [18] is an ABAC extension, and others.

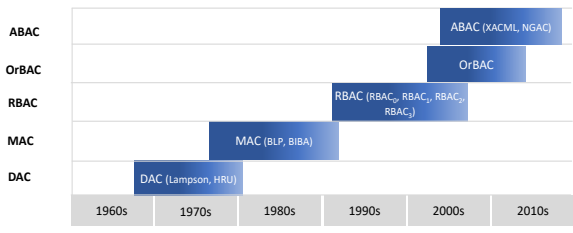


Fig. 2. Historical Evolution of common AC Models

2.2. Enhancing Features of Access Control Methods

The need to use enhanced AC methods imposes the necessity to find models with combined features from two or more models called hybrid AC models. Various hybrid AC models are presented in the literature, for example, hybrid RBAC and ABAC.

In RBAC it is difficult to set up an initial role structure in rapidly changing environments also it does not support dynamic attributes, Kuhn et al. in [19] address the idea of adding attributes

to RBAC. The aim is to find a model that supports dynamic attributes, especially in organizations to handle relationships between roles and attributes to provide better AC features in dynamic environments. As well, Rajpoot et al. in [20] propose Attribute Enhanced RBAC (AERBAC) model to enhance features from both RBAC and ABAC because both have complementary features to each other. Moreover, in [21] authors state that the integration of RBAC and ABAC still have some shortcomings in terms of AC flexibility and decision efficiency. For this purpose, they propose a more fine-grained, flexible, and efficient RABAC (RBAC/ABAC) model. To increase the flexibility of RBAC, an Emergency RBAC (E-RBAC) approach is proposed in [22]. In [23], an ABAC scheme integrated with controlled access delegation capabilities for collaborative e-Health environments is proposed.

2.3. Some Limitations of the Common AC Models

Table 1 summarizes the limitations common AC models [7, 8].

Table 1. Limitations of the common AC models

Model	Limitation(s)
DAC	<ul style="list-style-type: none"> in large systems granting permissions between subjects and objects are time consuming and difficult to manage. granted user allow others to read a file without asking the owner.
MAC	<ul style="list-style-type: none"> security levels assignment places limits on user actions which prevents dynamic modification of original policies. is difficult to implement due to dependence on trusted components.
RBAC	<ul style="list-style-type: none"> poor support for dynamic attributes (e.g., time of day) in large systems role inheritance and the need for customized privileges make administration potentially heavy.
OrBAC	<ul style="list-style-type: none"> poor support for dynamic attributes (e.g. time of day). inflexible in rapidly changing IT environments. it has some vulnerabilities to some kinds of attacks. e.g. covert channels.
ABAC	<ul style="list-style-type: none"> its implementations require significant time to run. often not possible to compute the set of users that may have access to a given resource. difficult to efficiently calculate the resulting set of permissions for a given user.

3. Access Control Metamodels

Access control models must consider the continuous developments and changes to answer the needed security requirements. The new technology trends (cloud computing, IoT, social networks...), the variety of platforms and applications, users’ types, etc. reflect the difficulty of controlling secure and private access to the needed resources in different areas. All this makes AC models and even combining some of them (hybrid models) are insufficient to handle the needed target. This fact forces the need to find models with a higher level of abstraction, called AC metamodels, that serve as unifying frameworks for specifying and enforcing any AC policy [8, 24]. However, metamodels are presented in the literature to concurrently handle multiple AC models. Different AC models can be derived as special instances from the same metamodel.

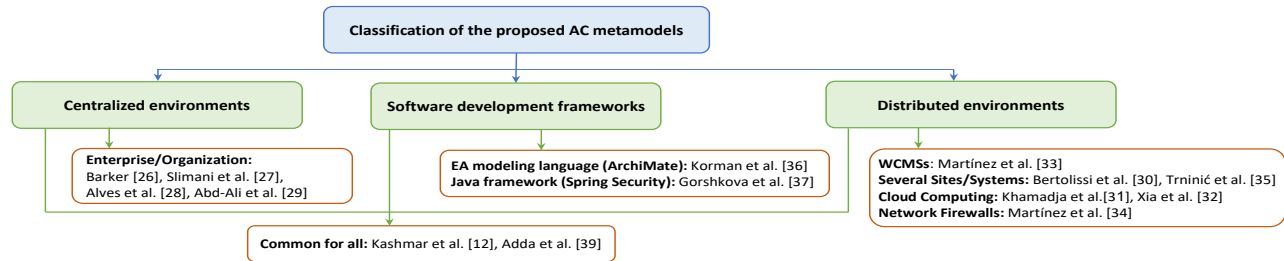


Fig. 3. Classification for the proposed AC Metamodels

3.1. General Definition of Metamodel Concept

The metamodel is defined as textual, graphical/visual, or formal representation of concepts in a certain domain and how they are linked together, these concepts might be rules, guidelines, etc. for an institution or organization. Moreover, metamodeling is defined as the modeling of a model to describe the permitted structure to which models must adhere. Also, models and metamodels need adaptable supporting tools due to changing requirements and policies. There are different metamodeling tools and languages such as Unified Modeling Language (UML), Eclipse Modeling Framework (EMF), ArchiMate, MetaEdit, etc. [8].

3.2. State-of-the-Art

Several AC metamodels are proposed for centralized computing environments, distributed computing environments, and for software development frameworks (Fig. 3). To the best of our knowledge, there is a limited number of recent works proposed in this domain. In this paper, we review them within a decade, analyze them to find if they are effective to follow technology upgrades. Ferraiolo et al. in [25], revise some concerns and raise some questions related to AC policy enforcement and focuses on the important role a metamodel might play when achieved.

3.2.1. Centralized Environments

To address the questions raised in [25], a paper published by Barker [26] demonstrates that multiple models can be derived as special cases from a defined AC metamodel called Category Based Access Control (CBAC) metamodel. A category is interpreted as a synonym for a role, a class, a group, security levels, etc. where entities (e.g., subjects) may be assigned. CBAC metamodel includes features of MAC, DAC, and RBAC where a wider range of constraints may be expressed based on it. Barker demonstrates that the presented AC models in the literature are based on a limited and small number of primitive notions. These notions are related to the concept of categories, relationships between categories and between categories and principals, and modalities. However, AC primitives are given a more general interpretation to allow developing many AC models by combining the primitives of AC models, hence a wider range of constraints may be expressed.

In [27], Slimani et al. extend Barker's metamodel to support resource and action hierarchies. They propose a Unified Access Control Modeling Language (UACML) to provide support for hybrid AC policies by allowing categories to be associated with other categories and finding hierarchical relationships between them. A CBAC metamodel extension is proposed by Alves et al. in [28] to expand a general notion of obligation for the existing AC

models and study the interaction between obligations and permission. The aim of their approach is to allow security administrators to check the consistency of a policy combining authorizations and obligations.

Furthermore, Abd-Ali et al. in [29] propose an integration metamodel for hybrid policies to concurrently handle multiple models. Their idea is based on the concept of abstracting each AC model (e.g., RBAC metamodel), then including a special element named DecisionHandler to determine AC decision. The AC decision depends on more than one AC metamodel (CW metamodel, BLP metamodel ...). Their approach depends on the idea of clustering the DecisionHandler instances of a hybrid policy, then apply them to combining algorithms (ComAl) to find one AC decision as output in response to multiple AC decisions as input. The integration of several AC models is based on a tree structure of AC decision systems named Ascending Decision Tree (ADT). ADT nodes are DecisionHandler instances or ComAl nodes. ADT has a unique root node and the decision it returns is the decision of the whole tree carrying out the hybrid AC policy.

3.2.2. Distributed Environments

Another approach based on CBAC metamodel is proposed by Bertolissi et al. in [30] for distributed environments that consist of several sites. A system of several sites might be composed of several policies at each site, and in the distributed metamodel the request can be passed to other sites and evaluated in a distributed manner. They demonstrate the expressive power of their metamodel by showing how a distributed, dynamic, event-based access control model (DEBAC) can be defined as an instance of the metamodel. In the context of cloud computing, saving data on cloud servers by cloud users raise security challenges to protect sensitive data. In [31] authors states that the classical AC models (DAC, MAC ...) are not adequately expressive for highly flexible and dynamic environments. For this purpose, they present a metamodel approach for cloud computing services called Category Based Access Control (CatBAC) framework, it has two stages at the different organization sites by considering the local constraints of each site. The first is achieved by the cloud provider (abstract stage), and the second is by network administrators (concrete stage). In the abstract level categories are connected to express authorizations and are named abstract authorizations. The concrete level represents AC decisions in relation to concrete level entities, which are subject, resources, action, and context, and are called concrete authorizations. Hence, this AC metamodel allows security administrators in the various company sites to find a concrete model with the constraints and specificities of each site. Xia et al. in [32] propose another metamodel approach to handle

security and privacy in cloud service development and operations, called the Cloud Security and Privacy Metamodel (CSPM). CSPM is proposed to address security and privacy in cloud services via integrating and extending the existing metamodels of cloud security together with newly added concepts.

Moreover, an approach is presented for web services by Martínez et al. in [33] to the representation of Web Content Management System (WCMS) AC policies to ease the analysis and manipulation of security requirements by abstracting them from vendor-specific details. Although AC methods are integrated with most WCMS systems (e.g., Wordpress, Drupal ...), some limitations still exist. For this purpose, the authors' aim to raise the level of abstraction of the AC implementation to be represented according to a vendor-independent metamodel. They propose a WCMS metamodel inspired from the RBAC concept, its abstract representation is developed using Model-Driven Engineering (MDE). The aim of their approach is to automatically extract the AC information in the domain of WCMSs. Also, Martínez et al. in [34] propose a model-driven approach to extract network AC policies enforced by firewalls within a network system. Their concept tackles the problem of filtering the traffic of a network with the presence of several filtering rules due to several firewalls. They suggest raising the level of abstraction of the information contained in the firewall configuration files, hence the AC policy would be easier to understand, analyze and manipulate. A model-driven approach is proposed to extract a model of the AC policy enforced by the firewalls within a network system, it consists of host and connection entities. The former represents a network host, e.g., IP address, and the latter represents connections between hosts, where the port and the protocol are specified to establish connections and specify if the connection is allowed or denied.

Trninić et al. in [35] present a generic AC management infrastructure for a broad set of systems, to provide a general method for specifying AC rules for different AC models. Their approach is based on models at three different abstraction levels defined by Meta-Object Facility (MOF) classification. The AC policy metamodel is defined at level M2 and used to derive different AC models at level M1 (e.g., RBAC). At level M0, PolicyDSL is used to specify concrete AC policies in a system. Their proposed metamodel is a Domain-Specific Language (DSL) with the syntax that is dynamically adapted to system features that are being modeled. Hence, a security expert would be able to express AC policies for a given AC model using the generated DSL.

3.2.3. Software Development Frameworks

Due to the lack of security features in software development frameworks, some metamodel extensions are proposed. In [36], a unified metamodel as a prospective extension for ArchiMate is proposed, the common Enterprise Architecture (EA) modeling language. The aim is to support the development of enterprises by extending their abilities to model authorization and AC in their architectures. They propose an extension to an established EA modeling language. The metamodel is developed based on the conceptual model of ABAC because of its ability to include most of the other AC models, then mapped to ArchiMate to enrich its existing models. Also, Gorshkova et al. in [37] introduce a fine-grained AC model and provide a metamodel extension for the Spring Security framework to meet modern security requirements. Spring Security is one of the major market players of open source security frameworks for Java. Gorshkova et al. focus on the

implementation of authorization frameworks with Java applications, their proposed framework defines a fine-grained extension of RBAC.

3.2.4. Any Computing Environment

The proposed metamodels reflect the importance of constructing more robust AC models in all computing environments, especially with the presence of heterogeneous technologies and platforms [38]. For this purpose, we propose a new generic AC metamodel approach in [12], it includes all AC models features by unifying a common set of AC concepts which can be used to instantiate the needed components and derive various instances of different AC models; also it can be used as a base to construct other essential metamodel features (section 5). Our approach is proposed for all computing environments and its components can be integrated with frameworks to support AC features. In the same way, Adda et Aliane proposed in [39] a generic ABAC AC model that is suitable for all computing environments.

Table 2 summarizes the proposed AC metamodels and their features.

4. Discussion and Critical Analysis

As shown in Table 2, AC metamodels are constructed based on some features of AC models where various models instances can be derived from them. They are defined as textual or visual, and some of the used tools are UML, Eclipse, and Java. Some of the used modeling languages are xtext, spring expression, etc. However, based on the historical evolution of AC methods, Fig.4 illustrates the era of developing AC metamodels. Some meta-

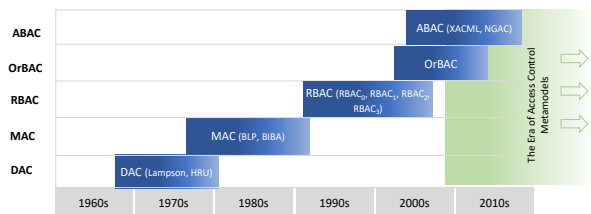


Fig. 4. The Era of Access Control Metamodels

models are proposed as generic, unifying, hybrid, and metamodel extensions for different distributed, centralized environments, and software development frameworks. Hence:

- 1- Some AC metamodels are constructed based on features of some AC models, and the only AC model(s) (also hybrid) instance(s) that can be derived are the one(s) that are employed in the core structure, for example, [27] and [29]. These metamodels are proposed as *Hybrid Metamodels*.
- 2- Some frameworks (for example, Drupal, ArchiMate, Spring Security, Network Firewalls ...) are extended to support AC features of one or more AC models, and the extracted AC policies belong to the model(s) that are used to extend the main framework, for example, [33, 34, 36, 37]. These metamodels are proposed as *Metamodel Extensions*.
- 3- Some AC metamodels are constructed based on a general notion that encompasses some (or all) AC features for some (or all)

Table 2. Summary of the Proposed Access Control Metamodels

ref.	Author	Year	Proposed for	Metamodel	Visual rep. Y/N	Tool	Type	Based on	instance(s)	Modeling lang.
Proposed AC metamodels for Centralized Environments										
[26]	Barker	2009	Enterprise	Barker's Metamodel	No	n/a	Unifying Metamodel	CBAC	RBAC,MAC	Rule/Logic Language
[27]	Slimani et al.	2011	Enterprise	UACML Metamodel	Yes	UML	Hybrid Metamodel	CBAC and Hybrid models	Group based, MAC, RBAC, hybrid model	Object constraint language (OCL)
[28]	Alves et al.	2014	Enterprise	Obligations in CBAC Metamodel	No	n/a	Metamodel Extension	CBAC	CBAC	rewrite-based operational semantics
[29]	Abd-Ali et al.	2015	Enterprise	Integration metamodel	Yes	UML	Hybrid Metamodel	CW,BLP,BIBA, RBAC	Hybrid models	First-order logic
Proposed AC metamodels for Distributed Environments										
[30]	Bertolissi et al.	2014	Distributed system of several sites	Distributed Metamodel	No	n/a	Generic Metamodel	CBAC	CBAC	rewrite-based operational semantics
[31]	Khamadja et al.	2013	Cloud Computing	CatBAC metamodel	Yes	UML	Generic Metamodel	CBAC	Hybrid models	First-order logic
[32]	Xia et al.	2018	Cloud services	cloud security & privacy (CSPM)	Yes	UML	Metamodel Extension	n/a	n/a	UML
[33]	Martinez et al.	2013	WCMSs	WCMS Metamodel	Yes	MDE	Metamodel Extension	RBAC	RBAC	UML
[34]	Martinez et al.	2012	Network Firewalls	Network Connection	Yes	Eclipse	Metamodel Extension	Network Firewalls	RBAC, OrBAC	Xtext
[35]	Trinić et al.	2013	Set of systems	PolisyDSL	Yes	UML	Generic Metamodel	n/a	RBAC	Textual DSL
Proposed AC metamodels for Software Development Frameworks										
[36]	Korman et al.	2016	Enterprise Architecture framework	Unified Metamodel	Yes	ArchiMate	Metamodel Extension	DAC,BLP,Biba, CW, RBAC, ABAC	DAC,BLP,CW, RBAC, ABAC	ArchiMate
[37]	Gorshkova et al.	2017	Enterprise application framework	Spring security framework	Yes	Java-ORM	Metamodel Extension	RBAC	RBAC	Spring expression lang.(SpEL)
Proposed AC metamodels for any Computing Environment										
[39]	Adda et al.	2020	any computing environment	Generalization of ABAC	Yes	UML	ABAC Metamodel	ABAC	ABAC models	UML
[12]	Kashmar et al.	2021	any computing environment	Generic with unified set of AC concepts	Yes	UML	Generic Metamodel	common models	AC common models & hybrid models	UML

models. Based on this metamodel, AC model instance(s) can be derived, for example [12, 26, 30, 31, 35]. These metamodels are proposed as *Generic Metamodels*.

4- Some of the existing AC metamodels are augmented with additional features to reflect a larger and more definitive set of possible rules to express AC policies, for example, [28, 32, 39]. This type of metamodels is proposed as *Metamodel Extensions*. Hence, the proposed works of AC metamodels in the literature can be classified into two concepts:

- In (1) and (3) the aim is to find a generic metamodel that encompasses most AC features where various AC models (and hybrid models) can be derived, Fig. 5 illustrates the idea of generic metamodels. With regard to this definition of generality, the existing AC metamodels are not generic¹, they have a hybrid structure with some AC features rather than a generic metamodel. This hybrid structure is employed to derive some AC models where their features are employed in the core structure. As shown in Fig. 5, if the metamodel includes features of DAC, MAC, and RBAC models, then the instances that can be derived are DAC, MAC, RBAC, and their combinations (hybrid models based on the existing features, e.g., hybrid MAC/ RBAC).

- In (2) and (4) the aim is to enhance features of the existing frameworks/metamodels by extending them to support AC features and express more AC policies, Fig. 6 illustrates the idea of metamodel extension where AC features are added to the core metamodel/framework to allow defining (more) AC policies. But the structure of the proposed AC metamodels is not extended, for example, no new components or attributes are defined, but AC features are added to the core metamodel structure. As shown in Fig. 6, AC features are implemented and added to an existing AC metamodel or framework to enhance its features and allow defining more AC policies. Then, the extended AC metamodel (or framework) can be used to derive various instances of AC models based on the features which are added to the core metamodel (or framework) structure.

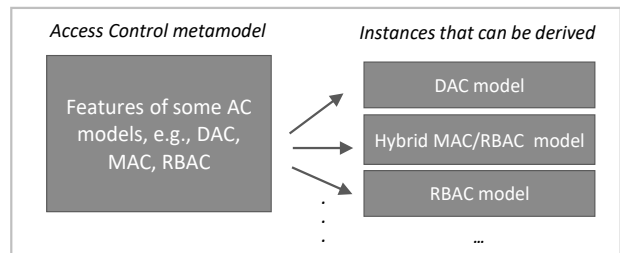


Fig. 5. Illustration for the concept generic metamodel

¹ Except the proposed metamodel in [12] since it includes most of the features of common models

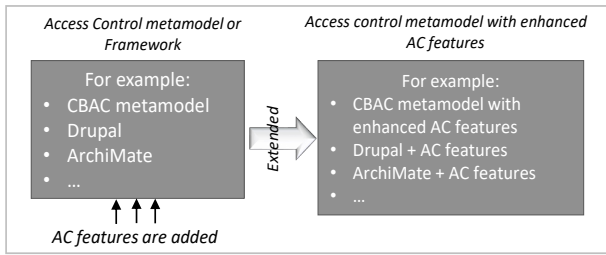


Fig. 6. Illustration for the concept of metamodel extension

However, the presented AC metamodels come with some advantages, and several combined features from AC models are implemented to enhance AC methods. But they also have several limitations especially in the light of new technologies, sections 4.1 and 4.2.

4.1. Limitations of the proposed AC metamodels

In this section, we highlight the limitations for each of the proposed AC metamodels to check out their effectiveness in the presence of new technologies. However, the proposed metamodel extension for Drupal framework in [33] is RBAC-inspired, it is for web contents and it is well known that such environment is rich of variants (time, system updates ...), in this metamodel extension the notion of variable attributes is not considered. Although authors in [30] provide a comprehensive theoretical description for their approach which is considered generic with no real case studies are explained or implemented. Hence, their proposed metamodel is still within the theoretical frame. Also, Khamadja et al. in [31] propose CatBAC metamodel to support various AC models in Cloud with no case study or testing result. In [31] and [32] authors have not explained or mentioned how access can be controlled in the context of several heterogeneous clouds (multi-clouds). In [31], authors mention that their proposed solution does not completely consider dynamic constraints, and this important issue should be considered to provide a general method for specifying AC rules for different AC models. Korman et al. in [36] present some of their metamodel limitations, such as the proposed approach misses the concept of logging, and the difficulty for potential implementation of automated analytical capabilities of the unified metamodel. In [27] and [29] the proposed metamodels are based on the concept of combining some models then instantiate one or more AC model(s) based on a hybrid structure, hence they are general templates to derive some AC models that are employed in the core structure rather than a metamodel. Barker’s approach [26] lacks the support of resource hierarchies and action hierarchies which are useful to specify high-level access rules [27]. The extension of CBAC metamodel in [28] is proposed to accommodate a general notion of obligation, authors adjust the notion of events and describe a set of core axioms for defining obligations with some examples to specify dynamic policies. Nevertheless, they have not explained how their approach could be dynamic in distributed contexts that are rich in events and variable attributes. The proposed metamodel extensions in [33, 34, 36, 37] tackle specific projects or frameworks to support some AC features without explaining how these extensions can be upgraded due to unexpected updates or changes, especially in distributed environments. The model proposed in [39] is limited to generate ABAC AC models. Moreover, although our proposed

AC metamodel in [12] is promising, many other phases are still missing and need to be handled and implemented, for example, developing DSL, a detailed case study, etc. Table 3 summarizes the objective(s) and limitation(s) for each of the proposed metamodels. Despite the proposed AC metamodels have gained the attention of researchers for a decade, they have common limitations. These limitations cannot be ignored, especially, with recent computing environments which are open to all kinds of attacks and threats.

4.2. The Common Limitations

Even with the advancements of implementing AC metamodels in various scenarios, each particularly has its limitation(s) in addition to some common limitations. They all lack some essential characteristics and can be enumerated as follows:

- Each metamodel is itself a case and does not encompass a general base concept to derive various instances for all AC models. In other words, they are planned for dedicated scenarios or case studies based on some features of AC models;
- They do not support the ability to define various types of attributes. So, they are not dynamic enough to follow the continuous technology upgrades.
- Neither the generic nor extended proposed metamodels is enough to address the needed target of enforcing AC policy, especially with the current technologies and continuous upgrades;
- No provided explanations about how the derived models could collaborate within the same computing architecture e.g., IoT;
- An essential aspect is not considered in all of the presented AC metamodels which is the migration of AC policy from one AC model to another. Having a metamodel should make it possible to translate an existing AC policy between the different AC models covered by the metamodel.

Fig. 7 summarizes the common limitations that should be addressed in the proposed AC metamodels. Accordingly, we are

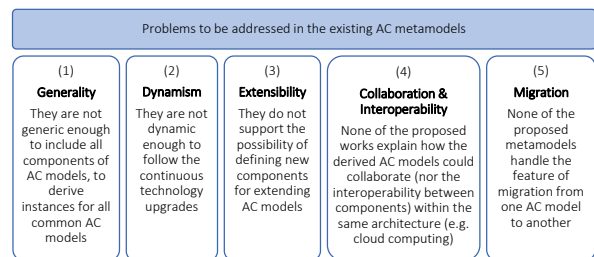


Fig. 7. The common limitations in the existing AC metamodels

constructing a unified and generic AC metamodel [12] that considers the continuous technology progressions, the variety of information systems, and the heterogeneity of AC models.

5. Research issues and open questions

The definition of security policies with the current computing environments, especially IoT, involves complexities and difficulties due to the following facts:

Table 3. Objective(s) and Limitation(s) of The Proposed Access Control Metamodels

Author(s)	Objective(s)	Limitation(s)
Barker [26]	Multiple models can be derived as special cases from CBAC metamodel.	- lacks the support of resource and action hierarchies.
Slimani et al. [27]	To provide support for hybrid AC policies by allowing categories to be associated with other categories and finding hierarchical relationships between them.	- hybrid structure to derive some AC models rather than a metamodel.
Alves et al. [28]	To allow security administrators to check the consistency of a policy combining authorizations and obligations.	- no explanation of how the approach could be dynamic in distributed contexts which are rich of events and variable attributes.
Bertolissi et al. [30]	To provide semantics for distributed AC mechanisms within distributed environments consisting of several sites.	- no real case studies are explained or implemented.
Khamadja et al.[31]	To develop a new cloud computing service named "Access Control as a Service".	- no case study or testing result, also they do not explain how access can be controlled in the context of multi-cloud.
Xia et al. [32]	To handle security and privacy in cloud service development and operations.	- have not explained how access can be controlled in the context of multi-cloud.
Martinez et al. [33]	To ease the analysis and manipulation of security requirements in WCMs.	- the notion of variable attributes is not considered, also no explanations of how Drupal framework can be upgraded.
Martinez et al. [34]	To extract network AC policies enforced by firewalls within a network system, then AC policy would be easier to understand, analyze and manipulate.	- no explanations of how the extended networks firewall systems can be upgraded.
Abd-Ali et al. [29]	To concurrently handle multiple AC models (CW, BLP, BIBA, and RBAC)	- hybrid structure to derive some AC models rather than a metamodel.
Trninić et al. [35]	to allow a security expert to express AC policies for a given AC model.	- does not consider dynamic constraints.
Korman et al. [36]	To provide support for architectures of enterprises by extending their abilities to model authorization and AC in their frameworks.	- difficulty for potential implementation of automated analytical capabilities, also no explanations of how the extended ArchiMate framework can be upgraded.
Gorshkova et al. [37]	To provide a metamodel extension for Spring Security framework to meet modern security requirements.	- they extend Spring Security framework to support some AC features without explaining how these extensions can be upgraded.
Adda et al. [39]	To provide a generic ABAC metamodel to generate a wide variety of AC models related to ABAC.	- limited to ABAC models.
Kashmar et al. [12]	To provide a generic AC metamodel with a unified set of AC concepts	- no case study or testing result, also no explanation of how access can be controlled in distributed environment.

- the heterogeneity of security strategies for information systems such as centralized, decentralized, or both
- the diversity of AC rights which might be raised from different information systems such as allow, deny, mixed, or undetermined, for different units such as subjects, roles, categories, groups, etc.
- the heterogeneity of security policies for different AC models and their extensions.
- the heterogeneity of security elements of various AC models such as objects, subjects, types, relations, etc.
- the heterogeneity of networks, platforms, applications, devices, etc. with multimillions of users

These facts and the complex structure of the recent technologies (cloud computing, IoT, ...) reflect the importance of developing an enhanced AC metamodel approach to adapt the continuous technology progressions and the existing heterogeneities in different domains. Through this review, we can find that there is a limited number of recent research proposals for AC metamodels. Yet, various research is still conducting for the AC metamodeling approach to find a more general metamodel that can be used to dynamically define AC policies.

However, finding a new generic metamodel that includes all AC models features, dynamic, and upgradable is a challenging topic. What makes it a critical need are the following:

- the heterogeneity and complexity in the structures of recent technologies and their environments;
- the continuous upgrades of the new technologies, especially IoT;
- the dynamic requirement for enforcing security issues;
- the need to find the collaboration between various AC models within the same architecture;

- the importance of migrating AC policies from one model to another.

Despite the proposed AC metamodels have some enhanced features, they lack some important characteristics that are essential to the current fact of technologies. Through this review we can find that some issues need to be addressed which are:

5.1. Generality

Generality is the first essential feature that must be considered in developing an AC metamodel. A generic AC metamodel should have the following characteristics:

- includes most of the features of the common AC models;
- can be oriented to derive various AC models and methods, and for specifying any AC policy in centralized and distributed computing environments;
- works as a base to construct other essential characteristics (e.g., a collaboration between AC models).

Note that, we address this issue in [10, 11, 38] then propose our metamodel approach in [12].

5.2. Dynamism

The term dynamism refers to the change within a system, model, etc., and being upgradable due to changing conditions or rules. A generic and dynamic AC metamodel should:

- describe how metamodel properties can be changed or modified over time along with technology progressions, for example, due to the changing environmental conditions.

- allow defining new types of attributes/entities, to describe a larger set of rules to express policies. Hence, various models can be formulated for static and dynamic policy enforcement.
- allow building relationships between its elements and describes the structural changes to reflect its dynamic characteristics.

5.3. Extensibility

Extensibility is the feature of being designed to allow adding new components, for an already defined model, with the relationships between them. Some of the proposed AC methods are based on (or extended from) the common AC models, while others are formulated based on the needed context. This reflects the diversity of the implemented AC models in different fields and the importance of upgrading them to follow technology progressions. The key components for the different AC methods are subjects, objects, actions, security levels, attributes, etc. The existing AC metamodels do not include the possibility of defining new components rather than the defined ones in the core structure. Hence, developing generic and dynamic metamodel is important to extend the existing AC methods, and to formulate and implement new ones.

5.4. Collaboration and interoperability

Collaboration is underlined as a goal for distributed computing environments, in collaborative computing environments, various collections of information systems and technologies are performed to support cooperation between organizations, individuals, etc. In these environments, organizations collaborate from remote locations, and users are allowed/denied to share information, upload content, communicate via applications such as video conferencing. To establish interoperability, various concepts must be studied such as autonomy, dynamism, and heterogeneity of systems, models, etc.; hence computational entities can collaborate to fulfill their mutual goals [40]. Collaborative environments need to control access to their assets to increase working cooperation efficiently and effectively. Finding a general basis for AC metamodel would allow handling multiple models to find advanced security features and operations, which would in turn, permit the collaboration between the obtained models and the interoperability between components of AC models.

5.5. Migration

Another interesting feature, that is missing in current AC metamodels, is the ease of migration from one model to another. In fact, having a metamodel should make it possible to translate an existing AC policy between different AC models covered by the metamodel. However, a metamodel with a generic, dynamic, and extendable structure can be implemented to allow migrating the AC policies from one model to another.

However, in this context we can raise the following questions:

- how a new generic and dynamic AC metamodel that considers the continuous technology progressions can be designed?
- what are the main features, components, etc. this AC metamodel can include?
- how its structure can be developed to handle collaboration/interoperability/extension/migration of AC models?
- how to construct a common set of AC concepts for the heterogeneous AC models?

- how heterogeneous AC models can interact to ensure privacy?

6. Conclusion and Future Perspectives

In this paper, we review and analyze the proposed AC metamodels, explain their objectives, their limitations especially with current technology progressions and upgrades. In this review, we provide a critical analysis, in addition to the potential research issues in this domain. The common limitations, which can also be considered as research issues in this domain, that have not been addressed yet are important to be implemented with the current heterogeneous computing environments.

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number 06351], Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT), and Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

References

- [1] EG Petrakis and Xenofon Koundourakis. ixen: Secure service oriented architecture and context information management in the cloud. *Journal of Ubiquitous Systems and Pervasive Networks (JUSPN)*, 14(2):01–10, 2021.
- [2] Kamalendu Pal and Ansar-Ul-Haque Yasar. Convergence of internet of things and blockchain technology in managing supply chain. *Journal of Ubiquitous Systems and Pervasive Networks (JUSPN)*, 14(2):11–19, 2021.
- [3] Jun Ho Huh, Rakesh B Bobba, Tom Markham, David M Nicol, Julie Hull, Alex Chernoguzov, Himanshu Khurana, Kevin Staggs, and Jingwei Huang. Next-generation access control for distributed control systems. *IEEE Internet Computing*, 20(5):28–37, 2016.
- [4] Sowmya Ravidas, Alexios Lekidis, Federica Paci, and Nicola Zanon. Access control in internet-of-things: A survey. *Journal of Network and Computer Applications*, 144:79–101, 2019.
- [5] Mehdi Sookhak, F Richard Yu, Muhammad Khurram Khan, Yang Xiang, and Rajkumar Buyya. Attribute-based data access control in mobile cloud computing: Taxonomy and open issues. *Future Generation Computer Systems*, 72:273–287, 2017.
- [6] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. *Access Control in Cybersecurity and Social Media*, chapter 4. Université Laval, 2021.
- [7] V.C. Hu, D.F. Ferraiolo, R. Chandramouli, and D.R. Kuhn. *Attribute-Based Access Control*. Artech House Publishers, 2017. ISBN 9781630814960.
- [8] Nadine Kashmar, Mehdi Adda, and Mirna Atieh. From access control models to access control metamodels: A survey. In *Future of Information and Communication Conference*, pages 892–911. Springer, 2019.
- [9] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. A review of access control metamodels. *Procedia Computer Science*, 184:445–452, 2021.
- [10] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. A new dynamic smart-ac model methodology to enforce access control policy in iot layers. In *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the*

- Internet of Things (SERP4IoT)*, pages 21–24. IEEE, 2019.
- [11] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. Smart-ac: A new framework concept for modeling access control policy. *Procedia Computer Science*, 155:417–424, 2019.
- [12] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. Access control metamodel for policy specification and enforcement: From conception to formalization. *Procedia Computer Science*, 184:887–892, 2021.
- [13] Mohammed Ennahbaoui and Said Elhajji. Study of access control models. In *Proceedings of the World Congress on Engineering*, volume 2, pages 3–5, 2013.
- [14] RS Sandhu, EJ Coyne, HL Feinstein, and CE Youman Role-Based. Access control models. *IEEE computer*, 29(2):38–47, 2013.
- [15] Edwin Okoampa Boadu and Gabriel Kofi Armah. Role-based access control (rbac) based in hospital management. *Int. J. Softw. Eng. Knowl. Eng.*, 3:53–67, 2014.
- [16] Abdeljebar Ameziane El Hassani, Anas Abou El Kalam, Adel Bouhoula, Ryma Abassi, and Abdellah Ait Ouahman. Integrity-orbac: a new model to preserve critical infrastructures integrity. *International Journal of Information Security*, 14(4):367–385, 2015.
- [17] Khalifa Toumi, César Andrés, and Ana Cavalli. Trust-orbac: A trust access control model in multi-organization environments. In *International Conference on Information Systems Security*, pages 89–103. Springer, 2012.
- [18] Linda Aliane and Mehdi Adda. Hobac: toward a higher-order attribute-based access control model. *Procedia Computer Science*, 155:303–310, 2019.
- [19] D Richard Kuhn, Edward J Coyne, and Timothy R Weil. Adding attributes to role-based access control. *Computer*, 43(6):79–81, 2010.
- [20] Qasim Mahmood Rajpoot, Christian Damsgaard Jensen, and Ram Krishnan. Integrating attributes into role-based access control. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 242–249. Springer, 2015.
- [21] Hui Qi, Xiaoqiang Di, and Jinqing Li. Formal definition and analysis of access control model based on role and attribute. *Journal of information security and applications*, 43:53–60, 2018.
- [22] Fatemeh Nazerian, Homayun Motameni, and Hossein Nematzadeh. Emergency role-based access control (e-rbac) and analysis of model specifications with alloy. *Journal of information security and applications*, 45:131–142, 2019.
- [23] Harsha S Gardiyawasam Pussewalage and Vladimir A Oleshchuk. Attribute based access control scheme with controlled access delegation for collaborative e-health environments. *Journal of information security and applications*, 37:50–64, 2017.
- [24] Saïd Assar. Meta-modeling: concepts, tools and applications. In *IEEE RCIS'15: 9th International Conference on Research Challenges in Information Science*, 2015.
- [25] David Ferraiolo and Vijay Atluri. A meta model for access control: why is it needed and is it even possible to achieve? In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 153–154, 2008.
- [26] Steve Barker. The next 700 access control models or a unifying meta-model? In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 187–196, 2009.
- [27] Nadera Slimani, Hemanth Khambhammettu, Kamel Adi, and Luigi Logrippo. Uacml: Unified access control modeling language. In *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, pages 1–8. IEEE, 2011.
- [28] Sandra Alves, Anatoli Degtyarev, and Maribel Fernández. Access control and obligations in the category-based metamodel: a rewrite-based semantics. In *International Symposium on Logic-Based Program Synthesis and Transformation*, pages 148–163. Springer, 2014.
- [29] Jamal Abd-Ali, Karim El Guemhioui, and Luigi Logrippo. A meta-model for hybrid access control policies. *JSW*, 10(7):784–797, 2015.
- [30] Clara Bertolissi and Maribel Fernández. A metamodel of access control for distributed environments: Applications and properties. *Information and Computation*, 238:187–207, 2014.
- [31] Salim Khamadja, Kamel Adi, and Luigi Logrippo. Designing flexible access control models for the cloud. In *Proceedings of the 6th International Conference on Security of Information and Networks*, pages 225–232, 2013.
- [32] Tian Xia, Hironori Washizaki, Takehisa Kato, Haruhiko Kaiya, Shinpei Ogata, Eduardo B Fernandez, Hideyuki Kanuka, Masayuki Yoshino, Dan Yamamoto, Takao Okubo, et al. Cloud security and privacy metamodel. In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, pages 379–386. SCITEPRESS-Science and Technology Publications, Lda, 2018.
- [33] Salvador Martínez, Joaquin Garcia-Alfaro, Frédéric Cuppens, Nora Cuppens-Boulahia, and Jordi Cabot. Towards an access-control metamodel for web content management systems. In *International Conference on Web Engineering*, pages 148–155. Springer, 2013.
- [34] Salvador Martínez, Jordi Cabot, Joaquin Garcia-Alfaro, Frédéric Cuppens, and Nora Cuppens-Boulahia. A model-driven approach for the extraction of network access-control policies. In *Proceedings of the Workshop on Model-Driven Security*, pages 1–6, 2012.
- [35] Branislav Trninić, Goran Sladić, Gordana Milosavljević, Branko Milosavljević, and Zora Konjović. Policydsl: Towards generic access control management based on a policy metamodel. In *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*. IEEE, 2013.
- [36] Matus Korman, Robert Lagerström, and Mathias Ekstedt. Modeling enterprise authorization: a unified metamodel and initial validation. *Complex Systems Informatics and Modeling Quarterly*, (7):1–24, 2016.
- [37] Ekaterina Gorshkova, Boris Novikov, and Manoj Kumar Shukla. A fine-grained access control model and implementation. In *Proceedings of the 18th International Conference on Computer Systems and Technologies*, pages 187–194, 2017.
- [38] Nadine Kashmar, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. Deriving access control models based on generic and dynamic metamodel architecture: Industrial use case. *Procedia Computer Science*, 177:162–169, 2020.
- [39] Mehdi Adda and Linda Aliane. Hobac: fundamentals, principles, and policies. *Journal of Ambient Intelligence and Humanized Computing*, 11(12):5927–5941, 2020. . URL <https://doi.org/10.1007/s12652-020-02102-y>.
- [40] Toni Ruokolainen. Modelling framework for interoperability management in collaborative computing environments. *Licentiate thesis, University of Helsinki, Department of Computer Science*, 2009.

CHAPTER 3

HEAD ACCESS CONTROL METAMODEL FOR DYNAMIC AND HETEROGENEOUS STRUCTURES

Published in Journal of Sensors (Special Issue: Security and Privacy in Software Based Critical Contexts), 2021; Volume-21(19); <https://doi.org/10.3390/s21196507>

Abstract: The evolution of ubiquitous computing and pervasive information systems, such as IoT and Industry 4.0 systems, has introduced significant challenges related to security and access control. To confront the challenge of accessing resources, various research works were conducted focusing on developing and enhancing AC modeling in five main directions, starting from (1) traditional models, (2) hybrid models, (3) extending models, (4) abstracting models, reaching to (5) AC metamodels which is the recent research issue in this domain. In this chapter, we propose a Hierarchical, Extensible, Advanced, and Dynamic (HEAD) AC metamodel, which takes into consideration the existing limitations of the proposed AC metamodels. Its meta-components are constructed after unifying the heterogeneous concepts of AC components. HEAD metamodel allows instantiating any needed component/attribute for any model (existing model or non-existing model), this makes it adaptable for dynamic and heterogeneous structures with the ability to encompass the heterogeneity of the existing AC models. Moreover, we show that HEAD metamodel is able to derive various AC models, and different static and dynamic policies can be generated using its components. We use Eclipse (xttext) to define the DSL of HEAD metamodel, and illustrate our approach with several successful instantiations for various models and hybrid models. Additionally, we provide some examples to show how some of the derived models can be implemented to generate AC policies.

Résumé: L'évolution de l'informatique omniprésente et des systèmes d'information omniprésents, tels que les systèmes IdO et Industrie 4.0, a introduit des défis importants liés à la sécurité et au contrôle d'accès. Pour relever le défi de l'accès aux ressources, divers travaux de recherche ont été menés en se concentrant sur le développement et l'amélioration de la modélisation CA dans cinq directions principales, à partir de (1) modèles traditionnels, (2) modèles hybrides, (3) modèles d'extension, (4) modèles abstraits, atteignant (5) les métamodèles CA qui est l'enjeu de recherche récent dans ce domaine. Dans ce chapitre, nous proposons un métamodèle CA hiérarchique, extensible, avancé et dynamique (HEAD), qui prend en considération les limitations existantes des métamodèles CA proposés. Ses méta-composants sont construits après unification des concepts hétérogènes de composants CA. Le métamodèle HEAD permet d'instancier n'importe quel composant/attribut nécessaire pour n'importe quel modèle (modèle existant ou modèle inexistant), ce qui le rend adaptable aux structures dynamiques et hétérogènes avec la capacité d'englober l'hétérogénéité des modèles CA existants. De plus, nous montrons que le métamodèle HEAD est capable de dériver divers modèles CA, et différentes politiques CA statiques et dynamiques peuvent être générées à l'aide de ses composants. Nous utilisons Eclipse (xttext) pour définir le DSL de notre métamodèle CA, et illustrons notre approche avec plusieurs instanciations réussies pour divers modèles et modèles hybrides. De plus, nous fournissons quelques exemples pour montrer comment certains des modèles dérivés peuvent être mis en œuvre pour générer des politiques CA.

Article

HEAD Metamodel: Hierarchical, Extensible, Advanced, and Dynamic Access Control Metamodel for Dynamic and Heterogeneous Structures

Nadine Kashmar ^{1,*}, Mehdi Adda ¹  and Hussein Ibrahim ² 

¹ Département de Mathématiques, Informatique et Génie, Université du Québec à Rimouski, 300 Allée des Ursulines, Rimouski, QC G5L 3A1, Canada; mehdi_adda@uqar.ca

² Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC G4R 5B7, Canada; hussein.ibrahim@itmi.ca

* Correspondence: nadine.kashmar@uqar.ca



Citation: Kashmar, N.; Adda, M.; Ibrahim, H. HEAD Metamodel: Hierarchical, Extensible, Advanced, and Dynamic Access Control Metamodel for Dynamic and Heterogeneous Structures. *Sensors* **2021**, *21*, 6507. <https://doi.org/10.3390/s21196507>

Academic Editors: Xabier Larrucea and Juan José Unzilla

Received: 2 September 2021

Accepted: 23 September 2021

Published: 29 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: The substantial advancements in information technologies have brought unprecedented concepts and challenges to provide solutions and integrate advanced and self-ruling systems in critical and heterogeneous structures. The new generation of networking environments (e.g., the Internet of Things (IoT), cloud computing, etc.) are dynamic and ever-evolving environments. They are composed of various private and public networks, where all resources are distributed and accessed from everywhere. Protecting resources by controlling access to them is a complicated task, especially with the presence of cybercriminals and cyberattacks. What makes this reality also challenging is the diversity and the heterogeneity of access control (AC) models, which are implemented and integrated with a countless number of information systems. The evolution of ubiquitous computing, especially the concept of Industry 4.0 and IoT applications, imposes the need to enhance AC methods since the traditional methods are not able to answer the increasing demand for privacy and security standards. To address this issue, we propose a Hierarchical, Extensible, Advanced, and Dynamic (HEAD) AC metamodel for dynamic and heterogeneous structures that is able to encompass the heterogeneity of the existing AC models. Various AC models can be derived, and different static and dynamic AC policies can be generated using its components. We use Eclipse (xttext) to define the grammar of our AC metamodel. We illustrate our approach with several successful instantiations for various models and hybrid models. Additionally, we provide some examples to show how some of the derived models can be implemented to generate AC policies.

Keywords: access control; metamodel; policy; hierarchy; security and privacy; IoT; Industry 4.0; heterogeneous

1. Introduction

The current generation of networking environments, referring to dynamic and ever-evolving environments, such as the Internet of Things (IoT), cloud computing, etc., with several millions of users who need access to information stored in distributed data centers and servers via various types of devices (wearable devices, mobile phones, tablets, . . .), makes the process of controlling access challenging and very complicated. Moreover, the emergence of ubiquitous computing with heterogeneous devices, platforms, etc., especially the concept of Industry 4.0 and IoT applications, releases new prospects to traditional information systems and access control (AC) methods by merging new technologies and services for seamless access to information sources at anytime and anywhere [1–3]. This reality, in addition to the heterogeneity of the AC models that are implemented in the different centralized and distributed computing environments, makes the process of controlling access even more complicated, as the method of AC should answer the needs

of any computing environment by including the heterogeneity of AC models, and being upgradable and dynamic to follow possible technology progressions [4,5].

Moreover, AC policies are among the most significant security mechanisms that are essential to increase the privacy and confidence of an information system. Up to the present time, AC research and real-world AC implementations to define and enforce AC policies broadly fall under one of the five stages:

1. Traditional AC models discretionary access control (DAC), mandatory access control (MAC), role-based access control (RBAC), attribute-based access control (ABAC) [4,6];
2. Hybrid models, by means of combining features of two or more AC models, for example, hybrid RBAC/ABAC model [7];
3. Extended AC models, by means of adding new component(s) to a model to enhance its features, for example, [8];
4. Abstract AC models, by means of abstracting a model and adding new components to it, then deriving different instances of it, for example, [9];
5. AC metamodels, by means of including all of the above, for example, [10].

In the literature, different AC models are implemented to define and enforce AC policies in order to specify users' access rights to resources and verify that they can only access resources they are allowed to in a given context. In Figure 1, we illustrate the notion of heterogeneous structures, which include heterogeneous systems, platforms, networks, and devices, in addition to the heterogeneity of the implemented AC models to define and enforce different AC policies, such as password policy, network access policy, remote access policy, etc. With the evolution of technology trends, it is realized that traditional models, hybrid models, extended AC models, and abstract AC models no longer meet the increasing demand of privacy and security standards; in other words, they are currently insufficient to handle all the AC requirements [5]. To enhance AC methods, the era of developing AC metamodels began within the decade to serve as unifying frameworks with advanced AC features that are able to include most features of AC models in order to define a larger set of AC policies and upgrade the defined policies [5,10,11]. The AC metamodel should allow security experts and system administrators to create the needed components to define/upgrade any static and dynamic AC policy since controlling users' access and the actions they perform on information cannot be ignored when developing strategies related to information security [6,12]. An information system (IS) must follow up the evolution of security threats by designing and including modern security concepts and practices and incorporating them with the information system development life cycle (SDL). With the evolution of SDL, various studies have focused on the importance of collaboration and communication between software operators and developers. Recently, the need for security prompted the collaboration between developers and operators by involving security experts from the start of SDL [13,14].

However, implementing AC methods in complicated and distributed environments with several millions of users who might be assigned to different levels of roles, categories, groups, etc., and who request access to millions of objects, which might be distributed also in levels in several sites, need a generic, dynamic, and extensible AC metamodel that supports the hierarchy of components, for example, objects, roles, categories, actions, and maybe conditions. Unfortunately, despite the existing metamodels tackling specific issues related to certain computing environments, they lack some essential features and have several limitations (summarized in Section 2) [5,6,11]. Our concern in this paper falls under the fifth stage of developing AC methods. For this purpose, we propose a Hierarchical, Extensible, Advanced, and Dynamic (HEAD) AC metamodel with unconventional features to assist developers and security experts to include its components in designing secure ISs that conform to organizational AC security policies. In this paper, we tackle the generic, dynamic, extensible, and hierarchical limitations of the existing metamodels. We propose the kernel elements of the HEAD metamodel (a preliminary and general instance of our metamodel is presented in [10]); we unify the concepts of the heterogeneous AC components and include them under a generic metamodel concept. We explain how different

AC components can be defined, instantiated to derive various models from the HEAD metamodel, and illustrate several scenarios to show its dynamism and extensibility. Additionally, we show how our metamodel supports the feature of hierarchy for all components, which is essential to define policies in, for example, a hierarchical organizational structure where, for example, several users might be assigned to several roles in a hierarchy and have permission to access several objects also in a hierarchy. However, the contribution of this paper can be summarized as follows:

- Proposing an advanced AC metamodel with generic, dynamic, and extensible characteristics; it can also be considered a foundation stone to solve other limitations in existing AC metamodels, including collaboration and interoperability between various models.
- Providing a solution for the need of component hierarchy in AC models (e.g., objects, actions, roles, categories, contexts, etc.).
- Developing a language to express AC requirements, and to serve as a basis for producing AC decisions for access requests.
- Providing a grammar language that is simple and flexible to appropriately express AC policy requirements.
- Assisting developers and security experts to include unified and generic components in designing secure ISs that conform to the organizational AC security policies.

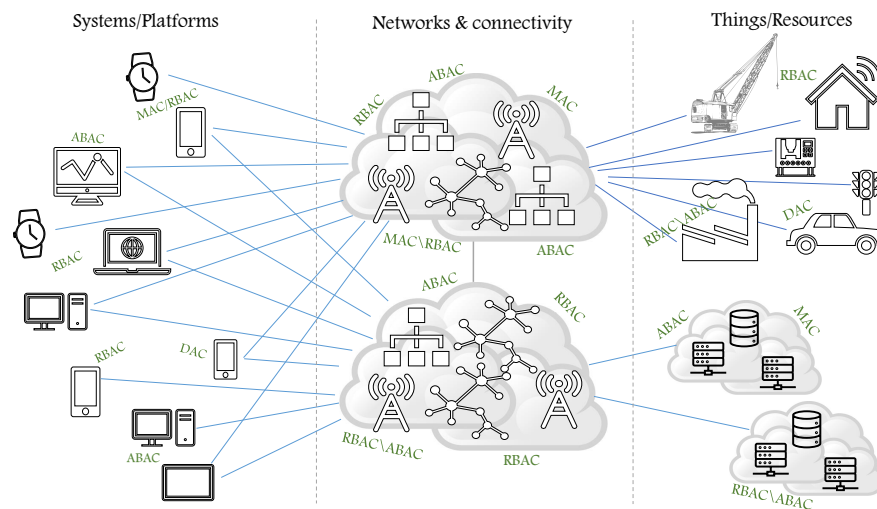


Figure 1. The dynamic and heterogeneous structures.

The remainder of this paper is organized as follows. In Section 2, we review the related works in this domain. In Section 3, we explain how we unify the common concepts of heterogeneous AC models, then we present the HEAD metamodel with its kernel elements based on the unified concepts. In Section 4, we design the tools for metamodel instantiation. We use the Eclipse Xtext framework to define the domain-specific language (DSL) of our metamodel, and then we explain the metamodel characteristics. In Section 5, we show how the HEAD metamodel provides a generic base to include all components of AC models and is able to derive various models (also hybrid models) instances. We also represent several scenarios to show its dynamism, extensibility, and how it supports the hierarchy of different components. In Section 6, we illustrate some examples of how AC policies can be generated, using the HEAD metamodel. Section 7 concludes this paper with future perspectives.

2. Related Works

As mentioned earlier, different AC implementations to define and enforce AC policies have been presented in the literature over the decades. In this section, we summarize

the proposed AC models, starting from traditional AC models reaching to the motif of AC metamodels.

The traditional AC models are DAC, MAC, RBAC, and ABAC [6]. In DAC, mapping users to operations on an object often takes the form of an access control matrix, while MAC is based on the concept of security levels assigned to subjects and objects to control the direction of information flow and users' operations. In these models, AC decisions are based on the identity of the user, and they are inadequate in dynamic computing environments where the sharing of information between systems and users from diverse security domains is common [5,8]. Thereafter, the RBAC model is proposed to provide a more generalized model than DAC and MAC. In RBAC, if users are assigned to a role, then they are granted a set of permissions. This model is insufficiently flexible for several scenarios and makes administration potentially heavy in very large systems [11,15], for example, systems with several local branches and abroad (e.g., banks), where hundreds of users need to be assigned to roles to determine their permissions. Additionally, it does not support dynamic attributes (e.g., time of day), for example, to prevent users from accessing any information after their working hours. Furthermore, due to continuous technology developments, the demand for a more generic and dynamic AC model has grown, and thus, the ABAC model is proposed. In ABAC, AC decisions are based on the attributes of users (subjects), attributes of objects being accessed, and attributes of environment (context). One of the ABAC model's limitations is the difficulty to calculate the resulting set of permissions for a given user [5,6,15,16].

Due to continuous technology growth and the appearance of distributed systems and because of some shortfalls in each of the traditional models [10,17], different approaches are proposed in the literature to enhance AC methods by combining features of two or more AC models, called hybrid models. A hybrid model is proposed by Kaiwen and Lihua in [18], based on attribute and role, to solve the shortage on the environment of large-scale dynamic users by solving the aspect of permissions assignment and policy management. Another hybrid RBAC/ABAC model approach is presented in [19] for multi-domain information systems. To find a model that supports dynamic attributes and handle relationship between roles and attributes to provide better AC features in dynamic environments, Kuhn et al. in [20] address the idea of adding attributes to RBAC. Moreover, Rajpoot et al. [7] integrate RBAC and ABAC models by associating attributes with subjects, objects and the environment, allowing the request context to be considered in making AC decisions. In [21], a hybrid model merging RBAC and ABAC is proposed to enhance the dynamic features of RBAC and to provide ease of administration, tight security, dynamic behavior, and efficient separation of duty implementation. Additionally, Kim et al. in [22] propose the MAC/RBAC model by configuring RBAC and MAC features to be applied in the domains where access has to be checked for both authorized roles and security levels together, for example, in the hospital domain, government domain, and military domain.

Moreover, to enhance an AC model in order to define additional rules, some AC models are extended by adding new component(s) to them. For example, in [23], a survey of an extended RBAC model in cloud computing is presented. In addition, an ABAC model extension is presented in [8], called hierarchical group and attribute-based access control (HGABAC), where groups and hierarchies of subjects and objects are added. Another ABAC extension is proposed in [9], named the higher-order attribute-based access control model (HoBAC); it extends the basic concepts (subjects, objects, and contexts) of the model with aggregation operations that provide hierarchies, and many other examples. As well, in [24], the RBAC model is abstracted to the meta-level in order to support the delegation of users' rights, which allows a user without any specific administrative privileges to delegate their access rights to another user; hence, a delegation metamodel is proposed for specifying RBAC and RBAC-based delegation features. In [25], Adda and Aliane proposed a generic ABAC metamodel to generate a wide variety of AC models related to the ABAC model that is suitable for all computing environments.

The current generation of networking environments impose the need to focus on developing more advanced AC features, especially since the existing AC models—hybrid models, extended AC models, and abstract AC models—have reached their limits and are currently insufficient to meet the needed AC requirements [5,17]. What makes this fact challenging is the heterogeneity of everything—networks, applications, devices, etc.—in addition to the heterogeneity of AC models. Hence, the notion of AC metamodels has existed for almost a decade [11]. They are proposed to work as frameworks to allow instantiating various models, and allow defining and enforcing a larger set of AC policies. These metamodels are proposed to include most of the features and components of AC models, which are employed in the core metamodel structure [5,11]. Hence, the more features they include, the more they allow defining a larger set of AC rules.

In the literature, several metamodel approaches are proposed for different computing environments. A unified AC modeling language is proposed in [26] as an extension for Barker's metamodel [27] to support object and action hierarchies. The aim of their language is to define hybrid AC policies by allowing categories to be associated with other categories and finding hierarchical associations between them. A subject may be assigned to a category, which could be a role, a class, a group, a security level, etc. Barker's approach includes features of MAC, DAC, and RBAC, and is named the category-based access control metamodel (CBAC). Another approach is proposed by Bertolissi et al. in [28] for distributed environments and is based on the CBAC metamodel, where the user's request can be passed to other sites and evaluated in a distributed manner. An integration metamodel for hybrid policies to concurrently handle DAC, MAC, and RBAC models is proposed by Abd-Ali et al. in [29]. Alves et al. in [30] propose a CBAC metamodel extension to study the interaction between obligations and permissions and expand a general notion of obligation for the existing AC models. A category-based access control (CatBAC) metamodel for highly flexible and dynamic environments, specifically for cloud computing services, is proposed by Khamadja et al. in [31], to allow security administrators in various company sites to find a concrete model with the constraints and specificities of each site. To handle security and privacy in cloud service development and operations, Xia et al. in [32] propose the cloud security and privacy metamodel (CSPM). To facilitate the analysis and manipulation of security requirements for web services for the representation of web content management systems (WCMS) AC policies and to automatically extract the AC information in the domain of WCMS, a metamodel approach is presented by Martínez et al. [33].

Generally, the metamodels are proposed to address the notions of including various hybrid AC models features in the core metamodel structure, to encompass AC models and allow extending them by adding new components, and to find a generic structure that could include the most possible features of AC models. Nevertheless, the proposed AC metamodels have several limitations and lack some essential characteristics [5,11,34], and can be summarized as follows:

- They do not include all features of the common AC models, so they are not generic enough;
- They do not support the ability of defining new entities and building relationships between them in order to describe larger set of AC rules, for example, due to changing environmental conditions, so they are not dynamic enough to follow technology upgrades;
- They do not include the possibility of defining new components and attributes in addition to the defined ones, so they are not extensible and the defined policies cannot be extended;
- They do not support the feature of hierarchy for components;
- They do not handle or support the feature of collaboration and interoperability between the various AC models;
- They do not tackle the issue of migrating AC policies from one model to another.

In this paper, our concern is to provide solutions for the limitations of generality, dynamism, extensibility, and hierarchy.

3. Formalization of Access Control Policies

A security policy is a definition of a set of rules and guidelines on which access is granted or denied for a user in any organization or industry sector. The following are some general examples of AC rules:

- Before check-in, each worker has to wear a face mask.
- The maximum number of visitors in each room is 15.
- Machine operators can only enter the labs during working hours.

To define a policy, a set of concepts are defined and formulated to form rules. For example, in the above rules we have subjects (e.g., worker, operator), objects (e.g., face-mask, machine, room), actions (e.g., wear, visit, enter), attributes (e.g., maximum number, working hours), and many other concepts could be included in a defined policy, such as permission, role, group, etc.

3.1. Unifying Access Control Concepts of Heterogeneous Security Policies

In the literature, several types of AC models are implemented; these models can be defined as frameworks for making authorization decisions. Each model is formulated based on AC concepts. The following are some examples:

- The DAC model includes subject, object, and action concepts.
- The MAC model includes subjects, object, security level, and operation concepts.
- The RBAC model includes subject, object, role, permission, and action concepts.
- The ABAC model includes subject attributes, object attributes, context attributes, condition, and action concepts.

Consequently, we can find that security policies include common concepts and attributes that are common to all AC models [10]. These concepts can be summarized as follows:

1. A set of concepts (and attributes) to describe subjects and objects.
2. A set of concepts (and attributes) that describe the authorized subjects.
3. A set of concepts (and attributes) that explain the different access rights.
4. A set of concepts (and attributes) that set various constraints and conditions.
5. A set of concepts (and attributes) that describe the context (environmental context) to access objects.

To unify them and make them adaptable to all AC models, we classify them into explicit, implicit (authorization units and procedural units), and setting concepts as illustrated in Figure 2. Note that each of the classified groups may include additional concepts. EXPLICIT concepts are those that refer to something that is real and exists (e.g., subjects and objects). IMPLICIT concepts are those that refer to something described or explained in the guidelines or rules. Implicit concepts include AUTHORIZATION UNITS (e.g., roles, security levels . . .) and PROCEDURAL UNITS (actions, permissions . . .). SETTING refers to concepts that are included to have more accurate and regulated access to resources (e.g., context, constraints . . .).

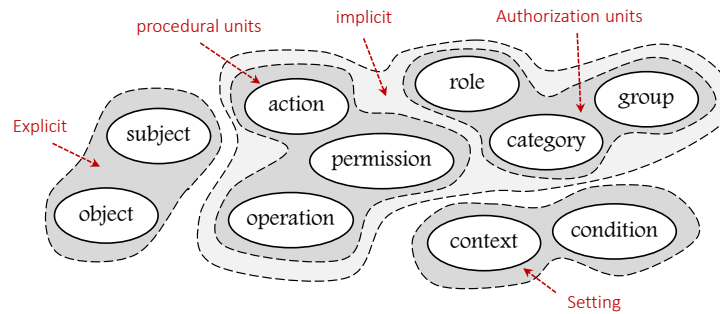


Figure 2. Unifying heterogeneous concepts of AC models.

3.2. HEAD Metamodel

The key responsibility of an AC metamodel is to define a language for specifying several AC models; usually, this level describes generic concepts. Examples of meta-objects (or meta-concepts) at the meta-modeling layer are the following: class/entity, attribute, component, and others. An AC model is an instance of the metamodel. The key responsibility of the model is to define a language that describes a security policy. Examples of objects (or concepts) at modeling layer are subjects, objects (or resources), actions, and other concepts; this level explains how these objects work together. At the system layer where users interact, the actual AC policies are expressed by a security expert for a given AC model instance(s). Note that in this section, we use the term “component” instead of “concept”. Table 1 gives a summary about the meta-modeling layers and details of our approach.

Table 1. Metamodeling layers and details.

Metamodeling Layers	Details	
	Description	Elements
Metamodel	Describes the models to be instantiated	explicit, implicit, authorization unit, procedural unit, setting
Model	Metamodel instance, e.g., RBAC model, ABAC model ...	subject, object, role, action, permission ...
System	Model instance, e.g., RBAC policy, hybrid policy ...	Alice, Bob, manager, nurse, prescription, device ...

Our AC metamodel aims first to describe the AC policy at the abstract level that is autonomous from the enforcement of this policy. Accordingly, instead of modeling the AC policy by using the concrete components of subject, object, permission, action, etc., we define the meta-policy by using the abstract components of explicit, implicit, and setting, then instantiate the concrete components to model the needed policy. The main characteristics of the HEAD metamodel are as follows (Figure 3):

- It unifies the heterogeneous components of AC models.
- It is generic enough to include the common AC models and other models.
- It is dynamic and includes the feature of defining components (and attributes for all components).
- It is extensible since it allows extending the already derived AC models.
- It supports the hierarchy for any type of components.

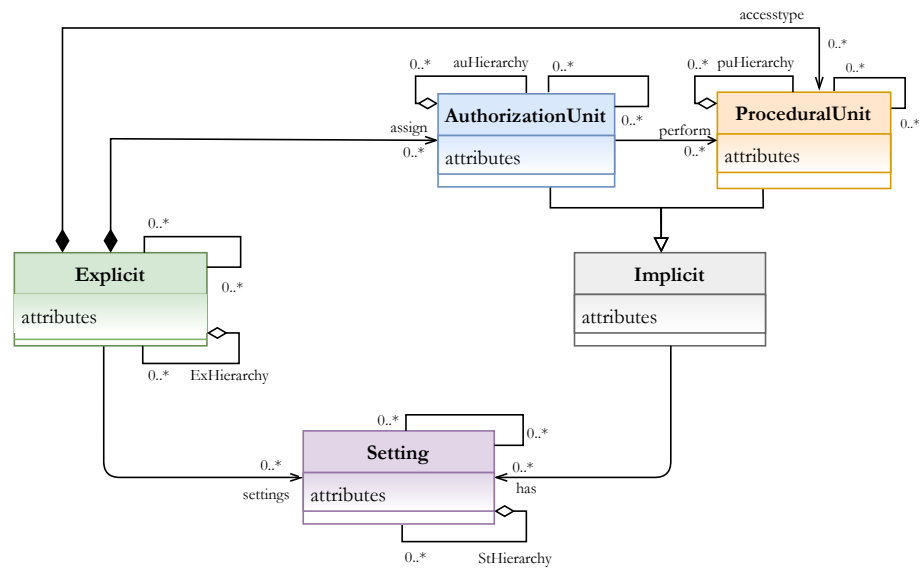


Figure 3. HEAD metamodel: the kernel elements.

Unifying the heterogeneous components of AC models and grouping them based on their functionality would allow instantiating an unlimited number of components related to the meta-component. Another essential characteristic is its generic structure, where all components of common AC models can be defined and new ones can also be defined. In other words, it is not restricted to the common models and it can also be used as a base to derive new models. Additionally, unlimited levels of hierarchy could be defined for all components whether they belong to explicit, implicit, or setting concepts. An AC metamodel that supports the hierarchy of components is an essential characteristic and cannot be ignored with the current distributed and complex structures of computing environments and the existing resources.

3.2.1. Kernel Elements: HEAD Metamodel

In this section, we present the kernel elements (meta-components) of our metamodel and the relationships between them.

- **EXPLICIT (E_x):** a set of explicit components that represent the real and the existing entities, such as subjects and objects in any organization or industry sector. The class EXPLICIT has a composition association with the sub-classes AUTHORIZATION UNIT and PROCEDURAL UNIT, which are the inheritance of the abstract IMPLICIT class.
- **IMPLICIT (I_m):** a set of implicit components that represent the described components. For example, subjects are classified or assigned to some other component(s) (e.g., roles), or the processes or functions that can be performed (e.g., actions). Two other sub-classes that are inherited from the Implicit super-class are the AUTHORIZATION UNIT (AU) components and the PROCEDURAL UNIT (PU) components.
 - **AUTHORIZATION UNIT (AU):** a set of authorization units. It is a subclass which should be specialized to create specific authorization units (au_i , sub-index i specifies the unit type), such as roles, categories, security levels, etc., to which some E_x units (e.g., subjects) can be assigned, An AU example is as follows:
 - au_i : role (manager, doctor, ...)
 - au_j : category (age > 18, temperature < 38°, ...)
 Hence, AU includes role and category components
 - **PROCEDURAL UNIT (PU):** a set of procedural units. It is a subclass which should be specialized to create specific procedural units (pu_i , sub-index i specifies the unit type) such as actions, permissions, operations, etc., to which some E_x units

(e.g., objects) can be assigned. In other words, it represents operations that can be performed by E_x units (e.g., subjects) on some other E_x units (e.g., objects), a PU example is as follows:

- pu_i : action (read, write, ...)
- pu_j : operation (turn on/off, open, close, ...)

Hence, PU includes action and operation components.

- **SETTING (S_t):** a set of setting components. It represents the concepts that are included to have more accurate and regulated access to resources, for example, context, contextual conditions, constraints, etc. The setting components actually provide our metamodel with high flexibility and expressiveness. They could include other components (explicit, implicit, or/and other setting) to construct the needed expression(s). For example, A context expression and contextual conditions can be expressed in terms of AU, PU, E_x , and S_t components.

3.2.2. Hierarchies and Associations

The concept of hierarchy is important to define multiple levels of components, such as roles, actions, objects, etc. It reflects the structure of an organization and, for example, the respective responsibilities/priorities of the hierarchical components. Figure 4 represents some examples of hierarchy in an organization or industry sector. In our metamodel, there are four basic sets of components: E_x (set of explicit entities/classes), AU (set of authorization unit entities/classes), PU (set of procedural unit entities/classes), and S_t (set of setting entities/classes). Our metamodel provides support for creating hierarchy for classes of AU (e.g., role hierarchy Figure 4a), PU (e.g., action hierarchy Figure 4b), E_x (e.g., object or resource hierarchy, Figure 4c), and S_t (e.g., context hierarchy, Figure 4d) by aggregating AU, PU, E_x , and S_t entities. These hierarchical relationships are depicted by an aggregation association in Figure 3.

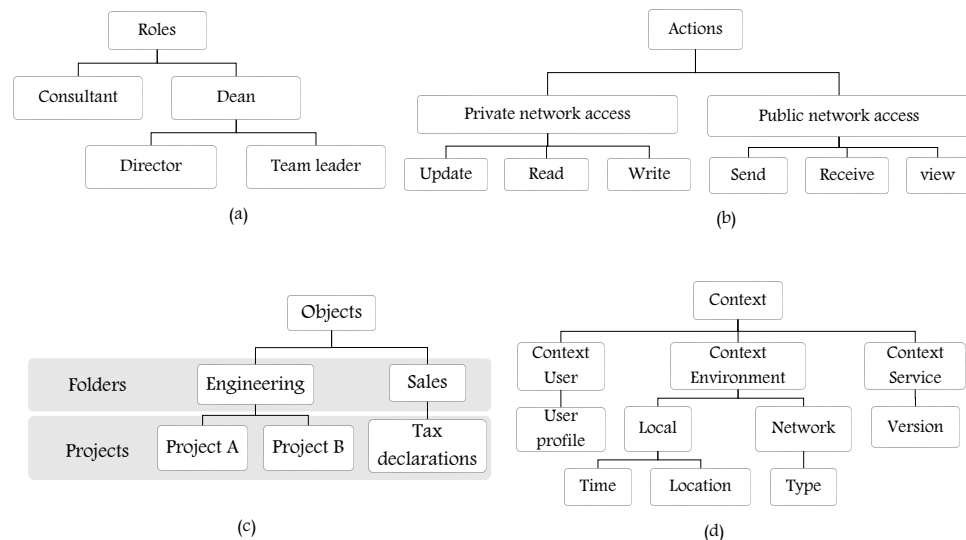


Figure 4. Examples for hierarchy of (a) roles; (b) actions; (c) objects; and (d) contexts.

The association between E_x and AU is to assign, for example, zero or many (0..*) subjects to roles, groups, categories or other AUs. The association between AU and PU and PU and E_x is to represent which AUs are able to perform zero or many PUs (e.g., actions, permissions ...) and access some, for example, objects or services. Note that I_m components (AUs and PUs) might have zero or many S_t (e.g., contextual and/or non-contextual constraints) before accessing/performing tasks on E_x components. Moreover, the metamodel provides support for formulating AC models and hybrid models for different policies by allowing AUs to be associated with other AUs, PUs to be associated with other PUs, E_x

components to be associated with other E_x components, and S_t components to be associated with other S_t components. As shown in Figure 3 a self-association edge exists on each of the classes. Note that in some models, we might have an empty set of AU, or S_t , for example, in the DAC model, AU is an empty set since explicit components are not assigned to AUs.

3.2.3. Meta-Policy and Policy

In this section, we explain the notions of meta-policy and policy of the HEAD meta-model; note that the theoretical foundations of the HEAD metamodel are not included in this paper. The meta-policy is expressed using the meta-components of E_x , I_m , and S_t , in the following way:

$$\text{Metapolicy} = \langle E_x, I_m, S_t \rangle$$

Based on this meta-policy, different AC policy definitions can be expressed as follows:

- To define RBAC policy:
 - $E_x = \{subject, object\}$
 - $I_m = \{AU = role, PU = permission\}$
 - Hence, $Policy = \langle subject, object, role, permission, action \rangle$
 - Meaning that a subject assigned to role has permission(s) to access object(s) and perform action(s).
- To define a hybrid MAC/RBAC policy:
 - $E_x = \{subject, object\}$
 - $I_m = \{AU = role, securitylevel, PU = permission, action\}$
 - Hence, $Policy = \langle subject, object, role, securitylevel, permissions, actions \rangle$
 - Meaning that, subjects who are assigned to specific roles and security levels have permissions to access objects that are classified to some security levels and perform some actions.

Thus, the policy is expressed using model components, which are derived from the meta-components of meta-policy.

An AC policy is a set of rules that determine users' access rights within a given information system. These rules constitute a definition of the AC requirements for the system. The process of implementing the AC mechanisms to make the system follow the defined rules is called enforcement. In this paper, our concern is to constitute the definition of AC requirements for a system.

4. Defining the Grammar of HEAD Metamodel

In the literature, several AC models, such as MAC, DAC, RBAC, ABAC, and many other hybrid models are formulated based on the definition of security rules. Depending on the model, the type of rules and the components (or entities) they employ are different. The remarkable advantage of our metamodel is that it supports the definition of AC policies for all these models and allows the implementation of generic tools to derive them. To handle this idea, the metamodel must allow defining the different components and attributes, then expressing models using them.

This section addresses the definition of the grammar of the DSL for our AC meta-model; the grammar we have created is listed in Figure 5. Our grammar definition can be interpreted as follows:

1. **Lines 1 to 39:** to instantiate the needed AC model(s) components, the hierarchies, and the attributes.
 - Lines 1 to 6: the block of defining all model components. 'Metamodel' is the root class for the definition of parser rules. The used keywords 'policy' and 'end', in line 3, are used to indicate the start and end of creating policy components. Note that our metamodel is able to create one or more policy types (e.g., MAC policy and RBAC policy). Each defined rule generates one decision (line 5).

- Lines 7 to 9: the declaration of attribute(s) name(s) and datatype(s); also, arrays can be declared.
- Lines 10 to 15: the definition of the policy name (e.g., RBAC) and the sub-blocks (inside the main block of policy) of the E_x , I_m , and S_t components. To create a policy, at least one or more explicit/implicit element(s) must be declared; also, we could have zero or more setting element(s). To define the sub-block of E_x and S_t elements, the keywords ‘explicit’ and ‘setting’ are used, respectively, at the beginning, and ‘end’ at the end. Note that the ‘Implicit’ parser rule (line 13) has two elements, ‘AuthorizationUnit’ and ‘ProceduralUnit’ (lines 23 to 27).
- Lines 16 to 18: the alternatives of attribute data types.
- Lines 19 to 22: the creation of E_x components, their attributes, and their hierarchies.
- Lines 23 to 27: the creation of sub-blocks of I_m elements (AUs and PUs). The keywords ‘authorization’ and ‘procedural’ are used to indicate the beginning of each sub-block, and ‘end’ at the ending.
- Lines 28 to 31: the creation of AU components, their attributes, and hierarchies.
- Lines 32 to 35: the creation of PU components, their attributes, and hierarchies.
- Lines 36 to 39: the creation of S_t components, their attributes, and hierarchies.
- Note that attributes could be defined for all components, and an unlimited number of levels for components hierarchy can be created.

```

1- Metamodel:
2-   {Metamodel}
3-   'policy' (policy+=Policy)+ 'end'
4-   'rule:'
5-   decision=Decision
6- ;
7- Attribute:
8-   name=ID (array ?=[' (length=INT)? '])? ":" type=AttType
9- ;
10- Policy:
11-   name=ID (('attributes+=Attribute+')?)
12-   'explicit' (explicit+=Explicit)+ 'end'
13-   ('implicit+=Implicit)+
14-   ('setting'(setting+=Setting)* 'end')?
15- ;
16- AttType:
17-   'String'|'int'|'boolean'|'char'|'float'
18- ;
19- Explicit:
20-   name=ID ((' attributes+=Attribute+')?)
21-   (('hierarchy+=Explicit+')?)
22- ;
23- Implicit:
24-   {Implicit}
25-   ('authorization' authunit+=AuthorizationUnit* 'end')?
26-   ('procedural' procunit+=ProceduralUnit* 'end'
27- ;
28- AuthorizationUnit:
29-   name=ID ((' attributes+=Attribute+')?)
30-   (('hierarchy+=AuthorizationUnit+')?)
31- ;
32- ProceduralUnit:
33-   name=ID ((' attributes+=Attribute+')?)
34-   (('hierarchy+=ProceduralUnit+')?)
35- ;
36- Setting:
37-   name=ID ((' attributes+=Attribute+')?)
38-   (('hierarchy+=Setting+')?)
39- ;
40- Decision:
41- ((' attributes+=Attribute*')?)
42-   {' (explicit+=[Explicit|Qualified Name]
43-   (' (wth+=[Attribute|Qualified Name]* ' '))?)
44-   (' (authunit+=[AuthorizationUnit|Qualified Name] ('(wth+=[Attribute|Qualified Name]* ' '))?) * ' ')}?
45-   (' (procunit+=[ProceduralUnit|Qualified Name] ('(wth+=[Attribute|Qualified Name]* ' '))?) *
46-   => '{'
47-   (explicit+=[Explicit|Qualified Name]
48-   ('(wth+=[Attribute|Qualified Name]* ' '))?)
49-   ('(authunit+=[AuthorizationUnit|Qualified Name] ('(wth+=[Attribute|Qualified Name]* ' '))?) * ' ')}?
50-   '{'
51-   (
52-   procunit+=[ProceduralUnit|Qualified Name] ('(wth+=[Attribute|Qualified Name]* ' '))?)
53-   ('( setting+=[Setting|Qualified Name] ('(wth+=[Attribute|Qualified Name]* ' '))?) * ' ')}?
54-   )+
55-   '}'
56-   )+
57-   ('}')?
58-   '}'
59- '}' ' ->' id+=ID)+
60- ;
61- QualifiedName:
62-   ID('.' ID)*
63- ;

```

Figure 5. HEAD Metamodel: The Grammar.

2. **Lines 40 to 63:** to define a policy (set of rules), based on the instantiated components and attributes with the access request decision. Note that, using our grammar definition, rules (and hybrid rules) can be expressed in different ways, for example, a subject can access object(s) and perform an operation(s), or an object can be accessed by a subject(s) and perform an action(s).
 - Line 40: the parser rule ‘Decision’, is the beginning of specifying and expressing a rule which ends with a decision (‘-> id+=ID, line 59).
 - Line 41: after using the keyword ‘rule’ (line 4), some attributes can be created, for example, ruletype, rulenummer, etc.
 - Lines 42 and 44: the block of rule definition starts with an open curly braces ‘{’. A rule is started by specifying an E_x component (e.g., subject or object) and its attributes. In some models, explicit components are assigned to some AUs, for example, in RBAC subjects are assigned to roles, and in MAC subjects/objects are assigned to security levels. Note that E_x -AU assignment is optional ‘?’ in expressing a rule, depending on the expressed model (line 44).
 - Line 45: it is optional to define a nested block for a procedural unit, for example permission in RBAC, to express a policy.
 - Line 46: the beginning of expressing another nested block after specifying an E_x component and assigning it to some AU(s), or defining some PU(s).
 - Lines 47 to 49: same interpretation of lines 42 to 44. Hence, the beginning of a rule could be expressed as follows, for example:
 - A subject can access object(s) . . .
 - An object can be accessed by a subject(s) . . .
 - A subject assigned to a role has permission to access object(s) . . .
 - A subject assigned to a security level can access object(s) assigned to some security levels . . .
 - Lines 47 to 56: the sign ‘+’ in ‘)+’, line 56, indicates that what is included between lines 47 to 56 can be expressed more than once within a rule.
 - Lines 50 to 55: the start and end of a sub-block of specifying what PUs (e.g., actions) an E_x unit can perform. Note that it is optional to include some S_t (e.g., conditions) while expressing a rule. The sign ‘+’ in ‘)+’, line 54, indicates that what is included between lines 51 and 54 can be expressed more than once.
 - Line 57 to 59: closing the main block with some of opened sub-blocks.
 - Line 59: indicates the end of rule expression with the decision ID. Note that the sign ‘+’ in ‘)+’ indicates that a set of rules can be defined within a policy.

In Section 5, we explain with examples how our metamodel grammar could be expressed to define different rules, and show how it is generic, dynamic, extensible, and supports a hierarchy of components.

5. Deriving Access Control Models

In this section, we show how our metamodel structure is (1) generic and able to derive instances of different models and hybrid models, (2) dynamic and allows defining new components in addition to the existing ones, also the relationships between them, (3) extensible to upgrade any defined policy, and (4) supportive of the feature of defining hierarchies.

The key responsibility of the model is to define a language that describes a security policy. Examples of entities at modeling layer are: subjects, objects (or resources), actions, and other entities. This metamodeling layer explains the way of how these entities work together. Henceforth, we use the term “entity” instead of “component”. Note that in this paper, we consider the environmental context.

5.1. Generality

In this section, common models (DAC, MAC, RBAC, and ABAC), in addition to some hybrid models, are instantiated, using the HEAD metamodel, and show how different AC rules can be expressed using the defined grammar (Figure 5). The models in Figures 6, 8, 10, 12, 14 and 16 illustrate class instances with the same colors of EXPLICIT, IMPLICIT, and SETTING classes of Figure 3.

5.1.1. Discretionary Access Control Model (DAC)

In DAC, subjects determine how some other subjects can access their objects [35]. It is based on the identity of three key entities shown in Figure 6, the E_x entities are subject and object, and the I_m entity is operation (PU instance). Subjects can control access rights to their objects by determining what operations can be performed by other subjects.

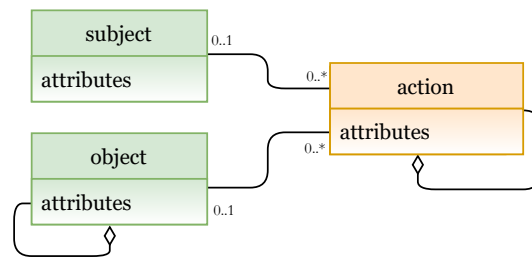


Figure 6. DAC model instance.

Based on our grammar definition, in Figure 7a (lines 1 to 4), we define DAC entities (and the needed attributes), and their instances are shown in Figure 7b. The DAC policy is expressed in lines 6 to 13 starting with the keyword 'rule', and since a policy is a set of rules, we define the attribute 'ruleid' to indicate the rule number. Note that any rule could have allow, deny, mixed, etc., decisions. Hence, the rule can be interpreted as follows:

A subject with name = name-value can access object with type = type-value and perform some operation(s) op = op-value(s).

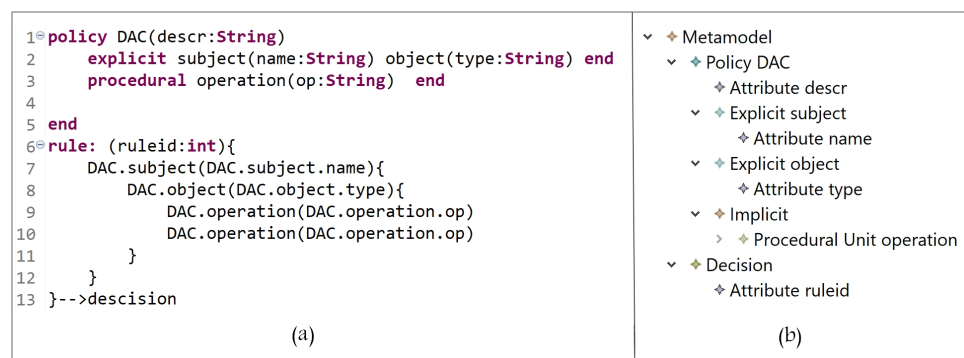


Figure 7. (a) DAC Policy Definition; (b) DAC entities.

5.1.2. Mandatory Access Control Model (MAC)

In MAC, access rights are based on the concept of security levels associated with each subject and object, where actions are derived. A security level for a subject is called the clearance level and for an object is called the classification level [6]. In Figure 8, the E_x entities are subject and object, and the I_m entities are security level (AU instance), and operation (PU instance). Clearance levels are assigned to subjects and objects, and based on these levels, AC rights are specified.

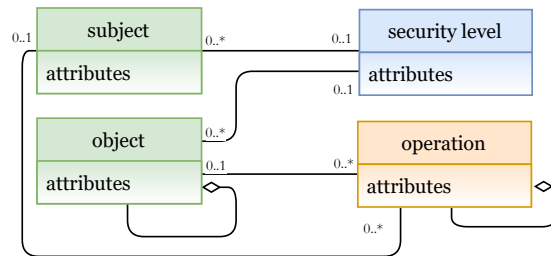


Figure 8. MAC model instance.

In Figure 9a (lines 1 to 5), we define MAC entities (and the needed attributes), and their instances are shown in Figure 9b. The MAC policy is expressed in lines 7 to 16, starting with the keyword ‘rule’, and the attribute ‘ruleid’ to indicate the rule number. Note that any rule could also have allow, deny, mixed, etc., decisions. Hence, the rule can be interpreted as follows:

A subject with name = name-value which is assigned to a securitylevel = clearancelevel-value can access object with securitylevel = classificationlevel-value and perform some operation(s) op = op-value(s).

In MAC, for example, the BLP (Bell–LaPadula) model, a subject is allowed to read an object if its clearance level is greater than or equal to the object’s classification level [6].

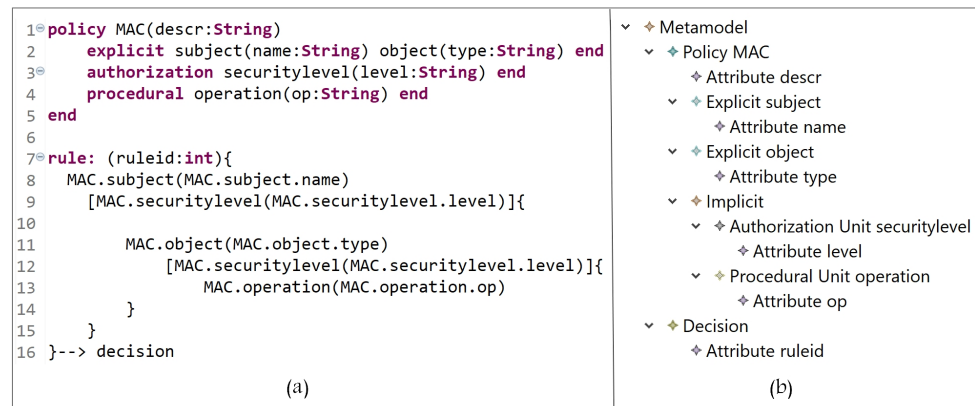


Figure 9. (a) MAC policy definition; (b) MAC entities.

5.1.3. Role-Based Access Control Model (RBAC)

In RBAC, subjects are given access based on their roles (e.g., engineer and doctor). In Figure 10, the E_x entities are subject and object, and the I_m entities are role (AU instance), and permission and action (PU instances). Subjects can be assigned to different roles (roles can be associated to several subjects), and each role is a group of permissions to perform some actions. As mentioned earlier, our metamodel provides support for creating hierarchies by aggregating some concepts. As shown in the figure, hierarchies for objects (resources), roles, and actions can be created [35].

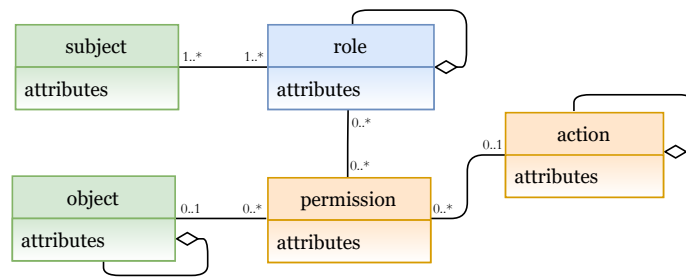


Figure 10. RBAC model instance.

In Figure 11a (lines 1 to 6), RBAC entities (and the needed attributes) are defined, using our defined metamodel language; their instances are shown in Figure 11b. RBAC policy is expressed in lines 8 to 23. RBAC rule can be interpreted as follows:

A subject with name = subjectname-value assigned to a role roletype = role-value has the permission to access object name = objectname-value and perform action act = act-value1, and action act = act-value2 if condition expr = condition-expression is true.

Note that the ‘act’ attribute might have read value for the first action and write for the second action.

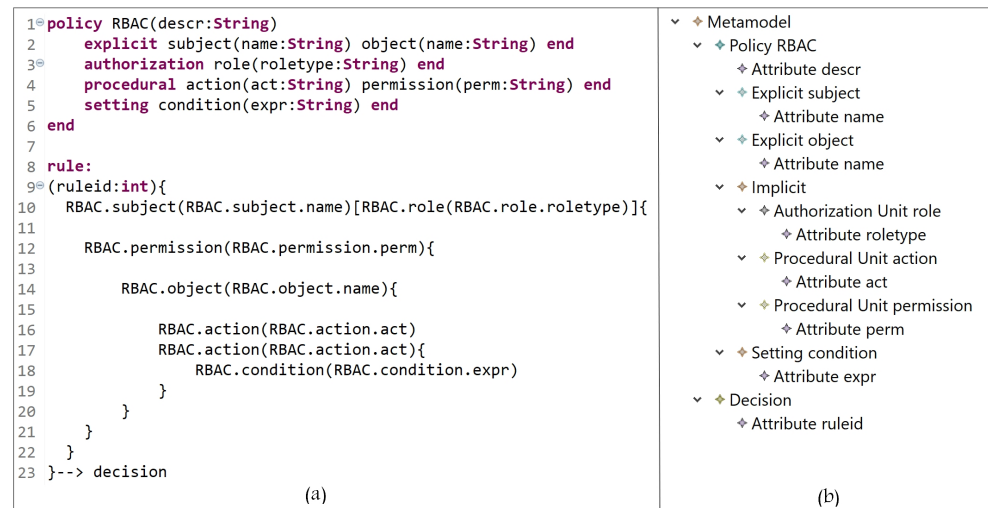


Figure 11. (a) RBAC policy definition; (b) RBAC entities.

5.1.4. Attribute-Based Access Control Model (ABAC)

In the ABAC model, AC rights are evaluated at the time that the actual request is made; it uses subject, object, and environmental (context) attributes to determine access decisions. In Figure 12, the E_x entities are the subject attributes and object attributes (which represent subjects and objects); the I_m entities are permission and action (PU instances); the S_t entities are context expressions and attributes. Subjects with some attributes are allowed to perform some actions on objects with some other attributes based on some conditions and constraints in the defined policy.

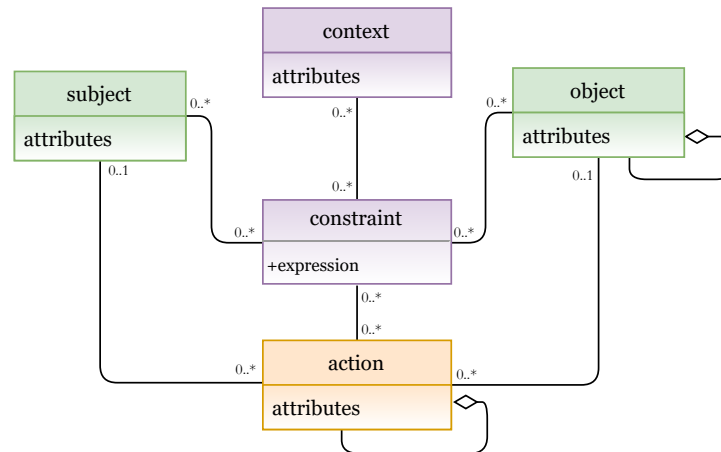


Figure 12. ABAC model instance.

In Figure 13a (lines 1 to 9), ABAC attributes, which represent subjects, objects, actions, and context, are defined; the instances are shown in Figure 13b. ABAC policy is expressed in lines 11 to 21. A rule can be interpreted as follows:

A subject with address = address value, and . . . attributes can access object with type = type value, and . . . attributes and perform an action act = act-value1, and act = act-value2 when expression if location = location-value is true.

Note that the ‘act’ attribute might have update for the first action and delete for the second action.

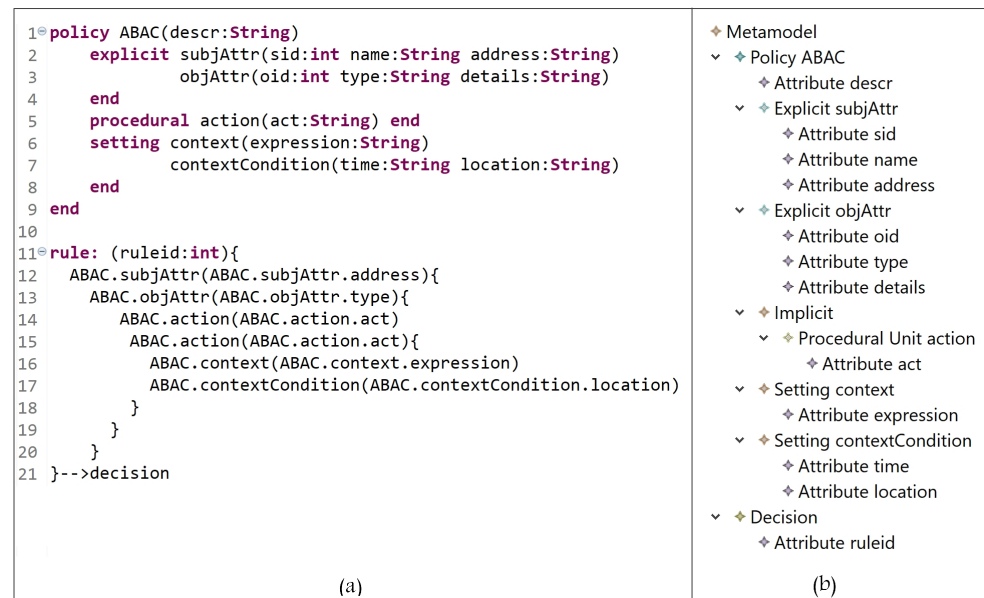


Figure 13. (a) ABAC policy definition; (b) ABAC entities.

5.1.5. Hybrid Models

A hybrid AC model combines features of two or more AC models. Using the grammar of the HEAD metamodel, various hybrid models can also be instantiated. Figure 14 represents a hybrid MAC/RBAC model instance.

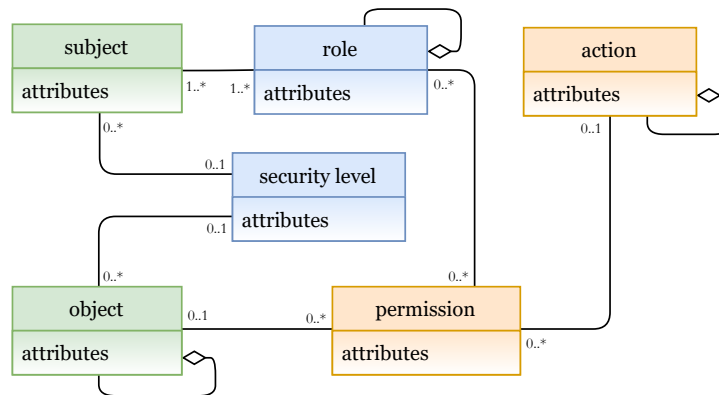


Figure 14. Hybrid MAC/RBAC model.

In Figure 15a (lines 1 to 5), we define MAC/RBAC entities (and the needed attributes), and their instances are shown in Figure 15b (subject, object, security level, role, action, and permission). A hybrid MAC/RBAC policy is expressed in lines 7 to 19, with the attribute ‘ruleid’ to indicate rule number. A MAC/RBAC rule can be interpreted as follows:

A subject which is assigned to role = roletype-value and securitylevel = clearancelevel-value has a permission to access object(s) with securitylevel = classificationlevel-value and perform some action(s) act = act-value.

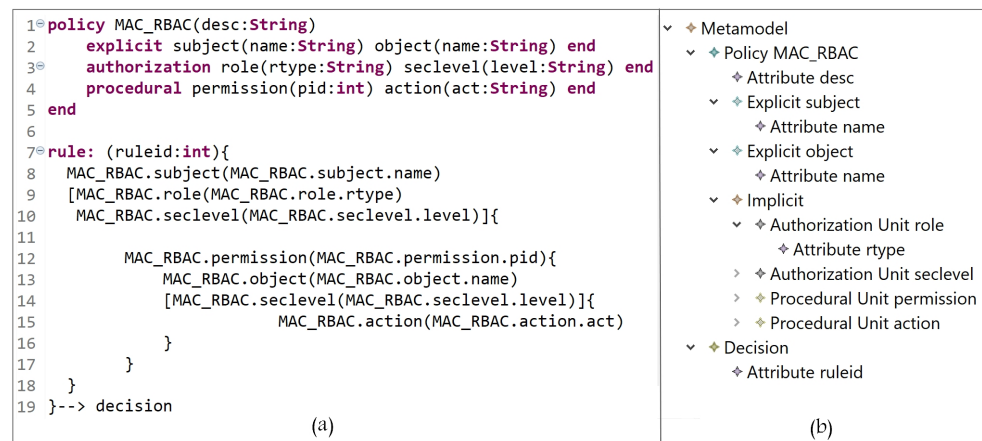


Figure 15. (a) MAC/RBAC policy definition; (b) MAC/RBAC entities.

Another hybrid model example is illustrated in Figure 16, which represents an instance of the hybrid RBAC/ABCA model. In this hybrid model, to determine subject’s role, a role is added as an attribute to the subject entity.

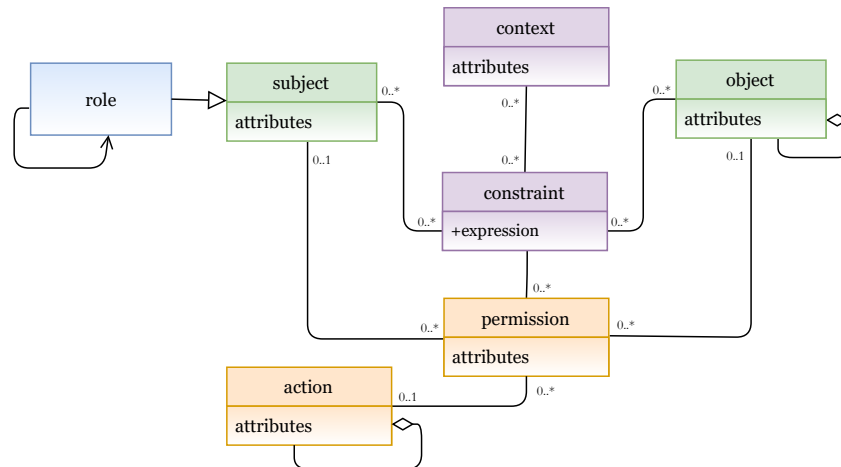


Figure 16. Hybrid RBAC/ABAC model.

In Figure 17a (lines 1 to 9), we define the RBAC/ABAC entities and attributes, and their instances are shown in Figure 17b (subject attributes, object attributes, action, permission, context, and contextual attributes). A hybrid RBAC/ABAC policy is expressed in lines 10 to 24. A rule can be interpreted as follows:

A subject with attributes address = address-value and role = role-value has a permission to access an object(s) with attribute(s) type = type-value and perform an action where act = act-value when context-expression if location = location-value and time = time-value are true.

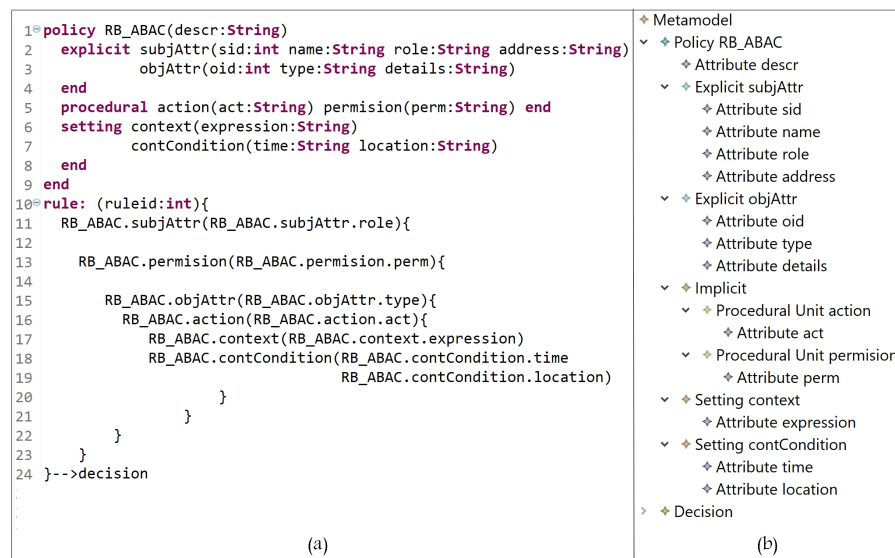


Figure 17. (a) RBAC/ABAC policy definition; (b) RBAC/ABAC entities.

As shown above, various AC models can be instantiated using the HEAD metamodel; hence, it is generic and able to include any AC feature for any model, and is also flexible enough to define the needed AC policies (also hybrid policies).

5.2. Dynamism

Along with technology upgrades, several security threats appear. To conquer them, security solutions must be regularly updated and stay amenable to follow and track the

evolution of these threats. Protecting resources against security threats has become a crucial concern in the development of IS and requires setting up trusted AC policies. The HEAD metamodel exceeds the features of the existing metamodels since it considers that AC is becoming more and more important for open, ubiquitous, and critical systems. An AC metamodel must be flexible and upgradable, due to changing conditions or updating rules. In other words, its structure should be dynamic and describe how its properties can be adjusted over time to define a larger set of static and dynamic AC policies. Hence, a dynamic metamodel allows defining new types of attributes, for example, contextual attributes, and new components, in addition to the relationships between them to upgrade and formulate different AC models. In this section, we assume some scenarios as examples to show the dynamism of the HEAD metamodel.

- Scenario 1: Assume that an RBAC model is already formulated to define a policy for an organization. Hereinafter, due to some organizational changes and updates, the following occurs:
 - a. Some users who are already assigned to certain roles need to be assigned, based on their roles, to security levels.
 - b. Some other users now should be directly assigned to levels.
 - c. Both users in (a) and (b) are only allowed to access some sensitive objects (e.g., documents), which are also classified into levels based on their sensitivity.

Hence, some of already defined rules must be updated, and new rules must be defined. Figure 18 is an example of how the HEAD grammar is dynamic. Lines 2 to 5 indicate the already defined RBAC entities (with the attributes), and lines 13 to 21 indicate the already defined RBAC rules. Lines 22 to 31 indicate that some RBAC rules are updated to express hybrid MAC/RBAC rules. As we can see, in line 24, some subjects who are assigned to some roles are now assigned to some security levels, and in line 26, some objects are assigned to some security levels. Lines 32 to 38 indicate the definition of new MAC rules.

```

1 policy
2  RBAC(desc:String)
3  explicit subject(name:String) object(name:String) end
4  authorization role(roletype:String) end
5  procedural permission(pid:int) action(act:String) end
6
7  MAC(desc:String)
8  explicit subject(name:String) object(name:String) end
9  authorization securitylevel(slevel:String) end
10 procedural operation(op:String) end
11 end
12 rule:
13 (rule1:int){
14   RBAC.subject(RBAC.subject.name)[RBAC.role(RBAC.role.roletype)]{
15     RBAC.permission(RBAC.permission.pid){
16       RBAC.object(RBAC.object.name){
17         RBAC.action(RBAC.action.act)
18       }
19     }
20   }
21 }--> decision
22 (rule2:int){
23   RBAC.subject(RBAC.subject.name)[RBAC.role(RBAC.role.roletype)
24     [MAC.securitylevel(MAC.securitylevel.slevel)]]{
25     RBAC.permission(RBAC.permission.pid){
26       RBAC.object(RBAC.object.name)[MAC.securitylevel(MAC.securitylevel.slevel)]{
27         RBAC.action(RBAC.action.act)
28       }
29     }
30   }
31 }--> decision
32 (rule3:int){
33   MAC.subject(MAC.subject.name)[MAC.securitylevel(MAC.securitylevel.slevel)]{
34     MAC.object(MAC.object.name)[MAC.securitylevel(MAC.securitylevel.slevel)]{
35       MAC.operation(MAC.operation.op)
36     }
37   }
38 }--> decision

```

Figure 18. Dynamic AC metamodel: Scenario 1.

- Scenario 2: Assume that an ABAC model is already formulated, and the policy is already defined in an organization. Suppose that the organization has departments dept1, dept2, and dept3. However, due to some changing conditions, a new static and dynamic AC rules must be defined and others must be updated. The updated policy, due to new changing conditions, states the following:
 - a. Dynamic rule: subjects (users) in dept2 and dept3 are not allowed to access some objects after three failed password attempts (assuming that another level of authentication is needed before accessing the objects).
 - b. Static rule: some subjects in dept1 can determine what operations, (i) other subjects can perform, and (ii) some other subjects with the specific role can perform, on their objects.

Clearly, some already defined rules must also be updated, and new rules must be defined. As shown in Figure 19, the red and blue indicators refer to the modifications and the new expressions for rules. To answer the needed updates, in line 3, two additional attributes are defined as well as the attribute 'countPW' in line 8 to count the number of failed attempts while entering the password 'PW'. Another S_t entity is instantiated named condition (line 9) to check the subject's department and role based on (a) and (b) of the above policy updates. Lines 12 to 22 indicate that some ABAC rules are updated to answer the needed modifications in (a). As we can see, an action can be performed on an object if the following are true:

- The answer of condition (line 17) is true (the condition is true if the value of the dept attribute is equal to that of dept2 or dept3).
- Another authentication level is verified by entering the correct password 'PW' and another condition (contextualCondition) must return true (contextualCondition is true if the value of 'countPW' is less than or equal to three).

Note that the condition is verified at a certain point in time, specifically when something occurs. Hence, this rule deals with the dynamic behavior of the subjects. Note that our grammar is able to express a rule in another way starting with the object. As well, to answer the requirements in (b)-(i) and (b)-(ii), a new rule is expressed in lines 24 to 37.

- (i) An object with attribute type = type-value can be accessed by subjects with attribute address = address-value and perform some action act = act-value1, and some other action act = act-value2 if their dept = dept-value (dept1).
- (ii) An object with attribute type = type-value can be accessed by subjects with attribute address = address-value and perform some action act = act-value1, and some other action act = act-value2 if their dept = dept-value and their role = role-value.

As expressed in Scenarios 1 and 2, we can find that new types of attributes and entities can be defined to describe a larger set of rules to express (then enforce) static and dynamic policies. The above scenarios show the Dynamism of the HEAD metamodel, which comes after its Generality feature.

```

1= policy ABAC(descr:String)
2=   explicit subjAttr(sid:int name:String address:String
3     dept:String PW:String role:String)
4     objAttr(oid:int type:String details:String)
5   end
6   procedural actionAttr(act:String) end
7   setting context(expression:String)
8     contextCondition(time:String location:String countPW:int)
9     condition(cid:int)
10  end
11 end
12= rule: (ruleid1:int){
13   ABAC.subjAttr(ABAC.subjAttr.address ABAC.subjAttr.dept){
14     ABAC.objAttr(ABAC.objAttr.type){
15       ABAC.actionAttr(ABAC.actionAttr.act)
16       ABAC.actionAttr(ABAC.actionAttr.act){
17         ABAC.condition(ABAC.subjAttr.dept)
18         ABAC.context(ABAC.context.expression)
19         ABAC.contextCondition(ABAC.contextCondition.location ABAC.contextCondition.countPW)
20       }
21     }
22   } --> decision
23 (ruleid2:int){
24   ABAC.objAttr(ABAC.objAttr.type){
25     ABAC.subjAttr(ABAC.subjAttr.address ABAC.subjAttr.dept){
26       ABAC.actionAttr(ABAC.actionAttr.act)
27       ABAC.actionAttr(ABAC.actionAttr.act){
28         ABAC.condition(ABAC.subjAttr.dept)
29       }
30     }
31     ABAC.subjAttr(ABAC.subjAttr.address ABAC.subjAttr.dept ABAC.subjAttr.role){
32       ABAC.actionAttr(ABAC.actionAttr.act)
33       ABAC.actionAttr(ABAC.actionAttr.act){
34         ABAC.condition(ABAC.objAttr.type ABAC.subjAttr.dept ABAC.subjAttr.role)
35       }
36     }
37 } --> decision

```

Figure 19. Dynamic AC metamodel: Scenario 2.

5.3. Extensibility

Developing a generic and dynamic AC metamodel enables developing other important features, such as extensibility. An extensible AC metamodel means that new entities (or attributes) could be defined and integrated with already derived models to support new AC features in addition to the previous ones.

In this section, we assume that an RBAC model is already defined, and the needed policy is expressed in an organization. Due to new procedures and upgrades in the organization, the users who are already assigned to specific roles need to be classified into groups. For example, subjects who are assigned to role1 are classified into two groups, (group11 and group12), and subjects who are assigned to role2 are classified into three groups, (group21, group22, and group23). Besides the already defined permissions for role1 and role2, other permissions need to be specified based on user-group assignments. Hence, users' permissions are specified based on their roles, groups, and roles and groups. Figure 20 illustrates an extension for RBAC policy to support the notion of groups. The red indicators show the newly defined entities/attributes in addition to the existing ones with rule expressions. Lines 9 to 18 express the permissions of subjects who are assigned to roles regardless of their groups. Lines 17 to 25 express a new rule of permissions for subjects who are assigned to groups regardless of their roles. Lines 26 to 36 express the permissions of subjects based on their roles and groups; in line 31, conditions are used to check the users' (subjects') role and group to allow/deny them performing action(s).

```

1@policy RBAC(descr:String)
2  explicit subject(name:String) object(name:String) end
3@  authorization role(roletype:String) [group(grpid:int)] end
4  procedural action(act:String) permission(perm:String) end
5  setting condition(cid:int) end
6 end
7 rule:
8@ (ruleid1:int){
9  RBAC.subject(RBAC.subject.name)[RBAC.role(RBAC.role.roletype)]{
10   RBAC.permission(RBAC.permission.perm){
11     RBAC.object(RBAC.object.name){
12       RBAC.action(RBAC.action.act)
13     }
14   }
15 }
16 }--> decision
17 (ruleid2:int){
18   RBAC.subject(RBAC.subject.name)[RBAC.group(RBAC.group.grpid)]{
19     RBAC.permission(RBAC.permission.perm){
20       RBAC.object(RBAC.object.name){
21         RBAC.action(RBAC.action.act)
22       }
23     }
24   }
25 }--> decision
26 (ruleid3:int){
27   RBAC.subject(RBAC.subject.name)[RBAC.role(RBAC.role.roletype) RBAC.group(RBAC.group.grpid)]{
28     RBAC.permission(RBAC.permission.perm){
29       RBAC.object(RBAC.object.name){
30         RBAC.action(RBAC.action.act){
31           ABAC.condition(RBAC.role.roletype RBAC.group.grpid)
32         }
33       }
34     }
35   }
36 }--> decision

```

Figure 20. Extensibility: RBAC example.

Hence, having an advanced AC metamodel that is able to extend the existing models is a substantial requirement with technology progressions and upgrades.

5.4. Hierarchy of Entities

Hierarchical authorization is the authorization determined based on the hierarchy. Within this structure, access rights are specified by an entity's place in the hierarchy. The hierarchy defines the relationships between specific types of entities (e.g., roles). This feature can be employed to extend the derived AC models. As we can see in Section 2, several models and metamodels in the literature are extended to support the feature of hierarchy since it provides additional, granular access to resources for an organization and helps reduce maintenance costs. For example, in complex scenarios (e.g., IoT), administrators can start with creating several entities and then add their hierarchy. This would help in managing access to data with less maintenance costs compared to creating a large number of nonhierarchical entities. In this section, we show how HEAD metamodel grammar is able to define a hierarchy for any type of entities. For example, we have the following:

- Creating a hierarchy of roles and objects in RBAC: assume that, after expressing an RBAC policy, an organization needs to update/define new rules that support the hierarchy of roles (two levels of hierarchy) and objects (three levels of hierarchy). In Figure 21a, in lines 2 to 4, we define three levels of object hierarchy: objectL1, objectL2, and objectL3. Note that L1, L2, and L3 are concatenated with the entity name to indicate the level number. In lines 6 to 7, two levels of role are defined (roleL1, and roleL2). In lines 14 to 28, the rule states the following:

A subject with name = name-value assigned to roleL1 = role-value has permission to access an objectL1 = name-value and perform action act = act-value if condition is true, an objectL2 = name-value and perform action act = act-value, and an objectL3 = name-value and perform action act = act-value.

In lines 30 to 39, a subject assigned to a role in second level of hierarchy can access an object(s) in a second level of hierarchy, and another object(s) in a third level of

hierarchy. The rule states the following:

A subject with name = name-value assigned to roleL2 = role-value has permission to access an objectL2 = name-value and perform action act = act-value, and an objectL3 = name-value and perform action act = act-value.

Figure 21b, shows the defined entities/attributes, and the hierarchy of roles and objects. Note that if a subject is assigned to one or more roles, the expression could be written as follows:

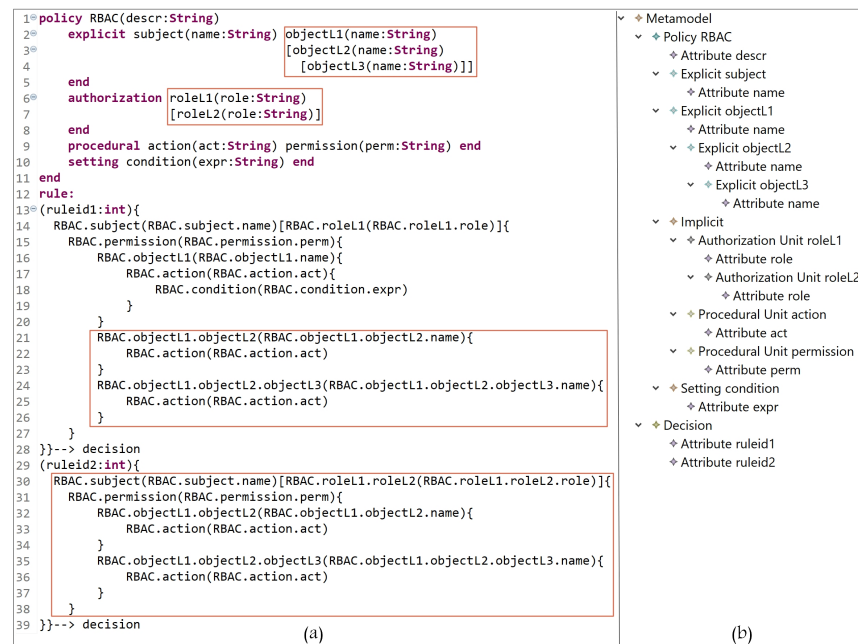
$$\begin{aligned} &RBAC.subject(RBAC.subject.name)[\\ &\quad RBAC.roleL1(RBAC.roleL1.role) \\ &\quad RBAC.roleL1.roleL2(RBAC.roleL1.roleL2.role) \\ &\quad RBAC.roleL1.roleL2.roleL3(RBAC.roleL1.roleL2.roleL3.role) \dots]\{ \dots \end{aligned}$$


Figure 21. RBAC: (a) definition of role/object hierarchy; (b) hierarchy of role/object entities.

6. Generating Policies: Examples and Illustrations

In the previous sections, we defined the grammar of the HEAD metamodel for specifying several AC models, then we defined a model language to describe AC security policies. In this section, the models expressed in the DSL are transformed to Java code in order to generate the AC policies. We provide examples of how the actual AC policies are generated for a given model(s) instances. AC policies are expressed at the system layer, where users interact.

- Example 1—RBAC policy: The doctors Mark and Joe in a hospital can read and write patients' prescriptions. The nurse Joyce is allowed to read these prescriptions.

In this example we have three rules:

1. Doctor Mark can read and write patients' prescriptions.
2. Doctor Joe can read and write patients' prescriptions.
3. The nurse Joyce can read patients' prescriptions.

In Figure 22, we illustrate a concrete model instance for the RBAC policy example, which is instantiated from the derived RBAC model based on the HEAD metamodel. For the above policy, we have the following entities/classes:

- E_x entities: subject (worker: name, dept, ...), and object (prescription: details, ...)
- AU entities: role (rType, ...)
- PU entities: action (aType, ...), permission (permId, ...)

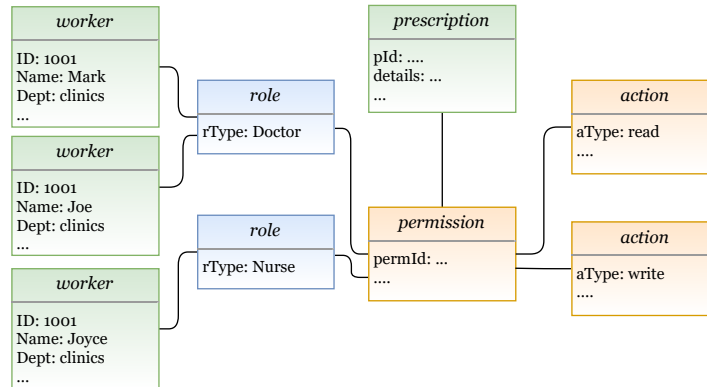


Figure 22. A model instance based on RBAC.

The generated RBAC policy in Figure 23 is modeled based on RBAC model, explained in Section 5.1.3. As shown in the Figure, the model includes several elements, and the three rules are generated based on them.

1. Three subjects (with attribute name): Mark, Joe, and Joyce.
2. One object (with attribute name): Prescription.
3. Two roles (with attribute rType): Doctor, and Nurse.
4. Two permission assignments (with attribute perm): DoctorPermission, and NursePermission.
5. Two actions (with attribute aType): Read, and Write.

```

RBAC Policy - organization:Clinic
--subject(s)--
  subject.name: Mark Joe Joyce
Any associations? (y/n) y
Mark -> role.rType: Doctor
Joe -> role.rType: Doctor
Joyce -> role.rType: Nurse
-----
--object(s)--
  object.name: Prescription
Any associations? (y/n) n
-----
--permission(s)--
  permission.permId: DoctorPermission NursePermission
--action(s)--
  DoctorPermission - action.aType: Read Write
  NursePermission - action.aType: Read

subject(s): [Mark [Doctor], Joe [Doctor], Joyce [Nurse]]
object(s): [Prescription]
permission(s): [DoctorPermission, NursePermission]
action(s): [Read Write, Read]

*****RULES*****
Rule#1
Mark [Doctor]{
  DoctorPermission{
    Prescription{Read Write}
  }->Allow
}
Rule#2
Joe [Doctor]{
  DoctorPermission{
    Prescription{Read Write}
  }->Allow
}
Rule#3
Joyce [Nurse]{
  NursePermission{
    Prescription{Read}
  }->Allow
}

```

Figure 23. Example 1: generating RBAC policy.

- Example 2—MAC/RBAC policy: In the clinics department of a hospital, the doctors Mark and Joe have a clearance level of “Top Secret” that is equal to the classification level of the object patient prescription. Hence, the doctors are allowed to read/write prescriptions. The nurse Joyce has the clearance level of “Secret” and can read patients’ prescriptions.

In this example we have the following rules:

1. Doctor Mark, whose clearance level is “Top Secret”, can read and write prescriptions that have a classification level equal to “Top Secret”.
2. Doctor Joe, whose clearance level is “Top Secret”, can read and write prescriptions that have a classification level equal to “Top Secret”.
3. Nurse Joyce, whose clearance level is “Secret”, can only read patients’ prescriptions.

In Figure 24, we illustrate a concrete model of hybrid MAC/RBAC policy example. Note that, in this example we use BIBA (developed by Kenneth J. Biba) as MAC variant. In the defined policy, some subjects are assigned to doctor and nurse roles. Subjects are permitted to read an object if their clearance level is \leq than the object’s classification level, and to write if it is greater than or equal (\geq). Note that if, for example, the clearance level for Doctor Joe is “secret”, then he is only allowed to read patients’ prescriptions. Hence, we have the following entities/classes:

- E_x class(es): subject (worker: name, dept, ...), and object (prescription: details)
- AU class(es): role (rType, ...), security level (level, ...)
- PU class(es): action (aType, ...), permission (permId, ...)

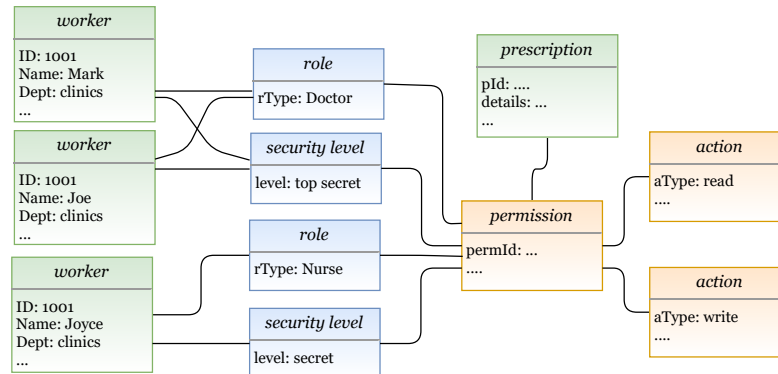


Figure 24. A model instance based on hybrid MAC/RBAC.

The generated MAC/RBAC policy in Figure 25 is modeled based on the hybrid MAC/RBAC model, explained in Section 5.1.5. As shown in the figure, the model includes several elements, and the three rules are defined based on them.

1. Three subjects (with attribute name): Mark, Joe, and Joyce.
2. One object (with attribute name): Prescription.
3. Two roles (with attribute rType): Doctor, and Nurse.
4. Two security levels (with attribute level): Top secret, and Secret.
5. Two permission assignments (with attribute perm): DoctorPermission, and NursePermission.
6. Two actions (with attribute aType): Read, and Write.

```

MAC_RBAC Policy org:Hospital
--subject(s)--
  subject.name: Mark Joe Joyce
subject-role associations? (y/n) y
Mark -> role.rtype: Doctor
Joe -> role.rtype: Doctor
Joyce -> role.rtype: Nurse

subject-seclevel associations? (y/n) y
Mark [Doctor] -> seclevel.level: TopSecret
Joe [Doctor] -> seclevel.level: TopSecret
Joyce [Nurse] -> seclevel.level: Secret
-----
--object(s)--
  object.name: Prescription
object-role associations? (y/n) n
object-seclevel associations? (y/n) y
Prescription -> seclevel.level: TopSecret
-----
--permission(s)--
  permission.perm: DoctorPermission NursePermission
--action(s)--
  DoctorPermission - action.atype: Read Write
  NursePermission - action.atype: Read

subject(s): [Mark [Doctor] [TopSecret], Joe [Doctor] [TopSecret], Joyce [Nurse] [Secret]]
object(s): [Prescription [TopSecret]]
permission(s): [DoctorPermission, NursePermission]
action(s): [Read Write, Read]

***RULES***
Rule#1
Mark [Doctor] [TopSecret]{
  DoctorPermission{
    Prescription [TopSecret]{Read Write}
  }->Allow
}
Rule#2
Joe [Doctor] [TopSecret]{
  DoctorPermission{
    Prescription [TopSecret]{Read Write}
  }->Allow
}
Rule#3
Joyce [Nurse] [Secret]{
  NursePermission{
    Prescription [TopSecret]{Read}
  }->Allow
}

```

Figure 25. Example 2: Generating MAC/RBAC policy.

7. Conclusions and Future Perspectives

The evolution of ubiquitous information systems has introduced significant challenges related to security and access control. Information systems should allow users to fulfill transparent access to resources at anytime, anywhere, and in any way, while protecting integrity and confidentiality within the creation of robust security policies. To confront the challenge of accessing resources, various research works were conducted, focusing on developing and enhancing AC modeling in five main directions, starting from (1) traditional access control models, (2) hybrid models, (3) extending AC models, (4) abstracting AC models, reaching to (5) AC metamodels.

On this basis, the objective of this paper is to provide an efficient AC metamodel that conforms to organizational (e.g., companies, industries and hospitals) AC security policies, and adapts the decision making, according to technology progressions to meet organizational and users' needs. Hence, we propose the HEAD AC metamodel, which takes into consideration the continuous technology changes and upgrades. Its meta-components are constructed after unifying the heterogeneous concepts of AC components. The DSL language of HEAD metamodel is defined for specifying any AC model; it is generic and able to create any component and attribute related to the traditional AC model or any new model. Furthermore, its structure is dynamic and able to define any new component (or attribute) and the relationships between all components; also, any derived model can be extended to follow any technological or organizational updates. Additionally, another powerful feature that exists in the HEAD metamodel is the hierarchy of components (any type of component) to meet hierarchical authorizations. We provide several scenarios to show its generality, dynamism, extensibility, and hierarchy; also, some examples are illustrated to show the generated rules of a policy. Despite providing many advantages, the metamodel may suffer from a drawback, which could be reflected in the vast amount of code that is needed to generate the required AC policies in large and complex systems where all

features must be implemented. Nevertheless, several approaches can be implemented to solve this issue.

The emergence of pervasive information systems and intelligent manufacturing has had an extensive impact on different directions, such as the future of the industry. Industry 4.0 is the modernization of traditional manufacturing using modern smart technology. It is based on smart industries where several physical and cyber technologies are merged with the aim of improving productivity, quality, performance, and management in the epoch of IoT. As progressions in technology in general, and IoT in particular, are taking place, the need for security has changed. Hence, organizations and industry sectors have now to rethink how to control access to resources through modern and enhanced AC methods. In Industry 4.0, smart sensors are used to collect huge amounts of environmental data, and a huge number of devices are connected to the internet, from sensors to factory machines, home appliances, hospital tools and equipment, and others. In the field of security and privacy, smart sensors are employed, for example, to send warning alarms to nearby areas in case of fire detection, use facial recognition technologies to send images of a thief to authorities within seconds of theft, and others. As a future perspective, we aim to explain how the HEAD metamodel can be implemented to specify and enforce AC policies, using a detailed case study for an industrial environment (non-IoT and IoT environments), with real applications and results.

Author Contributions: Conceptualization, N.K. and M.A.; formal analysis, N.K. and M.A.; visualization, N.K., M.A. and H.I.; software, N.K.; investigation, N.K., M.A. and H.I.; writing—original draft, N.K.; writing—review and editing, M.A. and H.I. supervision, M.A. and H.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant number 06351.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Acknowledgments: We acknowledge the support of Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT), and Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AC	Access Control
IoT	Internet of Things
HEAD	Hierarchical, Extensible, Advanced, and Dynamic
DAC	Discretionary Access Control Model
MAC	Mandatory Access Control Model
RBAC	Role-based Access Control Model
ABAC	Attribute-based Access Control Model
IS	Information System
HGABAC	Hierarchical Group and Attribute-based Access Control
HoBAC	Higher-order Attribute-based Access Control
CBAC	Category-based Access Control
CSPM	Cloud Security and Privacy Metamodel
WCMS	Web Content Management System
E _x	Explicit
I _m	Implicit
AU	Authorization Unit
PU	Procedural Unit
S _t	Setting

References

1. Zhang, Y.; Nakanishi, R.; Sasabe, M.; Kasahara, S. Combining IOTA and Attribute-Based Encryption for Access Control in the Internet of Things. *Sensors* **2021**, *21*, 5053.
2. Cruz-Piris, L.; Rivera, D.; Marsa-Maestre, I.; De La Hoz, E.; Velasco, J.R. Access control mechanism for IoT environments based on modelling communication procedures as resources. *Sensors* **2018**, *18*, 917. [[CrossRef](#)] [[PubMed](#)]
3. Kalsoom, T.; Ramzan, N.; Ahmed, S.; Ur-Rehman, M. Advances in sensor technologies in the era of smart factory and industry 4.0. *Sensors* **2020**, *20*, 6783. [[CrossRef](#)] [[PubMed](#)]
4. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Access Control in Cybersecurity and Social Media. *Cybersécurité et Médias Sociaux* **2021**, 69–105, ISBN 978-2-7637-5328-7.
5. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. A review of access control metamodels. *Procedia Comput. Sci.* **2021**, *184*, 445–452. [[CrossRef](#)]
6. Kashmar, N.; Adda, M.; Atieh, M. From Access Control Models to Access Control Metamodels: A Survey. In *Future of Information and Communication Conference*; Springer: Cham, Switzerland, 2019; pp. 892–911.
7. Rajpoot, Q.M.; Jensen, C.D.; Krishnan, R. Attributes enhanced role-based access control model. In *International Conference on Trust and Privacy in Digital Business*; Springer: Cham, Switzerland, 2015; pp. 3–17.
8. Servos, D.; Osborn, S.L. HGABAC: Towards a formal model of hierarchical attribute-based access control. In *International Symposium on Foundations and Practice of Security*; Springer: Cham, Switzerland, 2014; pp. 187–204.
9. Aliane, L.; Adda, M. HoBAC: Toward a higher-order attribute-based access control model. *Procedia Comput. Sci.* **2019**, *155*, 303–310. [[CrossRef](#)]
10. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Access control metamodel for policy specification and enforcement: From conception to formalization. *Procedia Comput. Sci.* **2021**, *184*, 887–892. [[CrossRef](#)]
11. Kashmar, N.; Adda, M.; Ibrahim, H. Access Control Metamodels: Review, Critical Analysis, and Research Issues. *J. Ubiquitous Syst. Pervasive Netw.* **2021**, *3*, in press.
12. Jaïdi, F.; Labbene Ayachi, F.; Bouhoula, A. A methodology and toolkit for deploying reliable security policies in critical infrastructures. *Secur. Commun. Netw.* **2018**, *2018*, 7142170. [[CrossRef](#)]
13. Myrbakken, H.; Colomo-Palacios, R. DevSecOps: A multivocal literature review. In *International Conference on Software Process Improvement and Capability Determination*; Springer: Cham, Switzerland, 2017; pp. 17–29.
14. Mao, R.; Zhang, H.; Dai, Q.; Huang, H.; Rong, G.; Shen, H.; Chen, L.; Lu, K. Preliminary findings about devsecops from grey literature. In Proceedings of the 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), Macau, China, 11–14 December 2020; pp. 450–457.
15. Hu, V.C.; Kuhn, D.R.; Ferraiolo, D.F.; Voas, J. Attribute-based access control. *Computer* **2015**, *48*, 85–88. [[CrossRef](#)]
16. Sandhu, R.; Coyne, E.; Feinstein, H.; Role-Based, C.Y. Access control models. *IEEE Comput.* **2013**, *29*, 38–47. [[CrossRef](#)]
17. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. A new dynamic smart-AC model methodology to enforce access control policy in IoT layers. In Proceedings of the 2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT), Montreal, QC, Canada, 25–31 May 2019; pp. 21–24.
18. Sun, K.; Yin, L. Attribute-role-based hybrid access control in the internet of things. In *Asia-Pacific Web Conference*; Springer: Cham, Switzerland, 2014; pp. 333–343.
19. Hasiba, B.A.; Kahloul, L.; Benharzallah, S. A new hybrid access control model for multi-domain systems. In Proceedings of the 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), Barcelona, Spain, 5–7 April 2017; pp. 0766–0771.
20. Kuhn, D.R.; Coyne, E.J.; Weil, T.R. Adding attributes to role-based access control. *Computer* **2010**, *43*, 79–81. [[CrossRef](#)]
21. Aftab, M.U.; Qin, Z.; Hundera, N.W.; Ariyo, O.; Son, N.T.; Dinh, T.V. Permission-based separation of duty in dynamic role-based access control model. *Symmetry* **2019**, *11*, 669. [[CrossRef](#)]
22. Kim, S.; Kim, D.K.; Lu, L.; Song, E. Building hybrid access control by configuring RBAC and MAC features. *Inf. Softw. Technol.* **2014**, *56*, 763–792. [[CrossRef](#)]
23. Li, H.; Wang, S.; Tian, X.; Wei, W.; Sun, C. A survey of extended role-based access control in cloud computing. In *Proceedings of the 4th International Conference on Computer Engineering and Networks*; Springer: Cham, Switzerland, 2015; pp. 821–831.
24. Nguyen, P.H.; Nain, G.; Klein, J.; Mouelhi, T.; Le Traon, Y. Model-driven adaptive delegation. In *AOSD'13: Proceedings of the 12th Annual International Conference on Aspect-Oriented Software Development*; ACM: New York, NY, USA, 2013; pp. 61–72.
25. Adda, M.; Aliane, L. HoBAC: Fundamentals, principles, and policies. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 5927–5941. [[CrossRef](#)]
26. Slimani, N.; Khambhammettu, H.; Adi, K.; Logrippo, L. UACML: Unified access control modeling language. In Proceedings of the 2011 4th IFIP International Conference on New Technologies, Mobility and Security, Paris, France, 7–10 February 2011; pp. 1–8.
27. Barker, S. The next 700 access control models or a unifying meta-model? In *SACMAT'09: Proceedings of the 14th ACM symposium on Access Control Models and Technologies*; ACM: New York, NY, USA, 2009; pp. 187–196.
28. Bertolissi, C.; Fernández, M. A metamodel of access control for distributed environments: Applications and properties. *Inf. Comput.* **2014**, *238*, 187–207. [[CrossRef](#)]

29. Abd-Ali, J.; El Guemhioui, K.; Logrippo, L. A Metamodel for Hybrid Access Control Policies. *J. Softw.* **2015**, *10*, 784–797. [[CrossRef](#)]
30. Alves, S.; Degtyarev, A.; Fernández, M. Access control and obligations in the category-based metamodel: A rewrite-based semantics. In *International Symposium on Logic-Based Program Synthesis and Transformation*; Springer: Cham, Switzerland, 2014; pp. 148–163.
31. Khamadja, S.; Adi, K.; Logrippo, L. Designing flexible access control models for the cloud. In *Proceedings of the 6th International Conference on Security of Information and Networks*, Aksaray, Turkey, 26–28 November 2013; pp. 225–232.
32. Xia, T.; Washizaki, H.; Kato, T.; Kaiya, H.; Ogata, S.; Fernandez, E.B.; Kanuka, H.; Yoshino, M.; Yamamoto, D.; Okubo, T.; et al. Cloud security and privacy metamodel. In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, Funchal, Portugal, 22–24 January 2018; pp. 379–386.
33. Martínez, S.; Garcia-Alfaro, J.; Cuppens, F.; Cuppens-Boulahia, N.; Cabot, J. Towards an access-control metamodel for web content management systems. In *International Conference on Web Engineering*; Springer: Cham, Switzerland, 2013; pp. 148–155.
34. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Deriving access control models based on generic and dynamic metamodel architecture: Industrial use case. *Procedia Comput. Sci.* **2020**, *177*, 162–169. [[CrossRef](#)]
35. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Smart-ac: A new framework concept for modeling access control policy. *Procedia Comput. Sci.* **2019**, *155*, 417–424. [[CrossRef](#)]

CHAPTER 4




INSTANTIATION AND IMPLEMENTATION OF HEAD METAMODEL IN INDUSTRIAL ENVIRONMENT: NON-IOT AND IOT CASE STUDIES

Submitted to Journal of Sensors (Special Issue: Cybersecurity in the Internet of Things), 2022

Abstract: The industry domain is one of the world's hugest heterogeneous organizations, it gains hugely from the increased adoption of ubiquitous computing and digital transformation which brings out new challenges to provide solutions and integrate advanced and self-ruling systems in critical and heterogeneous structures. Controlling access and ensuring security and cybersecurity in industry 4.0 environments is a challenging task due to the increasing distribution of resources. To preserve security and privacy, organizations need advanced AC methods. In this chapter, we use HEAD metamodel to specify the needed AC policies for two case studies inspired by the computing environment of Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada. The first is for ITMI's local environment and the second for the IoT environment. Using the DSL of HEAD metamodel the needed model is derived for each case study, then xtend notation is used to generate the concrete instance for each model. At the system level, AC rules are generated as Cypher queries to represent the Next Generation Access Control (NGAC) policy as a graph using Neo4j database. The results show that HEAD metamodel can be adapted and integrated with various local and distributed environments, able to serve as a unifying framework. Moreover, an administrative panel is implemented, as an additional example, using VB.NET and SQL to show that HEAD metamodel can be implemented to generate AC rules using other platforms.

Résumé: Le domaine de l'industrie est l'une des plus grandes organisations hétérogènes au monde, il profite énormément de l'adoption accrue de l'informatique omniprésente et de la transformation numérique qui fait apparaître de nouveaux défis pour fournir des solutions et intégrer des systèmes avancés et autonomes dans des structures critiques et hétérogènes. Contrôler l'accès et assurer la sécurité et la cybersécurité dans les environnements de l'industrie 4.0 est une tâche difficile en raison de la répartition croissante des ressources. Pour préserver la sécurité et la confidentialité, les organisations ont besoin de méthodes CA avancées. Dans ce chapitre, nous utilisons le métamodèle HEAD pour spécifier les politiques CA nécessaires pour deux études de cas inspirées de l'environnement informatique de l'Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada. Le premier est pour l'environnement local d'ITMI et le second pour l'environnement IoT. À l'aide du métamodèle DSL de HEAD, le modèle nécessaire est dérivé pour chaque étude de cas, puis la notation xtend est utilisée pour générer l'instance concrète de chaque modèle. Au niveau du système, les règles CA sont générées sous forme de requêtes Cypher pour représenter la politique de contrôle d'accès de nouvelle génération (NGAC) sous forme de graphique à l'aide de la base de données Neo4j. Les résultats montrent que le métamodèle HEAD peut être adapté et intégré à divers environnements locaux et distribués, capables de servir de cadre fédérateur. De plus, un panneau d'administration est implémenté, comme exemple supplémentaire, en utilisant VB.NET et SQL pour montrer que le métamodèle HEAD peut être implémenté pour générer des règles CA en utilisant d'autres plateformes.

Instantiation and Implementation of HEAD Metamodel in an Industrial Environment: non-IoT and IoT Case Studies

Nadine Kashmar ^{1,*} , Mehdi Adda ¹ , Hussein Ibrahim ² , Jean-François Morin ² and Tony Ducheman ²

¹ Département de Mathématiques, Informatique et Génie, Université du Québec à Rimouski, 300 Allée des Ursulines, QC G5L 3A1, Canada; mehdi_adda@uqar.ca

² Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC G4R 5B7, Canada; {houssein.ibrahim; Jean-Francois.Morin; Tony.Ducheman}@itmi.ca

* Correspondence: nadine.kashmar@uqar.ca

Abstract: Access to resources can take many forms, digital access via an onsite network, remote from an external site, website, etc.; or physical access to labs, machines, information repositories, etc. Whether access to resources is digital or physical, it must be allowed, denied, revoked, or disabled using robust and coherent access control (AC) models. What makes the process of AC more complicated is the emergence of digital transformation technologies and pervasive systems such as the internet of things (IoT) and industry 4.0 systems, especially with the increasing demand for transparency in users' interaction with various applications and services. Controlling access and ensuring security and cybersecurity in IoT and industry 4.0 environments is a challenging task due to the increasing distribution of resources and the massive presence of cyber-threats and cyber-attacks. To preserve the security and privacy of users and industry sectors, we need an advanced AC metamodel that is able to define all the needed components and attributes to derive various instances of AC models and follow the new and increasing demand AC requirements due to continuous technology upgrades. Due to several limitations in the existing metamodels and their inability to answer the recent AC requirements, we have developed a Hierarchical, Extensible, Advanced, Dynamic (HEAD) AC metamodel with notable features that address various recent AC requirements. In this paper, HEAD metamodel is employed to specify the needed AC policies for two case studies inspired by the computing environment of Institut Technologique de Maintenance Industrielle (ITMI)-Sept-Îles, QC, Canada; the first is for ITMI's local (non-IoT) environment and the second for ITMI's IoT environment. For each case study, the needed AC model is derived using the domain-specific language (DSL) of HEAD metamodel, then xtend notation (an expressive dialect of Java) is used to generate the needed java code which represents the concrete instance of the derived AC model. At the system level, to get the needed AC rules, Cypher queries are generated and then injected into Neo4j database to represent the Next Generation Access Control (NGAC) policy as a graph. NGAC framework is used as an enforcement point for the generated rules of each case study. The results show that HEAD metamodel can be adapted and integrated with various local and distributed environments, able to serve as a unifying framework, answer the current AC requirements and follow the needed policy upgrades. Moreover, we implement an administrator panel, as an additional example, using VB.NET and SQL to show that HEAD metamodel can be implemented to generate AC rules using other platforms.

Keywords: access control; metamodel; security and privacy; policy; DSL; cybersecurity; digital transformation; IoT; industry 4.0; NGAC; Neo4j, graph database; Cypher query language

Citation: Kashmar, N.; Adda, M.; Ibrahim, H.; Morin, J.; Ducheman, T. Instantiation and Implementation of HEAD Metamodel in an Industrial Environment: non-IoT and IoT Case Studies. *Sensors* **2021**, *11*, 0. <https://doi.org/>

Received:

Accepted:

Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Submitted to *Sensors* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The continuous advancement in technology in general and the internet of things (IoT) in particular has changed the need for security and privacy. Currently, organizations and industry sectors should rethink how to control access to their logical and physical resources through modern and enhanced access control (AC) methods. The

37 current pervasive and ubiquitous computing environments with the digital transforma-
38 tion, such as IoT, industry 4.0, etc., need to provide effective protection for resources
39 which are distributed everywhere and users can access anywhere, anytime, and anyhow.
40 Consequently, the evolution of AC policy languages should follow the development of
41 computing environments, highly ubiquitous technologies and information systems (IS),
42 especially with the concept of IoT and industry 4.0 (or Industrial IoT (IIoT)) applications.
43 With this reality, developing a new policy language is increasingly challenging due to
44 the dynamic and heterogeneous structures of the current networking generation. [1,2].
45 Moreover, an AC policy language should answer the increasing demand of authorizing
46 and controlling access to resources, it should be able to express a set of rules which are
47 derived from business requirements, plans, technical guidelines, and constraints. In
48 any organization or industry sector, the administrative responsibility of an IS is setting
49 policy rules for controlling access to objects which are accessed and administered by a
50 domain. To mitigate security threats that are accompanied by information technologies,
51 the definition of AC rules must consider the possible threats in a domain and should be
52 concise and easy to understand so that everyone can follow the guidelines set forth in it.
53 Hence, AC rules outline the way of working with information and resources, once they
54 are created the domain's AC policy is customized [3].

55 However, AC is defined as the process of restricting access to resources based
56 on a predefined set of AC rules known as AC policy, it consists of two main stages:
57 authentication and authorization. Authentication is the process where an unknown
58 subject (or user) after verifying his identity becomes a known user, and this process is
59 insufficient to protect resources. Authorization is the process of allowing/denying an
60 authenticated user to access a resource, a user is allowed or denied to perform some
61 action(s) on a resource after checking the regulations and policies of an organization or
62 industry sector [4,5], it includes the following phases [2,6]:

- 63 • defining a security policy (set of AC rules);
- 64 • selecting an AC model that matches the defined policy;
- 65 • implementing the model and enforcing the defined AC rules.

66 To control access to resources, the common AC models are implemented in dif-
67 ferent centralized and distributed computing environments are discretionary access
68 control (DAC), mandatory access control (MAC), role-based access control (RBAC), and
69 attribute-based access control (ABAC) models [5,7,8]. Moreover, various hybrid models,
70 extended models, and abstracted models are also implemented to enhance the features
71 of common models and allow the definition of a larger set of AC policies. The topic of
72 AC metamodels is a recent research issue in this domain, they are proposed to include
73 the features of all models and work as a unifying framework, their components usually
74 describe generic concepts. Accordingly, AC models are special instances of an AC meta-
75 model, they are used to specify and define AC policy languages that explain the way in
76 what, how, and when objects (resources) can be accessed by subjects (users) [1,8,9]. The
77 ubiquitous nature of the current computing environments, especially industry 4.0, urges
78 the need to develop advanced AC metamodels that are able to answer the increasing
79 demand of AC requirements. Due to the limitations of the existing AC metamodels
80 [6,8,10], we have proposed a Hierarchical, Extensible, Advanced, and Dynamic (HEAD)
81 AC metamodel [1]. HEAD metamodel considers the challenging requirements that
82 accompany the various technology progressions, for example, the need (1) to create
83 hierarchical AC components that fit the various hierarchical organizational structures,
84 (2) to define and upgrade a larger set of AC policies, (3) to have several AC solutions
85 within the same organization (e.g., local network, IoT, and cloud), and many other
86 challenges [9]. The domain-specific language (DSL) of HEAD metamodel is generic
87 enough to include the heterogeneity of AC models which allows for defining larger sets
88 of AC policies, it supports the hierarchy of all components to conform to the hierarchical
89 organizational structure, and the derived models can be extended to follow the needed

90 policy updates, also it is dynamic since various components/attributes can be added to
91 follow technology progressions and to include several AC solutions in an organization.

92 In this paper, we use the DSL of HEAD metamodel (explained in [1]) to derive the
93 needed AC models that fit the two case studies inspired by the computing environment
94 of Institut technologique de maintenance industrielle (ITMI)-Sept-Îles, QC, Canada. The
95 first case study is related to ITMI's local computing environment (in this paper we
96 consider it as a non-IoT environment), and the second case study is related to their
97 IoT environment. Then, the derived models of each case are encoded using Eclipse
98 Xtend notation (an expressive dialect of Java) to represent the concrete instance of AC
99 policies and generate the needed java code which expresses the AC rules as Cypher
100 queries. As an enforcement point and to verify the accuracy and the coherence of the
101 concrete instances of AC policies, Cypher queries are injected into the Neo4j database to
102 represent the generated rules in a form of Next Generation Access Control (NGAC) [11]
103 policy graph—the objects, the relationships between them, and the subjects that interact
104 with the system in a form that adheres the semantics of ITMI. NGAC policy graph is
105 constructed with the basic policy elements and a fixed set of relationships [11–13]. The
106 NGAC policy elements that are represented in the NGAC graph are:

- 107 • the authenticated users (U), and their user attributes (UA).
- 108 • the objects (O) that need to be accessed, and their attributes (OA).
- 109 • the policy class(es) (PC).
- 110 • the assignment relationships between U/O-UA/OA, UA/OA-UA/OA, and UA/OA,
111 in addition to the appropriate PC(s).
- 112 • the association relationships to define the access rights associated with some UAs
113 to perform operations on some objects specified by OAs.

114 Both cases studies are presented to show that HEAD AC metamodel can be adapted to an-
115 swer the AC requirements of non-IoT and IoT environments. However, the contribution
116 of this paper can be summarized as follows:

- 117 • Presents two new industrial case studies with the implementation of a new ad-
118 vanced metamodel, named HEAD metamodel; this metamodel (with its advanced
119 features) is itself a contribution to the domain.
- 120 • Provides modern AC tools and shows how they are applicable and can be adapted
121 within different computing environments, especially industrial environments.
- 122 • Helps researchers in the domain answer their questions of how AC metamodels
123 work in real industrial contexts (or any computing environment) starting from the
124 definition of informal policies reaching to the policy enforcement.

125 The remainder of this paper is organized as follows: section 2 reviews some of the
126 related work in this domain. A summary of HEAD metamodel and its characteristics
127 are explained in section 3. To illustrate the relevance of our metamodel, the subject of
128 study is described in detail in section 4, then the first case study—ITMI's local (non-
129 IoT) environment— is presented in section 5, and the second case study—ITMI's IoT
130 environment— is presented in section 6. In section 7, we show another example of how
131 HEAD metamodel can be implemented using VB.NET and SQL database to generate
132 the needed AC rules. In section 8 we explain the evaluation and validation of the HEAD
133 metamodel. In section ?? we present the limitations of the HEAD metamodel. Section 9
134 concludes this paper with the future perspectives.

135 2. Related Works

136 In the field of information security, the main concern for organizations and industry
137 sectors is to keep the secrecy of their information and prevent any illegal access to
138 their logical and physical resources, especially with the presence of cyber-criminals and
139 cyber-attacks [14,15]. Currently, resources are no longer located within local areas, they
140 are distributed on different sites and need to be accessed via various private and public
141 networks. Moreover, the emergence of new technologies and digital transformation

142 urges the need for organizations to protect their private and public networks, especially
143 with the evolution of industry 4.0 and the IoT concept. In this context, various AC
144 models and metamodels are implemented in different computing environments to
145 protect resources and information from any unauthorized access. Unfortunately, they
146 have limited features and are insufficient to meet the current AC requirements and
147 follow the needed technology upgrades [7,9,10]. In this section, we review some of the
148 proposed AC methods in this domain, and also summarize the existing AC metamodels
149 [7–9] that are proposed to instantiate different AC models.

150 To prevent insider threats in organizations a function-based AC method inspired
151 by functional encryption is proposed in [16]. It stores access authorizations as a three-
152 dimensional tensor where users can invoke authorized commands at different levels
153 such as data segments, they are authorized to use a command on an object, and are
154 forbidden to use the same command on another object. In [17], Qi et al. propose a scalable
155 industry data AC system for the RFID-enabled supply chain to provide an item-level
156 data AC mechanism that defines and enforces AC policies based on both the participants'
157 role attributes and the products' RFID tag attributes. For industrial automation and
158 control systems (IACS) and similar automation systems of the smart energy grid, an
159 eXtended Access Control Markup Language (XACML)-based AC system is described by
160 Ruland et al. [18] to protect connected devices and associated safety-relevant settings
161 from unauthorized access. Their proposed AC method is composed of a two-stage AC
162 schema. They first evaluate policies based on XACML, and the second uses information
163 about the system's behavior to prevent any malicious or accidental operations from
164 having a negative impact on system stability. The system design and implementation
165 consider safety requirements (e.g., timing requirements, the availability...) to enable
166 integration in safety-critical environments.

167 For secure information sharing in an IIoT, in [19], Ulltveit-Moe et al. investigate how
168 to secure information sharing with external vendors, and they identify the necessary
169 security requirements in this domain. Also, they propose a roadmap to improve security
170 in IIoT which investigates short-term and long-term solutions to IIoT devices. The
171 former is mainly based on integrating existing good practices such as firewalls, intrusion
172 detection systems, etc. The latter outlines a long-term solution with fine-grained AC
173 for sharing data between external entities that would support cloud-based data storage.
174 Ray et al. [12] demonstrate how AC for a remote healthcare monitoring (RHM) that uses
175 an IoT framework can be specified using ABAC model. They present the healthcare AC
176 requirements using a motivating example, then they explain how NGAC can be used for
177 specifying the AC policies. Moreover, a context-sensitive RBAC (CRABC) scheme is pro-
178 posed by Alagar et al. [20] for securing the environment of the healthcare IoT industry
179 which is composed of large-scale connectivity of medical devices, patients, physicians,
180 etc. with the vastness of information collection and sharing. CRBAC combines roles, at-
181 tributes, and context to formulate context constraints to enforce access and flow controls.
182 Another AC model for the IoT in healthcare, named Enhanced Context-aware Capability
183 based AC (ECCAPAC), is proposed by Ahamed et al. [21] to make the medical network
184 resilient against crypto attacks by taking into account the trust value of the object based
185 on relevance and node importance.

186 Despite that the proposed AC models in various industrial computing environments
187 come up with solutions for dedicated scenarios or case studies, they have limited features
188 compared to the increasing demand for security and privacy in the current computing
189 environments. The current fact of networking environments imposes the need to define
190 new components/attributes for the existing AC models in order to upgrade the defined
191 AC policies and allow defining and enforcing a larger set of static and dynamic AC
192 policies. This feature is not addressed in the proposed models, also the feature of
193 hierarchy to define multiple levels of components is not considered (e.g., role hierarchy).

194 Moreover, the evolution of pervasive ISs and intelligent manufacturing has had a
195 substantial influence on the future of the industry. Industry 4.0 (or IIoT) is the moderniza-

tion of conventional manufacturing using modern smart technology. In smart industries, several physical and cyber technologies are integrated to improve productivity, quality, performance, and management in the age of IIoT [22]. All of this raises the challenge to develop AC metamodels that serve as unifying frameworks for deriving advanced AC models able to follow the needed technology upgrades and allow defining and enforcing a larger set of static and dynamic AC policies. In [9], we summarize the development stages of AC methods starting from common AC models, hybrid models, extending AC models, abstracting AC models, and reaching AC metamodels which is the recent research issue in this domain. In [7,8], we find that some metamodels are developed based on a general notion that encompasses the AC features of some common AC models, and they are proposed as generic metamodels. Some other metamodels are developed based on the concept of combining some AC models, these metamodels are proposed as hybrid metamodels to provide a generic base metamodel concept. Other metamodels are proposed as metamodel extensions for some of the existing metamodels and software development frameworks. The proposed metamodels provide some development approaches in the field, but they have several limitations since (1) they are not generic enough to derive the needed AC models (common models, hybrid models, and other models); (2) they are not flexible and dynamic enough to follow technology upgrades; (3) they are not extensible, also the derived models and the defined policies cannot be extended; (4) they do not support the hierarchy feature for all model components (e.g., role, action, context, etc.), this feature is essential for nowadays computing environments to fit the hierarchical organizational structures where, for example, a number of users are assigned to various roles in a hierarchy and have the right to access various objects also in a hierarchy; (5) the issues of collaboration and interoperability between AC models are not addressed; and (6) they do not consider the concept of migrating AC policies from one model to another. To address these limitations, we have designed and developed HEAD metamodel [1] where its advanced features allow deriving different models (existing and even non-existing models). In this paper, we use this HEAD metamodel to derive the needed AC models for two case studies inspired by ITMI's environment in order to show how it can be adapted to various industrial and organizational computing environments. Also, as to show how various components and attributes for static and dynamic AC policies can be defined in addition to the common ones, and to illustrate how it supports the feature of the component hierarchy.

3. HEAD Metamodel

As mentioned earlier, one of the recent research directions in the field of security and privacy is developing advanced AC metamodels [6,8]. To answer the current AC requirements, especially with the emergence of digital transformation technologies and pervasive systems (e.g., industry 4.0), an AC metamodel must consider the dynamic and heterogeneous nature of computing environments, in addition to the continuous technology progressions. In [1], we propose a hierarchical, extensible, advanced, and dynamic AC metamodel, named HEAD metamodel, with advanced features compared to other proposed AC metamodels in the literature. Its distinct design and the new opportunities it opens in the domain are described in [9]. In this section, we briefly explain its meta-components (or meta-classes) and features to show, later in this paper, how it can be employed to derive the needed AC models for two case studies (explained in sections 5 and 6) inspired from ITMI's non-IoT and IoT computing environments, then generate the needed AC rules in order to enforce them. The meta-components (or meta-classes) of HEAD metamodel are the EXPLICIT (E_x), IMPLICIT (I_m), and SETTING (S_t), shown in Figure 1 [1]:

- E_x represents the actual and the existing entities/classes, such as subjects and objects of any organization or industry sector.

- 247 • I_m represents the described entities/classes. I_m includes AUTHORIZATION UNIT
- 248 (AU) entities/classes such as roles, security levels, etc., and PROCEDURAL UNIT
- 249 (PU) entities/classes such as actions, permissions, etc;
- 250 • S_t represents the concepts which are included to have more regulated access to
- 251 resources, such as entities/classes of context, contextual conditions, etc.

252 The relationships between HEAD metamodel components can be described as follows:

- 253 1- between E_x and AU is to allow assigning zero or many, for example, subjects to
- 254 roles, groups, etc.;
- 255 2- between AU and PU, and PU and E_x is to describe which AUs (e.g., groups) can
- 256 perform zero or many PUs (e.g., actions) on some other E_x (e.g., objects);
- 257 3- E_x and I_m could have zero or many S_t (e.g., condition) to fulfill access requests;
- 258 4- the self-association edge on each of the meta-classes is to allow formulating hybrid
- 259 models by associating, for example, AUs with other AUs, PUs with other PUs, etc;
- 260 5- the aggregation association for each of the meta-classes is to allow the creation of
- 261 hierarchies of the derived components.

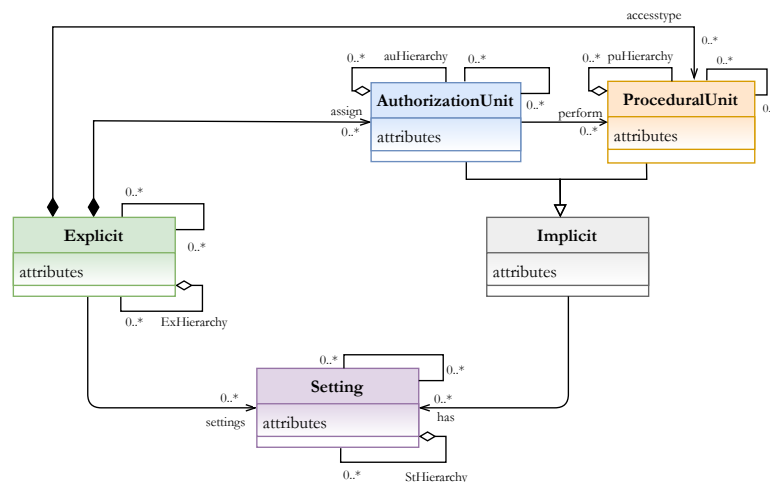


Figure 1. HEAD Metamodel [1]

262 The DSL of the HEAD metamodel is explained in detail in [1], it can be used to

263 derive various AC models (common AC, hybrid AC, and other models). For example:

- 264 • to derive MAC model, the subject and object classes must be instantiated from E_x ,
- 265 the security level must be instantiated from AU, and the operation class must be
- 266 instantiated from PU.
- 267 • in Figure 2 (a), we illustrate an RBAC example showing the derived classes of subject
- 268 and object from E_x , the derived role class from AU, and the derived permission and
- 269 action classes from PU. Note that, the derived instances (subjects, objects ...) are
- 270 indicated with the same colors of E_x , AU, and PU meta-classes of Figure 1. The
- 271 definition of E_x (subject and object), AU (role), and PU (permission and action)
- 272 classes, in addition to the rule expression, is shown in Figure 2 (b). The keywords
- 273 explicit, authorization, and procedural are used to define the needed instances with
- 274 their attributes; and the keyword rule is used to express the needed AC rule(s) of
- 275 the RBAC policy.

276 Using HEAD metamodel, the access to resources is determined by the attributes

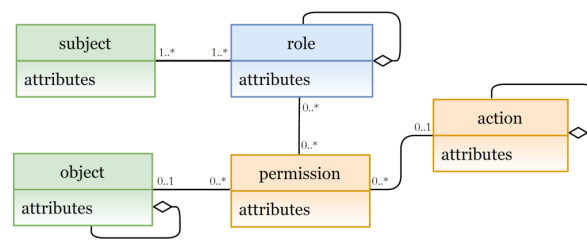
277 of the defined entities over the access request including subject, object, action, context,

278 etc. It allows enforcing complex AC policies that involve an arbitrary combination of

279 attributes with static, dynamic, and relationship values, thus extending beyond the

280 proposed AC metamodels in the literature [1,9]. Also, it allows deriving models of high

281 granularity for access policies, hence fitting the various needs of AC requirements.



(a)

```

policy RBAC(descr:String)
  explicit subject(name:String) object(name:String) end
  authorization role(roletype:String) end
  procedural action(act:String) permission(perm:String) end
end
rule:
(ruleid:int){
  RBAC.subject(RBAC.subject.name)[RBAC.role(RBAC.role.roletype)]{
    RBAC.permission(RBAC.permission.perm){
      RBAC.object(RBAC.object.name){
        RBAC.action(RBAC.action.act)
      }
    }
  }
}
}--> decision
  
```

(b)

Figure 2. Example: (a) RBAC model [1]; (b) RBAC policy definition

282 4. The Subject of Study: Technological Institute for Industrial Maintenance (ITMI)

283 ITMI is a technology transfer center affiliated with the Cégep de Sept-Îles, QC,
 284 Canada specializing in industrial maintenance. In an industrial field where the reli-
 285 ability of machines/devices is of strategic importance, ITMI offers tailor-made support
 286 to North Shore and Quebec industries and provides technical services to reduce down-
 287 time, minimize maintenance costs, optimize productivity, lessen training costs, and
 288 increase the success rate. Since its creation in 2008, ITMI has continued to grow and
 289 expand its fields of expertise to adapt to the needs of industries on the one hand and
 290 to technological changes on the other. The research efforts are centered on industry
 291 4.0, IoT, embedded systems, artificial intelligence, energy intelligence, computer-aided
 292 design, and others. ITMI has laboratories with extensive research infrastructure and
 293 regularly updated machines/devices which include industrial machinery, hydraulic and
 294 pneumatic, prototyping and 3D printing, and drones.

295 4.1. ITMI Departments

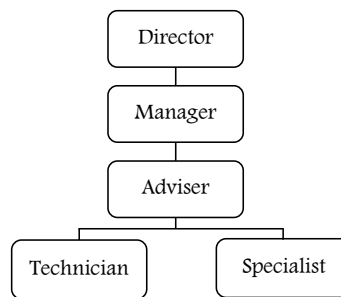
- 296 • Design and Maintenance: forms the core of ITMI's activities, it aims to provide
 297 predictive and proactive maintenance to optimize the efficiency of production assets
 298 and reduce operating costs. To achieve this, all the related software/hardware
 299 concepts and designs are handled.
- 300 • Digital Audit: is accredited by the Government of Quebec, its expertise in industry
 301 4.0 is employed to diagnose an industry sector and provides it with a digital plan
 302 to become an industry 4.0. The main role of this department is to provide reports
 303 or dashboards with the level of digital maturity, strengths, and weaknesses, then
 304 produce the best digital solutions to implement for an enterprise.
- 305 • IoT and Embedded Systems: helps enterprises evolve in the industrial world
 306 where systems cooperate and communicate with each other and with an enterprise.
 307 Embedded systems are defined as hardware/software packages integrated for the
 308 purpose of performing specific functional tasks. Department's expertise in IoT and
 309 embedded systems translates concretely into products developed for companies in
 310 different sectors: mining industry, rail transport, metallurgy, fishing, etc. Digital

311 technologies give the chance to decentralize the decision chain, opt for remote
 312 monitoring, generate SMS and email alerts and ultimately reorganize the value
 313 chain for an enterprise or industry.

- 314 • Information Technology (IT): is responsible for the implementation of new technolo-
 315 gies. Its role has been evolving over time as the digital transformation began to bear
 316 a greater significance within enterprises. As well, this department performs essen-
 317 tial tasks such as managing and assigning accounts and resources/machines/devices
 318 to users, technical support to all other departments, ensuring the privacy and secu-
 319 rity of ITMI's systems, carrying out the tasks of machine maintenance and support,
 320 etc.

321 4.2. ITMI Workers

322 In general, ITMI has four types of workers (users) in each department, in addition
 323 to the main director, they are specified into four roles: manager, adviser, specialist,
 324 and technician. Note that, ITMI is a sub-division of the Department of Research and
 325 Innovation (DRI) at Sept-Îles and it is directed by the main director of DRI. Figure 3
 326 illustrates the hierarchy of roles at ITMI; the manager who is directed by the main director
 327 of DRI directs the advisers of design and maintenance, digital audit, IoT and embedded
 328 systems, and information technology departments. Advisers of each department direct
 329 their specialists and technicians. Note that, advisers, specialists, and technicians can
 330 share their expertise among the departments based on the running project(s). Their
 access rights are explained in the case studies of sections 5 and 6.



331 **Figure 3.** ITMI - Role heirarchy

332 4.3. ITMI Resources

333 There are two types of resources at ITMI, logical and physical.

334 4.3.1. Logical Resources

- 335 • Local Database: two basic kinds of data are stored in the local database for each
 336 worker or machine. The first is static data entered into the system when a worker
 337 is newly hired, this includes some personal details such as name, address, dob,
 338 experience, and other information; or when a machine/device is newly installed
 339 such as specs, installation location, etc. The second kind of data is dynamic that is
 340 used and updated in the normal day-to-day running of ITMI, for example, user's
 341 login/logout information, machine performance, project status, etc.
- 342 • Public Database: at ITMI's cloud server to store data collected from IoT sensors and
 343 devices.

344 4.3.2. Physical Resources

- 345 • Labs: each lab has some machine(s)/device(s) that can be operated/used based on
 346 some tasks related to some running projects.
- 347 • Machines/device: include industrial machinery, hydraulic and pneumatic, proto-
 348 typing and 3D printing, and drones.

349 For the case studies in sections 5 and 6, our concern is local and public databases, labs,
350 and machines/devices. Note that, in the proposed case studies we consider that all the
351 identified users are able to log into the database(s) using their user names and passwords,
352 and the operations they are able to perform on some entities/attributes are specified in
353 the AC rules.

354 4.4. Access Control Requirements at ITMI

355 To better understand what kind of AC method(s) is needed for ITMI, in this section
356 we summarize its specific AC requirements illustrated by two case studies. The first
357 case study explains how resources are accessed via ITMI's local network (non-IoT), and
358 the second case study explains how resources are accessed via IoT. Also, we describe
359 the existing challenges in both situations. An AC mechanism must satisfy all users'
360 needs to fulfill specific access over logical/physical resources to avoid, for example, any
361 malfunction while using some device/machine which could lead to hazardous results
362 and financial loss. For example:

- 363 • 3D printing: any misbehavior while using 3D printers could cause several mal-
364 functions which lead to financial loss. Also, they must be controlled by specialists,
365 since during the printing process they emit toxic particles that may be harmful to
366 humans.
- 367 • Hydraulic systems: in hydraulic systems, fluid is stored under high pressure,
368 misadjusting any of its components without releasing the pressure could lead to
369 several hazards such as burns from hot fluid, bruises or cuts, and others. To avoid
370 such accidents, it is important to prevent non-specialist users from accessing these
371 machines.

372 The following are the privacy and security requirements that are identified as essential
373 to the ITMI environment:

- 374 1. The task of managing AC policies should not be complicated to preserve trust in
375 the system and ensure the system's performance.
- 376 2. Each department should have the autonomy to design its own security policy. To
377 enforce it, it must be revised by the manager and then confirmed and activated by
378 the system administrator at the IT department.
- 379 3. The manager should be able to hide specific fields of information contained in some
380 projects from some users who are associated with a specific role(s).
- 381 4. It is important that some authorized uses of data are not blocked after working
382 hours, for example, they need access to some devices to monitor a machine's
383 performance, temperature, etc., and need-to-know data access requirements in
384 accidents or emergencies.

385 Ensuring the privacy of users and the security of resources is essential for ITMI
386 since any exposure or intrusion to logical or physical resources could result in serious
387 consequences such as financial loss, and loss of reputation. Considering that the main
388 role of ITMI is to follow up, handle, and find solutions related to projects for some insti-
389 tutions, any intervention creates the possibility of ITMI's entire data being compromised
390 by a single action. Accordingly, ITMI needs an AC method that is able to answer all
391 the needed requirements based on different situations and conditions. Knowing that
392 the needed AC method is assumed always to be updated due to the nature of the given
393 projects which are related to recent and different technologies, this might impose new
394 AC requirements and/or modify some of the already defined rules. Moreover, ITMI is
395 enhancing the method of controlling access to resources to effectively manage building
396 services such as managing projects, maintenance requests, and others. In the following
397 sections, we present non-IoT and IoT case studies to show how HEAD metamodel
398 can be implemented to derive the needed AC models that fit their AC requirements,
399 generate the needed AC policies and enforce them. Figure 4 summarizes the needed
400 implementation phases to handle each case study and achieve the needed results. The

401 first phase is investigating the case study and identifying the informal AC policy, the
 402 second phase is specifying and deriving the formal policy model and generating the AC
 403 policy using the DSL of the HEAD metamodel, and the third phase is using the NGAC
 404 framework as an enforcement point for the generated rules.

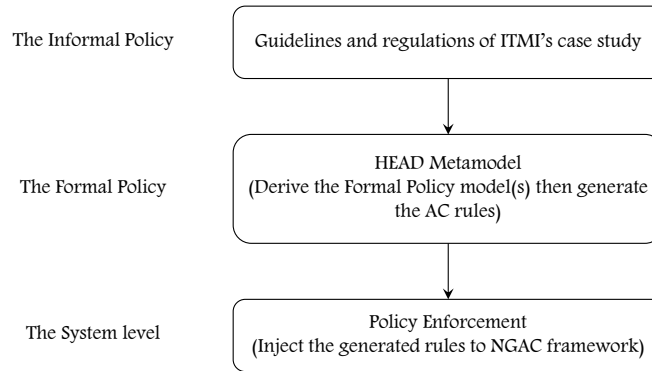


Figure 4. The implementation phases

405 5. Case Study 1— ITMI: non-IoT

406 ITMI shares a working environment of about 35 workers with different expertise
 407 related to mechanics, artificial intelligence (AI), informatics, machine learning, and other
 408 expertise co-working across various projects. Figure 5 illustrates the system architecture
 409 of ITMI's (non-IoT) local computing environment. All users need to be authenticated
 410 by identifying their identities using passwords to access the database, fingerprints to
 411 access the labs, and a PIN code to operate the machines/devices. Authenticated users
 412 are authorized, after checking the defined AC rules in the policy database, to access
 resources and perform operations according to their roles, and groups.

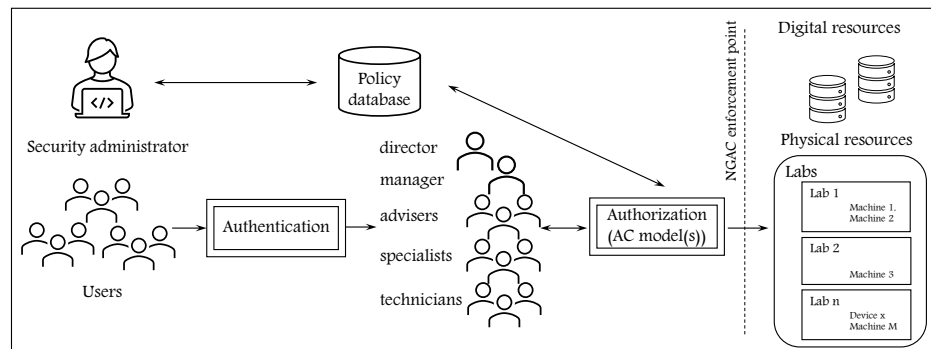
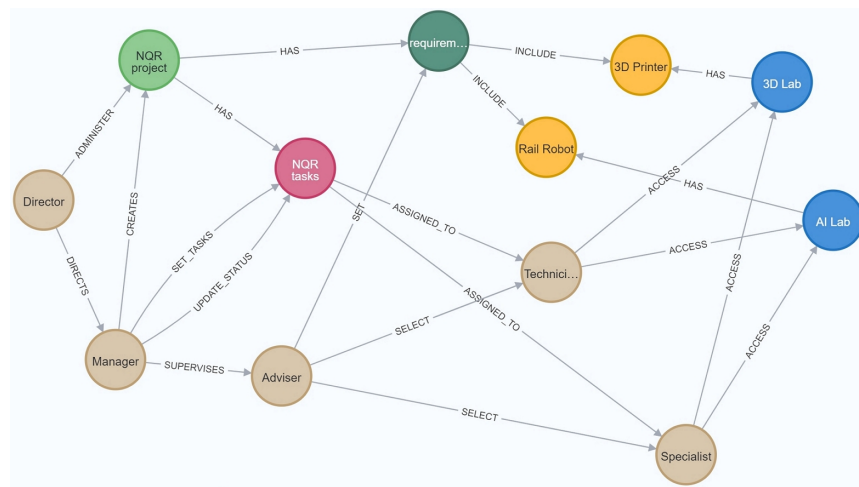


Figure 5. The system architecture of ITMI: non-IoT environment

413 To easily understand the connections between users and resources, we use Neo4j to
 414 express the relatedness among them in the form of a graph. In Figure 6, we illustrate
 415 the flow of information for the case study scenario. A rail robot needs to be implemented for
 416 a North Quebec Rail (NQR) project which is administered by the director of ITMI who
 417 manages the priority of project tasks and directs the manager in setting up the needed
 418 methodologies to implement them. The manager, who supervises advisers, creates the
 419 project record in the database and set the project's tasks based on the given guidelines
 420 from the director. Likewise, he updates the status of each project task (in progress,
 421 completed, on hold ...) depending on the project's performance. Advisers select the ap-
 422 propriate specialists and technicians based on their expertise and assign the needed tasks
 423 to them, also they select the project requirements of machines and devices. Likewise,
 424 they provide support with customized solutions to the existing obstacles and problems
 425 of the running project. Specialists are researchers with one or more specializations in
 426

427 a domain, they work with technicians to fulfill the project tasks. Technicians perform
 428 technical tasks such as maintenance, installation of machines/devices, etc. However, for
 429 the NQR project, the specialists and the technicians who are selected by the adviser, have
 430 access to the 'AI lab' where the 'Rail Robot' machine is located, and the '3D lab' where
 431 the '3D printing' device exists. To handle some of the project tasks, the 'Rail Robot' is
 432 accessed by a group of specialists and technicians to fulfill testing and assessing tasks
 433 and then record the needed results into the database. Another group of specialists and
 434 technicians is responsible for analyzing the obtained results to check if any flaw exists
 435 which might affect the rail robot's performance. If so, the third group of specialists and
 436 technicians has to redesign and implement another prototype for the flawed part of the
 robot, this prototype is produced using a '3D printer'.



437 **Figure 6.** A graph model representing the information flow of ITMI's non-IoT environment

438 5.1. The challenge

439 ITMI frequently has new projects with unexpected AC requirements. In this context,
 440 ITMI needs to ensure that the right access to data, device(s), and machine(s) is provided
 441 to the right user(s) with a focus on improving efficiency and collaboration, especially that
 442 users who are assigned to the same role might have different tasks since they might be
 443 also assigned to different groups. Thus, users' permissions need to be specified based on
 444 user-role, and user-group assignments. The solution has to be scalable to accommodate
 445 ITMI's future growth.

446 5.2. The Informal Policy: the access rights

447 AT ITMI's local computing environment, AC is based on the role, group, and other
 448 attributes of the users. Note that, each department has its labs which can be accessed by
 449 workers (users) of the same department or workers from other departments. Workers
 450 are allowed to access some resources to fulfill project tasks within the start and end of
 451 project dates. Based on the projects' requirements, some users assigned to a certain role
 452 may also be assigned to different groups and share some common tasks. Particularly,
 453 access rights grant access to users with a certain role/group to some resources and allow
 454 them to perform some actions on the database (read, write, update ...), and on machines
 455 (switching on/off, monitoring, modifying settings ...). An access decision could be
 456 allowed or denied based on the defined policy. In the following we summarize the access
 457 rights associated with the different roles and groups in this case study:

- 458 • The director 'Roy' has full access to databases and labs. Note that, the project details
 459 and tasks which are specified by the manager and the advisers must be confirmed
 460 by him (through the column 'confirm' of table projects) before running it.

- 461 • The manager ‘Thomas’ has full access to tables Projects (except the column ‘con-
462 firm’), and Tasks. Also, he has full access to the labs.
- 463 • The advisers ‘John’ and ‘Sophia’ have full access to tables ‘GroupTasks’ and ‘Re-
464 quirements’ to specify the tasks of each group of specialists and the technicians,
465 and set the project requirements of machines/devices.
- 466 • The specialists ‘Bob’, ‘Cathy’, and ‘Marc’ can access ‘AI Lab’ and operate the ‘Rail
467 Robot’ machine to test its functionality, assess its performance, analyze the findings,
468 then write the obtained findings in table ‘Results’. Likewise, they can access ‘3D
469 Lab’ to design prototypes and print the needed objects using ‘3D printer’.
- 470 • The technicians ‘Peter’ and ‘Eva’ can access labs and machines/devices to support
471 specialists with their experimental tests, perform maintenance, install new prod-
472 ucts/machines/devices and test them to ensure that they are working correctly,
473 check the needed connections, and some other tasks.

474 Note that, not all specialists/technicians are allowed to perform the same tasks, they are
475 classified by an adviser into three groups to perform the needed tasks:

- 476 • GroupA: the specialists ‘Bob’ and ‘Cathy’ perform the tasks of testing the robot,
477 and assessing then writing the obtained results into table ‘Results’. The technician
478 ‘Peter’ supports them to troubleshoot the robot hardware/software issues.
- 479 • GroupB: the specialists ‘Bob’ and ‘Marc’ read and analyze the results to check if any
480 flaw(s) exists and might affect the rail robot’s performance. The technician ‘Eva’
481 operates the robot to match the analyzed results with the robot’s performance.
- 482 • GroupC: the specialists ‘Marc’ and ‘Cathy’ redesign and implement another pro-
483 totype for the flawed part(s) of the robot. The needed part(s) is created using the
484 3D printer. The technicians ‘Peter’ and ‘Eva’ support them to replace the flawed or
485 damaged part(s), then check the robot system to ensure its compatibility.

486 Advisers, specialists, and technicians are allowed to access the specified resources during
487 the working hours via ITMI’s private network, their access rights must be revoked once
488 the project is over.

489 5.3. The Solution: HEAD Metamodel

490 The main question here is: what is the AC model that best fits the above (informal)
491 AC requirements, to be formalized and generate the needed AC rules?

492 The DSL of HEAD metamodel allows deriving various instances of E_x , I_m (AUs and
493 PUs), and S_t entities. By considering the above informal AC rules, we can figure out the
494 following:

- 495 • $E_x = \{\text{subject (name, ...); object (title, ...)}\}$.
- 496 • I_m are AUs = $\{\text{role (type, ...); group (number, ...)}\}$, and PUs = $\{\text{permission (ptype, ...); action (type, ...)}\}$
- 497 • $S_t = \{\text{contextual-constraint (date, time, loginlocation ...); constraint (projectconfirm, task_status ...)}\}$.

500 5.3.1. The Formal Policy Model

501 In this section, we need to investigate the best AC model that fits our case study.
502 Due to the above derived E_x , I_m , and S_t entities, we have the following possibilities:

- 503 • RBAC model: the main entities of RBAC are subject, object, role, permission, and
504 action. However, although users are assigned to roles at ITMI, and this is an
505 essential feature to consider in order to implement an RBAC model, in several
506 situations the use of attributes and environmental conditions is mandatory. As
507 well, users are assigned to groups, and the group entity does not belong to the
508 RBAC model. Hence, the RBAC model is not enough to answer the needed security
509 requirements.
- 510 • ABAC model: ABAC entities are subject, object, action, and environmental (con-
511 text) attributes. Users at ITMI are assigned to roles and groups, hence with the

512 ABAC model, the notion of some other models should be considered (e.g., hybrid
513 RBAC/ABAC).
514 • Hybrid model: the notion of role reflects the importance of considering the RBAC
515 model; the notion of attributes and the need to express dynamic AC rules reflect
516 the importance of considering the ABAC model; also the demand to define a group
517 entity with its attributes imposes the need not to restrict the solution to RBAC or
518 ABAC models only. Note that, the solution should consider that maybe some other
519 entity(ies) in the future needs to be created/added due to some upgrades.

520 Hence, our solution, in this case, is to derive a hybrid model based on user-groups,
521 RBAC, and ABAC. Using the DSL of HEAD metamodel the formal user-groups, RBAC,
522 and ABAC hybrid model can be derived by first instantiating the needed entities of E_x ,
523 AU, PU, and S_t components, then expressing the rules as shown in Figure 7. In Figure
524 7(a) we define the following:

- 525 • line 1: the specification of policy model/class.
- 526 • lines 2 to 10: the block of creating E_x entities, starts and ends with the keywords
527 explicit and end.
 - 528 – line 3: the creation of subject entity with the attribute(s).
 - 529 – lines 4 to 9: four levels of object hierarchy are created to represent the hierarchy
530 of logical and physical resources. Note that, we actually have three levels of the
531 object hierarchy, but since we are using NGAC as an enforcement framework
532 we create an additional level to have four levels. ‘object1’ level is to express
533 the main objects which will be assigned to different object containers, and the
534 other three levels are defined to represent the hierarchy object containers.
- 535 • lines 12 to 21: the block of creating AU entities, starts and ends with the keywords
536 authorization and end.
 - 537 – line 13: the creation of group entity with the attribute(s).
 - 538 – line 14 to 20: four levels of role hierarchy are created. For example, role1 refers
539 to the director, role2 refers to the manager, etc.
- 540 • lines 23 to 26: the block of creating PU entities, starts and ends with the keywords
541 procedural and end.
 - 542 – line 24: the creation of permission entity with the needed attribute(s).
 - 543 – line 25: the creation of action entity with the attribute(s).
- 544 • lines 28 to 33: the block of creating S_t entities, starts and ends with the keywords
545 setting and end.
 - 546 – line 29 and 30: the creation of context entity with the context attribute(s).
 - 547 – line 31 and 32: the creation of constraint entity with the attribute(s).

548 Moreover, in Figure 7(b) we provide examples of three possible rule expressions:

- 549 • lines 39 to 47: this rule expression is, for example, to express the access rights
550 for users who are assigned to the first-level of role hierarchy (e.g., director) and
551 implicitly connected with the actions associated with their role (e.g., manager)
552 without the administrator having to explicitly list the manager actions on objects,
553 and this is also applied on the lower levels of hierarchy.
- 554 • lines 48 to 58: to express, for example, the rule for users who are assigned to the
555 second-level of role hierarchy and implicitly connected with the actions associated
556 with their role(s) and perform some actions on objects with some constraints.
- 557 • lines 59 to 73: to express, for example, the rule for users who are assigned to a
558 lower-level of role hierarchy, and also assigned to some group to perform some
559 actions (with some constraints) on some objects of a certain level of hierarchy in
560 some context with some contextual and non-contextual constraints.

561 The substantial advantage of the DSL of the HEAD metamodel is that whenever a system
562 administrator creates an attribute for any entity, this attribute can be flexibly included
563 with the policy rules with some other entity. For example, context and constraint entities

<pre> 1 policy Hybrid (industry:String) 2 explicit 3 subject (sname:String) 4 object1 (oaname:String)[5 object2 (oaname:String)[6 object3 (oaname:String)[7 object4(oaname:String)] 8] 9] 10 end 11 12 authorization 13 group(title:String) 14 roleL1(title:String)[15 roleL2(title:String)[16 roleL3(title:String)[17 roleL4(title:String) 18] 19] 20] 21 end 22 23 procedural 24 permission(perm:String) 25 action(act:String) 26 end 27 28 setting 29 context(time:String loginLocation:String 30 sDate:String eDate:String) 31 constraint(prjConfirm:String 32 task_status:String) 33 end 34 end 35 36 37 </pre>	<pre> v Metamodel v Policy Hybrid Attribute industry Explicit subject Attribute sname Explicit object1 Attribute oaname Explicit object2 Attribute oaname Explicit object3 Attribute oaname Explicit object4 Implicit Authorization Unit group Attribute title Authorization Unit roleL1 Attribute title Authorization Unit roleL2 Attribute title Authorization Unit roleL3 Procedural Unit permission Attribute perm Procedural Unit action Attribute act Setting context Attribute time Attribute loginLocation Attribute sDate Attribute eDate Setting constraint Decision </pre>
<pre> 38 rule: 39 (ruleid1:int) { 40 Hybrid.subject(Hybrid.subject.sname)[Hybrid.roleL1(Hybrid.roleL1.title)]{ 41 Hybrid.permission(Hybrid.permission.perm){ 42 Hybrid.object1(Hybrid.object1.oaname){ 43 Hybrid.action(Hybrid.action.act) 44 } 45 } 46 } 47 }-->decision 48 (ruleid2:int) { 49 Hybrid.subject(Hybrid.subject.sname)[Hybrid.roleL1.roleL2(Hybrid.roleL1.roleL2.title)]{ 50 Hybrid.permission(Hybrid.permission.perm){ 51 Hybrid.object1.object2(Hybrid.object1.object2.oaname){ 52 Hybrid.action(Hybrid.action.act){ 53 Hybrid.constraint(Hybrid.constraint.prjConfirm) 54 } 55 } 56 } 57 } 58 }-->decision 59 (ruleid3:int) { 60 Hybrid.subject(Hybrid.subject.sname) 61 [Hybrid.roleL1.roleL2.roleL3.roleL4 (Hybrid.roleL1.roleL2.roleL3.roleL4.title) 62 Hybrid.group(Hybrid.group.title) 63]{ 64 Hybrid.permission(Hybrid.permission.perm){ 65 Hybrid.object1.object2.object3(Hybrid.object1.object2.object3.oaname){ 66 Hybrid.action(Hybrid.action.act){ 67 Hybrid.constraint(Hybrid.constraint.prjConfirm 68 Hybrid.context.loginLocation Hybrid.context.time) 69 } 70 } 71 } 72 } 73 }-->decision </pre>	<pre> (a) </pre>
<pre> (b) </pre>	<pre> (b) </pre>

Figure 7. Case Study 1: HEAD metamodel instance: (a) A hybrid model based on user-groups, RBAC and ABAC entities/attributes; (b) rule expressions

564 (with their attributes) are created in lines 29 and 32, and in lines 67 and 68 of rule
565 expression, the attributes of the context entity are included with the constraint entity.

566 5.3.2. The Formal Generated Policy

The meta-policy is expressed in terms of the meta-components E_x , I_m , and S_f [1,9]:

$$\text{Metapolicy} = \langle E_x, AU, PU, S_f \rangle$$

567 Before expressing the formal policy, we need to describe the policy elements:

- 568 • Users: a set of entities who have accounts in the ITMI's IS. Users = {Roy, Thomas,
569 John, Sophia, Marc, Bob, Cathy, Peter, Eva}
- 570 • User Attributes: each user is associated with a set of attributes (e.g., Id, name ...).
- 571 • Roles: specifies the user's role. Role = {director, manager, adviser, specialist, techni-
572 cian}.
- 573 • Groups: specifies user's group. Group = {groupA, groupB, groupC}.
- 574 • Objects: a set of logical and physical objects that need protection. For example,
575 database entities (projects, appliances ...), labs, machines, devices, etc.
- 576 • Object attributes: object attributes include the project name, machine#, etc.
- 577 • Actions: a set of actions on logical objects in the database (e.g., read, write, update,
578 delete, etc.), and on physical objects (e.g., operate machines, test, troubleshoot, etc.).
- 579 • Permissions: users are given permission(s) to perform actions on objects based on
580 their roles (or groups), and if constraints and contextual constraints are true.
- 581 • Context: for a given context—for example, log in via private/public network,
582 machine malfunction, system failure, etc.—context attributes include datetime,
583 login location, failed password attempts, etc., contextual constraint expressions
584 include context attributes which are important for authorization decisions.
- 585 • Constraints: expressions include the other various types of attributes, for example,
586 subject attributes, object attributes, etc.

Based on the meta-policy expression, we express the policy of this case study as follows:

$$\text{Policy} = \langle \text{subject}(\text{name}, \dots); \text{object}(\text{title}, \dots); \text{role}(\text{type}, \dots); \text{group}(\text{number}, \dots); \\ \text{permission}(\text{ptype}, \dots); \text{action}(\text{type}, \dots); \text{context}(\text{date}, \dots); \\ \text{constraints}(\text{roletype}, \text{title}, \dots) \rangle$$

587 Since our aim is to use NGAC as an enforcement point, we need Cypher queries as
588 java output which then will be injected into the Neo4j database to represent the graph
589 of NGAC policy. To represent the concrete instance of user-groups, RBAC and ABAC
590 model, xtend notation is used to generate the needed java code at the system level. In
591 Figure 8, we show a sample of xtend code to generate the E_x entities.

- 592 • Lines 3 to 16 of Figure 8 (a) generate the non-hierarchical entities (e.g., subject).
- 593 • Lines 18 to 31 to generate the root entities of the hierarchical components (e.g., the
594 root entities of objects).
- 595 • To create users (U) and objects (O) nodes, the needed Cypher code is expressed and
596 added to the array list 'rules' as shown in lines 13 and 28 of Figure 8 (a).
- 597 • Since in our case study we have object hierarchy, of E_x , 'explicitroot' and 'ex-
598 plicitrarchy' (lines 1 and 2) of Figure 8 (b) represent objects at the root level and their
599 hierarchies. Note that, for NGAC policy we define the code to generate hierarchical
600 attribute containers. If object or object attributes/containers at the root level (lines 4
601 to 8), or any lower level (lines 9 to 20), 'include' or 'assigned_to' other containers
602 we create the needed attribute containers (lines 32 to 61) and express the Cypher
603 code to represent Os and OAs of NGAC policy (lines 37 to 48).

604 In addition to object hierarchy, we also have hierarchy of roles and the same is applied
605 to create the I_m entities/attributes. The sample code of xtend notation (in Figure 8)
606 generates the sample java code in Figure 9 for the subject and object entities. However:

```

1 /*-----Formation of AC Rules-----*/
2 //Explicit Entities
3 «FOR j : 0 ..< explicitlist.size»
4
5 System.out.println("explicitlist.get(j).name.toUpperCase(s)-----");
6 «FOR a : explicitlist.get(j).attributes»
7 System.out.print("\texplicitlist.get(j).name.«a.name»: ");
8 temp = sc.nextLine().split(" ");
9
10 for (int i=0; i < temp.length; i++) {
11     explicitlist.get(j).name.toString.toLowerCase().set«a.name.toFirstUpper»(temp[i]);
12     e_«explicitlist.get(j).name.toString.toLowerCase».add("explicitlist.get(j).name.toString.toLowerCase.«a.name.toFirstUpper»());
13     rules.add("CREATE (:O {«a.name»: '"+temp[i]+'"}");
14 }
15 «ENDFOR»
16 «ENDFOR»
17
18 «FOR j : 0 ..< explicitroot.size»
19
20 System.out.println("explicitroot.get(j).name.toUpperCase(s)-----");
21 «FOR a : explicitroot.get(j).attributes»
22 System.out.print("\texplicitroot.get(j).name.«a.name»: ");
23 temp = sc.nextLine().split(" ");
24
25 for (int i=0; i < temp.length; i++) {
26     explicitroot.get(j).name.toString.toLowerCase().set«a.name.toFirstUpper»(temp[i]);
27     r_«explicitroot.get(j).name.toString.toLowerCase».add("explicitroot.get(j).name.toString.toLowerCase.«a.name.toFirstUpper»());
28     rules.add("CREATE (:O {«a.name»: '"+temp[i]+'"}");
29 }
30 «ENDFOR»
31 «ENDFOR»

```

(a)

```

1 «FOR j : 0 ..< explicitroot.size»
2 «FOR hr : 0 ..< explicitrchy.size»
3
4 «IF hr == 0»
5 for (int i=0; i < r_«explicitroot.get(j).name.toString.toLowerCase».size(); i++) {
6 System.out.print("Does "+ r_«explicitroot.get(j).name.toString.toLowerCase».get(i)+" has attribute container(s)? (y/n) ");
7 exp_hrchy.add(r_«explicitroot.get(j).name.toString.toLowerCase».get(i));
8 answer = sc.nextLine();
9 «ELSEIF hr >= 1»
10 for (int i=0; i < h_«explicitrchy.get(hr-1).name.toString.toLowerCase».size(); i++) {
11     if(h_«explicitrchy.get(hr-1).name.toString.toLowerCase».get(i) == null){
12         continue;
13     }else{
14         System.out.print("Does "+ h_«explicitrchy.get(hr-1).name.toString.toLowerCase».get(i)+" has attribute container(s)? (y/n) ");
15     }
16     exp_hrchy.add(h_«explicitrchy.get(hr-1).name.toString.toLowerCase».get(i));
17 }
18 «ENDFOR»
19 exp_hrchy.add(h_«explicitrchy.get(hr-1).name.toString.toLowerCase».get(i));
20 answer = sc.nextLine();
21 }
22 «ENDIF»
23 if(answer.equals("Y") || answer.equals("y")){
24     exp_hrchy.add(">");
25 System.out.print("1-include_hierarchical or 2-assigned_to attribute container(s)? (1/2) ");
26 answer1 = sc.nextLine();
27 if(answer1.equals("1")){relation="include"; } else {relation="assigned_to"; }
28 «IF hr == 0»
29 System.out.print(" "+ relation + " Container(s) "+ r_«explicitroot.get(j).name.toString.toLowerCase».get(i)+" ": ";
30 «ELSEIF hr >= 1»
31 System.out.print(" "+ relation + " Container(s) "+ h_«explicitrchy.get(hr-1).name.toString.toLowerCase».get(i)+" ": ";
32 «ENDIF»
33 «FOR a : explicitrchy.get(j).attributes»
34 temp = sc.nextLine().split(" ");
35 for (j=0; j < temp.length; j++) {
36     explicitrchy.get(hr).name.toString.toLowerCase().set«a.name.toFirstUpper»(temp[j]);
37     h_«explicitrchy.get(hr).name.toString.toLowerCase».add("explicitrchy.get(hr).name.toString.toLowerCase.«a.name.toFirstUpper»());
38     exp_hrchy.add(temp[j]+" ");
39     rules.add("MERGE (:OA {«a.name»: '"+temp[j]+'"}");
40 }
41 «IF hr == 0»
42 strp="MATCH (er«hr»"+i+j+":O {«FOR at : explicitroot.get(j).attributes»at.name: '"+ENDFOR»
43 strpc strp + " MATCH (eh«hr»"+i+j+":OA {«a.name»: '"+temp[j]+'"}" + " MERGE (er«hr»"+i+j+":-[:'+relation']->(eh«hr»"+i+j+":)";
44 «ELSEIF hr >= 1»
45 strp="MATCH (er«hr»"+i+j+":DA {«FOR at : explicitrchy.get(hr-1).attributes»at.name: '"+ENDFOR»
46 strpc strp + " MATCH (eh«hr»"+i+j+":OA {«a.name»: '"+temp[j]+'"}" + " MERGE (er«hr»"+i+j+":-[:'+relation']->(eh«hr»"+i+j+":)";
47 «ENDIF»
48 rules.add(strc);
49 }
50 exp_hrchy.add("\n");
51 explicitrchy.get(hr).name.toString.toLowerCase().set«a.name.toFirstUpper»(null);
52 h_«explicitrchy.get(hr).name.toString.toLowerCase».add("explicitrchy.get(hr).name.toString.toLowerCase.«a.name.toFirstUpper»());
53 }
54 }
55 else {
56     exp_hrchy.add(" Nothing\n");
57     explicitrchy.get(hr).name.toString.toLowerCase().set«a.name.toFirstUpper»(null);
58     h_«explicitrchy.get(hr).name.toString.toLowerCase».add("explicitrchy.get(hr).name.toString.toLowerCase.«a.name.toFirstUpper»());
59     continue;
60 }
61 //end (if answer)
62 //end (for i)
63 «ENDFOR»
64 «ENDFOR»
65 «ENDFOR»

```

(b)

Figure 8. Case Study 1: A sample of the Xtend notation to generate the java code for the Explicit entities and their hierarchies.

```

1  /*-----Formation of AC Rules-----*/
2  //Explicit Entities
3  System.out.println("SUBJECT(s)-----");
4  System.out.print("\tsubject.name: ");
5  temp = sc.nextLine().split(" ");
6  for (int i=0; i < temp.length; i++) {
7      subject.setName(temp[i]);
8      e_subject.add(subject.getName());
9      rules.add("CREATE (:U {name:'"+temp[i]+"}");
10 }
11 System.out.println("OBJECT1(s)-----");
12 System.out.print("\tobject1.label: ");
13 temp = sc.nextLine().split(" ");
14
15 for (int i=0; i < temp.length; i++) {
16     object1.setLabel(temp[i]);
17     r_object1.add(object1.getLabel());
18     rules.add("CREATE (:O {label:'"+temp[i]+"}");
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33 for (int i=0; i < r_object1.size(); i++) {
34     System.out.print("Does "+ r_object1.get(i)+" has attribute container(s)? (y/n) ");
35     exp_hrchy.add(r_object1.get(i));
36     answer = sc.nextLine();
37     if(answer.equals("Y") || answer.equals("y")){
38         exp_hrchy.add(">");
39         System.out.print("1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) ");
40         answer1 = sc.nextLine();
41         if(answer1.equals("1")){relation="include"; } else {relation="assigned_to"; }
42         System.out.print(r_object1.get(i) + " " + relation + " Container(s): ");
43         temp = sc.nextLine().split(" ");
44         for (j=0; j < temp.length; j++) {
45             object2.setName(temp[j]);
46             h_object2.add(object2.getName());
47             exp_hrchy.add(temp[j]+" ");
48             rules.add("MERGE (:OA {oname:'"+temp[j]+"}");
49             strp="MATCH (er0'+i+j'+:O {oname:'"+
50                 r_object1.get(i)+"}");
51             strc= strp + " MATCH (eh0'+i+j'+:OA {oname:'"+temp[j]+"}') + " MERGE (er0'+i+j+)-[:'+relation+']->(eh0'+i+j+);";
52             rules.add(strc);
53         }
54         exp_hrchy.add("\n");
55         object2.setName(null);
56         h_object2.add(object2.getName());
57     }
58     else{
59         exp_hrchy.add(": Nothing\n");
60         object2.setName(null);
61         h_object2.add(object2.getName());
62         continue;
63     }
64     //end (if answer)
65     //end (for i)
66 }
67 for (int i=0; i < h_object2.size(); i++) {
68     if(h_object2.get(i) == null){
69         continue;
70     }
71     else{
72         System.out.print("Does "+ h_object2.get(i)+" has attribute container(s)? (y/n) ");
73         exp_hrchy.add("\t");
74         exp_hrchy.add(h_object2.get(i));
75         answer = sc.nextLine();
76         if(answer.equals("Y") || answer.equals("y")){
77             exp_hrchy.add(">");
78             System.out.print("1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) ");
79             answer1 = sc.nextLine();
80             if(answer1.equals("1")){relation="include"; } else {relation="assigned_to"; }
81             System.out.print(h_object2.get(i) + " " + relation + " Container(s): ");
82             temp = sc.nextLine().split(" ");
83             for (j=0; j < temp.length; j++) {
84                 object3.setName(temp[j]);
85                 h_object3.add(object3.getName());
86                 exp_hrchy.add(temp[j]+" ");
87                 rules.add("MERGE (:OA {oname:'"+temp[j]+"}");
88                 strp="MATCH (er1'+i+j'+:OA {oname:'"+
89                     h_object2.get(i)+"}");
90                 strc= strp + " MATCH (eh1'+i+j'+:OA {oname:'"+temp[j]+"}') + " MERGE (er1'+i+j+)-[:'+relation+']->(eh1'+i+j+);";
91                 rules.add(strc);
92             }
93             exp_hrchy.add("\n");
94             object3.setName(null);
95             h_object3.add(object3.getName());
96         }
97         else {
98             exp_hrchy.add(": Nothing\n");
99             object3.setName(null);
100            h_object3.add(object3.getName());
101            continue;
102        }
103        //end (if answer)
104        //end (for i)

```

Figure 9. Case Study 1: A sample of the generated java code for the subject and object entities (with object hierarchy) of group-based, RBAC and ABAC model.

- 607 • the xtend sample code of Figure 8 (a) generates the sample java code of Figure 9
608 (a). Lines 3-10 define subject entities/attributes and lines 12-20 create the objects (at
609 object1 level). Lines 9 and 18 express the Us and Os nodes of the Cypher code.
- 610 • the xtend sample code of Figure 8 (b) generates the sample java code of Figure 9 (b).
611 Lines 2 to 32 create the object containers (at object2 level) where objects (at object1
612 level) are 'assigned_to'/'include' them. As well, lines 33 to 67 create the object con-
613 tainers (at object3 level) where objects (at object2 level) are 'assigned_to'/'include'
614 them. Hence, we would have some object containers and some hierarchical object
615 containers with U/O-UA/OA, and UA/OA-UA/OA assignments to represent the
616 NGAC policy. Lines 16-20 and lines 53-57 express the needed Cypher expressions.

617 Before presenting the NGAC graph we have to explain the NGAC policy configuration
618 where users and objects may be included in one or more containers, and containers may
619 be included by or include other containers. However:

- 620 • in Figure 10 (a), we show user containers that represent the assignment of Us
621 (Roy, Thomas ...) to UAs (Director, Manager, GroupA ...). An example of a user
622 container named Adviser includes two users 'John' and 'Sophia'.
- 623 • In Figure 10 (b), we show object containers with the representation of relational
624 database tables with some distinguished rows/columns where they are represented
625 as containers of data objects corresponding to the row/column fields. For example,
626 a container named ProjectDetails includes the fields prjName, startDate, endDate,
627 and prjConfirmation.
- 628 • In Figure 10 (c), users' access rights to perform operations are formulated through
629 associations. For example, in (1) and (2) the director is allowed to {r:read, w:write/
630 insert, u:update, d:delete} the data in FinancialDetails container, also he is allowed
631 to {d: delete, c: confirm} data of ProjectDetails container; in (3) the manager is
632 allowed to {r, w, u} data of ProjectDetails; etc. (the operations 's' and 'o' of 6 and
633 9 stand for select and operate). However, Figure 10 (c) lists thirteen association
634 relations in terms of the user and object attributes/containers illustrated in Figure
635 10 (a) and Figure 10 (b).
- 636 • In Figure 10 (d), we have two prohibition relations which express user attribute-
637 deny (ua_deny) to deny the technicians 'Peter' and 'Eva' of performing {w, u, d} on
638 GrpATskRslt/GrpCTskRslt and GrpBTskRslt/GrpCTskRslt respectively.
- 639 • In Figure 10 (e), we represent six obligations that are defined as event-response
640 relations to define constraints under which policy state data is obligated to change.
641 For example, in (1) when the director confirms the ProjectDetails information,
642 the manager wouldn't be able to {u, d} this information; another example in (3)
643 expresses an obligation due to some context where the adviser is not allowed to {u,
644 d} information in Requirements container if he logged into the system via a public
645 network or if the current date is greater than the end date of NQR project; also in (4,
646 5, and 6) users of Groups A, B, and C are not allowed to {w, u, d} information of
647 containers GrpATskRslt, GrpBTskRslt, and GrpCTskRslt during the non-business
648 hours and before and after the start and end of project dates.

649 In Figure 11 we show a sample output of the java code (in Figure 9) which expresses
650 in part (a) a sample of subject and object entities/attributes which are configured by a
651 system administrator with setting up the relationships between object containers. In
652 (b), the Cypher code to create policy class (PC), U, and O nodes based on the system
653 entities/attributes which are configured in part (a). In part (c), some of the generated
654 Cypher code which represents O-OA and OA-OA assignments and hierarchies. In (d), a
655 sample of Cypher code that expresses the association relationships to define the access
656 rights associated with some UAs (e.g., Director, GroupB ...) to perform operations on
657 some objects specified by OAs (e.g., FinancialDetails, GrpBTskRslt ...). For example, in
658 the following Cypher expression:

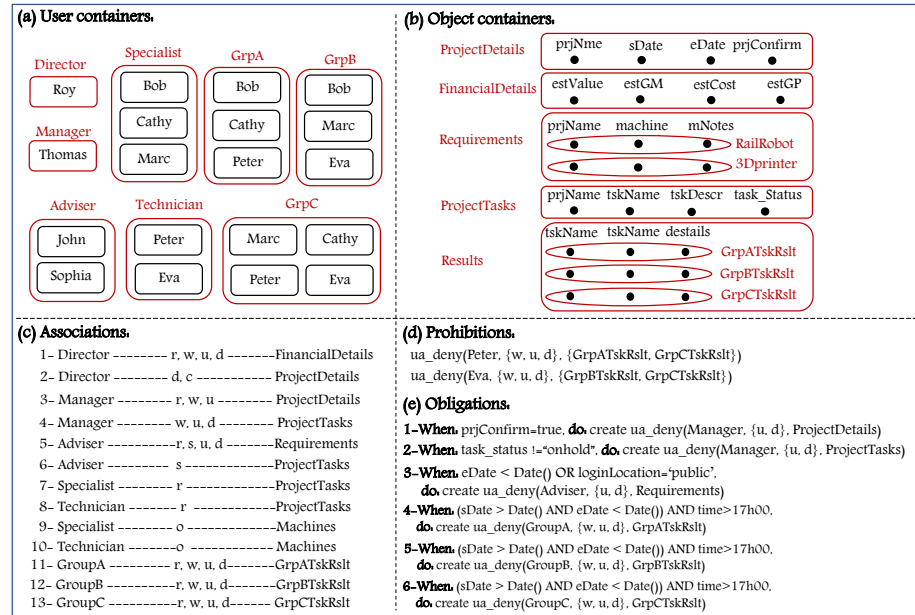


Figure 10. Case Study 1: NGAC Policy Configuration.

```

559 MATCH (auh00:UA{title:'Manager'}) MATCH (eh00:OA {oaname:'ProjectDetails'})
560 MERGE(auh00)-[:Permission {perm:'ManPermission', act:'r,w,u', prjConfirm:'false'}]-> (eh00);

```

561 The manager with ManPermission is able to perform {r, w, u} operations on ProjectDetails
562 as long as the information is not confirmed by the director, in other words, as long as the
563 value of prjConfirm = False. Since the basic elements of NGAC are U/O, UA/OA, and
564 assignment/association relationships, we configure the constraints and the contextual
565 constraints as properties in the association relationship. Note that, to simplify and
566 minimize the number of the association relationships between some UAs and OAs in
567 the graph of Figure 12, we use a single association relationship instead of two or three.
568 For example, in the first association relationship which represents ManPermission the
569 operation {r} on ProjectDetails can unconditionally be performed by the manager, and
570 for the second one, the manager cannot perform {u,d} if the PrjConfirm value is true. In
571 our illustration, we use a single association relationship and apply the constraints on all
572 operations.

5.4. NGAC: the policy enforcement point

574 This section presents the AC policy enforcement over Cypher queries issued from
575 a system using Cypher statements as NGAC inputs and having NGAC authorization
576 responses based on them. In this paper, the system represents the generated java code
577 (Figure 9) where Cypher queries are issued after the administrative configuration of
578 AC policy. However, the generated Cypher code is injected into the Neo4j database to
579 represent the AC policy as NGAC graph policy. As shown in Figure 12, in the left section
580 the nodes indicated in light and dark green color represent the assignment of users
581 (Us) to their roles/groups (UAs), in addition to the role hierarchy which is indicated
582 by the red arrows and 'has_child_content' relationship. The nodes indicated in light
583 and dark blue color in the right section of the graph represent the assignment of objects
584 (Os) to object containers (OAs) (and some OAs to other OAs), in addition to object
585 hierarchy where the hierarchy of object containers is indicated by the red arrows and
586 'include' relationship. The association relationships, the yellow arrows, represent users'
587 permission based on their roles/groups.

588 By assigning a user to a role (with the inheritance relation between roles), the user is
589 indirectly associated with the access rights of that role's lower roles. Figure 13 illustrates

```

Hybrid Policy for industry:ITMI
SUBJECT(s)-----
  subject.sname: Roy Thomas John Sophia Bob Cathy Marc Peter Eva
OBJECT1(s)-----
  object1.oname: nqrName nqrDetails nqrDuration nqrTasks Labs
Does nqrName has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 2
nqrName 'assigned_to' Container(s): FinancialDetails ProjectDetails
Does nqrDetails has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 2
nqrDetails 'assigned_to' Container(s): FinancialDetails ProjectDetails
Does nqrDuration has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 2
nqrDuration 'assigned_to' Container(s): ProjectDetails
Does nqrTasks has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 2
nqrTasks 'assigned_to' Container(s): ProjectTasks
Does Labs has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 1
Labs 'include' Container(s): Machines

```

(a)

```

CREATE (:PC {industry:'ITMI'})
CREATE (:U {sname:'Roy'})
CREATE (:U {sname:'Thomas'})
CREATE (:U {sname:'John'})
CREATE (:U {sname:'Sophia'})
CREATE (:U {sname:'Bob'})
CREATE (:U {sname:'Cathy'})
CREATE (:U {sname:'Marc'})
CREATE (:U {sname:'Peter'})
CREATE (:U {sname:'Eva'})
CREATE (:O {oname:'nqrName'})
CREATE (:O {oname:'nqrDetails'})
CREATE (:O {oname:'nqrDuration'})
CREATE (:O {oname:'nqrTasks'})
CREATE (:O {oname:'Labs'})

```

(b)

```

MERGE (:OA {oname:'FinancialDetails'});
MATCH (er000:O {oname:'nqrName'}) MATCH (eh000:OA {oname:'FinancialDetails'}) MERGE (er000)-[:assigned_to]->(eh000);
MERGE (:OA {oname:'ProjectDetails'});
MATCH (er001:O {oname:'nqrName'}) MATCH (eh001:OA {oname:'ProjectDetails'}) MERGE (er001)-[:assigned_to]->(eh001);
MERGE (:OA {oname:'FinancialDetails'});
MATCH (er010:O {oname:'nqrDetails'}) MATCH (eh010:OA {oname:'FinancialDetails'}) MERGE (er010)-[:assigned_to]->(eh010);
MERGE (:OA {oname:'ProjectDetails'});
MATCH (er011:O {oname:'nqrDetails'}) MATCH (eh011:OA {oname:'ProjectDetails'}) MERGE (er011)-[:assigned_to]->(eh011);
MERGE (:OA {oname:'ProjectDetails'});
MATCH (er020:O {oname:'nqrDuration'}) MATCH (eh020:OA {oname:'ProjectDetails'}) MERGE (er020)-[:assigned_to]->(eh020);
MERGE (:OA {oname:'ProjectTasks'});
MATCH (er030:O {oname:'nqrTasks'}) MATCH (eh030:OA {oname:'ProjectTasks'}) MERGE (er030)-[:assigned_to]->(eh030);
MERGE (:OA {oname:'Machines'});
MATCH (er040:O {oname:'Labs'}) MATCH (eh040:OA {oname:'Machines'}) MERGE (er040)-[:include]->(eh040);
MERGE (:OA {oname:'Requirements'});
MATCH (er110:OA {oname:'ProjectDetails'}) MATCH (eh110:OA {oname:'Requirements'}) MERGE (er110)-[:include]->(eh110);
MERGE (:OA {oname:'ProjectTasks'});
MATCH (er111:OA {oname:'ProjectDetails'}) MATCH (eh111:OA {oname:'ProjectTasks'}) MERGE (er111)-[:include]->(eh111);
MERGE (:OA {oname:'GrpATskRslt'});
MATCH (er180:OA {oname:'ProjectTasks'}) MATCH (eh180:OA {oname:'GrpATskRslt'}) MERGE (er180)-[:include]->(eh180);
MERGE (:OA {oname:'GrpBTskRslt'});
MATCH (er181:OA {oname:'ProjectTasks'}) MATCH (eh181:OA {oname:'GrpBTskRslt'}) MERGE (er181)-[:include]->(eh181);
MERGE (:OA {oname:'GrpCTskRslt'});
MATCH (er182:OA {oname:'ProjectTasks'}) MATCH (eh182:OA {oname:'GrpCTskRslt'}) MERGE (er182)-[:include]->(eh182);
MERGE (:OA {oname:'RailRobot'});
MATCH (er1100:OA {oname:'Machines'}) MATCH (eh1100:OA {oname:'RailRobot'}) MERGE (er1100)-[:include]->(eh1100);
MERGE (:OA {oname:'3D_Printer'});
MATCH (er1101:OA {oname:'Machines'}) MATCH (eh1101:OA {oname:'3D_Printer'}) MERGE (er1101)-[:include]->(eh1101);
MERGE (:OA {oname:'GrpATskRslt'});
MATCH (er2110:OA {oname:'RailRobot'}) MATCH (eh2110:OA {oname:'GrpATskRslt'}) MERGE (er2110)-[:assigned_to]->(eh2110);
MERGE (:OA {oname:'GrpCTskRslt'});
MATCH (er2120:OA {oname:'3D_Printer'}) MATCH (eh2120:OA {oname:'GrpCTskRslt'}) MERGE (er2120)-[:assigned_to]->(eh2120);

```

(c)

```

MATCH (aur00:UA {title:'Director'}) MATCH (er00:OA {oname:'FinancialDetails'})
MERGE (aur00)-[:Permission{perm:'DirPermission', act:'r,w,u,d'}]->(er00);
MATCH (auh00:UA {title:'Manager'}) MATCH (eh00:OA {oname:'ProjectDetails'})
MERGE (auh00)-[:Permission{perm:'ManPermission', act:'r,w,u', prjConfirm:'false'}]->(eh00);
MATCH (auh00:UA {title:'Adviser'}) MATCH (eh00:OA {oname:'Requirements'}) MATCH (eh01:OA {oname:'ProjectTasks'})
MERGE (auh00)-[:Permission{perm:'AdvPermission', act:'r,s,u,d', eDate:'08-08-2022', loginLocation:'local'}]->(eh00)
MERGE (auh00)-[:Permission{perm:'AdvPermission', act:'r'}]->(eh01);
MATCH (aur00:UA {title:'GroupA'}) MATCH (er00:OA {oname:'GrpATskRslt'})
MERGE (aur00)-[:Permission{perm:'grpAPermission', act:'r,w,u,d', sDate:'08-01-2022',
eDate:'08-08-2022',time1:'8', time2:'17'}]->(er00);

```

(d)

Figure 11. Case Study 1: A sample java code output (a) to configure system values; (b) generated Cypher code to create PC, U, and O nodes; (c) O-OA assignments and hierarchies to express AC rules; and (d) a sample of access rights associated to some AUs to perform operations on some objects specified by OAs.

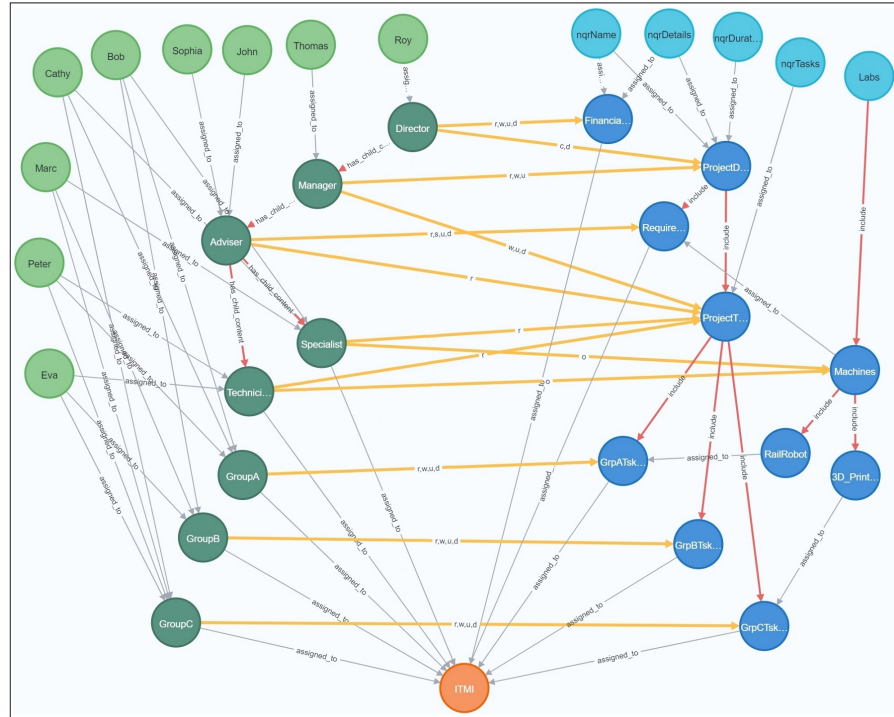


Figure 12. Case Study 1: NGAC graph.

690 permissions associated for some users on some objects such as: in (a) Roy is assigned to
 691 the role of director and also has the permission of the lower roles (manager . . .); in (b)
 692 Thomas is assigned to role manager also has the permission of the lower roles (adviser
 693 . . .); in (c) Sophia is assigned to role adviser also has the permission of the lower roles;
 694 in (d) Marc is assigned to role specialist and (e) Eva is assigned to role technician, in
 695 addition to the group permission for users who are also assigned to groups. For example,
 696 the user Roy has the permission DirPermission to confirm and delete {c, d} ProjectDetails
 697 data through the Roy-Director assignment, Roy also has the permission ManPermission
 698 to {r, w, u} ProjectDetails through the Director-Manager assignment which is expressed
 699 as 'has_child_content' relationship to represent role hierarchy. Moreover, he would have
 700 the permission of the lower roles to perform the needed operations on the needed objects.
 701 Consequently, a manager with ManPermission also has AdvPermission, SpePermission,
 702 TecPermission, and so on.

703 Figure 14 illustrates two examples of the association relationship properties for
 704 the manager and adviser permissions, note that we express the contextual and non-
 705 contextual attributes with the permission relationship properties. Consequently, the
 706 manager has ManPermission to {r, w, u} ProjectDetails if the object attribute PrjConfirm
 707 value is false, and the adviser has AdvPermission to {r, s, u, d} project Requirements
 708 when the context attributes LoginLocation is local and if the current date is before the
 709 endDate of the project. In Figure 15 we run some Cypher queries with some constraints
 710 using context, user, and object attributes to show NGAC authorization responses to
 711 Cypher statements. In (a) we show that the manager is able to perform the needed
 712 operations on ProjectDetails, but when the value of prjConfirm is updated (by the
 713 director) to 'true' (b), he would not be able to perform any operation on the specified
 714 object container (c). In Figure 15 (d) and (e) we show that an adviser is able to {r, w, u,
 715 d} project Requirements if the current date is less than the end project date. In (f) and
 716 (g), we show what roles are allowed to perform some operations on ProjectDetails and
 717 ProjectTasks objects.



Figure 13. Case Study 1: Users' Permissions based on their roles (and role hierarchy).

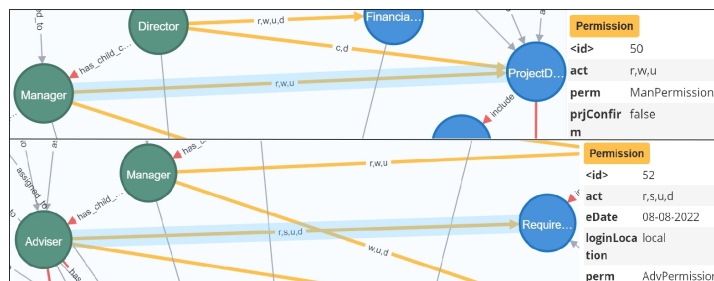


Figure 14. Case Study 1: Examples of association relationship properties for role permissions.

<pre> 1 MATCH (u:U)-[:assigned_to]-(ua:UA)-[r:Permission]-(oa:OA) 2 Where ua.title='Manager' and oa.oaname='ProjectDetails' and r.prjConfirm='false' 3 Return u.sname As User,r.perm As Permission,r.act As Operation order by User Asc </pre>	<table border="1"> <thead> <tr> <th>"User"</th> <th>"Permission"</th> <th>"Operation"</th> </tr> </thead> <tbody> <tr> <td>"Thomas"</td> <td>"ManPermission"</td> <td>"r,w,u"</td> </tr> </tbody> </table>	"User"	"Permission"	"Operation"	"Thomas"	"ManPermission"	"r,w,u"	(a)														
"User"	"Permission"	"Operation"																				
"Thomas"	"ManPermission"	"r,w,u"																				
<pre> 1 MATCH (auh02:UA {title:'Manager'})-[r:Permission]-(oa:OA{oaname:'ProjectDetails'}) 2 set r.prjConfirm = 'true' return r.act As Operation, r.prjConfirm As Confirmation </pre>	<table border="1"> <thead> <tr> <th>"Operation"</th> <th>"Confirmation"</th> </tr> </thead> <tbody> <tr> <td>"r,w,u"</td> <td>"true"</td> </tr> </tbody> </table>	"Operation"	"Confirmation"	"r,w,u"	"true"	(b)																
"Operation"	"Confirmation"																					
"r,w,u"	"true"																					
<pre> 1 MATCH (u:U)-[:assigned_to]-(ua:UA)-[r:Permission]-(oa:OA) 2 Where ua.title='Manager' and oa.oaname='ProjectDetails' and r.prjConfirm='false' 3 Return u.sname As User,r.perm As Permission,r.act As Operation order by User Asc </pre>	(no changes, no records)	(c)																				
<pre> 1 MATCH (u:U)-[:assigned_to]-(ua:UA)-[r:Permission]-(oa:OA) 2 Where ua.title='Adviser' and oa.oaname='Requirements' and r.eDate > date() 3 Return u.sname As User, r.perm As Permission, r.act As Operation, 4 r.eDate As EndDate,date() As CurrentDate order by User Asc </pre>	<table border="1"> <thead> <tr> <th>"User"</th> <th>"Permission"</th> <th>"Operation"</th> <th>"EndDate"</th> <th>"CurrentDate"</th> </tr> </thead> <tbody> <tr> <td>"John"</td> <td>"AdvPermission"</td> <td>"r,s,u,d"</td> <td>"2022-08-08"</td> <td>"2022-05-11"</td> </tr> <tr> <td>"Sophia"</td> <td>"AdvPermission"</td> <td>"r,s,u,d"</td> <td>"2022-08-08"</td> <td>"2022-05-11"</td> </tr> </tbody> </table>	"User"	"Permission"	"Operation"	"EndDate"	"CurrentDate"	"John"	"AdvPermission"	"r,s,u,d"	"2022-08-08"	"2022-05-11"	"Sophia"	"AdvPermission"	"r,s,u,d"	"2022-08-08"	"2022-05-11"	(d)					
"User"	"Permission"	"Operation"	"EndDate"	"CurrentDate"																		
"John"	"AdvPermission"	"r,s,u,d"	"2022-08-08"	"2022-05-11"																		
"Sophia"	"AdvPermission"	"r,s,u,d"	"2022-08-08"	"2022-05-11"																		
<pre> 1 MATCH (u:U)-[:assigned_to]-(ua:UA)-[r:Permission]-(oa:OA) 2 Where ua.title='Adviser' and oa.oaname='Requirements' and r.eDate < date() 3 Return u.sname As User, r.perm As Permission, r.act As Operation, 4 r.eDate As EndDate,date() As CurrentDate order by User Asc </pre>	(no changes, no records)	(e)																				
<pre> 1 MATCH (u:U)-[:assigned_to]-(ua1:UA)-[r:Permission]-(oa:OA) 2 where oa.oaname='ProjectDetails' RETURN distinct ua1.title As Role, 3 r.perm As Permission,r.act AS Operation,oa.oaname As Object </pre>	<table border="1"> <thead> <tr> <th>"Role"</th> <th>"Permission"</th> <th>"Operation"</th> <th>"Object"</th> </tr> </thead> <tbody> <tr> <td>"Manager"</td> <td>"ManPermission"</td> <td>"r,w,u"</td> <td>"ProjectDetails"</td> </tr> <tr> <td>"Director"</td> <td>"DirPermission"</td> <td>"c,d"</td> <td>"ProjectDetails"</td> </tr> </tbody> </table>	"Role"	"Permission"	"Operation"	"Object"	"Manager"	"ManPermission"	"r,w,u"	"ProjectDetails"	"Director"	"DirPermission"	"c,d"	"ProjectDetails"	(f)								
"Role"	"Permission"	"Operation"	"Object"																			
"Manager"	"ManPermission"	"r,w,u"	"ProjectDetails"																			
"Director"	"DirPermission"	"c,d"	"ProjectDetails"																			
<pre> 1 MATCH (u:U)-[:assigned_to]-(ua1:UA)-[r:Permission]-(oa:OA) 2 Where oa.oaname='ProjectTasks' RETURN distinct ua1.title As Role, 3 r.perm As Permission, r.act AS Operation,oa.oaname As Object </pre>	<table border="1"> <thead> <tr> <th>"Role"</th> <th>"Permission"</th> <th>"Operation"</th> <th>"Object"</th> </tr> </thead> <tbody> <tr> <td>"Technician"</td> <td>"TecPermission"</td> <td>"r"</td> <td>"ProjectTasks"</td> </tr> <tr> <td>"Specialist"</td> <td>"SpePermission"</td> <td>"r"</td> <td>"ProjectTasks"</td> </tr> <tr> <td>"Adviser"</td> <td>"AdvPermission"</td> <td>"r"</td> <td>"ProjectTasks"</td> </tr> <tr> <td>"Manager"</td> <td>"ManPermission"</td> <td>"w,u,d"</td> <td>"ProjectTasks"</td> </tr> </tbody> </table>	"Role"	"Permission"	"Operation"	"Object"	"Technician"	"TecPermission"	"r"	"ProjectTasks"	"Specialist"	"SpePermission"	"r"	"ProjectTasks"	"Adviser"	"AdvPermission"	"r"	"ProjectTasks"	"Manager"	"ManPermission"	"w,u,d"	"ProjectTasks"	(g)
"Role"	"Permission"	"Operation"	"Object"																			
"Technician"	"TecPermission"	"r"	"ProjectTasks"																			
"Specialist"	"SpePermission"	"r"	"ProjectTasks"																			
"Adviser"	"AdvPermission"	"r"	"ProjectTasks"																			
"Manager"	"ManPermission"	"w,u,d"	"ProjectTasks"																			

Figure 15. Case Study 1: Examples of NGAC authorization responses to Cypher statements.

718 Controlling access to resources in industrial organizations is often managed at the
719 application level, which is sufficient in static computing environments where changes in
720 the data sources or system are not expected. On the other hand, industry 4.0 applications
721 are generally dynamic, which means that new machines, devices, sensors, users, etc. are
722 frequently added or changed, and the AC policies need to be frequently added and up-
723 dated. Accordingly, industry 4.0 environment, and other highly dynamic environments,
724 need flexible and frequently upgradable AC models to answer the frequent changes in
725 AC requirements. In the following section, we propose a simplified case study for the
726 industry 4.0 environment.

727 6. Case Study 2— ITMI: IoT

728 Due to the immense value that IoT brings to every organization, ITMI refers to the
729 use of IoT in improving the existing systems and processes and enabling it to increase
730 operational efficiency, create better experiences, and unlock additional value for the
731 running projects. For ITMI, IoT and industry 4.0 reflect a growing focus on driving
732 results using sensor-based data and creating analytically rich data sets.

733 Figure 16 illustrates the system architecture of ITMI's IoT environment. Note that,
734 this case study is a subsequent project related to the previous case study. One of the
735 important IoT projects that are maintained by ITMI, is the Inspection of the Railways
736 of Quebec (IRQ) in Canada to detect any unrevealed crack in railway tracks in order

737 to avoid accidents. To achieve this, the rail robot (which is accomplished in case study
 738 1) and a drone are connected or paired together so that they are synchronized and
 739 well geolocated. Both are connected to the internet via cellular or satellite networks
 740 depending on where the inspection is taking place. The sensor nodes attached to the
 741 rail robot are used to inspect the railways and capture images for the crack detection on
 742 railway tracks as well as any technical problem. Synchronously, the drone also captures
 743 images for the geographical locations along with the robot where cracks in railway tracks
 744 exist, and all the data and captured images are sent to ITMI's cloud server. To complete
 745 this process the specialists 'Bob' and 'Cathy' and the technician 'Peter' need to travel to
 746 the sites having the 'Drone' and the 'Rail Robot' where they are released on a specific
 747 point to start the inspection process. Once the process is done, 'Bob', 'Cathy', and 'Peter'
 748 have to travel back and place the machines back in the labs. Thereafter, the Adviser
 749 would be able to access the public database after verifying his identity to analyze the
 750 obtained data and the captured images, then write the needed report(s) of the inspection
 751 results and save them in table Results in the database. The report(s) need to be confirmed
 752 by the manager before emailing them to the company which maintains these railways.
 753 Figure 17 illustrates the flow of information for the case study scenario using the Neo4j
 graph database.

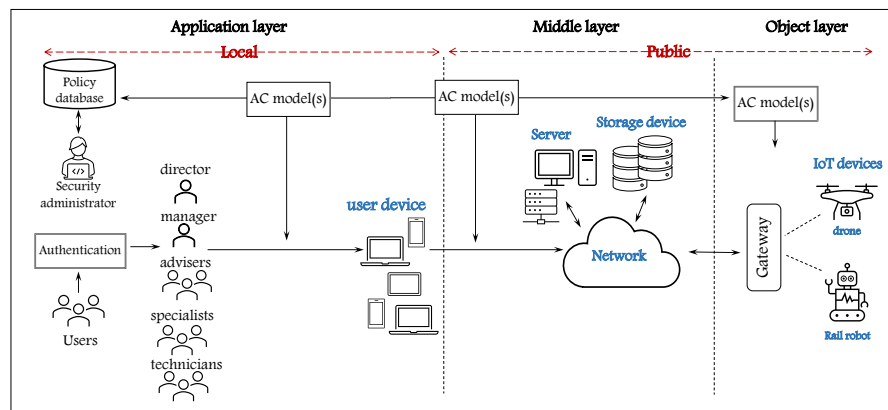


Figure 16. The system architecture of ITMI: non-IoT environment

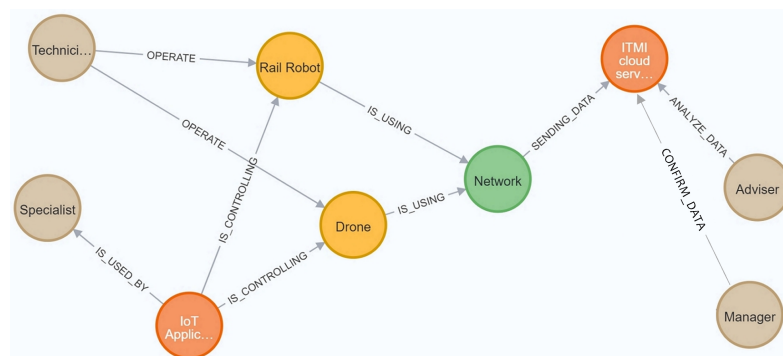


Figure 17. A graph model representing the information flow of ITMI's IoT environment

754

755 **6.1. Challenge**

756 The rail robot and the drone machines could generate privacy-sensitive information,
 757 for example, for some logistic locations. It is important to provide an efficient AC model
 758 that is able to consider all the needed factors to avoid operating the devices for any
 759 illegal use by any illegal user. With a set of physical identities (ITMI workers, company
 760 employees, and maybe others) in some of the sites where the inspection process for

761 the railways is assumed to take place, operating the rail robot and the drone must be
 762 done by trusted users. Moreover, the machines are objects that need to be controlled by
 763 authorized users and are also considered users when they send and insert data into the
 764 public database at ITMI's cloud server, in other words, they are objects and subjects at
 765 the same time.

766 6.2. *The Informal Policy: the access rights*

767 In this case study, AC is based on users' roles, in addition to other attributes of
 768 subjects, objects, and environment. Since some users having the same role are not
 769 allowed to access the labs, we need to specify users who are allowed to take out the
 770 'Rail Robot' from 'AI Lab' and the 'Drone' from embedded systems (ES) 'ES Lab', and
 771 operate them onsite. Note that, the allowed users should be able to access the labs 24/7
 772 when the status of this task is 'in progress' and within the start and end date of the
 773 running project for the purpose of taking out and returning back the machines after the
 774 inspection process. Onsite, users must log into the machines using PIN codes, they have
 775 three possible attempts to use the correct PIN code in order to operate and control them;
 776 otherwise, the machines, and via the use of a GPS tracking system send alert messages
 777 to the manager, the specialists and the technician with the geographical coordinate of
 778 the location to take the needed actions. In the following we summarize users' access
 779 rights specified for this case study:

- 780 • The specialists 'Bob' and 'Cathy' with the technician 'Peter' are allowed to operate
 781 the 'Rail Robot' and the 'Drone' using the IoT device controller to control the
 782 machines and read the collected data and images.
- 783 • The adviser 'John' have full access to database tables at the ITMI cloud server where
 784 the collected data and images from the site are received from the 'Rail Robot' and
 785 the 'Drone'. He analyzes the obtained information, then writes/inserts the needed
 786 reports to table 'Results'.
- 787 • Before emailing the reports to the company which has to maintain the railways, the
 788 manager 'Thomas' must read (and modify if needed), and confirm the report(s).

789 All users are allowed to perform the specified actions if the task status is 'in progress'
 790 and within the start and end date of the project.

791 6.3. *The Solution: HEAD Metamodel*

792 In complex computing environments, variants of AC policies need to be defined and
 793 enforced at different levels of application logic. To prevent confidentiality and integrity
 794 breaches, system administrators and security experts must ensure that the various types
 795 of policies are defined and enforced. HEAD metamodel supports the feature of deriving
 796 various models using the same DSL language, it allows the creation of various types of
 797 policy elements with the relationships between them. As shown in Figure 16, to protect
 798 the resources there are various AC models need to be implemented. To control access
 799 and protect the resources in the local environment we provide case study 1. In this case
 800 study, our concern is to derive the needed model(s) to protect the resources of ITMI's
 801 public environment. However, for this case scenario, we need also to know what is the
 802 AC model(s) that best fits the (informal) AC requirements to be formalized and generate
 803 the needed AC rules. For the needed solution, we have to consider that the physical
 804 objects are also subjects (users)—for example, the 'RailRobot' and the 'Drone' machines
 805 are objects that need to be protected from any illegal access, they are also subjects when
 806 they need to connect to ITMI's public database in order to insert/write the collected data.
 807 For this purpose we have to derive a hybrid model with two PCs, PC1 considers the
 808 machines as objects and PC2 considers them as subjects. However, the DSL of HEAD
 809 metamodel allows instantiating different PCs of E_x , I_m , and S_t entities. Hence:

- 810 • PC1:
- 811 – $E_x = \{\text{subject (sname, ...); object (oname, ...)}\}.$

- 812 – I_m : AUs = {role (type ...)} and PUs = {permission (perm ...); action (type ...)}
- 813 – S_t = {context (loginLocation, time, pw-attempts ...); constraint (inspectionState,
- 814 confirmation ...)}.
- 815 • PC2:
- 816 – E_x = {subject (sname, ...); object (oname, ...)}.
- 817 – I_m is PU = {action (type, ...)}
- 818 – S_t = {context (Location, pwAttempts ...)}.

819 6.3.1. The Formal Policy Model

820 To implement the needed solution, we have to find the best AC model that fits the
821 AC requirements of this case study. Due to the above derived E_x , I_m , and S_t entities, we
822 have the following possibilities:

- 823 • Hybrid RBAC/ABAC model: due to the above entities for PC1, the notion of role
824 reflects the importance of considering the RBAC model, and the need to express
825 static and dynamic AC rules based on subject, object, and context attributes reflect
826 the importance of considering ABAC model.
- 827 • ABAC model: the above entities for PC2 match the entities of ABAC.

828 Hence, our solution, in this case, is to derive a hybrid model with two PCs (hybrid
829 RBAC/ABAC and ABAC). Using the DSL of the HEAD metamodel, we derive the
830 needed model as shown in Figure 18. In Figure 18(a) we define the following:

- 831 • lines 1 to 38: the block of specifying two PCs.
- 832 • lines 1 to 22: the specification of PC1.
- 833 • lines 2 to 8: the block of creating E_x entities, subject and object entities, it starts
834 and ends with the keywords explicit and end. Three levels of object hierarchy are
835 created to represent the hierarchy of objects. Note that, we actually have two levels
836 of the object hierarchy, but since we are using NGAC as an enforcement framework
837 we create an additional level. object1 level is to express Os which will be assigned
838 to different containers, and the other two levels represent the hierarchy of objects
839 containers.
- 840 • lines 9 to 12: the block of creating AU entities, starts and ends with the keywords
841 authorization and end. Two levels of role hierarchy are created for the manager
842 and the adviser.
- 843 • lines 13 to 16: the block of creating PU entities, starts and ends with the keywords
844 procedural and end.
- 845 • lines 17 to 21: the block of creating S_t entities, context and constraint entities and
846 attributes, starts and ends with the keywords setting and end.
- 847 • lines 23 to 37: the specification of PC2.
- 848 • lines 24 to 29: the block of creating E_x entities, subject, and object.
- 849 • lines 30 to 32: the block of creating PU entity, action entity, and attributes.
- 850 • lines 33 to 36: the block of creating S_t entities, context, and constraint.

851 In Figure 18 (b) we express some AC rules based on the defined entities/attributes:

- 852 • lines 40 to 51: the rule expression of the hybrid RBAC/ABAC model. It expresses
853 the access rights for users using the defined components/attributes of PC1.
- 854 • lines 52 to 61: the rule expression of ABAC model. It expresses the access rights for
855 users using the defined components/attributes of PC2.
- 856 • lines 62 to 71: the expression of a hybrid rule using the defined components/attributes
857 of PC1 and PC2.

858 Note that, for each PC the names of the components must be unique, for example, we
859 create two different names for the subject (for PC1 we create 'subject' entity and for PC2
860 we create 'subject' entity). A considerable advantage of the DSL of the HEAD metamodel
861 is that it allows specifying an unlimited number of PCs with the flexibility of defining
862 entities/attributes for each PC and expressing hybrid rules.

<pre> 1 policy RBAC_ABAC (industry1:String) 2 explicit 3 subject (sname:String) 4 object1 (oaname:String)[5 object2 (oaname:String)[6 object3 (oaname:String)] 7] 8 end 9 authorization 10 role1(title:String)[11 role2(title:String)] 12 end 13 procedural 14 permission(perm:String) 15 action(act:String) 16 end 17 setting 18 context(loginLocation:String Date:String) 19 constraint(Confirm:String 20 tsk_status:String Inspec_status:String) 21 end 22 end 23 policy ABAC (industry2:String) 24 explicit 25 subjct (sname:String) 26 objct (oaname:String)[27 objct1(oaname:String) 28] 29 end 30 procedural 31 acton(act:String) 32 end 33 setting 34 contxt(loginLocation:String 35 pwAttemps:String Date:String) 36 end 37 end 38 39 rule: 40 (ruleid1:int) { 41 RBAC_ABAC.subject(RBAC_ABAC.subject.sname)[RBAC_ABAC.role1(RBAC_ABAC.role1.title)]{ 42 RBAC_ABAC.permission(RBAC_ABAC.permission.perm){ 43 RBAC_ABAC.object1(RBAC_ABAC.object1.oaname){ 44 RBAC_ABAC.action(RBAC_ABAC.action.act){ 45 RBAC_ABAC.constraint(RBAC_ABAC.constraint.Confirm 46 RBAC_ABAC.constraint.Inspec_status RBAC_ABAC.context.Date 47 RBAC_ABAC.context.loginLocation) 48 } 49 } 50 } 51 }-->decision 52 (ruleid2:int) { 53 ABAC.subjct(ABAC.subjct.sname){ 54 ABAC.objct(ABAC.objct.oaname){ 55 ABAC.acton(ABAC.acton.act) 56 ABAC.acton(ABAC.acton.act){ 57 ABAC.contxt(ABAC.contxt.loginLocation ABAC.contxt.Date ABAC.contxt.pwAttemps) 58 } 59 } 60 } 61 }-->decision 62 (ruleid2:int) { 63 RBAC_ABAC.subject(RBAC_ABAC.subject.sname){ 64 RBAC_ABAC.permission(RBAC_ABAC.permission.perm){ 65 ABAC.objct(ABAC.objct.oaname){ 66 ABAC.acton(ABAC.acton.act) 67 RBAC_ABAC.action(RBAC_ABAC.action.act) 68 } 69 } 70 } 71 }-->decision </pre>	<ul style="list-style-type: none"> ✦ Metamodel ✦ Policy RBAC_ABAC <ul style="list-style-type: none"> ✦ Attribute industry1 ✦ Explicit subject <ul style="list-style-type: none"> ✦ Attribute sname ✦ Explicit object1 <ul style="list-style-type: none"> ✦ Attribute oname ✦ Explicit object2 ✦ Implicit <ul style="list-style-type: none"> ✦ Authorization Unit role1 ✦ Procedural Unit permission ✦ Procedural Unit action ✦ Setting context <ul style="list-style-type: none"> ✦ Attribute loginLocation ✦ Attribute Date ✦ Setting constraint ✦ Policy ABAC <ul style="list-style-type: none"> ✦ Attribute industry2 ✦ Explicit subjct <ul style="list-style-type: none"> ✦ Attribute sname ✦ Explicit objct <ul style="list-style-type: none"> ✦ Attribute oname ✦ Explicit objct1 ✦ Implicit <ul style="list-style-type: none"> ✦ Procedural Unit acton ✦ Setting contxt <ul style="list-style-type: none"> ✦ Attribute loginLocation ✦ Attribute pwAttemps ✦ Attribute Date ✦ Decision
(a)	
(b)	

Figure 18. Case Study 2: HEAD metamodel instance: (a) A hybrid model with two PCs (hybrid RBAC/ABAC and ABAC); (b) rule expressions.

863 6.3.2. The Formal Generated Policy

864 In this section we describe the policy elements for each PC:

- 865 • PC1: Hybrid RBAC/ABAC

- 866 – Users = {Thomas, John, Bob, Cathy, Peter}
- 867 – User Attributes: each user is associated with a set of attributes (e.g., name ...).
- 868 – Role = {manager, adviser}.
- 869 – Objects: set of logical (e.g., tables rows/columns at the cloud server) and
- 870 physical (e.g., rail robot and drone) objects need protection.
- 871 – Object attributes: object attributes include inspection_status, confirmation, etc.
- 872 – Actions: a set of actions on logical objects in the database include read, write,
- 873 update, delete, etc.; and on physical objects include control, operate, etc.
- 874 – Permissions: users are given permission(s) to perform actions on objects based
- 875 on their roles.
- 876 – Context attributes: for a given context—example, log in via private/public net-
- 877 work, machine failure, etc.—context attributes include the date, time, location,
- 878 password attempts, etc., contextual constraint expressions are important for
- 879 authorization decisions.
- 880 – Constraints: expressions include the other various types of attributes, for
- 881 example, subject attributes, object attributes, etc.

The policy expression for PC1 can be expressed as follows:

$$\text{Policy} = \langle \text{subject}(\text{name} \dots); \text{object}(\text{title} \dots); \text{role}(\text{type} \dots); \text{permission}(\text{ptype} \dots); \\ \text{action}(\text{type} \dots); \text{context}(\text{date} \dots); \text{constraints}(\text{confirm} \dots) \rangle$$

- 882 • PC2: ABAC

- 883 – Users = {RailRobot, Drone}, and users' attributes (e.g., machineName).
- 884 – Objects: the logical resources (e.g., tables rows/columns at the cloud server),
- 885 with their attributes, e.g., coordinates, image, etc.
- 886 – Actions: set of actions on logical objects in the database, e.g., write.
- 887 – Contextual and non-contextual constraint expressions with the attributes of
- 888 context, subject, and object.

The policy expression for PC2 can be expressed as follows:

$$\text{Policy} = \langle \text{subject}(\text{name} \dots); \text{object}(\text{title} \dots); \text{action}(\text{type} \dots); \text{context}(\text{time} \dots) \rangle$$

889 To represent the concrete instance of the hybrid model, xtend notation is used to
 890 generate the needed java code at the system level. The output of the java code, the
 891 Cypher queries, is injected into the Neo4j database to represent the graph of NGAC
 892 policy. In Figure 19 we show a sample of xtend code:

- 893 • (a) to specify the PCs (lines 1 to 9), in this case study we have two PCs as instantiated
 894 (as shown in Figure 18 RBAC/ABAC for PC1, and ABAC for PC2), then the E_x , AU,
 895 PU, and S_t entities are created for each PC;
- 896 • (b) to define AUs and their hierarchies (e.g., role and role hierarchy), lines 1-11 to
 897 generate the java code for the root node(s) of AUs and express their Cypher code
 898 (they represent the user attributes of NGAC policy), lines 15-36 to generate the java
 899 code for defining the children of the root node(s), and lines 37-57 to express the
 900 Cypher code and the hierarchy relationship between them; and
- 901 • (c) assign E_x entities to AUs (e.g, user-role assignment), and express the Cypher
 902 statement of their relationships.

903 The sample code of xtend notation (in Figure 19 (b) and (c)) generates the sample java
 904 code in Figure 20:

- 905 • part (a) is a sample for the AU root entities and their child entities (in this sample
 906 the generated code is for role entity) of the instantiated Hybrid model;

```

1  /*---Specify Policy---*/
2  «FOR pl:0..<plist.size»
3  «FOR a : plist.get(pl).attributes»
4  System.out.println("«plist.get(pl).name» Policy for «a.name»:");
5  «a.type» «a.name.toString.toLowerCase» = sc.nextLine();
6  rules.add("CREATE (:PC«pl+1» {«a.name»: "+ «a.name.toString.toLowerCase»+"}");
7  pc.add(«a.name.toString.toLowerCase»);
8  «ENDFOR»
9  «ENDFOR»
(a)

1  «FOR j : 0 ..< auroot1.size»
2  System.out.println("«auroot1.get(j).name.toUpperCase(s)-----");
3  «FOR a : auroot1.get(j).attributes»
4  System.out.println("«auroot1.get(j).name»«a.name»: ");
5  temp = sc.nextLine().split(" ");
6  for (int i=0; i < temp.length; i++) {
7  «auroot1.get(j).name.toString.toLowerCase».set«a.name.toFirstUpper»(temp[i]);
8  r_«auroot1.get(j).name.toString.toLowerCase».add(«auroot1.get(j).name.toString.toLowerCase».get«a.name.toFirstUpper»());
9  rules.add("CREATE (:UA1 {«a.name»: "+ temp[i] + "});");
10 }
11 «ENDFOR» «ENDFOR»
12 «FOR j : 0 ..< auroot1.size»
13 «FOR hr : 0 ..< auhrchyl1.size»
14 «IF hr == 0»
15 for (int i=0; i < r_«auroot1.get(j).name.toString.toLowerCase».size(); i++) {
16 System.out.println("Does "+ r_«auroot1.get(j).name.toString.toLowerCase».get(i) + " has children? (y/n) ");
17 au_hrchy.add(r_«auroot1.get(j).name.toString.toLowerCase».get(i));
18 answer = sc.nextLine();
19 «ELSEIF hr >= 1»
20 for (int i=0; i < h_«auhrchyl1.get(hr-1).name.toString.toLowerCase».size(); i++) {
21 if(h_«auhrchyl1.get(hr-1).name.toString.toLowerCase».get(i) == null){
22 continue;
23 }else{
24 System.out.println("Does "+ h_«auhrchyl1.get(hr-1).name.toString.toLowerCase».get(i)+ " has children? (y/n) ");
25 «FOR x: 0..< hr»
26 «ENDFOR»
27 answer = sc.nextLine();
28 «ENDIF»
29 if(answer.equals("Y") || answer.equals("y")){
30 «IF hr == 0»
31 System.out.println("Children of "+ r_«auroot1.get(j).name.toString.toLowerCase».get(i)+ " ": ");
32 «ELSEIF hr >= 1»
33 System.out.println("Children of "+ h_«auhrchyl1.get(hr-1).name.toString.toLowerCase».get(i)+ " ": ");
34 «ENDIF»
35 «FOR a : auhrchyl1.get(j).attributes»
36 temp = sc.nextLine().split(" ");
37 for (j=0; j < temp.length; j++) {
38 «auhrchyl1.get(hr).name.toString.toLowerCase».set«a.name.toFirstUpper»(temp[j]);
39 h_«auhrchyl1.get(hr).name.toString.toLowerCase».add(«auhrchyl1.get(hr).name.toString.toLowerCase».get«a.name.toFirstUpper»());
40 rules.add("CREATE (:UA1 {«a.name»: "+temp[j]+"});");
41 «IF hr == 0»
42 strp="MATCH (e«hr»"+i+j+" :UA1 {«FOR at : auroot1.get(j).attributes»«at.name»: "+
43 «ENDFOR»r_«auroot1.get(j).name.toString.toLowerCase».get(i)+"});");
44 «ELSEIF hr >= 1»
45 strp="MATCH (e«hr»"+i+j+" :UA1 {«FOR at : auhrchyl1.get(hr-1).attributes»«at.name»: "+
46 «ENDFOR»h_«auhrchyl1.get(hr-1).name.toString.toLowerCase».get(i)+"});");
47 «ENDIF»
48 strc= strp + " MATCH (e«hr»"+i+j+" :UA1 {«a.name»: "+temp[j]+"}); + " MERGE (e«hr»"+i+j+"-[include]->(e«hr»"+i+j+"));");
49 rules.add(strc);
50 }
51 «auhrchyl1.get(hr).name.toString.toLowerCase».set«a.name.toFirstUpper»(null);
52 h_«auhrchyl1.get(hr).name.toString.toLowerCase».add(«auhrchyl1.get(hr).name.toString.toLowerCase».get«a.name.toFirstUpper»());
53 }else{
54 «auhrchyl1.get(hr).name.toString.toLowerCase».set«a.name.toFirstUpper»(null);
55 h_«auhrchyl1.get(hr).name.toString.toLowerCase».add(«auhrchyl1.get(hr).name.toString.toLowerCase».get«a.name.toFirstUpper»());
56 continue;
57 } //end (if answer) //end (for i)
58 «ENDFOR»
59 «ENDFOR»
(b)

1  System.out.println("\n-----Explicit - AuthorizationUnit Assignments-----");
2
3  «FOR j : 0 ..< explicitlist1.size»
4  «FOR jj : 0 ..< aulist1.size»
5  System.out.println("***ASSIGN «explicitlist1.get(j).name.toFirstUpper(s) to «aulist1.get(jj).name.toFirstUpper(s)***");
6  «ENDFOR»
7  for (int i=0; i < e_«explicitlist1.get(j).name.toString.toLowerCase».size(); i++){
8  «FOR a : explicitlist1.get(j).attributes»
9  «FOR jj : 0 ..< aulist1.size»
10 for (int k=0; k < au_«aulist1.get(jj).name.toString.toLowerCase».size(); k++){
11 «FOR at : aulist1.get(jj).attributes»
12 System.out.println(e_«explicitlist1.get(j).name».get(i)+"->"+au_«aulist1.get(jj).name».get(k)+"? (y/n)");
13 answer = sc.nextLine();
14 if(answer.equals("Y") || answer.equals("y")){
15 rules.add("MATCH (eau"+i+k+":U {«a.name»: "+ e_«explicitlist1.get(j).name».get(i)+"});
16 MATCH (au"+i+k+":UA {«at.name»: "+ au_«aulist1.get(jj).name».get(k)+"}) MERGE (eau"+i+k+"-[assigned_to]->(au"+i+k+"));");
17 }
18 «ENDFOR»
19 }
20 «ENDFOR»
21 «ENDFOR»
22 «ENDFOR»
23 }
(c)

```

Figure 19. Case Study 2: A sample of the Xtend notation to generate the java code for the (a) PC, (b) AU (and AU hierarchy) entities, (c) the assignment of E_x -AUs entities.

- 907 • part (b) is a sample of the generated java code for user-role (root nodes) assignment,
908 and the Cypher statement expressions.

```

System.out.println("ROLE(s)-----");
System.out.println("\trole1.title: ");
temp = sc.nextLine().split(" ");
for (int i=0; i < temp.length; i++) {
    role1.setTitle(temp[i]);
    r_role1.add(role1.getTitle());
    rules.add("CREATE (:UA1 {title:'" + temp[i] + "'})");
}
for (int i=0; i < r_role1.size(); i++) {
    System.out.println("Does " + r_role1.get(i) + " has children? (y/n) ");
    au_hrchy.add(r_role1.get(i));
    answer = sc.nextLine();
    if(answer.equals("Y") || answer.equals("y")){
        System.out.println("Children of " + r_role1.get(i)+ " ");
        temp = sc.nextLine().split(" ");
        for (j=0; j < temp.length; j++) {
            role2.setTitle(temp[j]);
            h_role2.add(role2.getTitle());
            rules.add("CREATE (:UA1 {title:'"+temp[j]+''})");
            strp="MATCH (er0'+i+j':UA1 {title:'"+
                r_role1.get(i)+''})";
            strc= strp + " MATCH (eh0'+i+j':UA1 {title:'"+temp[j]+''})" + " MERGE (er0'+i+j')-[:include]->(eh0'+i+j+)";
            rules.add(strc);
        }
        role2.setTitle(null);
        h_role2.add(role2.getTitle());
    }else {
        role2.setTitle(null);
        h_role2.add(role2.getTitle());
        continue;
    }
}
}

System.out.println("\n-----Explicit - AuthorizationUnit Assignments-----");

System.out.println("***ASSIGN Subject(s) to Role (s)***");
for (int i=0; i < e_subject.size(); i++){
    for (int k=0; k < r_role1.size(); k++){
        System.out.println(e_subject.get(i)+" --> "+r_role1.get(k)+"? (y/n)");
        answer = sc.nextLine();
        if(answer.equals("Y") || answer.equals("y")){
            rules.add("MATCH (erau'+i+k':U1{sname:'"+ e_subject.get(i)+''}) "
                + "MATCH (rau'+i+k':UA1{title:'"+r_role1.get(k)+''}) MERGE (erau'+i+k')-[:assigned_to]->(rau'+i+k+)");
        }
    }
}
}

```

(a)

(b)

Figure 20. Case Study 2: A sample of the generated java code for (a) AU root entities and their child entities; and (b) a sample of the assignment of user to root role nodes (Us to UAs).

909 In Figure 21 we show a sample output of the java code (of Figure 20):

- 910 • in part (a) a sample of specified policy classes and the configuration of their entities/attributes which are configured by a system administrator with setting up the relationships between object containers.
911
912
913 • in part (b), the creation of instances of role entities and role hierarchy, in addition to the assignment of E_x -AUs which represents in this case user-role assignment.
914

915 After configuring the policy, Cypher queries are generated as an output from the java code (of Figure 21) in a form the result of them matches the NGAC policy graph with two PCs. Different output samples of Cypher queries are shown in Figure 22:

- 918 • in part (a) the Cypher code to create PCs, Us, and Os.
919 • in part (b) a sample of the generated the Cypher code for PC1, to create UAs with the needed assignment and hierarchies. Note that, we consider Workers as a role in this case study for U-UA assignment.
920
921
922 • in part (c) we show a sample of Cypher code to create UAs/OAs nodes of PC2 (e.g., IoTM1/ RailwayData) and assign the appropriate Us/Os (e.g., MRailRobot/ Machine1Data) to them.
923
924

925 Before presenting the NGAC graph, we have to explain the NGAC policy configuration of this IoT case study. However in Figure 23:

- 927 • in (a), we show the configuration of user containers. Red containers represent the assignment of Us to UAs of the first PC1, and blue ones represent the assignment of Us to UAs of the second PC2. U of PC1 stands for Thomas, John, etc., and UA of PC1 stands for the role (Manager, Adviser, etc.). U of PC2 stands for 'RailRobot' and 'Drone', and UA of PC2 stands for first and second IoT machines.
928
929
930
931

```

RBAC_ABAC Policy for industry1:ITMIiot1
ABAC Policy for industry2:ITMIiot2
*****
Configuration of: ITMIiot1
-----
SUBJECT(s)-----
  subject.sname: Thomas John Bob Cathy Peter
OBJECT1(s)-----
  object1.oname: CollectedInfo CollectedImages Machine1 Machine2
Does CollectedInfo has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2)
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2)
*****
Configuration of: ITMIiot2
-----
SUBJECT(s)-----
  subjct.sname: MRailRobot MDrone
OBJECT(s)-----
  objct.oname: Machine1Data Machine2Data
Does Machine1Data has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 2
Machine1Data 'assigned_to' Container(s): RailwayData
Does Machine2Data has attribute container(s)? (y/n) y
1-include_hierarchical/2-assigned_to attribute container(s)? (1/2) 2
Machine2Data 'assigned_to' Container(s): GeolocationData

ROLE(s)-----
  role1.title: Manager Workers
Does Manager has children? (y/n) y
Children of Manager: Adviser
Does Workers has children? (y/n) n
*****
----- Explicit - AuthorizationUnit Assignments-----
***ASSIGN Subject(s) to Role hierarchy***
Thomas --> Manager? (y/n)y
Thomas --> Workers? (y/n)n
John --> Manager? (y/n)n
John --> Workers? (y/n)n
Bob --> Manager? (y/n)n
Bob --> Workers? (y/n)y
Cathy --> Manager? (y/n)n
Cathy --> Workers? (y/n)y
Peter --> Manager? (y/n)n
Peter --> Workers? (y/n)y
Thomas --> Adviser? (y/n)n
John --> Adviser? (y/n)y
Bob --> Adviser? (y/n)n
Cathy --> Adviser? (y/n)n
Peter --> Adviser? (y/n)n

```

Figure 21. Case Study 2: A sample of java output (a) to configure PC1 and PC2; (b) user-role assignment.

```

CREATE (:PC1 {industry1:'ITMIiot1'})
CREATE (:U1 {sname:'Thomas'})
CREATE (:U1 {sname:'John'})
CREATE (:U1 {sname:'Bob'})
CREATE (:U1 {sname:'Cathy'})
CREATE (:U1 {sname:'Peter'})
CREATE (:O1 {oname:'CollectedInfo'})
CREATE (:O1 {oname:'CollectedImages'})
CREATE (:O1 {oname:'Machine1'})
CREATE (:O1 {oname:'Machine2'})

CREATE (:PC2 {industry2:'ITMIiot2'})
CREATE (:U2 {sname:'MRailRobot'})
CREATE (:U2 {sname:'MDrone'})
CREATE (:O2 {oname:'Machine1Data'})
CREATE (:O2 {oname:'Machine2Data'})

CREATE (:UA1 {title:'Manager'});
CREATE (:UA1 {title:'Workers'});
CREATE (:UA1 {title:'Adviser'});
MATCH (er000:UA1 {title:'Manager'}) MATCH (eh000:UA1 {title:'Adviser'}) CREATE (er000)-[:has_child_content]->(eh000);
MATCH (er000:UA1 {sname:'Thomas'}) MATCH (rau000:UA1 {title:'Manager'}) CREATE (er000)-[:assigned_to]->(rau000);
MATCH (ehau10:U1 {sname:'John'}) MATCH (hau10:UA1 {title:'Adviser'}) CREATE (ehau10)-[:assigned_to]->(hau10);
MATCH (ehau20:U1 {sname:'Bob'}) MATCH (hau20:UA1 {title:'Workers'}) CREATE (ehau20)-[:assigned_to]->(hau20);
MATCH (ehau30:U1 {sname:'Cathy'}) MATCH (hau30:UA1 {title:'Workers'}) CREATE (ehau30)-[:assigned_to]->(hau30);
MATCH (ehau40:U1 {sname:'Peter'}) MATCH (hau40:UA1 {title:'Workers'}) CREATE (ehau40)-[:assigned_to]->(hau40);

CREATE (:UA2 {saname:'IoTMI1'});
MATCH (er010:U2 {sname:'MRailRobot'}) MATCH (eh010:UA2 {saname:'IoTMI1'}) MERGE (er010)-[:assigned_to]->(eh010);
CREATE (:UA2 {saname:'IoTMI2'});
MATCH (er011:U2 {sname:'MDrone'}) MATCH (eh011:UA2 {saname:'IoTMI2'}) MERGE (er011)-[:assigned_to]->(eh011);
CREATE (:OA2 {oaname:'RailwayData'});
MATCH (er010:O2 {oname:'Machine1Data'}) MATCH (eh010:OA2 {oaname:'RailwayData'}) MERGE (er010)-[:assigned_to]->(eh010);
CREATE (:OA2 {oaname:'GeolocationData'});
MATCH (er011:O2 {oname:'Machine2Data'}) MATCH (eh011:OA2 {oaname:'GeolocationData'}) MERGE (er011)-[:assigned_to]->(eh011);

```

Figure 22. Case Study 2: A sample of Cypher code (a) PC, U, and O nodes; (b) UAs of roles and Workers with U-UA assignment; and (c) a sample U/O-UA/OA assignment of PC2.

- 932 • in (b), we show object containers for physical (RailRobot and Drone) resources—
 933 where access to them is controlled via the IS—indicated in red for PC1, and logical
 934 resources (database tables at ITMI’s cloud server) with the representation of tables
 935 with some distinguished rows/columns indicated in red for PC1, and blue for PC2.
- 936 • in (c), access rights of users to perform operations are formulated through asso-
 937 ciations. For example, in (1) and (2) the workers are allowed to {o: operate, ct:
 938 control} the RailRobot and Drone objects, in (5) the manager should confirm that
 939 the inspection process is done before allowing the adviser in (7) to {r:read, cp:copy}
 940 the collected data in order to analyze it, also in (10) and (11) the subjects of PC2, the
 941 ‘Rail Robot’ and the ‘Drone’ which are assigned to IoTM1 and IoTM2 UAs, need
 942 the permission to connect the cloud server to {w:write/insert} data and images into
 943 RailwayData and GeolocationData containers.
- 944 • in (d), three prohibition relations which express ua_deny to deny all users perform-
 945 ing {w, u} on ‘IoTData’.
- 946 • in (e), four obligations that are defined as event-response relations to define con-
 947 straints under which policy state data is obligated to change. For example, the
 948 workers onsite are allowed to access the cloud server and delete the collected IoT-
 949 Data due to unexpected events, e.g., machine malfunction or system error, in order
 950 to repeat the inspection process. In (1), they are not allowed to {d} IoTData if they
 951 are logging into the system locally, if InspectionStatus=“inprogress”, and not within
 952 the start and end dates of the IRQ project.

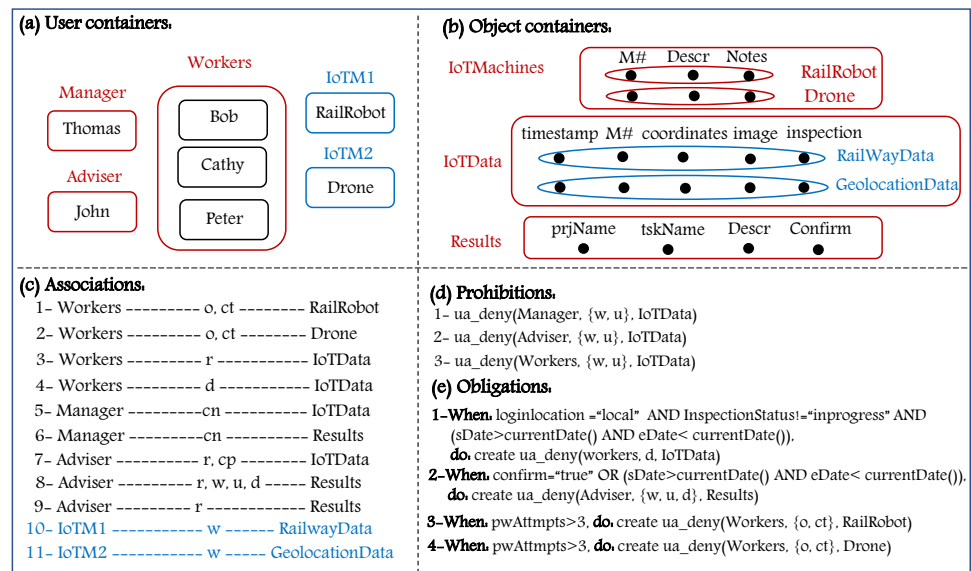
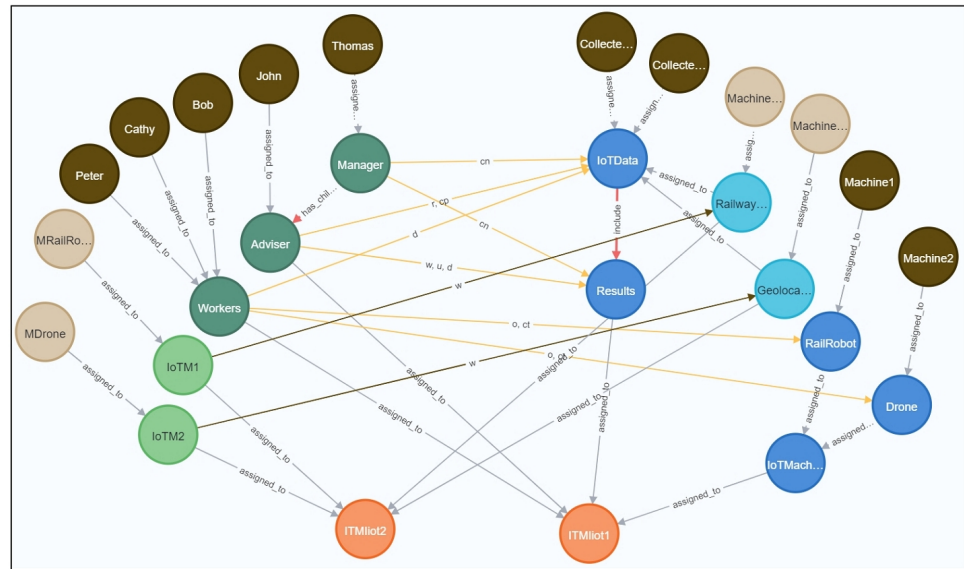


Figure 23. Case Study 2: NGAC Policy Configuration.

953 6.4. NGAC: policy enforcement

954 Similar to the steps we follow in case study 1, in this section we use Cypher
 955 statements as NGAC inputs and then have NGAC authorization responses based on
 956 them. However, the generated Cypher code is injected into the Neo4j database to
 957 represent the AC policy as NGAC graph policy. As shown in Figure 24 the dark brown,
 958 green, and blue colors refer to PC1, and the light brown, green, and blue colors refer
 959 to PC2. In the left section of the graph, the dark/light brown with dark/light green
 960 nodes represents the assignment of users (Us) to their UA containers, in addition to role
 961 hierarchy which is indicated by the red arrow and ‘has_child_content’ relationship. In the
 962 left section of the graph, the dark/light brown with dark/light blue nodes represents the
 963 assignment of objects (Os) to their OA containers, in addition to object hierarchy which
 964 is indicated by the red arrow and ‘include’ relationship. The association relationships,

965 the yellow arrows for PC1 and dark brown arrows for PC2, represent users' access rights.



966 **Figure 24.** Case Study 2: NGAC graph.

967

968 Due to the inheritance relationship between manager and adviser roles, In Figure
969 25 we show some examples of the access rights associated with the manager Thomas
970 and the users who are assigned to the Workers (UA) container. As shown in Figure 25:

- 970 • in part (a), Thomas has ManPermission to {cn: confirm} data in Results and IoTData
971 containers, also he has the permission AdvPermission to {w, u, d} Results and
972 {r, cp} IoTData through the Manager-Adviser assignment which is expressed as
973 'has_child_content' relationship to represent role hierarchy;
- 974 • in part (b), we show the access rights associated to the workers Bob, Cathy, and
975 Peter to {d} IoTData in a certain context (explained in Figure 26(a)) and {o:operate,
976 ct:control} the 'Rail Robot' and the 'Drone';
- 977 • in part (c), we show the set of permissions associated with users to perform operations
978 on the 'IoTData' object container.

979

980 In Figure 26 (a) we show an example of the association relationship properties for Work-
981 ers' permission where they are allowed to {d} IoTData—for example, due to machine
982 malfunction or system failure—if the value LoginLocation is 'public' and the Inspection-
983 Status is 'inprogress' and before the end project date. In this case study, we also express
984 the contextual and non-contextual attributes with the permission relationship properties
985 to have more accurate and regulated AC rules, since NGAC does not graphically illus-
986 trate the context/constraint entity/attributes. Similar to case study 1, to avoid multiple
987 association relationships between UA and OA containers, in Figure 24 we create a single
988 association relationship between Adviser and Results to define his access rights {w, u, d}
989 with the needed contextual and non-contextual constraints. In this case, the operation
990 {r} is not defined to avoid preventing the Adviser from reading the Results when the
991 constraints are true. In Figure 26 (b) we represent double association relationships
992 between Adviser and Results, the first association is to express the {w, u, d} operations
993 and the second association is to express the {r} operation and express the constraints
994 with each association.

994

995 Moreover, in Figure 27 we run some Cypher queries with some contextual and
996 non-contextual constraints using some context, subject, and object attributes to show
997 NGAC authorization responses to Cypher statements.

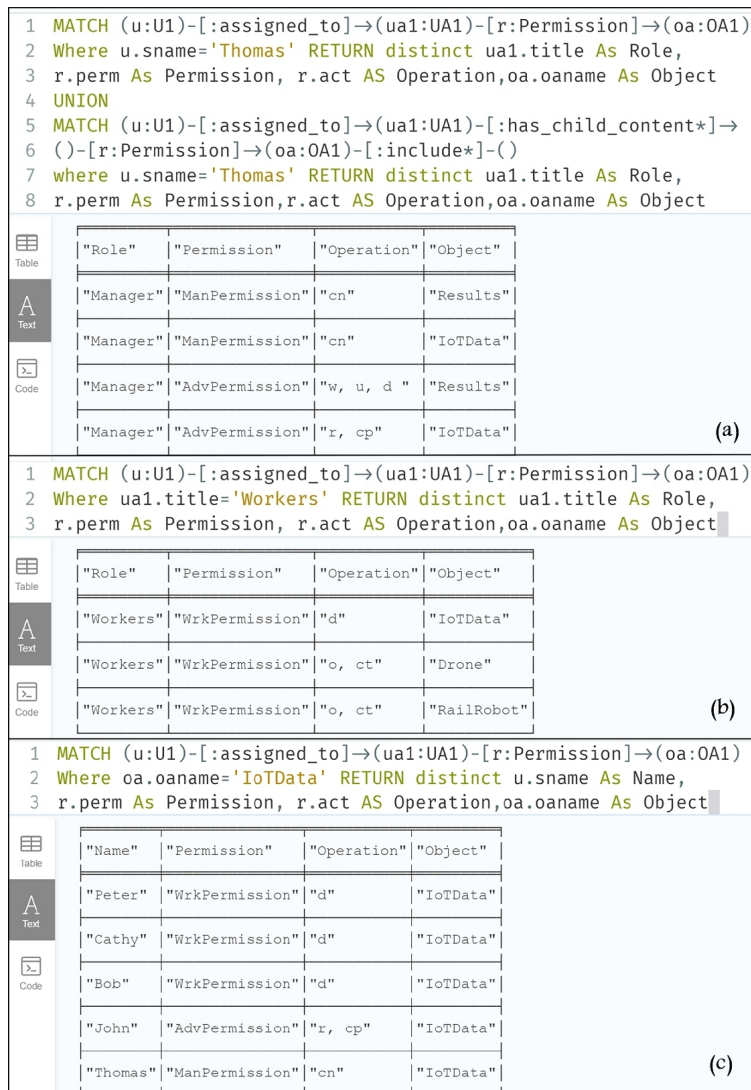


Figure 25. Case Study 2: Examples of Users' access rights.

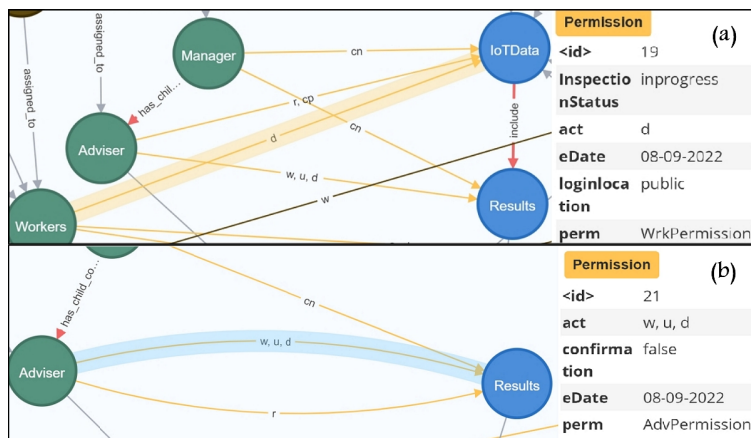


Figure 26. Case Study 2: Example of association relationship properties for Workers permission.

- 997 • In (a) we show that the Workers are able to {d} IoTData when the value of InspectionStatus='inprogress', which means the inspection process is not completed and
 998 more data need to be collected, but when the value of InspectionStatus is updated
 999 (by the manager) to 'complete' as shown in (b), the workers would not be able to
 1000 delete the IoTData even if they are logging into the system via public network as
 1001 shown in (c).
 1002
 1003 • In Figure 27 (d) we show that a Worker would be able to {o, ct} the Rail Robot or
 1004 the Drone if the pwAttmpmts are less than or equal three attempts, otherwise (e) he
 1005 would not.

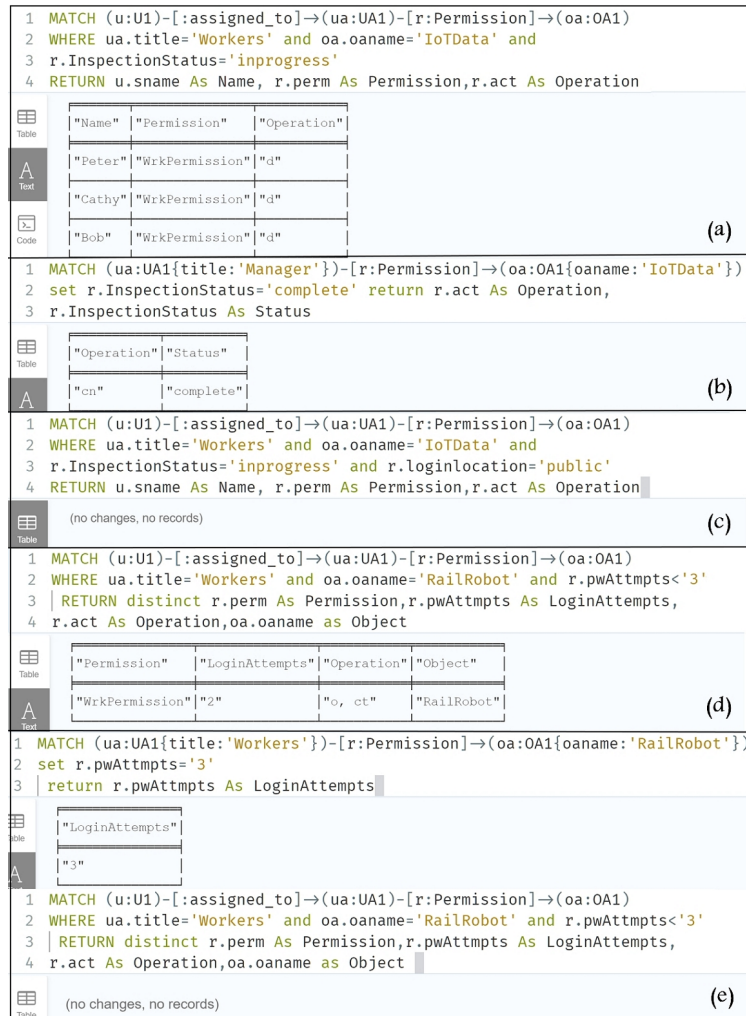


Figure 27. Case Study 2: Examples of NGAC authorization responses to Cypher statements.

1006 7. HEAD administrative panel

1007 In this section, we provide another example for the implementation of HEAD
 1008 metamodel using VB.net and SQL, for decision-making to access industrial resources
 1009 (or any other resources in any computing environment) according to components of
 1010 the derived AC model(s). For example, in case study 1 (section 5), the AC decision
 1011 occurs according to the user's role/group, the contextual information (location, time,
 1012 etc.), and the attributes of subjects/objects, and some constraints. In Figure 28, we show
 1013 an administrative panel example where the AC model can be derived after creating
 1014 E_x , AU, PU, and S_t entities/attributes. In Figure 29 we show the steps to create model
 1015 components:

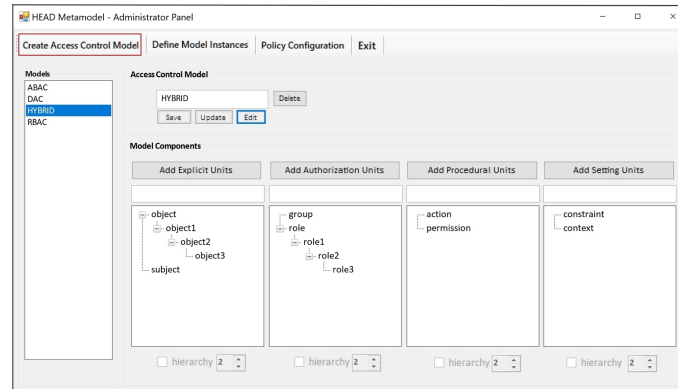


Figure 28. HEAD metamodel: Administrative Panel example

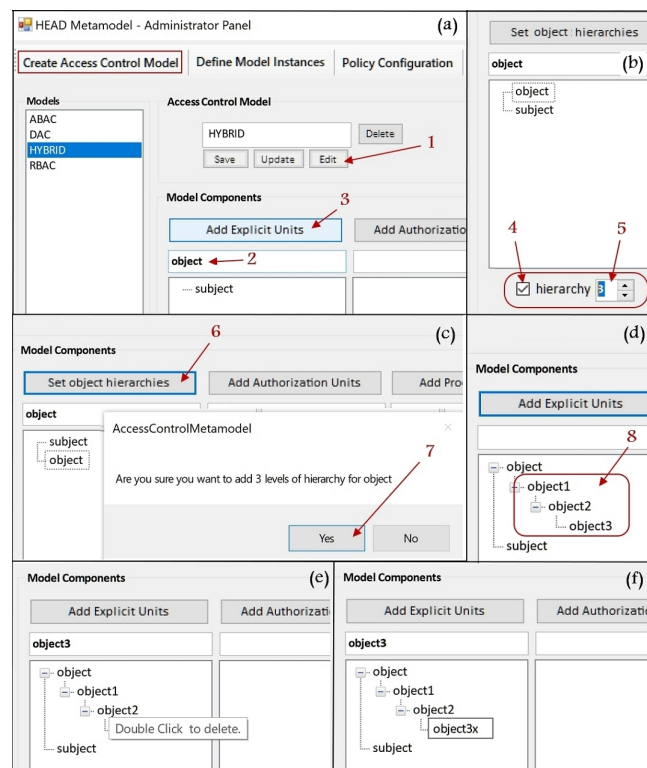


Figure 29. HEAD Administrative Panel: Instantiation of AC model

- 1016 • in (a), after defining the needed model (1) the administrator would be able to edit
 1017 the model in order to update, delete, or (2) create and (3) add model entities.
 1018 • in (b), the needed entity (e.g., object) is (4) selected to (5) define the number of
 1019 hierarchy levels.
 1020 • in (c), the administrator (6) sets and (7) confirms the hierarchical levels of an entity.
 1021 • in (d), we show that the root node object (8) has three levels of the object hierarchy.
 1022 • in (e) and (f), the nodes can be deleted and their names can be updated.

1023 After specifying the model entities, in Figure 30 we show another form of the panel to
 1024 define the model entities/attributes. The left part shows the defined model entities (and
 1025 the hierarchies), the right part shows the created instances of policy elements based on
 1026 model entities, and in between, we create the entities and summarize the attributes for
 1027 each selected element of the right part (e.g., context). Note that, by default, each entity

1028 has a name attribute and later on we explain how additional attributes can be added for
 1029 each entity. In Figure 31 we explain the AC policy configuration steps:

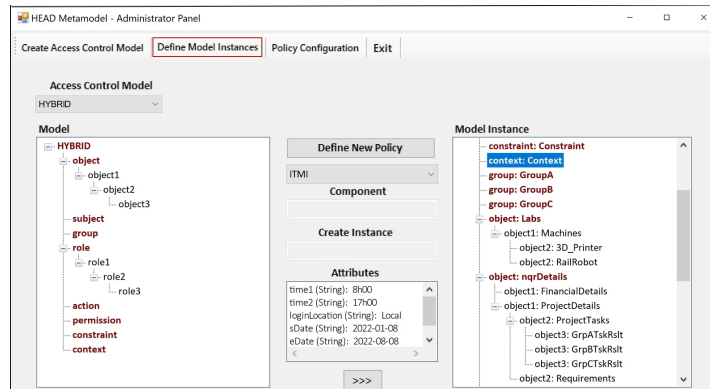


Figure 30. HEAD Administrative Panel: AC policy configuration

- 1029
- 1030 • in (a)(1) we configure a hybrid model for ITMI. To create model instances, for
- 1031 example, (2) for object entities (at root level) we write the objects names (3) separated
- 1032 by ‘;’ instead of defining each object individually, then (4) we confirm the creation.
- 1033 All other elements of subjects, roles, etc. are created in the same way.
- 1034 • In (b), we show how a child node is created for (5) the Director, (6) named Manager.

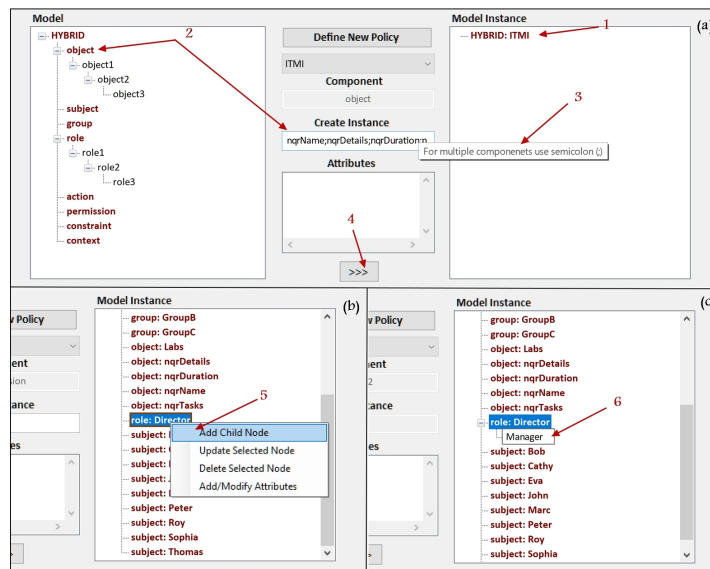


Figure 31. HEAD Administrative Panel: AC policy configuration steps

1035 In Figure 32, we show how additional attributes can be created by selecting and right-
 1036 clicking on a specific element, then choosing ‘Add/Modify Attributes’ where a popup
 1037 window opens to define and create various types of attributes (with their values), in our
 1038 example, we define the context attributes to specify if the user needs to access resources
 1039 within the business hours, the user’s login location, and if the current date is within the
 1040 start/end date of the project. Moreover, the selected node can be updated, or deleted.

1041 To configure the AC rules, in Figure 33 we show the configuration steps. Note that,
 1042 different models might be implemented for an organization, hence in Figure 33(a):

- 1043 (1) the policy with model type is selected to configure the needed rules.
- 1044 (2) the administrator assigns E_x to AUs, in our example, (2) subjects to roles/groups.
- 1045 (3) the administrator associates AUs with PUs, in our example, the role permission.

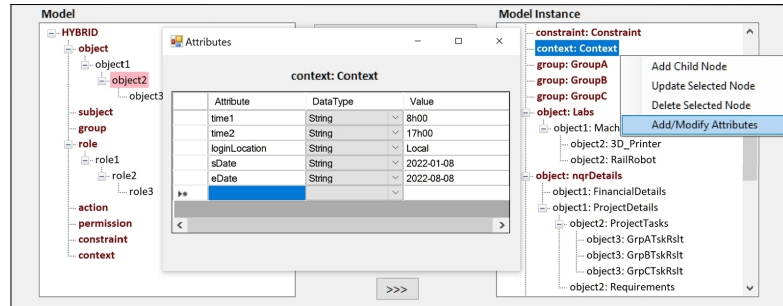


Figure 32. HEAD Administrative Panel: Adding Attributes to Model Components

- 1046 After assigning subjects to roles/groups and specifying their permission(s), in
 1047 Figure 33(b):
 1048 (1) the administrator, based on the permission type, needs to select the objects,
 1049 (2) and associate the actions that can be performed on each of the selected objects
 1050 (FinancialDetails and ProjectDetails).
 1051 (3) to apply constraints, the administrator should select the row (permission, object,
 1052 and action), and selects the needed attributes to include them with the rule.
 1053 (4) Finally, the rules can be exported as Cypher, json, or any other format of code. In
 1054 this example, we also generate Cypher code as explained in section 5.

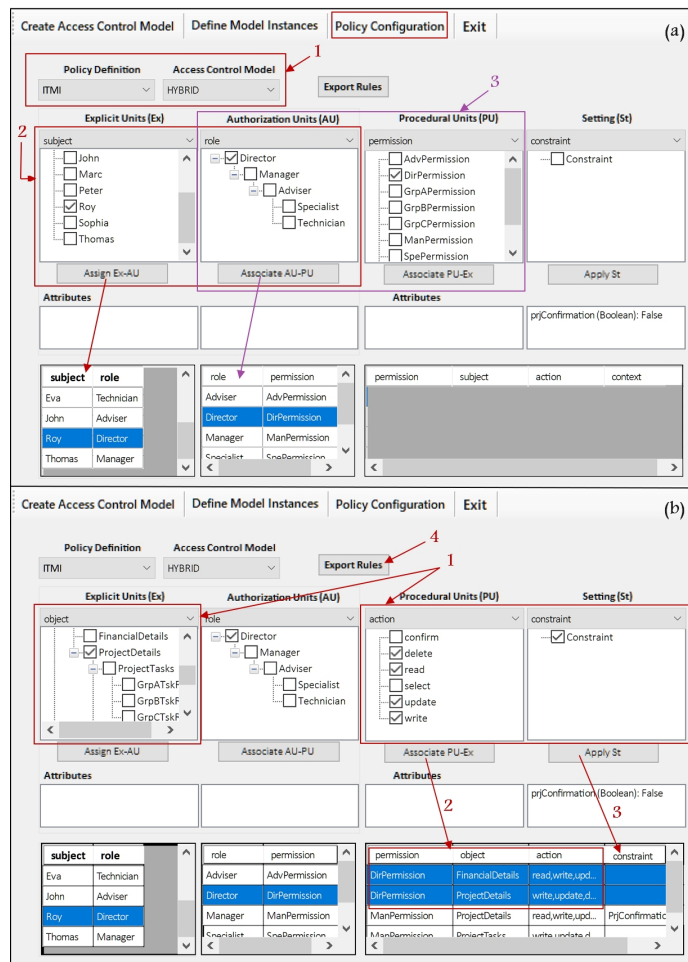


Figure 33. HEAD Administrative Panel: Formulation of AC Rules

1055 8. Evaluation and Validation of HEAD metamodel

1056 After presenting the above two case studies and showing how it is possible to
 1057 use HEAD metamodel to derive AC models, and how it can be adapted in different
 1058 computing environments. In this section, we present the comparison and the evaluation
 1059 and validation of the HEAD metamodel with the proposed AC metamodels in the
 1060 literature.

1061 8.1. Comparison

1062 In Table 1 we summarize the features of the HEAD metamodel compared to the
 1063 other proposed metamodels in the literature [7–9].

Table 1. Comparison between HEAD Metamodel and the other AC metamodels.

Metamodel	Access Control Metamodels	
Features	HEAD Metamodel	Other Metamodels
Unify components	Unify all the heterogeneous components of heterogeneous AC models.	Some metamodels unify some of the heterogeneous components under the notion of ‘category’ which includes roles, groups, security levels, etc.
Generality	Include all features and components of common AC models and allow deriving various instances of various models.	Hybrid structures to derive some AC models rather than generic metamodels
Dynamism	Allows defining and adding any type of components and attributes for existing models and non-existing ones to express various static and dynamic rules	None of the existing metamodels support this feature, and they are not dynamic enough to define static and dynamic AC policies.
Extensibility	New components can be defined and integrated with already derived models to support new AC features in addition to the previous ones.	Some metamodels are extended but not extensible, and none of the existing metamodels support this feature.
Hierarchical	It allows defining multi-levels of all components (e.g., role, context) to conform to the hierarchical organizational structures.	Some metamodels support the feature of hierarchy for some components, and none of them consider the context hierarchy which is an important feature in complex and highly dynamic environments.
Upgradability	Able to follow technology upgrades and update any policy.	None of the existing metamodels support this feature, and they have reached their limits.
Unified framework	Allows the creation of any model, in addition to any hybrid model with different policy classes (e.g., case study 2), and hybrid models with hybrid components (e.g., case study 1)	Allows the creation of some models based on features employed in their hybrid structures.
Adaptability	Can be implemented in different centralized and distributed environments, especially IoT.	They provide solutions for specific cases and scenarios.
Novelty	A new development in the domain with advanced features (all of the above).	The last AC metamodel was proposed in 2015 [23].

1064 To sum up, the derived models of HEAD metamodel are flexible and upgradable;
 1065 they can be extended, updated, and easily define any new component/attribute to
 1066 follow the technology upgrades and define and enforce larger sets of static and dynamic
 1067 policies. Compared to the existing metamodels, system designers and security experts
 1068 need to redesign a model and rethink how it could be enhanced instead of being able to
 1069 extend/upgrade it, which means additional time and cost.

1070 8.2. Evaluation and Validation

1071 The main value of HEAD metamodel is that it contributes to several essential
 1072 advancements in the domain compared to the proposed metamodels in the literature.
 1073 The advancements can be identified as follows:

- 1074 • It allows the specification, the formalization, the generation, and the verification of
 1075 AC policies. HEAD metamodel draws a complete strategy instead of tackling each
 1076 issue independently. In our experience so far, AC research and practice focus on
 1077 one phase and confuses issues that cut across multiple phases.

- 1078 – As for the specification, HEAD metamodel is flexible enough to allow identifying the E_x , I_m , and S_t entities (with their attributes) based on the AC requirements of a system. Using HEAD metamodel, system administrators and security experts are not restricted to defining some entities for some models, they are able to create any entity (and attribute) for any model whether it is an existing AC model or a non-existing model.
 - 1082 – For the formalization, after specifying the needed AC entities, the DSL language of HEAD metamodel allows formalizing the specified AC models (common AC models, hybrid models, and other models) with the ability to follow the technology upgrades by allowing the definition of new entities and attributes. The power of the DSL language of the HEAD metamodel is simple and flexible to appropriately express any AC policy requirements, it overcomes the complication of existing language expressions, also it is independent of specific AC models.
 - 1092 – For the generation of AC policies, the nature of meta-components of HEAD metamodel which allows deriving any model, can also be adapted to represent the concrete instance of any AC model and generate the needed AC policies. In this paper, we use Eclipse Xtend notation to represent the concrete instances of the derived models, then the AC rules are expressed and generated as a format of Cypher queries. Another generator could be used, for example, to represent the needed models and then generate AC rules in as 'json' format.
 - 1099 – For the formal verification of the generated AC policies is to formally verify the accuracy and the coherence of the concrete instance of the AC policy before policy enforcement. For example, using Cypher queries as NGAC inputs to represent the AC rules of a system in a graph, then verify the objects, the relationships between them, and the subjects that interact with the system in a way that adheres to the semantics of an organization.
 - 1105 • It serves as a unifying framework and it is the only metamodel that provides to the literature all the design phases and steps starting from the conception [1,6,24–26] to policy enforcement which is addressed in this paper, through the implementation of two case studies, in addition to the new research opportunities this metamodel opens in the domain [9].
 - 1110 • The effectiveness of HEAD metamodel is reflected in the proof of the concept since we show that the metamodel idea with the theoretical foundations can be implemented and applied using different tools.
 - 1113 • If the above case studies are implemented using one of the proposed metamodels in the literature, the solution would be insufficient to answer ITMI's AC requirements, since the existing metamodels only include features of DAC, MAC, and RBAC models, and neither attributes can be defined nor new components can be added. As well, they are not flexible to allow defining different PCs.
 - 1118 • HEAD metamodel is an essential development in the field since the last proposed metamodel was in 2015 [9], and it is a hybrid metamodel only includes the features of DAC, MAC, and RBAC, not dynamic, does not support the hierarchy of all components, and the derived models cannot be extended.
- 1122 All of the above reflect the advancement of HEAD metamodel over the other proposed metamodels in the literature, and ensure that it is can be considered as a centerpiece towards enhancing other essential characteristics, and elaborating in different research directions in the domain (explained in [9]).

1126 9. Conclusion and Future Perspectives

1127 Despite over the past decade security researchers have proposed a variety of policy models and metamodels to address real-world security problems, the limited ability of the existing AC methods to generically specify, upgrade, and enforce policy persists. 1128 Moreover, the need for secure information sharing has dramatically increased with the 1129 1130

1131 explosion of the IoT and industry 4.0, with the digital transformation, where resources
1132 are highly distributed and need to be accessed from everywhere, anyhow, and at any
1133 time. As well, the evolution of pervasive ISs and intelligent manufacturing has had an
1134 extensive impact on different directions, such as the future of the industry. In smart
1135 industries, several physical and cyber technologies are combined to improve productivity,
1136 performance, quality, management, etc. In this context, controlling access to protect
1137 resources from unauthorized use is a complicated and challenging task, especially
1138 with the presence of cybercriminals and cyberattacks. All this has motivated us to
1139 design and implement a new and advanced AC metamodel, named HEAD metamodel,
1140 that addresses the limitations of the existing metamodels—for example, generality, the
1141 hierarchy of components, dynamism, and extensibility of AC models—and works as a
1142 base to develop other essential features—for example policy migration, and collaboration
1143 and interoperability between AC models.

1144 The AC policy is concerned with what AC rules need to be enforced, while the AC
1145 mechanism is concerned with how AC rules are being enforced. In this paper, we present
1146 two case studies inspired by ITMI's, local (non-IoT) and IoT, computing environments
1147 to show that our metamodel can be adapted in different computing environments and
1148 various AC models can be instantiated with the needed components/attributes to fit the
1149 AC requirements of an organization (or industry sector). For the AC policy, we use the
1150 DSL of HEAD metamodel to derive the needed model for each case study, then we use
1151 xtend notation to represent the concert instance of the derived model. The AC rules are
1152 generated as Cypher queries which are then injected into Neo4j to represent the NGAC
1153 policy as a graph. NGAC framework is used as an enforcement point for the generated
1154 rules of each case study. The results show that HEAD metamodel:

- 1155 • overcomes the limitations of the existing AC metamodels and could be considered
1156 as a base to develop other essential features.
- 1157 • is able to serve as a unifying framework and encompass the heterogeneity of the
1158 existing models.
- 1159 • can be adapted and integrated with various local and distributed computing envi-
1160 ronments.
- 1161 • is able to answer the current AC requirements and follow the needed policy up-
1162 grades.
- 1163 • can be used to generate the needed format of AC rules (e.g., Cypher code, json...).

1164 Besides, the HEAD metamodel reveals some limitations of the NGAC framework. NGAC
1165 theoretically considers the obligations and prohibitions which express the context and
1166 constraint elements, but practically it does not include them among the basic elements of
1167 the NGAC graph. Hence, during implementation, this would prevent having accurate
1168 access control decisions. As described in the case studies, we have to create multiple
1169 association relationships between the UA node and OA node if some operations need to
1170 be performed without constraints by UA, and some other operations can be performed
1171 if some contextual (or non-contextual) constraints are true/false depends on the defined
1172 rule.

1173 Policies with minimum quality may lead to unacceptable situations and decisions,
1174 for example preventing users from accessing resources they are allowed to, or allowing
1175 some users to access resources they are not allowed to. As future perspectives, we
1176 aim to develop the needed tools to analyze and assess the obtained policies at the run-
1177 time before enforcing them in order to avoid uncertainties concerning the obtained AC
1178 decision. Policy analysis and assessment are for assuring the quality of the generated
1179 AC policies and making sure that they are consistent, relevant, minimal, complete,
1180 and correct with respect to the required actions by subjects on some objects. This
1181 process is of major importance while implementing AC policies in highly dynamic and
1182 heterogeneous environments, especially IoT. Moreover, we also aim to extend HEAD
1183 metamodel to support additional features and services, for example, developing the
1184 needed algorithms and tools to migrate AC policies from one model to another, and for

1185 collaboration and interoperability between different AC models. Moreover, extending
1186 the metamodel to support packages of predefined AC models (e.g., common models) to
1187 minimize administrative efforts and technical implementation to specify models and
1188 define policies. Additionally, one of our future perspectives is to extend the NGAC
1189 framework to adapt context and constraint in addition to its policy elements.

1190 **Author Contributions:** Conceptualization, N.K. and M.A.; methodology, N.K., M.A. and H.I.;
1191 software, N.K.; validation, N.K., M.A., H.I., J.M. and T.D.; formal analysis, N.K. and M.A.;
1192 investigation, N.K., M.A., H.I., J.M. and T.D.; resources, N.K., M.A., H.I., J.M. and T.D.; data
1193 curation, N.K. and J.M.; writing—original draft preparation, N.K.; writing—review and editing,
1194 N.K., M.A. and H.I.; visualization, N.K., M.A. and H.I.; supervision, M.A. and H.I.; project
1195 administration, M.A. and H.I. All authors have read and agreed to the published version of the
1196 manuscript.

1197 **Funding:** This research was funded by the Natural Sciences and Engineering Research Council of
1198 Canada (NSERC), grant number 06351.

1199 **Institutional Review Board Statement:** Not applicable.

1200 **Informed Consent Statement:** Not applicable.

1201 **Data Availability Statement:** The study did not report any data.

1202 **Acknowledgments:** We acknowledge the support of Fonds Québécois de la Recherche sur la
1203 Nature et les Technologies (FRQNT), Réseau Québécois sur l'Énergie Intelligente (RQEI), and
1204 Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

1205 **Conflicts of Interest:** The authors declare no conflict of interest.

1206 References

- 1207 1. Kashmar, N.; Adda, M.; Ibrahim, H. HEAD Metamodel: Hierarchical, Extensible, Advanced,
1208 and Dynamic Access Control Metamodel for Dynamic and Heterogeneous Structures. *Sensors*
1209 **2021**, *21*, 6507.
- 1210 2. Jaïdi, F.; Labbene Ayachi, F.; Bouhoula, A. A methodology and toolkit for deploying reliable
1211 security policies in critical infrastructures. *Security and Communication Networks* **2018**, *2018*.
- 1212 3. Mishra, A.; Alzoubi, Y.I.; Gill, A.Q.; Anwar, M.J. Cybersecurity Enterprises Policies: A
1213 Comparative Study. *Sensors* **2022**, *22*. doi:10.3390/s22020538.
- 1214 4. Antunes, M.; Maximiano, M.; Gomes, R.; Pinto, D. Information Security and Cybersecurity
1215 Management: A Case Study with SMEs in Portugal. *Journal of Cybersecurity and Privacy* **2021**,
1216 *1*, 219–238.
- 1217 5. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Access Control in Cybersecurity and Social
1218 Media. *Cybersécurité et médias sociaux* **2021**.
- 1219 6. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Access control metamodel for policy
1220 specification and enforcement: From conception to formalization. *Procedia Computer Science*
1221 **2021**, *184*, 887–892.
- 1222 7. Kashmar, N.; Adda, M.; Ibrahim, H. Access Control Metamodels: Review, Critical Analysis,
1223 and Research Issues. *J. Ubiquitous Syst. Pervasive Netw* **2021**, *3*.
- 1224 8. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. A review of access control metamodels.
1225 *Procedia Computer Science* **2021**, *184*, 445–452.
- 1226 9. Kashmar, N.; Adda, M.; Ibrahim, H. HEAD Access Control Metamodel: Distinct Design,
1227 Advanced Features, and New Opportunities. *Journal of Cybersecurity and Privacy* **2022**,
1228 *2*, 42–64. doi:10.3390/jcp2010004.
- 1229 10. Kashmar, N.; Adda, M.; Atieh, M. From Access Control Models to Access Control Metamodels:
1230 A Survey. Future of Information and Communication Conference. Springer, 2019, pp.
1231 892–911.
- 1232 11. Ferraiolo, D.; Chandramouli, R.; Kuhn, R.; Hu, V. Extensible access control markup language
1233 (XACML) and next generation access control (NGAC). Proceedings of the 2016 ACM
1234 International Workshop on Attribute Based Access Control, 2016, pp. 13–24.
- 1235 12. Ray, I.; Alangot, B.; Nair, S.; Achuthan, K. Using attribute-based access control for remote
1236 healthcare monitoring. 2017 Fourth International Conference on Software Defined Systems
1237 (SDS). IEEE, 2017, pp. 137–142.

- 1238 13. Ferraiolo, D.; Gavrila, S.; Katwala, G.; Roberts, J. Imposing Fine-Grain Next Generation
1239 Access Control over Database Queries. Proceedings of the 2nd ACM Workshop on Attribute-
1240 Based Access Control; Association for Computing Machinery: New York, NY, USA, 2017;
1241 ABAC '17, p. 9–15. doi:10.1145/3041048.3041050.
- 1242 14. Antunes, M.; Maximiano, M.; Gomes, R. A Client-Centered Information Security and
1243 Cybersecurity Auditing Framework. *Applied Sciences* **2022**, *12*. doi:10.3390/app12094102.
- 1244 15. Quader, F.; Janeja, V.P. Insights into Organizational Security Readiness: Lessons Learned
1245 from Cyber-Attack Case Studies. *Journal of Cybersecurity and Privacy* **2021**, *1*, 638–659. doi:
1246 10.3390/jcp1040032.
- 1247 16. Desmedt, Y.; Shaghghi, A. Function-Based Access Control (FBAC): Towards Preventing
1248 Insider Threats in Organizations. In *From Database to Cyber Security*; Springer, 2018; pp.
1249 143–165.
- 1250 17. Qi, S.; Zheng, Y.; Li, M.; Liu, Y.; Qiu, J. Scalable industry data access control in RFID-enabled
1251 supply chain. *IEEE/ACM Transactions on Networking* **2016**, *24*, 3551–3564.
- 1252 18. Ruland, C.; Sassmannshausen, J. Access control in safety critical environments. 2018 12th
1253 International Conference on Reliability, Maintainability, and Safety (ICRMS). IEEE, 2018, pp.
1254 223–229.
- 1255 19. Erdödi, L.; Ulltveit-Moe, N.; Nergaard, H.; Gj, T.; Kolstad, E.; others. Secure information
1256 sharing in an industrial internet of things. In *arXiv*; Cornell University, 2016; pp. 1–12.
- 1257 20. Alagar, V.; Alsaig, A.; Ormandjiva, O.; Wan, K. Context-based security and privacy for
1258 healthcare IoT. 2018 IEEE International Conference on Smart Internet of Things (SmartIoT).
1259 IEEE, 2018, pp. 122–128.
- 1260 21. Ahamed, J.; Khan, F. An enhanced context-aware capability-based access control model for
1261 the internet of things in healthcare. 2019 Sixth HCT Information Technology Trends (ITT).
1262 IEEE, 2019, pp. 126–131.
- 1263 22. Mrabet, H.; Alhomoud, A.; Jemai, A.; Trentesaux, D. A Secured Industrial Internet-of-Things
1264 Architecture Based on Blockchain Technology and Machine Learning for Sensor Access Con-
1265 trol Systems in Smart Manufacturing. *Applied Sciences* **2022**, *12*. doi:10.3390/app12094641.
- 1266 23. Abd-Ali, J.; El Guemhioui, K.; Logrippo, L. A Metamodel for Hybrid Access Control Policies.
1267 *JSW* **2015**, *10*, 784–797. doi:10.17706/jsw.10.7.784-797.
- 1268 24. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. A new dynamic smart-AC model methodol-
1269 ogy to enforce access control policy in IoT layers. 2019 IEEE/ACM 1st International Work-
1270 shop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT).
1271 IEEE, 2019, pp. 21–24.
- 1272 25. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Smart-ac: A new framework concept for
1273 modeling access control policy. *Procedia Computer Science* **2019**, *155*, 417–424.
- 1274 26. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Deriving access control models based on
1275 generic and dynamic metamodel architecture: Industrial use case. *Procedia Computer Science*
1276 **2020**, *177*, 162–169.

CHAPTER 5

HEAD ACCESS CONTROL METAMODEL: DISTINCT DESIGN, ADVANCED FEATURES, AND NEW OPPORTUNITIES

Published in Journal of Cybersecurity and Privacy, 2022; Volume-2(1), pp. 42-64;
<https://doi.org/10.3390/jcp2010004>

Abstract: The revolution of industry 4.0, the adoption of telework due to COVID-19 pandemic, the digital transformation, and many other facts bring out new trends, concepts, and challenges to integrate more advanced and intelligent systems in critical and heterogeneous structures. All this have prompted a greater need than ever for access control. To minimize security risks of any unauthorized access to resources, several AC approaches are proposed to find a common specification for security policy. Unfortunately, the proposed AC models and metamodels have limited features and are insufficient to meet the current AC requirements. In this chapter, we present a novel approach in five main phases based on HEAD metamodel: metamodel development, deriving models, generating policies, policy analysis and assessment, and policy enforcement. We describe the achieved and the remaining steps and how they can be employed to develop more advanced features in order to open new opportunities and answer the various challenges of technology progression, and the impact of the pandemic in the domain. The aim of this is to draw a complete strategy instead of tackling each issue separately, and to explain and clarify the functionality of all phases to close the enormous gaps between them. This approach can be employed to assist security experts and system administrators to design secure systems that comply with the organizational security policies that are related to access control.

Résumé: La révolution de l'industrie 4.0, l'adoption du télétravail en raison de la pandémie de COVID-19, la transformation numérique et bien d'autres faits font ressortir de nouvelles tendances, concepts et défis pour intégrer des systèmes plus avancés et intelligents dans des structures critiques et hétérogènes. Tout cela a suscité un besoin plus important que jamais de contrôle d'accès. Pour minimiser les risques de sécurité de tout accès non autorisé aux ressources, plusieurs approches CA sont proposées pour trouver une spécification commune pour la politique de sécurité. Malheureusement, les modèles et métamodèles CA proposés ont des fonctionnalités limitées et sont insuffisants pour répondre aux exigences CA actuelles. Dans ce chapitre, nous présentons une nouvelle approche en cinq phases principales basée sur le métamodèle HEAD : développement du métamodèle, dérivation de modèles, génération de politiques, analyse et évaluation des politiques, et application des politiques. Nous décrivons les étapes réalisées et restantes et comment elles peuvent être utilisées pour développer des fonctionnalités plus avancées afin d'ouvrir de nouvelles opportunités et de répondre aux différents défis des progression technologiques et de l'impact de la pandémie dans le domaine. Le but est de dessiner une stratégie complète au lieu d'aborder chaque problème séparément, et d'expliquer et de clarifier la fonctionnalité de toutes les phases pour combler les énormes écarts entre elles. Cette approche peut être utilisée pour aider les experts en sécurité et les administrateurs système à concevoir des systèmes sécurisés conformes aux politiques de sécurité organisationnelles liées au contrôle d'accès.

Article

HEAD Access Control Metamodel: Distinct Design, Advanced Features, and New Opportunities

Nadine Kashmar ^{1,*}, Mehdi Adda ¹ and Hussein Ibrahim ²

¹ Département de Mathématiques, Informatique et Génie, Université du Québec à Rimouski, 300 Allée des Ursulines, Rimouski, QC G5L 3A1, Canada; mehdi_adda@uqar.ca

² Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC G4R 5B7, Canada; hussein.ibrahim@itmi.ca

* Correspondence: nadine.kashmar@uqar.ca

Abstract: Access control (AC) policies are a set of rules administering decisions in systems and they are increasingly used for implementing flexible and adaptive systems to control access in today's internet services, networks, security systems, and others. The emergence of the current generation of networking environments, with digital transformation, such as the internet of things (IoT), fog computing, cloud computing, etc., with their different applications, bring out new trends, concepts, and challenges to integrate more advanced and intelligent systems in critical and heterogeneous structures. This fact, in addition to the COVID-19 pandemic, has prompted a greater need than ever for AC due to widespread telework and the need to access resources and data related to critical domains such as government, healthcare, industry, and others, and any successful cyber or physical attack can disrupt operations or even decline critical services to society. Moreover, various declarations have announced that the world of AC is changing fast, and the pandemic made AC feel more essential than in the past. To minimize security risks of any unauthorized access to physical and logical systems, before and during the pandemic, several AC approaches are proposed to find a common specification for security policy where AC is implemented in various dynamic and heterogeneous computing environments. Unfortunately, the proposed AC models and metamodels have limited features and are insufficient to meet the current access control requirements. In this context, we have developed a Hierarchical, Extensible, Advanced, and Dynamic (HEAD) AC metamodel with substantial features that is able to encompass the heterogeneity of AC models, overcome the existing limitations of the proposed AC metamodels, and follow the various technology progressions. In this paper, we explain the distinct design of the HEAD metamodel, starting from the metamodel development phase and reaching to the policy enforcement phase. We describe the remaining steps and how they can be employed to develop more advanced features in order to open new opportunities and answer the various challenges of technology progressions and the impact of the pandemic in the domain. As a result, we present a novel approach in five main phases: metamodel development, deriving models, generating policies, policy analysis and assessment, and policy enforcement. This approach can be employed to assist security experts and system administrators to design secure systems that comply with the organizational security policies that are related to access control.

Keywords: access control; metamodel; heterogeneous systems; security and privacy; IoT; industry 4.0; COVID-19; digital transformation



Citation: Kashmar, N.; Adda, M.; Ibrahim, H. HEAD Access Control Metamodel: Distinct Design, Advanced Features, and New Opportunities. *J. Cybersecur. Priv.* **2022**, *2*, 42–64. <https://doi.org/10.3390/jcp2010004>

Academic Editor: Danda B. Rawat

Received: 21 December 2021

Accepted: 8 February 2022

Published: 14 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Security and privacy in general and the protection of logical and physical resources from unauthorized access, in particular, are critical and essential issues for today's organizations. Their resources are accessed via various types of private and public networks; this depends on the type of service(s) their information systems (IS) provide. The new generation of networking environments with digital transformation such as the internet

of things (IoT), fog computing, cloud computing, etc., is emerging and has various applications related to several, critical domains. This networking generation has several interconnections by means of multiple layers of public and private networks constituting critical cyber-physical infrastructures where an unlimited number of resources are distributed and accessed from everywhere, anytime, and anyhow. With this fact, controlling access is a critical objective, especially with the presence of cyber-criminals and cyber-attacks [1], and a big challenge for deploying secure information ISs. For this purpose, several AC models have been implemented to control what resources can be accessed by users, when, and how after checking a predefined set of AC rules. What makes this process even more challenging is the orientation of the world toward the concept of telework due to the COVID-19 pandemic and the urgent need to access sensitive and vital resources related to various domains [2].

With the evolution of technology trends, it is realized that controlling users' access and the operations they perform on information cannot be overlooked when developing approaches related to information security. Moreover, the existing AC models have reached their limits and are insufficient to answer the increasing demand for security and privacy standards [3,4]. Common AC models that are implemented to control users' access to resources are discretionary access control (DAC), mandatory access control (MAC), role-based access control (RBAC), and attribute-based access control (ABAC) models [4–7]. Unfortunately, two main issues prevent an IS from following up the evolution of security threats. First, the focus is on defining the functional IS requirements, while the non-functional requirements such as authentication and authorization are almost barely handled at the termination of the development process. Second, the existing AC metamodels, which are proposed to serve as unifying frameworks to include the heterogeneous AC models, only provide support of defining some AC components for some models which prevent defining a larger set of organizational AC policies or upgrading the defined ones [4,6,8]. Hence, one of the main steps to solve the first issue is to tackle the second one, in other words, while developing an IS it is important to set up trusted security policies and provide the essential tools to define all the needed AC components. This, in turn, would allow system administrators to set up and manage the defined AC policies in an organization.

Security solutions must be manageable and adaptable to track the evolution of security threats which accompany technology upgrades. Along with technology progressions, various research works have been conducted in this domain (in Section 2, we summarize their development stages). Unfortunately, common AC models have various limitations; the hybrid models (Section 3.2.1), the extended models (Section 3.2.2), and the abstracted models (Section 3.2.3) cannot answer all the needed AC requirements; moreover, the existing AC metamodels have several shortcomings (explained in Section 4) and are not able to follow technology progressions [5,6,9]. To address this issue, we have developed a Hierarchical, Extensible, Advanced, and Dynamic (HEAD) AC metamodel with substantial and advanced features [3] compared to the existing AC metamodels [5,9]. It is generic and able to include all features of common models (and any other model), dynamic and able to create the needed components/attributes to express a larger set of static and dynamic AC rules, extensible since the already defined models can be extended, and it supports the feature of hierarchy for any component.

In addition to all the above, the emergence of COVID-19 as a global pandemic has had a huge impact on business and work strategies; moreover, it put the spotlight on access control. It forced a greater requirement than ever for access control, due to widespread telework and the need to access resources and data related to critical domains such as government, healthcare, industry, and others. As stated in Security Distributing and Marketing (SDM) magazine [10], "The importance of controlling who has access to specific areas of a facility and the knowledge of who actually accessed specific areas became much more important" due to the pandemic.

All the aforementioned issues force the need to implement more robust, coherent, and advanced AC models, which are flexible, dynamic, and able to meet the AC requirements

and follow technology upgrades. In this paper, we present the new opportunities and the various research directions which would enhance our HEAD AC metamodel to meet the expected AC requirements with the presence of various technology trends. The characteristics of the HEAD metamodel open various research directions in the domain, for example, its unifying structure would facilitate developing many other essential features (e.g., collaboration and interoperability between various models and migration of AC policies), developing the necessary tools for policy analysis and assessment, integrating artificial intelligence techniques to support the necessary intelligent services with the current era of technologies, and many other directions. HEAD metamodel characteristics can be summarized as follows [3]:

- Unify the heterogeneous concepts of policy models.
- A new generic metamodel that is able to include the heterogeneity of the existing models and the proposed metamodels.
- Dynamic with the ability to create any needed component/attribute and the relationships between them.
- Extensible metamodel where any derived AC model can be extended and upgraded.
- Unlimited levels of components hierarchy.
- Allow instantiating non-existing AC models.

However, the contribution of this paper can be summarized as follows:

- Provides a complete plan for a distinct AC framework starting from the metamodel development phase and reaching to the policy enforcement phase.
- Opens new opportunities to develop and enhance the necessary services to answer the challenging AC requirements of today's computing environments based on distinct design and advanced AC metamodel.
- Assists developers and security experts to include unified and generic components in designing secure ISs that conform to the organizational AC security policies.

The remainder of this paper is organized as follows. In Section 2, we present the AC challenges of today's computing environments. The development stages of AC methods starting from common models reaching to AC metamodels, which is a recent research topic in the domain, are summarized in Section 3. In Section 4, we illustrate and explain the issues and limitations of the existing AC metamodels. In Section 5, we describe our development approach, the achieved steps, and the remaining ones, starting from the HEAD metamodel development phase and reaching to the policy enforcement phase. In Section 6, we explain the new opportunities this approach opens and what essential services can be developed based on the HEAD metamodel. Section 7 concludes this paper.

2. Access Control Challenges within Dynamic and Heterogeneous Environments

Providing security and privacy, using AC mechanisms, is a very crucial metric within current ubiquitous computing and pervasive systems. Herein is a set of existing challenges in this domain [2,3,11–14]:

- Access control serves as a protective shield for the existing resources in organizations, industries, homes, etc., against security risks that accompany transparency, shareability, and interoperability while answering users' access requests within distributed and heterogeneous computing environments (e.g., IoT environments) where data is generated dynamically and on a real-time basis.
- Access control is highly essential to ensure secure communications in open, dynamic, and heterogeneous environments, especially with the existence of several contexts (network type, time, location, machine characteristics, etc.). For example, to provide an access decision, several or multi-level contexts must be considered. The context itself is a challenge since it could be defined, expressed, and interpreted in different ways according to the application domain, the needed objectives, and the existing techniques.

- Access control mechanism should be flexible and controllable enough to deal with various situations that confront users while requesting access to different types of resources, for example, rapid progressions, unexpected events, system failures, highly dynamic environments that need very quick and controlled response, various hierarchical organizational structures, the different contexts, dynamism of IoT devices, huge number of devices, etc.
- With the large adoption of telework, especially with the COVID-19 pandemic, employees can work anytime, anywhere, and sometimes using their own personal devices. With this fact, security and AC issues have been raised, especially that AC mechanisms should be adapted to users' context, their profiles, their devices, the existing conditions, and many other parameters to improve system usability. With this fact, AC needs to be dynamically managed to minimize human intervention.

Besides, the following are the key AC enforcement challenges:

- The heterogeneity of implemented AC models in heterogeneous structures.
- Deciding upon the most relevant AC model(s) to implement in an organization that conform to its AC rules based on the type and sensitivity of resources they have.
- The necessity to enforce persistent AC policies with the current generation of dynamic and heterogeneous structures where information flows from the clouds and/or servers to everywhere (companies, hotels, homes, cars, etc.) without traditional borders.
- The possible need for several AC solutions within the same organization (or industry sector) where various technologies (e.g., local network, IoT, and cloud) may need to work in concert to fulfill the needed requirements of AC.
- The importance of upgrading AC policies and enforcing them in accordance with technology upgrades and due to dynamically changing conditions.

Due to the above challenges, and the limitations of common models, hybrid models, extended AC models, and abstract AC models in answering the needed AC requirements of the current computing environments, and since they are unable to follow up the continuous technology progressions [5,6,12], the research interest for developing AC metamodels has gained attention in the last decade [9].

3. Access Control Models

Over the decades, various information technologies (IT) have been developed, and this imposed the need to implement various AC methods to find secure communication environments. In the following sections, we summarize the background of the proposed AC models in the literature and the development stages to enhance them.

3.1. The Background

To ensure security and privacy, different AC models are implemented in the literature to enforce AC policy and prevent any illegal access to resources. Figure 1 summarizes the features of common AC models, DAC, MAC, RBAC, and ABAC, and their major components which are used to formulate and enforce organizational AC policies [6,15].

3.2. The Development Stages

With the evolution of technology trends, it is realized that the aforementioned models are insufficient to answer the needed AC requirements, and computing environments need more enhanced AC models. Hence, various works are proposed to (1) combine features of two or more AC models called hybrid models, (2) extend AC models, (3) raise their level of abstraction, and (4) develop more advanced and abstract models, called AC metamodels.

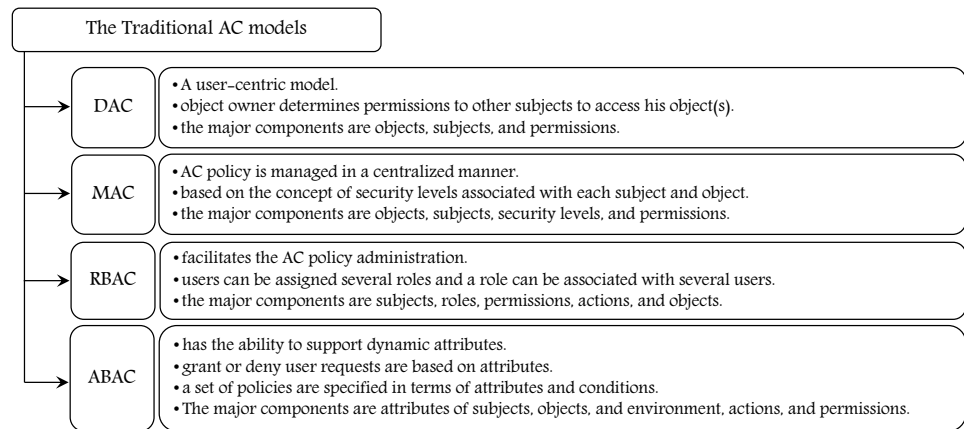


Figure 1. The common AC models.

3.2.1. Combining AC Models

With the continuous technology upgrades and distributed computing environments, the presence of security threats also increases. This imposes the need to find hybrid AC models by combining features of two or more AC models to allow defining more AC policies and enhance the process of access management [5]. Several hybrid AC models are proposed in the literature that combine features of RBAC and ABAC, for example, [16–18]; DAC, MAC and RBAC, for example, [19]; MAC and RBAC, for example, [20]; and many other hybrid models. Figure 2 illustrates the idea of hybrid models.

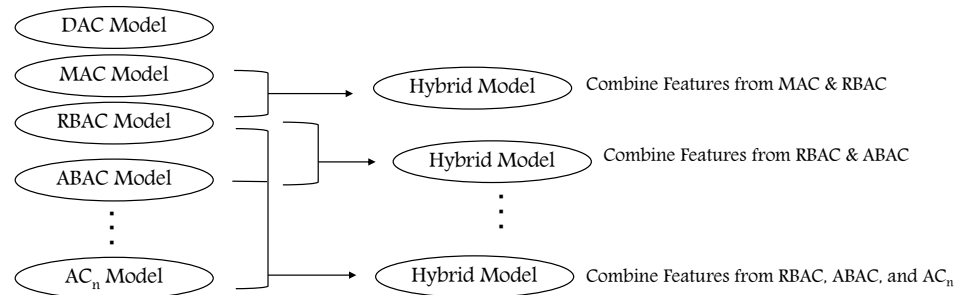


Figure 2. Hybrid AC models.

3.2.2. Extending AC Models

Other works extend AC models by adding new components to them such as new types of roles, permissions, and relationships. For example, the core or flat RBAC model with major components such as subjects, objects, permissions, actions, and roles is extended to hierarchical RBAC where a new component is added to support role hierarchy. Moreover, it is extended to constrained RBAC where a new component is added to enforce the separation of duties. Then, symmetric RBAC which includes hierarchical RBAC and constrained RBAC [21]. In [22], a higher-order attribute-based access control (HoBAC) model is proposed as an extension for the ABAC model, which extends the basic concepts of ABAC with aggregation operations that yield hierarchies. Another ABAC model extension is presented in [23], called the hierarchical group and attribute-based access control (HGABAC), where groups and hierarchies of subjects and objects are added. Moreover, several other AC model extensions are proposed in this domain, for example, [24]. Figure 3 illustrates the concept of AC model extension.

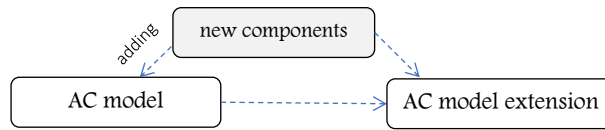


Figure 3. Illustration for AC model extension concept.

3.2.3. Abstracting AC Models

Some AC models are abstracted, then new components are added to enhance their features and allow expressing a larger set of AC policies. Subsequently, the derived AC model is an extended model with the old and new AC features. For example, Nguyen et al., in [25], add a delegation component to the abstracted RBAC model (RBAC metamodel) to have an RBAC-based delegation model. In addition to defining RBAC permission rules, their approach would allow defining delegation rules to specify which actions are accessible to users by delegation. Moreover, in [26], a business and system role-based access control (B&S-RBAC) metamodel is proposed where business and system roles are defined and mapped to overcome the weakness of business role definitions and RBAC models. Hence, the RBAC model is abstracted, a system and business role component is added, then it is extended for business usage. Moreover, Adda et al. [27] propose a generalization for the ABAC model where the core concepts of HoBAC [22] are first revisited and refined, then present new concepts to complete and reinforce its theoretical foundations. Figure 4 illustrates the concept of AC model abstraction.

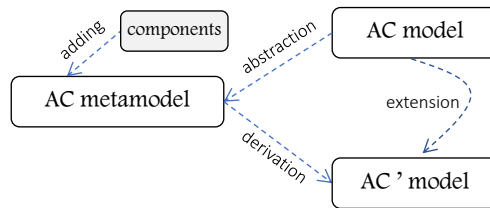


Figure 4. Illustration for model abstraction concept.

3.2.4. Access Control Metamodels

Access control metamodels are proposed to serve as frameworks that unify and include most or all features of AC components to derive various instances of AC models [5,6]. Figure 5 illustrates the concept of the AC metamodel. However, AC metamodels should handle all of the above concepts (combining AC models, extending AC models, and abstracting AC models). In other words, having an abstract model (metamodel) that is able to include all AC models components, or have the ability to define the needed ones, would allow combining and extending different AC models.

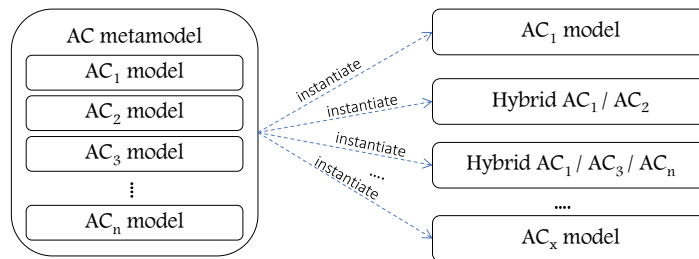


Figure 5. Illustration for AC metamodel concept.

The continuous technology progressions impose developing different stages of AC methods; the AC metamodels stage is the recent development stage in this domain.

The current generation of networking environments is composed of heterogeneous networks, platforms, applications, devices, etc. Controlling access to resources in such

environments is a challenging task and needs advanced AC frameworks. Within the decade, a limited number of AC metamodels are proposed in the literature (Table 1) and they can be summarized as follows [5,9]:

- Some AC metamodels are proposed as generic metamodels [28–31].
- Some others are proposed as hybrid metamodels to provide a generic base metamodel concept [7,32].
- Others are proposed as metamodel extensions for some of the existing metamodels and some software development frameworks [33,34].

Unfortunately, the proposed metamodels lack some key features and are insufficient to answer the AC requirements of the current networking generation (explained in Section 4) [5,9].

Table 1. Access control metamodels: the state-of-the-art [5,9].

ref.	Year	Proposed for	Metamodel	Based on	Instances	Modeling Language
AC metamodels are proposed as generic metamodels						
Barker [28]	2009	Enterprise	Barker’s metamodel	DAC, MAC, RBAC	RBAC, MAC	Rule/logic language
Bertolissi et al. [29]	2014	Distributed system of several sites	Distributed metamodel	DAC, MAC, RBAC	Hybrid models of DAC, MAC, RBAC	Rewrite-based operational semantics
Khamadja et al. [30]	2013	Cloud computing	CatBAC metamodel	DAC, MAC, RBAC	Hybrid models of DAC, MAC, RBAC	First-order logic
Trninić et al. [31]	2013	Set of systems	PolicyDSL	RBAC models	RBAC and hybrid models	Textual DSL
AC metamodels are proposed as hybrid AC metamodels						
Slimani et al. [32]	2011	Enterprise	UACML metamodel	DAC, MAC, RBAC	Group based, MAC, RBAC, hybrid model	Object constraint language (OCL)
Abd-Ali et al. [7]	2015	Enterprise	Integration metamodel	CW, BLP, BIBA, RBAC	Hybrid models	First-order logic
AC metamodels are proposed as metamodel extension for some of the existing metamodels						
Alves et al. [33]	2014	Enterprise	Obligations in CBAC metamodel	DAC, MAC, RBAC	Hybrid models of DAC, MAC, RBAC	Rewrite-based operational semantics
Korman et al. [34]	2016	Enterprise architecture framework	Unified metamodel	DAC, BLP, Biba, CW, RBAC, ABAC	DAC, BLP, CW, RBAC, ABAC	ArchiMate

4. Issues and Limitations of the Existing AC Metamodels

Although the proposed AC metamodels in the literature come with some advancement for some scenarios and use cases, they have several limitations and shortcomings which are explained and illustrated in the following:

4.1. Generality

In the literature, the metamodels proposed as generic are not generic enough and they do not include all features of AC models. They are hybrid templates to derive some AC models that are employed in their core structure rather than metamodels [9]. A generic AC metamodel should include all features and components of common AC models and other models. In Figure 6, we illustrate the concept of generic metamodel where all AC components are included, with the relationships between them.

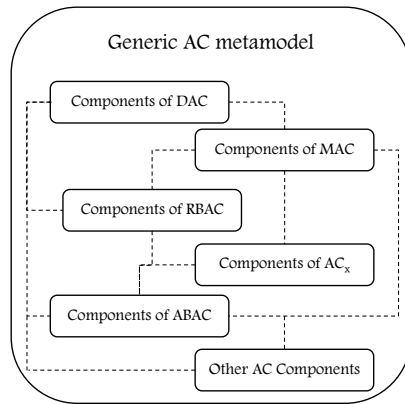


Figure 6. The concept of generic AC metamodel.

Figure 7 summarizes AC model features which are employed in the core structure for each of the proposed metamodels. As we can see, in Table 1, the metamodels by Barker et al. [28], Bertolissi et al. [29], Khamadja et al. [30], and Trninić et al. [31] are proposed as generic by including DAC, MAC, and RBAC features and components, and by Slimani et al. [32] and Abd-Ali et al. [7] as hybrid by combining DAC, MAC, and RBAC AC features and components. In [34], Korman et al. extend the ArchiMate development framework to support features and components of DAC, MAC, RBAC, and ABAC models. Alves et al. [33] extend the category-based AC (CBAC) metamodel which includes features of DAC, MAC, and RBAC to support obligations. Hence, the proposed approaches are not generic enough and do not include or combine all AC features and components. As illustrated in Figure 7, the generic metamodel should include the features and components of common AC models and other models.

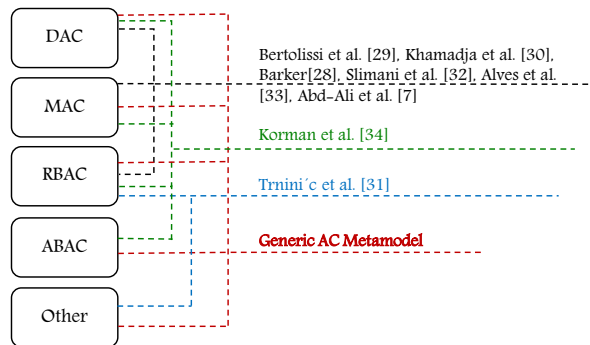


Figure 7. AC features in the core structure for each of the proposed metamodels.

4.2. Dynamism

Security solutions must be manageable and adaptable to track the evolution of security threats which come along with technology upgrades. In this context, an AC metamodel must be upgradable due to changing conditions or rules. The structure of a metamodel should be dynamic and describe how its properties can be modified over time, for example, due to changing conditions (e.g., system, environment, etc.). The metamodel is dynamic if it allows defining new types of attributes, for example, contextual attributes, and components with the relationships between them in order to update and formulate different models; hence, it allows defining a larger set of policies for static and dynamic enforcement. In reference to Table 1, the concept of a dynamic metamodel is not considered since none of them allow defining new components/attributes. Consequently, dynamism is an additional feature that can be implemented for a generic metamodel. Figure 8 illustrates the concept of a dynamic metamodel.

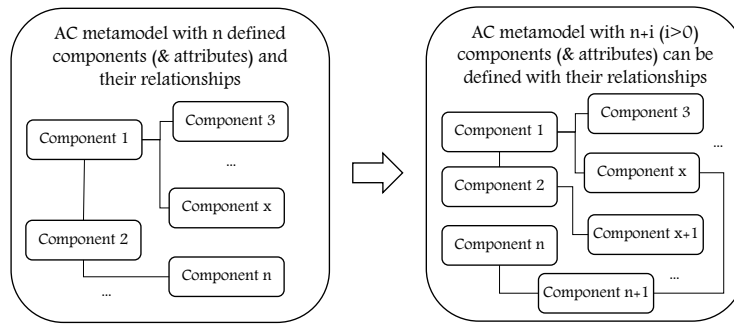


Figure 8. The concept of dynamic AC metamodel.

4.3. Extensibility

Beside designing a generic and dynamic AC metamodel, it is important to consider the feature of extensibility. Extensible metamodel means that new components could be defined and integrated with already existing models and frameworks to support new AC features in addition to the previous ones. However, the proposed metamodel extensions, Table 1, in [33,34], extend some of the existing AC metamodels and software development frameworks to support some AC features of some AC models, but they do not explain how their approaches could be extended beyond the proposed limit. In other words, they are extended but not extensible. In Figure 9, we illustrate the concept of the extensible AC metamodel.

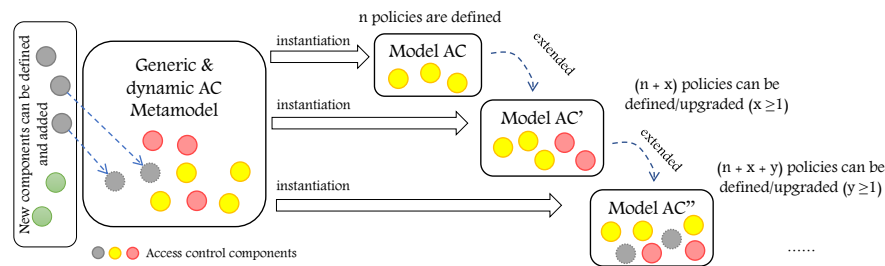


Figure 9. The concept of extensible AC metamodel.

4.4. Hierarchy of Components

The concept of hierarchy is important to define multi-level components (e.g., roles, actions, objects, etc.). It reflects the structure of an organization and, for example, the respective responsibilities/priorities of the hierarchical components. A component hierarchy, e.g., role, defines roles that have unique attributes and may contain other roles; one role may implicitly include the actions that are associated with another role. Within this structure, access rights are determined by an entity’s place in the hierarchy, for example, in complex scenarios (e.g., IoT), administrators can start with creating a number of entities then add their hierarchy. This would help in controlling access to data with less maintenance costs compared to creating a large number of non-hierarchical entities. Hence, hierarchical authorization is the authorization determined based on the hierarchy. Figure 10 represents examples of hierarchy in an organization or industry sector (e.g., role hierarchy, Figure 10a; action hierarchy, Figure 10b; object hierarchy, Figure 10c; context hierarchy, Figure 10d). For example, in an academic and research situation, a role called Dean could contain the roles of Director and Team Leader, Figure 10a. This means that subjects (users) of the role Dean are implicitly connected with the actions associated with their roles as Director and Team Leader without the administrator having to explicitly list the Director and Team Leader actions. In the literature, several models and metamodels are extended to support the feature of hierarchy for some components, but in complex scenarios and highly dynamic environments, it is important to consider this feature for all components, for example,

none of the proposed metamodels consider context hierarchy. The metamodels proposed in [29,30] support hierarchy of category (e.g., role, groups, etc.); [32] supports hierarchy of category, action, object; and [7,31,34] support hierarchy of roles.

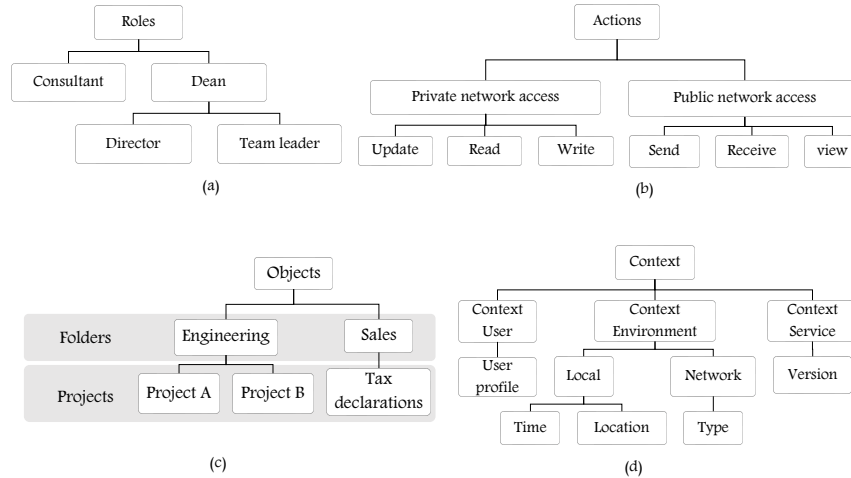


Figure 10. Examples for hierarchy of (a) roles, (b) actions, (c) objects, and (d) contexts [3].

4.5. Collaboration and Interoperability

In pervasive computing environments, the software enables the connection between various heterogeneous devices and users within a dynamic and heterogeneous environment; it carries out the needed mappings between each task and the required services that users need. In the field of AC, and with the current technologies, AC policies are employed to administer decisions in systems. They are increasingly used for implementing flexible and adaptive systems to control access to resources in today’s internet services, networks, security systems, and others. An AC metamodel should deal with the dynamicity of devices and objects used, events, situations encountered, users accessing systems and the environments from which they are connected, and others. It should be highly adaptable and flexible, and it should be able to integrate many devices and information systems to provide the needed services for users (and organizations) and ensure homogeneity and collaboration between its components. Moreover, it should provide the administrative required tasks and enable interoperable interactions between several derived AC models. Figure 11 illustrates the collaboration and interoperability concept an advanced AC metamodel should provide.

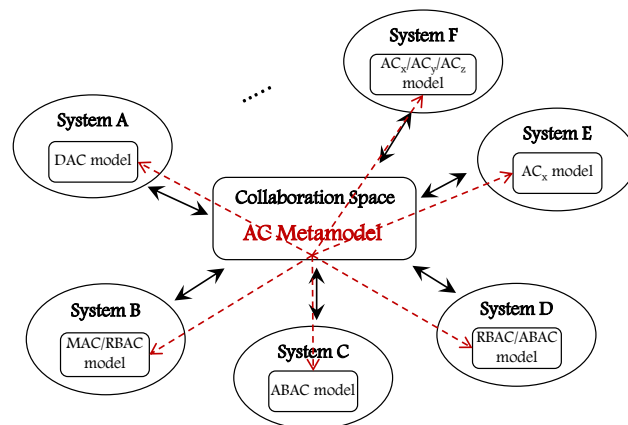


Figure 11. The concept of collaboration and interoperability of AC models.

4.6. Migration

Due to continuous technology upgrades, it is essential to consider the concept of migration where different software components are transferred from one computing environment to another. This concept must also be considered for AC models where the AC migration is a kind of modernization for AC policies (set of rules that are generated using different AC components) from one model to another covered by a generic, dynamic, and extensible AC metamodel. Though, none of the proposed approaches tackle the concept of migrating AC policies. In Figure 12, we illustrate the concept of policy migration from one model to another.

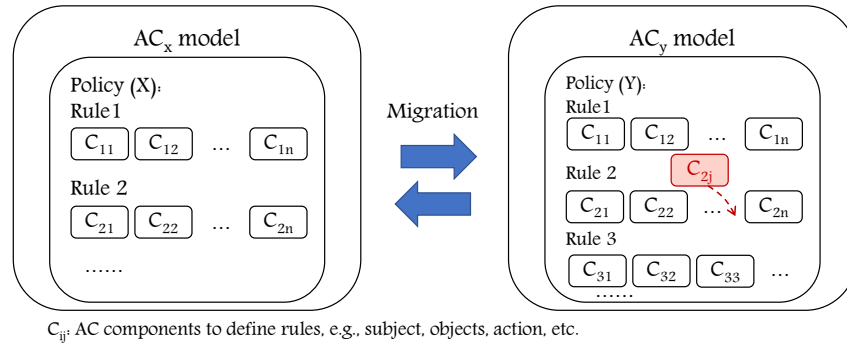


Figure 12. The concept of migration from one AC model to another.

5. HEAD Metamodel: Development Approach to Access Control in Dynamic and Heterogeneous Environments

Security issues are dynamic and ever-changing due to continuous technology progressions and unexpected conditions, so designing, implementing, and testing software for security is a challenging task. Accordingly, a security policy should be updatable and modifiable at any time, and it must be kept aligned with software extension or development. In Section 3, we present the development stages of AC methods and how AC models are combined, extended, abstracted, reaching to the concept of AC metamodels. Developing an advanced AC metamodel is a recent research issue; in Section 4, we explain why the proposed metamodels are not enough to answer the AC needs of the current networking generation.

The world of AC is changing fast [10] and, as shown in Table 1, the last metamodel was proposed in 2015 [7]. Moreover, the last extended framework to support AC features was ArchiMate in 2016 [34]. This reflects the lack of having an advanced AC metamodel in the domain and the importance of developing and having a new and advanced one with essential features that is able to face the existing challenges which accompany the various variants of technology and digital transformation. For this purpose, we have developed the HEAD metamodel [3] and, in this paper, we explain the new opportunities and research directions it opens in the domain. As we can see in Figure 13, the generic and hybrid metamodels which are proposed by Barker et al. [28], Bertolissi et al. [29], Khamadja et al. [30], Slimani et al. [32], Abd-Ali et al. [7], and Alves et al. [33] include features of DAC, MAC, and RBAC so they are not generic enough and do not support the feature of adding/defining new components/attributes. The metamodel proposed by Trninic et al. [31] is not generic since it only includes RBAC and RBAC extensions; also, the metamodel which is extended to support the features of common models is ArchiMate framework [34], it does not allow defining new components to formulate new AC models, and it is extended but not extensible.

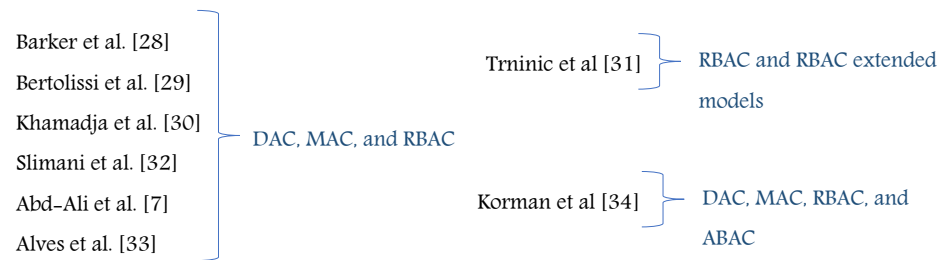


Figure 13. The components that the proposed metamodells include.

Until today, several approaches were proposed to develop and enhance AC models in order to solve various AC problems and to also enhance model features by combining or extending them. Nevertheless, these approaches are insufficient since they focus on some AC models and upgrading them requires modifying the core model design then following all the necessary steps of implementation and testing. This process increases the complexity in controlling access within organizations due to loss of time and cost, especially in the current age of digital transformation, and ubiquitous computing and pervasive systems. Hence, it is very essential for organizations and industry sectors now to rethink how to control access to resources through modern and advanced AC methods. In this section, we present a complete development approach to AC in today's dynamic and heterogeneous computing environments, starting from the AC metamodell development phase and reaching to the policy enforcement phase. Figure 14 summarizes the main phases and the steps. Note that the gray squares indicate the achieved phases and steps, which are explained in detail in [3], and we provide a summary, in this paper, in order to link them with the remaining steps and explain the new opportunities and research directions that could be achieved in this domain based on the HEAD metamodell.

5.1. Metamodell Development

In the literature, heterogeneous AC models with heterogeneous components (e.g., subject, object, role, category, permission, action, etc.) have been implemented. Figure 15 shows an example of heterogeneous models where AC policies are expressed differently. To develop an advanced AC metamodell, the main step is to encompass the heterogeneity of AC models (common models and others). In [3], we have developed a Hierarchical, Extensible, Advanced, and Dynamic AC metamodell, named HEAD metamodell, that addresses the limitations of the existing metamodells. After reviewing and investigating different AC policies in different computing environments [5,6,15], we have realized that the first step to develop a generic metamodell is to unify the heterogeneous components of AC models. In the following, we explain the achieved steps of this phase.

5.1.1. Unify AC Components of Heterogeneous Models

To unify the heterogeneous components and make them adaptable to all AC models, we categorize them into [3]:

- EXPLICIT (E_x) components, those that refer to something that is real and exists (e.g., subjects and objects).
- IMPLICIT (I_m) components, those that refer to something described or explained in the guidelines or rules, and they include:
 - AUTHORIZATION UNITS (AU) (e.g., roles, security levels, categories, etc.).
 - PROCEDURAL UNITS (PU) (e.g., actions, permissions, etc.).
- SETTING (S_t), which refers to components that are included to have more accurate and regulated access to resources (e.g., context, constraints, etc.).

Metamodel Development

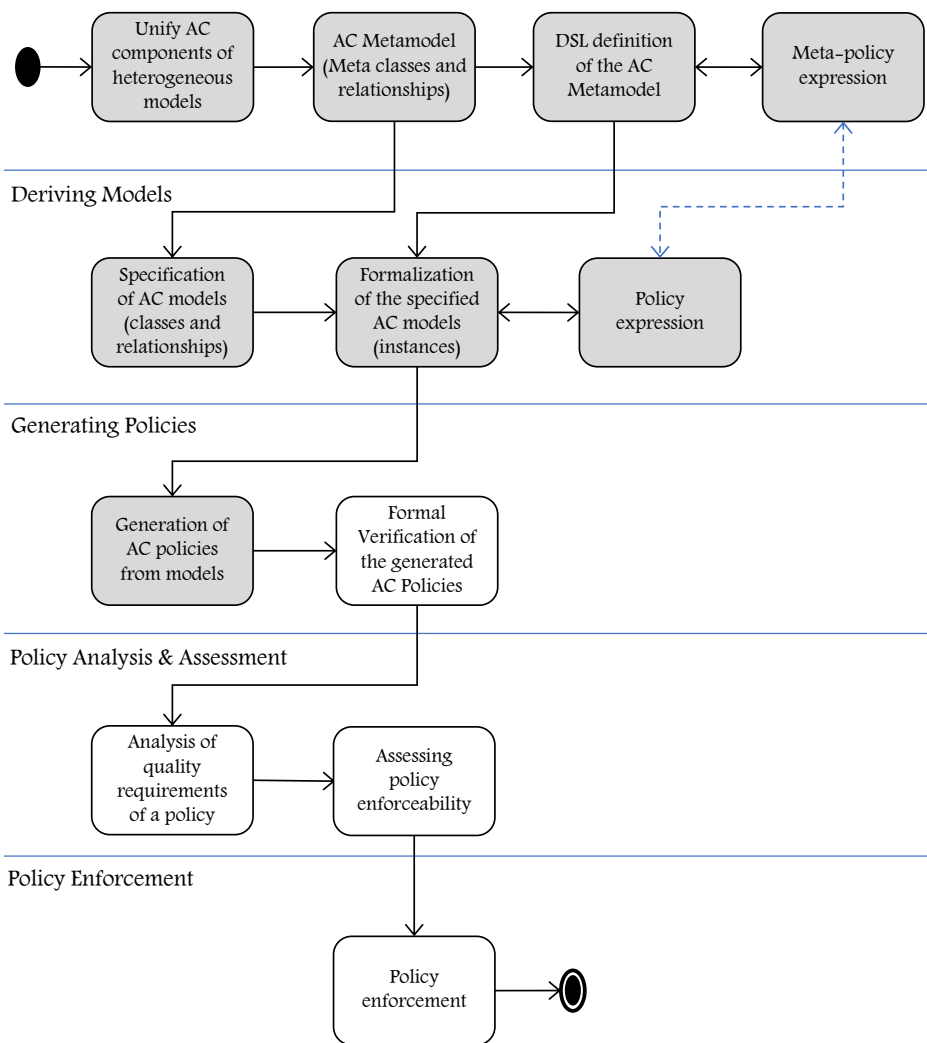


Figure 14. The development approach.

DAC	policy = ⟨Subject, Object, Action⟩
MAC	policy = ⟨Subject, Object, Security Level, Action⟩
RBAC	policy = ⟨Subject, Object, Role, Permission, Action⟩
ABAC	policy = ⟨SubjectAttr, ObjectAttr, ContextAttr, ActionAttr⟩
Hybrid MAC/RBAC	policy = ⟨Subject, Object, Role, SecurityLevel, Permission, Action⟩
Extended RBAC	policy = ⟨Subject, Object, Role, Group, Permission, Action⟩
⋮	

Figure 15. Heterogeneous models with different policy expressions.

Figure 16 shows a hybrid policy example where heterogeneous components need to be expressed in an AC policy. Hence, a metamodel must allow deriving a model that includes all these components.

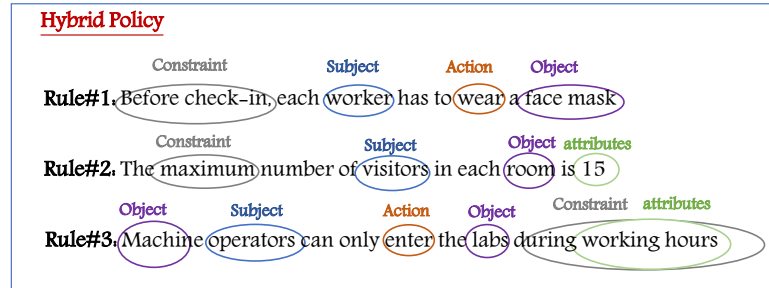


Figure 16. An example of hybrid policy.

5.1.2. HEAD Metamodel: Meta-Components and Relationships

The unified AC components of the previous step have been used to develop the needed metamodel. The meta-components E_x , I_m (AU and PU), and S_t and the relationships between them form the HEAD metamodel. As shown in Figure 17 [3]:

- The relationship between E_x and AU is to assign, for example, zero or many (0..*) subjects to roles, groups, categories, or any other authorization unit.
- The relationship between AU and PU, and PU and E_x , is to represent which AUs are able to perform zero or many PU (e.g., actions, permissions, etc.) and access some, for example, objects or services.
- E_x and I_m components might have zero or many S_t (e.g., contextual and/or non-contextual constraints) before accessing/performing tasks on some other E_x components.
- The self-association edge exists on each of the classes to allow formulating AC models and hybrid models for different policies by allowing AUs to be associated with other AUs, PUs to be associated with other PUs, and so on.
- The hierarchical relationships are depicted by aggregation association for each of the entities, E_x , AU, PU, and S_t , to create a hierarchy of classes.

The importance of the HEAD metamodel is that it even allows defining any new component/attribute for non-existing model(s) to formulate new model(s), besides its ability to derive the common models.

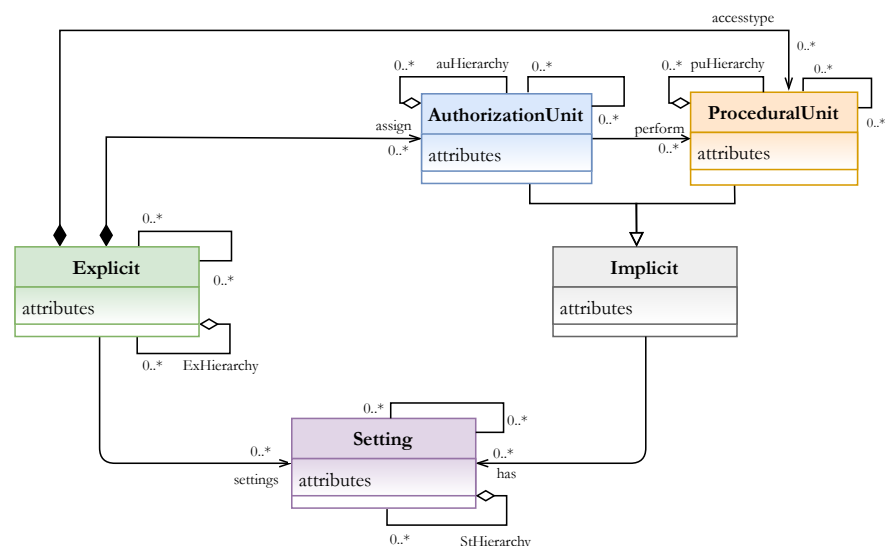


Figure 17. HEAD metamodel [3].

5.1.3. The DSL Definition of HEAD Metamodel

The domain-specific language (DSL), using Eclipse Xtext, for the HEAD metamodel is able to represent a variety of AC models in a generic way by defining any component/attribute for any AC model (common models, hybrid models, new models, etc.). Figure 18 shows part of the DSL which is used to define the explicit, implicit, and setting components and their attributes (the detailed DSL definition is explained in [3]).

```

Policy:
  name=ID (('attributes+=Attribute+')?)
  'explicit' (explicit+=Explicit)+ 'end'
    (implicit+=Implicit)+
  ('setting'(setting+=Setting)* 'end')?
;
AttType:
  'String'|'int'|'boolean'|'char'|'float'
;
Explicit:
  name=ID (('attributes+=Attribute+')?)
    (['heirarchy+=Explicit+']?)?
;
Implicit:
  {Implicit}
  ('authorization' authunit+=AuthorizationUnit* 'end')?
  'procedural' procunit+=ProceduralUnit* 'end'
;
AuthorizationUnit:
  name=ID (('attributes+=Attribute+')?)
    (['heirarchy+=AuthorizationUnit+']?)?
;
ProceduralUnit:
  name=ID (('attributes+=Attribute+')?)
    (['heirarchy+=ProceduralUnit+']?)?

```

Figure 18. Sample of DSL of HEAD metamodel [3].

5.1.4. Meta-Policy Expression

The meta-policy is expressed in terms of the meta-components E_x , I_m , and S_t :

$$\text{Metapolicy} = \langle E_x, I_m, S_t \rangle$$

5.2. Deriving Models

HEAD metamodel is (1) generic and allows deriving instances of different models and hybrid models, (2) dynamic and allows defining new components with the relationships between them, (3) extensible to upgrade any defined policy, and (4) supportive of defining hierarchies.

5.2.1. Specification of AC Models: Components and Relationships

This step consists of analyzing and expressing the AC requirements of a system and identifying the E_x (e.g., subject, object, etc.), I_m (e.g., AU = role, group, etc., and PU = action, permission, etc.), and S_t (e.g., context, constraints, etc.) model components. The inputs of this phase are the identified components for modeling a policy, and the outputs are the specification diagrams (UML diagrams).

5.2.2. Formalization of the Specified AC Models

In this step, we encode the obtained AC models from the previous step, after specifying the needed AC components, using the DSL of the HEAD metamodel as an AC modeling language. The power of the DSL language of the HEAD metamodel is that it is simple and flexible to appropriately express any AC policy requirements, overcomes the complication of existing language expressions, and it is also independent of specific AC models. The flowchart in Figure 19 explains how this DSL works to instantiate various policy models. It can be interpreted as follows:

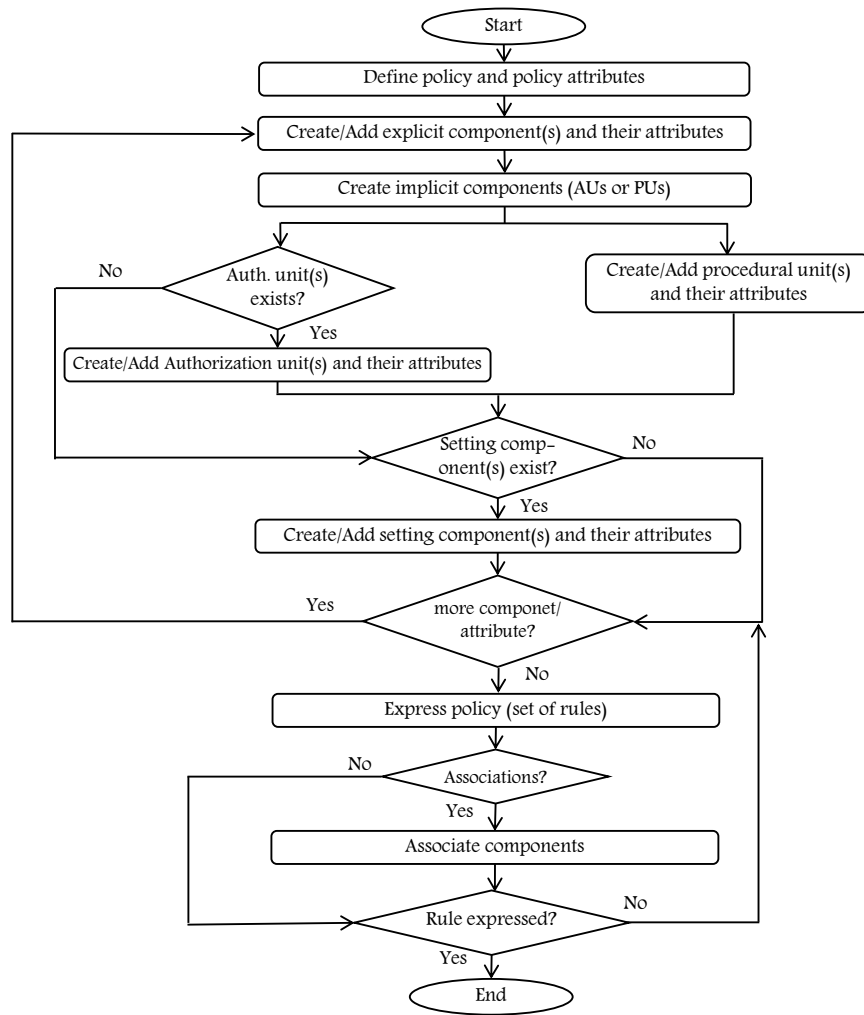


Figure 19. Instantiation of AC models using the DSL of HEAD metamodel.

- Before defining any component(s), policy model should be specified with the needed attributes, for example, policy type (e.g., RBAC policy, ABAC policy, etc.), organization, etc.
- At least one (or more) E_x component must be created (with the needed attributes) since any AC model must include E_x entities (e.g., subject, object).
- I_m components (AUs and/or PUs) should be identified and created. Note that an AC model might not include an AU component(s) if, for example, the needed AC model is DAC. If the model to be instantiated is MAC or RBAC, then there exists an AU component (security level or role). Moreover, an AC model must include at least one PU component, for example, action, and it should be created.
- An AC model might not include S_t component(s), for example, in the DAC model context, and constraint components do not exist.
- If there exists additional components/attributes, they should be created, for example, to upgrade a policy. The HEAD metamodel allows for adding them.
- Finally, if all the needed components/attributes are created, then associations of components are handled while expressing the needed rule(s).

Figure 20 shows an example of MAC instance as a result of this step. MAC components/attributes are defined and then the needed rule is expressed in terms of MAC elements (subject, object, security level, and operation).

```

1 policy MAC(descr:String)
2   explicit subject(name:String) object(type:String) end
3   authorization securitylevel(level:String) end
4   procedural operation(op:String) end
5 end
6
7 rule: (ruleid:int){
8   MAC.subject(MAC.subject.name)
9   [MAC.securitylevel(MAC.securitylevel.level)]{
10
11     MAC.object(MAC.object.type)
12     [MAC.securitylevel(MAC.securitylevel.level)]{
13       MAC.operation(MAC.operation.op)
14     }
15   }
16 }--> decision

```

Figure 20. Example: MAC instance [3].

5.2.3. Policy Expression

Based on the meta-policy expression (Section 5.1.4), different AC policy definitions can be expressed as shown in Figure 21.

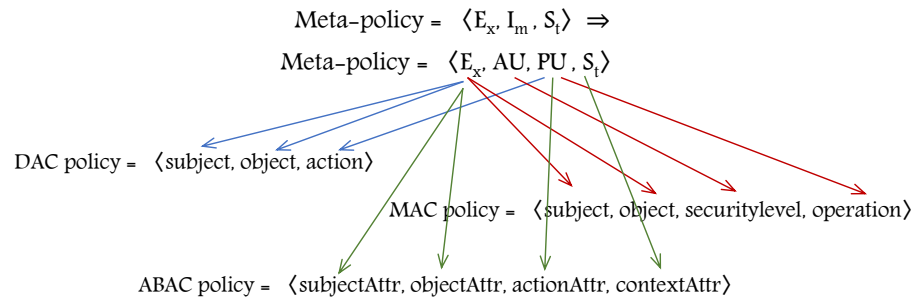


Figure 21. Examples of policy expressions using the meta-policy of HEAD metamodel.

5.3. Generating Policies

After specifying and defining the needed policy (or policies), this phase aims to generate then verify the defined policies before analyzing and assessing them.

5.3.1. Generation of AC Policies from the Specified Models

The input of this step is the outputs of the previous one where the derived models are encoded using Eclipse Xtend notation (Figure 22 shows a sample), an expressive dialect of Java, to represent the concrete instance of an AC policy and generate the needed java code. This step is a preliminary step that allows formally verifying the policy concepts and properties.

5.3.2. Formal Verification of the Generated AC Policies

The input of this step is the generated java code from the previous step. This step consists of formally verifying the accuracy and the coherence of the concrete instance of the AC policy, which is formalized in the previous step, before proceeding to its implementation. This can be achieved, for example, by injecting the generated java code of the previous step into the Next Generation Access Control (NGAC) framework [35] to represent the AC rules of a system in a graph—the objects, the relationships between them, and the subjects that interact with the system in a way that adheres the semantics of an organization.

```

«FOR j : 0 ..< explicitlist.size»
import explicit«explicitlist.get(j).name.toFirstUpper»;
«ENDFOR»
import authorization.*;
import procedural.*;
import setting.*;
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;

public class «p.name.toFirstUpper»{
    enum Decision {
        Allow,
        Deny,
        Mixed,
        Undetermined
    }

«FOR a : p.attributes»
    private «a.type» «a.name»;
«ENDFOR»

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        «FOR j : 0 ..< plist.size»
        «FOR a : p.attributes»
            System.out.print("«p.name» Policy «a.name»:");
            «a.type» «a.name.toString.toLowerCase» = sc.nextLine();
        «ENDFOR»
        «ENDFOR»

        /* ---Explicit components--- */
        «FOR j : 0 ..< explicitlist.size»
        «explicitlist.get(j).name.toFirstUpper» «explicitlist.get(j).name.toString.toLowerCase» = new «explicitlist.get(j).name.toFirstUpper»();
        List<String> e_«explicitlist.get(j).name.toString.toLowerCase» = new ArrayList<String>();
        «ENDFOR»
        /* ---Implicit components--- */
        /* Authorization Units */
        «FOR k : 0 ..< aulist.size»
        «aulist.get(k).name.toFirstUpper» «aulist.get(k).name.toString.toLowerCase» = new «aulist.get(k).name.toFirstUpper»();
        String «aulist.get(k).name.charAt(0)»«k» = new String();
        List<String> au_«aulist.get(k).name.toString.toLowerCase» = new ArrayList<String>();
        «ENDFOR»
        /* Procedural Units */
        «FOR k : 0 ..< pulist.size»
        «pulist.get(k).name.toFirstUpper» «pulist.get(k).name.toString.toLowerCase» = new «pulist.get(k).name.toFirstUpper»();
        String «pulist.get(k).name.charAt(0)»«k» = new String();
        List<String> pu_«pulist.get(k).name.toString.toLowerCase» = new ArrayList<String>();
        «ENDFOR»
        /* ---Setting components--- */
        «FOR j : 0 ..< slist.size»
        «slist.get(j).name.toFirstUpper» «slist.get(j).name.toString.toLowerCase» = new «slist.get(j).name.toFirstUpper»();
        String «slist.get(j).name.charAt(0)»«j» = new String();
        List<String> st_«slist.get(j).name.toString.toLowerCase» = new ArrayList<String>();
        «ENDFOR»
    }
}

```

Figure 22. A sample of Eclipse Xtend notation.

5.4. Policy Analysis and Assessment

Policies with minimum quality may lead to situations, for example, preventing users from accessing data they are allowed to access or releasing data to unauthorized parties. In some other cases, the AC model may not even have a policy concerning some requests which may lead to uncertainties concerning the obtained AC decision. For higher confirmation of security for data and resources, it is important to investigate whether such policies fit for their purposes [36,37]. This phase is to analyze and assess the obtained policies before enforcing them. It consists of two main sub-phases or steps.

5.4.1. Analysis of Quality Requirements of a Policy

This step consists of assuring the quality of the obtained AC policies and making sure that they are consistent, relevant, minimal, complete, and correct with respect to the actions needed to be performed by subjects on some objects [13,36].

- Consistency means investigating that the obtained policies do not include both an allow and deny decision to the same subject for the action of an object.
- Relevance means checking if the obtained policies do not contain rules that do not apply to any action performed by subjects. In other words, making sure that subjects do not have any authorization to access object(s) and perform action(s) that they are not expected to execute.
- Minimality refers to investigating that the obtained policies do not include redundant or unnecessary policies.
- Completeness means that, for a given access request to access an object, there must be a corresponding policy, and any action to be executed by the subjects on any object the default decision that should be taken by AC model is to deny the access.
- Correctness refers to the process of checking that the policies conform with their intended goals and are in compliance with the system requirements.

Hence, this phase is of major importance while implementing AC rules in highly dynamic and heterogeneous computing environments, especially IoT environments. Unfortunately, although several methods for policy analysis and assessment are proposed [13,38], they have major shortcomings. For example, they do not address all the quality requirements, e.g., they only address consistency and minimality, they analyze and assess the obtained policies of only some AC models (e.g., RBAC policies), and they also require all possible AC requests as input [13]. Hence, the proposed methods are not suitable for dynamic and distributed environments. Moreover, developing an approach for analyzing the quality requirements of a policy at runtime is of major importance, especially for today's environments such as IoT and industry 4.0 where data are generated in real-time basis.

5.4.2. Assessing Policy Enforceability

Policy assessment is the process of predicting and evaluating the possible impacts of policy options by, for example, testing policies with respect to a set of scenarios to determine their enforceability. The enforceability of AC policies is a challenging task due to its high dependence on contexts, for example, some policies can be easily enforced within an organization, and the same policy may not be enforceable in the IoT context. Assessing enforceability of policies may require monitoring the intended system and gathering information about some unexpected events, delays, system failures, etc., and based on this information, the AC system could assess the difficulty or impossibility of enforcing certain policies [13,39]. Moreover, AC systems should also allow identifying constant policy abusers and provide the required evidence to exclude users who do not respect the best practices of an organization. For example, by controlling who goes where in a certain context, to ensure that accessing sensitive areas are limited to users with the needed permission, the AC system should hence be able to detect intruders and prevent them from accessing these areas.

Assessing policy enforceability is critical for a new networking generation such as IoT computing environments due to the huge number of interconnected devices, millions of users, the various contexts, and other facts. The ability to detect inappropriate or unusual behavior, assess it as a real event, and use the evidence to support additional system training or enforcement is of tremendous value for any organization.

5.5. Policy Enforcement

Policy enforcement is where all the previous phases and steps come together to serve an organization and where the final decision is triggered to subsequently enforce the valid AC decisions. One of the fundamental capabilities of organization security systems is their ability to support security, organization standards, and AC policy enforcement. A policy enforcement entity could be a network device on which policy decisions are enforced. This phase is usually handled by software or hardware that serves as proxy, gateway, or any other centralized control point in a network. The defined AC policies must specify the needed action(s) if breaches occur. If so, the software or hardware, which works as a policy enforcement point in the network, (1) detects the breaches by comparing the request status with the assessed policy in Section 5.4 (previous phase), then (2) takes the needed action by allowing or denying the access request [40].

6. Open Issues and New Opportunities

The emergence of new technologies, especially IoT and industry 4.0 systems, needs the application of robust and advanced security mechanisms. However, security models and AC solutions were not developed to address the current challenges of highly dynamic environments. They are not able to meet the needs of transparency, scalability, shareability, interoperability, and end-to-end security. Accordingly, several studies confirm that a deep revision and adaptation of those mechanisms need to be considered [41,42]. In this paper, we present a development approach in AC for complex, dynamic, and heterogeneous environments. The characteristics of the HEAD metamodel, and the development approach

for its phases, explained in Section 5, allow developing essential issues in the domain and it opens new opportunities and various research directions which will be summarized in this section. Hence, having an advanced AC metamodel, that is generic, dynamic, extensible, and supports the hierarchy of components, allows for developing and integrating various other important issues such as:

- HEAD metamodel could be enhanced to dynamically generate AC policies according to users' context, profile, device, etc., since is based on a multi-level rule engine.
- In collaborative computing environments, distributed multiple cyber-physical areas interoperate with each other to provide an intelligent environment for users to achieve their collaborative activities. Hence, the features of the HEAD metamodel would facilitate fulfilling the collaboration and interoperability between the derived and heterogeneous AC models and their components among distributed multiple cyber-physical areas.
- The continuous technology upgrades impose the need to migrate AC policies from one model to another. The HEAD metamodel supports all the necessary tools to enhance with this new feature, in addition to its features.
- HEAD metamodel could be enhanced to implement and include built in packages or set of tools with predefined AC models, for example, the most used models DAC, MAC, RBAC, and ABAC. This could minimize the technical implementation efforts and facilitate administrative efforts to define policies.
- Since the definition for hundreds of thousands of attributes, expressing rules and policies, and performing implementation require a lot of time and resources, artificial intelligence (AI) techniques could be used, for example, object recognition to identify objects and define their attributes. Hence, the administrator could modify, add, or remove some attributes instead of creating hundreds of objects and thousands of attributes, especially in very large and complex systems.
- It could be enhanced and implemented to extract the data flow of an organization to define the rules after extracting the entities and attributes, then the administrator could be notified if there are some missing rules that must be defined.
- The features of the HEAD metamodel allow integrating AC to other systems, for example, intrusion detection systems.
- The flexible, upgradable, and dynamic nature of the HEAD metamodel makes it adaptable and applicable to distributed systems; hence, it allows developing and implementing blockchain-based AC systems.

Each of the above points is itself an opportunity (in addition to many other opportunities) for enhancing AC in today's computing environments using a unifying framework. In summary, the HEAD metamodel contributes to the development of knowledge in the field and can be considered as a paradigm shift toward reshaping the existing models and AC methods, especially in the field of industry.

7. Conclusions

The ability to control access to sensitive data and resources in conformity with AC policy is a fundamental security requirement. Despite the access control research over the decade, the significant technology progressions and the limited ability of the existing AC mechanisms force the need to develop and enhance AC methods. Hence, the development stages of AC models over the decades produce heterogeneous AC models which impose the need to develop AC metamodels to encompass the heterogeneity of the existing models and find more advanced AC features. Unfortunately, the proposed AC metamodels are not generic enough to define and enforce the essential AC requirements, especially with the presence of various challenges such as the emergence of ubiquitous computing and pervasive systems, the revolution of industry 4.0, the adoption of telework due to the COVID-19 pandemic, the digital transformation, and many other facts. All these oblige the need for defining and enforcing a larger set of static and dynamic AC policies and the requirement to set up a new and general approach.

However, as explained in this paper, there is a limited number of the proposed AC metamodels, and the existing ones have several limitations and cannot follow the needed upgrades which must accompany the various technology progressions. For this purpose, we have proposed the HEAD AC metamodel with advanced features. In this paper, we explain our development approach starting from the metamodel development phase and reaching to the policy enforcement phase. We explain the achieved steps and the remaining ones in order to employ this metamodel in developing many other essential phases in this domain. The aim of this is to draw a complete strategy instead of tackling each issue separately. In our experience so far, AC research and practice focuses on one phase and confuses issues that cut across multiple phases. The purpose of this paper is to explain and clarify the functionality of all phases to close the enormous gaps between them. To the best of our knowledge, this is the first paper that considers all the needed phases and steps and opens new opportunities in the domain based on an advanced AC metamodel, especially since the last AC metamodel was proposed in 2015 and another extended one was in 2016. This reflects the importance of setting up new strategies, methods, and opportunities to share with researchers and experts in the domain, a new approach that tackles the different facets and is based on an unconventional AC metamodel.

Author Contributions: Conceptualization, N.K., M.A. and H.I.; formal analysis, N.K. and M.A.; visualization, N.K., M.A. and H.I.; software, N.K.; investigation, N.K., M.A. and H.I.; writing—original draft, N.K.; writing—review and editing, M.A. and H.I. supervision, M.A. and H.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), grant number 06351.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Acknowledgments: We acknowledge the support of Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT), and Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Quader F, Janeja V P. Insights into Organizational Security Readiness: Lessons Learned from Cyber-Attack Case Studies. *J. Cybersecur. Priv.* **2021**, *1*, 638–659. [[CrossRef](#)]
2. Krehling, L.; Essex, A. A Security and Privacy Scoring System for Contact Tracing Apps. *J. Cybersecur. Priv.* **2021**, *1*, 597–614. [[CrossRef](#)]
3. Kashmar, N.; Adda, M.; Ibrahim, H. HEAD Metamodel: Hierarchical, Extensible, Advanced, and Dynamic Access Control Metamodel for Dynamic and Heterogeneous Structures. *Sensors* **2021**, *21*, 6507. [[CrossRef](#)] [[PubMed](#)]
4. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Access Control Metamodel for Policy Specification and Enforcement: From Conception to Formalization. *Procedia Comput. Sci.* **2021**, *184*, 887–892. [[CrossRef](#)]
5. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. A Review of Access Control Metamodels. *Procedia Comput. Sci.* **2021**, *184*, 445–452. [[CrossRef](#)]
6. Kashmar, N.; Adda, M.; Atieh, M. From Access Control Models to Access Control Metamodels: A Survey. In *Future of Information and Communication Conference*; Springer: Cham, Switzerland, 2019; pp. 892–911. [[CrossRef](#)]
7. Abd-Ali, J.; El Guemhioui, K.; Logrippo, L. A Metamodel for Hybrid Access Control Policies. *J. Softw.* **2015**, *10*, 784–797. [[CrossRef](#)]
8. Abramov, J.; Anson, O.; Dahan, M.; Shoval, P.; Sturm, A. A methodology for integrating access control policies within database development. *Comput. Secur.* **2012**, *31*, 299–314. [[CrossRef](#)]
9. Kashmar, N.; Adda, M.; Ibrahim, H. Access Control Metamodels: Review, Critical Analysis, and Research Issues. *J. Ubiquitous Syst. Pervasive Netw.* **2021**, *16*, 2. [[CrossRef](#)]
10. Wolfe, C. State of the Market: Access Control. *Security Distributing and Marketing (SDM) Magazine*, 5 April 2021.
11. Al Kukhun, D. Steps Towards Adaptive Situation and Context-Aware Access: A Contribution to the Extension of Access Control Mechanisms within Pervasive Information Systems. Ph.D. Thesis, Université de Toulouse, Toulousen, France, 2012.

12. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H. Deriving Access Control Models based on Generic and Dynamic Metamodel Architecture: Industrial Use Case. *Procedia Comput. Sci.* **2020**, *177*, 162–169. [[CrossRef](#)]
13. Bertino, E.; Jabal, A.A.; Calo, S.; Verma, D.; Williams, C. The challenge of access control policies quality. *J. Data Inf. Qual.* **2018**, *10*, 1–6. [[CrossRef](#)]
14. Soltani, N.; Jalili, R. Enforcing Access Control Policies over Data Stored on Untrusted Server. In Proceedings of the 2017 14th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), Shiraz, Iran, 6–7 September 2017; pp. 119–124. [[CrossRef](#)]
15. Kashmar, N.; Adda, M.; Atieh, M.; Ibrahim, H., Access Control in Cybersecurity and Social Media. In *Cybersécurité et Médias Sociaux*; Presses de l'Université: Laval, QC, Canada, 2021; Chapter 4.
16. Hasiba, B.A.; Kahloul, L.; Benharzallah, S. A new hybrid access control model for multi-domain systems. In Proceedings of the 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), Barcelona, Spain, 5–7 April 2017; pp. 766–771. [[CrossRef](#)]
17. Rajpoot, Q.M.; Jensen, C.D.; Krishnan, R. Integrating attributes into role-based access control. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy, Fairfax, VA, USA; Springer: Cham, Switzerland, 13 July 2015; pp. 242–249. [[CrossRef](#)]
18. Kaiwen, S.; Lihua, Y. Attribute-role-based hybrid access control in the internet of things. In Proceedings of the Asia-Pacific Web Conference, Cham, Switzerland, 5 September 2014; pp. 333–343. [[CrossRef](#)]
19. Oh, S. Permission-Centric Hybrid Access Control. In *Advances in Web and Network Technologies, and Information Management*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 694–703. [[CrossRef](#)]
20. Kim, S.; Kim, D.K.; Lu, L.; Song, E. Building hybrid access control by configuring RBAC and MAC features. *Inf. Softw. Technol.* **2014**, *56*, 763–792. [[CrossRef](#)]
21. Ennahbaoui, M.; Elhajji, S. Study of access control models. *Proc. World Congr. Eng.* **2013**, *2*, 3–5.
22. Aliane, L.; Adda, M. HoBAC: Toward a higher-order attribute-based access control model. *Procedia Comput. Sci.* **2019**, *155*, 303–310. [[CrossRef](#)]
23. Servos, D.; Osborn, S.L. HGABAC: Towards a formal model of hierarchical attribute-based access control. In *International Symposium on Foundations and Practice of Security*; Springer: Cham, Switzerland, 2014; pp. 187–204. [[CrossRef](#)]
24. Layouni, F.; Pollet, Y. Fi-orbac: A model of access control for federated identity platform. In Proceedings of the IADIS International Conference Information Systems, Barcelona, Spain, 27 February 2009.
25. Nguyen, P.H.; Nain, G.; Klein, J.; Mouelhi, T.; Le Traon, Y. Model-driven adaptive delegation. In Proceedings of the 12th Annual International Conference on Aspect-Oriented Software Development, New York, NY, USA, 24 March 2013; pp. 61–72. [[CrossRef](#)]
26. Klarl, H.; Molitorisz, K.; Emig, C.; Klinger, K.; Abeck, S. Extending Role-based Access Control for Business Usage. In Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies, Athens, Greece, 18–23 June 2009; pp. 136–141. [[CrossRef](#)]
27. Adda, M.; Aliane, L. HoBAC: fundamentals, principles, and policies. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 5927–5941. [[CrossRef](#)]
28. Barker, S. The next 700 access control models or a unifying meta-model? In Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, New York, NY, USA, 3 Jun 2009; pp. 187–196. [[CrossRef](#)]
29. Bertolissi, C.; Fernández, M. A metamodel of access control for distributed environments: Applications and properties. *Inf. Comput.* **2014**, *238*, 187–207. [[CrossRef](#)]
30. Khamadja, S.; Adi, K.; Logrippo, L. Designing flexible access control models for the cloud. In Proceedings of the 6th International Conference on Security of Information and Networks, Aksaray, Turkey, 26–28 November 2013; pp. 225–232. [[CrossRef](#)]
31. Trninić, B.; Sladić, G.; Milosavljević, G.; Milosavljević, B.; Konjović, Z. Policydsl: Towards generic access control management based on a policy metamodel. In Proceedings of the 2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT), Budapest, Hungary, 22–24 September 2013; [[CrossRef](#)]
32. Slimani, N.; Khambhammettu, H.; Adi, K.; Logrippo, L. UACML: Unified access control modeling language. In Proceedings of the 2011 4th IFIP International Conference on New Technologies, Mobility and Security, Paris, France, 7–10 February 2011; pp. 1–8. [[CrossRef](#)]
33. Alves, S.; Degtyarev, A.; Fernández, M. Access control and obligations in the category-based metamodel: a rewrite-based semantics. In *International Symposium on Logic-Based Program Synthesis and Transformation*; Springer: Cham, Switzerland 2014; pp. 148–163. [[CrossRef](#)]
34. Korman, M.; Lagerström, R.; Ekstedt, M. Modeling enterprise authorization: a unified metamodel and initial validation. *Complex Syst. Inform. Model. Q.* **2016**, *7*, 1–24. [[CrossRef](#)]
35. Ferraiolo, D.; Chandramouli, R.; Kuhn, R.; Hu, V. Extensible access control markup language (XACML) and next generation access control (NGAC). In Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control, New Orleans, LA, USA, 11 March 2016; pp. 13–24. [[CrossRef](#)]
36. Bertino, E.; Jabal, A.A.; Calo, S.; Makaya, C.; Touma, M.; Verma, D.; Williams, C. Provenance-based analytics services for access control policies. In Proceedings of the 2017 IEEE World Congress on Services (SERVICES), Honolulu, HI, USA, 25–30 June 2017; pp. 94–101. [[CrossRef](#)]

37. Hu, V.C.; Kuhn, D.R.; Xie, T. Property verification for generic access control models. In Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Shanghai, China, 17–20 December 2008; Volume 2, pp. 243–250. [[CrossRef](#)]
38. Hu, V. C.; Kuhn, R.; Yaga, D. *Verification and Test Methods for Access Control Policies/Models*; NIST Special Publication: Gaithersburg, MD, USA, 2017; Volume 800, p. 192. [[CrossRef](#)]
39. Vanickis, R.; Jacob, P.; Dehghanzadeh, S.; Lee, B. Access control policy enforcement for zero-trust-networking. In Proceedings of the 2018 29th Irish Signals and Systems Conference (ISSC), Belfast, UK, 21–22 June 2018; pp. 1–6. [[CrossRef](#)]
40. Norman, T. 5-Electronics Elements (High-Level Discussion). In *Integrated Security Systems Design*, 2nd ed.; Norman, T., Ed.; Butterworth-Heinemann: Boston, MA, USA, 2014; pp. 49–55. [[CrossRef](#)]
41. Ouaddah, A.; Mousannif, H.; Abou Elkalam, A.; Ouahman, A.A. Access control in the Internet of Things: Big challenges and new opportunities. *Comput. Netw.* **2017**, *112*, 237–262. [[CrossRef](#)]
42. Ravidas, S.; Lekidis, A.; Paci, F.; Zannone, N. Access control in Internet-of-Things: A survey. *J. Netw. Comput. Appl.* **2019**, *144*, 79–101. [[CrossRef](#)]

CHAPTER 6

GENERAL CONCLUSION

The ability to control access to sensitive data in conformity with policy is perhaps the most essential security requirement, especially with the massive presence of new paradigms and technologies, also with the emergence of digital transformation and intelligent solutions based on the industry 4.0 or smart industry concept. To prevent any unauthorized access to logical or physical resources, several AC methods are implemented to control what users can access, when, and how by enforcing the defined organizational AC policies. In this context, various research issues are conducted in this domain, and AC metamodels is the most recent topic in this domain. AC metamodels are developed to serve as unifying frameworks that are able to include the heterogeneity of all AC features and allow deriving different instances of AC models. However, AC policy languages should follow the continuous development of highly ubiquitous technologies and information systems, especially with the concept of IoT and industry 4.0 applications. Developing a new policy language is increasingly challenging due to the dynamic and heterogeneous structures of the current networking generation.

The main objective of this research project is to design new and advanced AC metamodel that conforms to organizational (e.g., companies, industries, and hospitals) AC security policies, and adapts the decision making, according to technology progression to meet organizational and users' needs. On this basis, a Hierarchical, Extensible, Advanced, and Dynamic AC metamodel is developed, named HEAD metamodel, which takes into consideration the continuous technology changes and upgrades. Its meta-components unify the heterogeneous components of AC models, it allows specifying and deriving any AC model (existing and non-existing AC models). It is dynamic and allows defining any new component/attribute with the relationships between components, and allows extending the already derived models to follow technology upgrades. Additionally, another powerful feature that exists in the HEAD metamodel is the hierarchy of components (any type of component) to meet hierarchical authorizations, hence administrators can start with creating a number of entities then add their hierarchy which would help in controlling access to data with less maintenance costs compared to creating a large number of non-hierarchical entities. Moreover, the DSL language of HEAD metamodel allows specifying unlimited number of policy classes with the flexibility of defining entities/attributes for each policy class, also it allows expressing combined and uncombined rules for the same hybrid model. For the implementation, Eclipse (xttext) is used to define the DSL of HEAD metamodel. We illustrate our approach with several successful instantiations of various models to show how it supports advanced features compared to other metamodels. For the evaluation and validation, HEAD metamodel is employed to specify the needed AC policies for two case studies inspired by the computing environment of ITMI; the first is for ITMI's local (non-IoT) environment and the second for ITMI's IoT environment.

The results show that HEAD metamodel is feasible and can be adapted and integrated with various local and distributed environments, able to serve as a unifying framework, answer the current AC requirements and follow the needed policy upgrades. Also, we show that HEAD metamodel can be implemented to generate AC rules using other platforms.

6.1 ACHIEVED OBJECTIVES

Within the framework of this thesis, the main objective is to develop a new AC metamodel with advanced features adaptable to centralized and computing environments, and able to work as a base to overcome the limitations of the existing metamodels. However, the most significant contribution of this work lies in features of the proposed AC metamodel. To achieve this, our work was divided into different phases reflected in different chapters (journal papers) throughout this thesis.

In Chapter 3, the formalization of AC metamodel with the needed steps are presented. To achieve this, the heterogeneous components of different AC models are unified after reviewing the common models and the proposed metamodels in the domain (Chapter 2), then different examples of policy expressions related to different domains (local, social media, IoT . . .) are analyzed. Thereafter, the unified components are used as meta-components, with the relationships between them, for the HEAD metamodel and they are classified into three main components. The first is Explicit which refers to something that is real and exists in an organization (e.g., subjects and objects), the second is Implicit which refers to something described or explained in the AC rules (e.g., actions), and the third is Setting which refers to components that are included to have more accurate and regulated access to resources (e.g., constraints). Hereinafter, the relationships between the meta-components are depicted to allow creating different hierarchical levels of components, and instantiating various hybrid models. The meta-policy is expressed in terms of Explicit, Implicit, and Setting components, then any policy is expressed in terms of the derived components of the meta-policy. Accordingly, we designed a generic metamodel, able to create any needed component for any model, able to create different levels of hierarchy, able to extend the derived models since it flexibly allows creating new components/attributes, and dynamic to express larger set of static and dynamic AC policies. Moreover, the DSL language of HEAD metamodel is expressed using Eclipse-xttext, the syntax of the language is simple and flexible to appropriately express AC policy requirements and overcomes the complication of existing language expressions.

In Chapter 4, to verify the technical feasibility of HEAD metamodel, the DSL of HEAD metamodel is used to specify the needed AC policies for two case studies related to industrial environments inspired by ITMI's local (non-IoT) and IoT environments. The needed AC model(s) for each case study is derived, then xtend notation is used to transform the DSL and generate the needed java code which represents the concrete instance of the derived AC model. At the system level and after policy configuration, the output of java is Cypher statements which express the AC rules for each case study. For policy enforcement, the Cypher statements are injected into Neo4j database to represent the NGAC policy as a graph, then we run some Cypher queries to get NGAC authorization responses. The results show that HEAD metamodel can be adapted and integrated with local and distributed computing environments, able to serve as a unifying framework, answer the current AC requirements and follow the needed policy upgrades. Additionally, we implemented an administrative panel, using VB.NET and SQL, to show that HEAD metamodel can be implemented to generate AC rules using other platforms.

6.2 COMPARISON BETWEEN HEAD METAMODEL AND OTHER AC METAMODELS

In Table 8 we summarize the features of HEAD metamodel which distinguish it from the other existing metamodels.

Table 8
Comparison between HEAD Metamodel and other AC metamodels.

Metamodel	Access Control Metamodels	
Features	HEAD Metamodel	Other Metamodels
Unify components	Unify all the heterogeneous components of heterogeneous AC models.	Some metamodels unify some of the heterogeneous components under the notion of ‘category’ which includes roles, groups, security levels, etc.
Generality	Include all features and components of common AC models and allow deriving various instances of various models.	Hybrid structures to derive some AC models rather than generic metamodels
Dynamism	Allows defining and adding any type of components and attributes for existing models and non-existing ones.	None of the existing metamodels support this feature, and they are not dynamic enough to define static and dynamic AC policies.
Extensibility	New components can be defined and integrated with already derived models to support new AC features in addition to the previous ones.	Some metamodels are extended but not extensible, and none of the existing metamodels support this feature.
Hierarchical	It allows defining multi-levels of all components (e.g., role, context) to conform to the hierarchical organizational structures.	Some metamodels support the feature of hierarchy for some components, and none of them consider the context hierarchy which an important feature in complex and highly dynamic environments.
Upgradability	Able to follow technology upgrades and update any policy.	None of the existing metamodels support this feature, and they have reached their limits.
Unified framework	Allows creating any model, in addition to any hybrid model with different policy classes (e.g., case study 2), and hybrid models with hybrid components (e.g., case study 1)	Allow creating some models based on features employed in their hybrid structures.
Adaptability	Can be implemented in different centralized and distributed environments, especially IoT.	They provide solutions for specific cases and scenarios.
Novelty	A new development in the domain with advanced features (all of the above).	The last AC metamodel was proposed in 2015 (Abd-Ali et al., 2015).

6.3 FUTURE PERSPECTIVES

Throughout this thesis we show that the distinct design of HEAD metamodel and how it could be used to solve various AC issues associated with the emergence of ubiquitous computing and pervasive information systems. Its features allow developing essential issues in the domain and it opens new opportunities and various research directions. As future perspectives we aim to develop several theoretical and practical tools to extend its features in order to meet the different AC requirements with the presence of various technology trends.

As theoretical future perspectives, HEAD metamodel can be extended to address the following:

- In collaborative computing environments, distributed multiple cyber-physical areas interoperate with each other to provide an intelligent environment for users to achieve their collaborative activities. Hence, the features of the HEAD metamodel would facilitate fulfilling the collaboration and interoperability between the derived and heterogeneous AC models and their components among distributed multiple cyberphysical areas.
- The continuous technology upgrades impose the need to migrate AC policies from one model to another. The HEAD metamodel supports all the necessary tools to enhance with this new feature, in addition to its features.
- To develop the needed tools to analyze and assess the obtained policies at the run-time before enforcing them in order to avoid uncertainties concerning the obtained AC decision. Policy analysis and assessment are for assuring the quality of the generated AC policies and making sure that they are consistent, relevant, minimal, complete, and correct with respect to the required actions by subjects on some objects. This process is of major importance while implementing AC policies in highly dynamic and heterogeneous environments, especially IoT.

As practical future perspectives, HEAD metamodel can be extended to support the following:

- HEAD metamodel could be enhanced to dynamically generate AC policies according to users' context, profile, device, etc., since is based on a multi-level rule engine.
- HEAD metamodel could be enhanced to include built in packages or set of tools with predefined AC models, for example, the most used models DAC, MAC, RBAC, and ABAC. This could minimize the technical implementation efforts and facilitate administrative efforts to define policies.
- Since the definition for hundreds of thousands of attributes, expressing rules and policies, and performing implementation require a lot of time and resources, artificial intelligence (AI) techniques could be used, for example, object recognition to identify objects and define their attributes. Hence, the administrator could modify, add, or remove some attributes instead of creating hundreds of objects and thousands of attributes, especially in very large and complex systems.
- It could be enhanced and implemented to extract the data flow of an organization to define the rules after extracting the entities and attributes, then the administrator could be notified if there are some missing rules that must be defined.
- The features of the HEAD metamodel allow integrating AC to other systems, for example, intrusion detection systems. The flexible, upgradable, and dynamic nature of the HEAD metamodel makes it adaptable and applicable to distributed systems; hence, it allows developing and implementing blockchain-based AC systems.

Each of the above points is itself an opportunity (in addition to many other possible opportunities) for enhancing AC in today's computing environments based on a unified framework. In summary, the HEAD metamodel contributes to the development of knowledge in the field and can be considered as a paradigm shift toward reshaping the existing models and AC methods, especially in the field of industry.

APPENDIX I

Future of Information and Communication Conference, 14-15 March 2019, San Francisco, US
pp. 892-911. Springer, Cham, 2019, https://doi.org/10.1007/978-3-030-12385-7_61

From Access Control Models to Access Control Metamodels: A Survey

Nadine Kashmar¹, Mehdi Adda² and Mirna Atieh³

^{1,2} Université du Québec à Rimouski, Rimouski QC G5L 3A1, Canada

kasn0002@uqar.ca¹, mehdi_adda@uqar.ca²

³ Lebanese University, Hadat, Lebanon

matieh@ul.edu.lb

Abstract. Access control (AC) is a computer security requirement used to control, in a computing environment, what the user can access, when and how. Policy administration is an essential feature of an AC system. As the number of computers are in hundreds of millions, and due to the different organization requirements, applications and needs, various AC models are presented in literature, such as: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC), etc. These models are used to implement organizational policies that prevent the unauthorized disclosure of sensitive data, protecting the data integrity, and enabling secure access and sharing of information. Each AC model has its own methods for making AC decisions and policy enforcement. However, due to the diversity of AC models and the various concerns and restrictions, its essential to find AC metamodels with higher level of abstraction. Access control metamodels serve as a unifying framework for specifying any AC policy and should ease the migration from an AC model to another. This study reviews existing works on metamodels descriptions and representations. But, are the presented metamodels sufficient to handle the needed target of controlling access especially in the presence of the current information technologies? Do they encompass all features of other AC models? In this paper we are presenting a survey on AC metamodels.

Keywords: Metamodel, model, access control, policy, security.

1 Introduction

As long as there is an increase and development in the use for internet services, there is also an increase and a critical need for computer security solutions [1, 2]. Computer security and the related topics were and still the main issues in the world of Information Technology (IT).

Over the years until today, IT security and privacy are critical concerns for academic, economic, social, industrial, and governmental organizations. Access Control (AC) is one of the most important and critical aspects of IT security. For this purpose, different AC models and policies are developed. In literature, various AC

models are proposed, such as: Discretionary Access Control (DAC) [3, 4], Mandatory Access Control (MAC) [3, 5, 6], Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC) [3, 4, 5, 7], and Organization Based Access Control (OrBAC) [5]. Each AC model is developed either to overcome limitations found in previous models or as a solution for a specific use case and application.

Finding a unified AC model becomes a significant issue due to the need to include all features offered by the existing AC models, which in some cases, are incompatible and irreconcilable. Also, due to the widespread and heterogeneity of interconnected networks and distributed systems, heterogeneity of platforms and applications, and due to diversity of users, the necessity to design a well coherent AC architecture for enterprises becomes a must. In this context, different AC metamodels, to address these issues, are presented in literature to serve as a unifying framework for specifying and enforcing different AC policies. In the following sections we will summarize existing AC metamodels. Nevertheless, before describing and comparing these metamodels, we have to explore some basic concepts related to AC.

The remaining of this paper is organized as follows: Section II summarizes the existing AC models, their advantages and limitations. The AC usage in different system levels are presented in section III. Section IV describes the concept of AC metamodels, the metamodeling tools, and provides a comprehensive overview about the existing AC metamodels. Section V presents the potential research issues. Section VI concludes this paper.

2 Access Control Models

2.1 A Brief Overview

Access control is defined as an essential security requirement in the field of IT. Each organization has its own information system where a set of policies is defined based on conditions where users can access all or some system resources. Implementing these policies is essential to protect resources. AC methods are carried out at different IT infrastructure levels. They are used in operating systems, databases, networks, and information systems. The goal is to protect files, directories, regulate access to database objects and fields, and protect applications' information (payroll processing, e-health...) etc. However, the primary objective of access controls is the fulfillment of the defined AC policies [1, 2].

Generally speaking [3, 4, 5], AC models and mechanisms are defined in terms of subjects, objects and access rights. The subject concept usually refers to a user or program; the object concept refers to an entity a user wants to have access to such as a file, a table or a class. However, a subject may or may not have an access right to an object. Access right means that a subject is able or perform an operation on an object. The operation may be read, write, execute, etc. In other words, a user (subject) can perform an operation (read, write...) on an object (file, class...) if he has a permission to do so. To carry out an operation, access rights are required. To manage who and how operations must be carried out, privileges or AC must be defined. Data resources are protected under different access policies. A model is the projection of the scope of

policies and the needed behavior between subject and object entities. Policies are a set of guidelines, which are generalized, abstracted, formally, or semi-formally described [3]. Several research surveys are presented in literature with detailed descriptions about AC models. In the following sections we summarize the concept of each model.

2.2 Discretionary Access Control (DAC)

In late 1960s, Discretionary Access Control (DAC) model was first introduced by Lampson, a member of a curriculum design team. In DAC, the system protection notion includes three major components: a set of objects, a set of domains, and a matrix. Lampson's work was then extended by Graham and Denning, where the term "subject" was included instead of the domain. Then, the extended Lampson's work was developed by Harrison, Ruzzo and Ullman (HRU) to find a formal proof that tracking privilege propagation was undecidable in general [3].

DAC is defined as a user-centric AC model in the sense that a file owner determines the permissions that are assigned to other users requiring access to the file [4]. DAC mechanism allows users control the access rights (read, write...) to their files without the need of a pre-specified set of rules. The access rights are specified by Access Control Matrix (ACM), where AC rights of subject(s) over object(s) are specified. Other variations of implementing AC matrix include Capability Lists (CLs) and Access Control Lists (ACLs). In the concept of CLs the user access rights are stored by rows, whereas in ACLs the access rights for various users on a file are stored by columns. Lampson and Harrison Ruzzo Ullman (HRU) are two DAC models [3].

DAC model is very flexible to assign access rights between subjects and objects. But it also has limitations where the system maintenance and verification of security principles are extremely difficult, since users control access rights to their owned objects. Also, the possible attacks for Trojan horses [3, 5].

2.3 Mandatory Access Model (MAC)

In 1970s, Mandatory Access Control (MAC) protection was presented to include the use of a security kernel. In 1987, a paper was published in IEEE Symposium of Security and Privacy, where crucial differences between commercial and military security requirements were presented by Clark and Wilson [3].

In MAC users cannot define AC rights by themselves. The AC policy is managed in a centralized manner. MAC model is based on the concept of security levels associated with each subject and object, where permissions and actions are derived. Security classes have hierarchical and nonhierarchical components. The hierarchical components include types: unclassified (U), confidential (C), secret (S), and top-secret (TS) where $TS \geq S \geq C \geq U$, to classify subjects and objects into levels of trust and sensitivity. For objects a security level is called the classification level and for subjects it is called clearance level. The nonhierarchical component is represented by a set of categories. Security labels indicate security levels for classification of objects and clearance of subjects, and uses two security properties, simple security property and *-property. Bell and LaPadula (BLP) of multi-level security and BIBA are two

MAC variants. In BLP, a subject is allowed to read an object if the subject's clearance is \geq than the object's classification, and to write if it is \leq . In BIBA, a subject is allowed to read an object if the subject's clearance is \leq than the object's classification, and to write if it is \geq [3, 5].

This model is presented to overcome the limitations of DAC model, which is the Trojan Horse attacks, by centralizing access control. In [6], it is mentioned that MAC model is relatively straightforward and is considered to be a good model for commercial systems that operate in hostile environments such as web servers and financial institutions where the risk of attack is very high. Also, it has limitations, since it is difficult to implement due to the dependence on trusted components, and the necessity for applications to be rewritten to adhere to MAC labels and properties. Similarly, the assignment of security levels by the system places limits on user actions which prevents dynamic modification of the original policies.

2.4 Role-Based Access Control (RBAC)

Based on historical practices, Role-Based Access Control (RBAC) was defined as a job a user performs, and he/she can be assigned one or more roles to indirectly associate permissions with users [3].

RBAC model is considered as an alternative approach to MAC and DAC. In RBAC, users can be assigned several roles and a role can be associated with several users [4]. A role means a collection of permissions to use objects to perform a job function that combines the authority and responsibility assigned to a subject who plays this role, e.g. accountant, director, engineer, etc. each role is associated with privileges or permissions [3]. The aim of RBAC is to facilitate the administration of the AC policy. It governs the access of a user to information through roles for which the user is authorized to perform. RBAC model is based on several entities, which are, users, roles, permissions, actions, operations, and objects. Each role can have many permissions, and permissions may be assigned to many roles. A subject can operate or play many roles and a role can be performed by different subjects [5]. Several RBAC models are proposed in the literature, Flat RBAC (RBAC0), Hierarchical RBAC (RBAC1), Constrained RBAC (RBAC2), and Symmetric RBAC (RBAC3) [5, 7].

This model has many benefits, it has central administration of role memberships and ACs. It may also be applied in distributed areas because it is based on the concept of constraints and inheritance [5, 6]. In RBAC, role hierarchy specifies which roles and permissions are available to subjects based on different inheritance mechanisms. Role hierarchies and permission inheritance in RBAC models are explained in [8, 9]. Also, in distributed areas where different resources are shared among users, RBAC has powerful means of specifying AC decisions [10]. Conversely, it also has some drawbacks. It is frequently criticized for the difficulty of setting up an initial role structure and for inflexibility in rapidly changing IT technologies. For example, RBAC provide poor support for dynamic attributes such as time of day, which might be needed when determining user permission [11]. Another drawback is reflected in large systems, where role inheritance and the need for customized privileges make administration potentially heavy [6].

2.5 Organization Based Access Control (OrBAC)

Organization Based Access Control (OrBAC) is first presented in 2003. The aim of this model is to solve some problems in the previous AC models (DAC, MAC and RBAC), to find a more abstract control policy. It is designed to address the subject, object and action, in such a way that the policy determines what subject(s) has some action(s) to access some object(s). Each organization (clinic, banks, hospitals...) is comprised of a structured group of subjects having certain roles, or entities. This model exceeds the concept of only granting permissions to subjects, it also addresses the concept of prohibitions, obligations and recommendations [5]. A role may have a permission, prohibition or obligation to do some activity on some view given an associated context. In this model seven entities are defined, a) the abstract level or organizational (1- Role, 2- Activity, 3- View) and b) the concrete level (4- Subject, 5- Action, 6- Object). The seventh entity is Context lies between the two levels to have a correspondence between the elements of each level. The context is presented in OrBAC to express dynamic rules for relations between entities, for example, Permission, Prohibition, Isprohibited, Recommendation, Ispermitted, Isobligatory, Isrecommended, Obligation [5, 12].

The OrBAC has an advantage in eliminating conflicts between security rules. It also has some vulnerabilities to some kinds of attacks, e.g. covert channels [5].

2.6 Attribute-Based Access Control (ABAC)

Attribute Based Access Control (ABAC) concepts have paralleled that of RBAC. It has some advantages over RBAC, because of its benefits in authorization management and its ability to support dynamic attributes. The main idea behind ABAC is to grant or deny user requests based on arbitrary attributes of the user and selected attributes of the object that may be globally recognized [3, 11]. It enables precise AC which allows a higher number of discrete inputs into an AC decision. This provides a larger set of possible combinations of variables to reflect a larger set of possible rules to express policies [8]. Recently, this model gained attention from businesses, academia and organizations due to the limitations in RBAC model. Two standards that widely address the ABAC framework are: The Extensible AC Markup Language (XACML) and Next Generation AC (NGAC) with AC facility for applications and other important features [3].

In ABAC, subjects are enabled to access a wider range of objects without specifying individual relationships between each subject and each object. As well as, there are three types of attributes: subject, object and environmental attributes. The first two are common to all ABAC models. The third type used in some models that depend on the availability of system sensors that can detect and report values, they may include the current time or day of the week. Despite the benefits of ABAC model over the other models and its flexibility to assign policies and security features, it has a number of limitations such as [2]:

- The problem of determining the current set of permissions available to all users.
- Its implementations require significant time to run.

- It is often not possible to compute the set of users that may have access to a given resource.
- It is difficult to efficiently calculate the resulting set of permissions for a given user as all objects would need to be checked against all relevant policies.

The historical AC models are summarized to provide an overview of their concepts, benefits and limitations. This overview gives a conception about creating new AC models, combining some of their features, or finding models with higher level of abstraction (metamodels).

3 Access Control Usage

This section explains how AC models are generally integrated in any Information System (IS). In general, the IS is composed of six components: hardware, software, data, procedures, people and communication. IT security concepts are related to each component of any IS. For a secure environment, security must be carefully managed. Moreover, any connected computer to the internet is vulnerable to attacks. Thus, the IS components are also under threat. Subsequently, various AC models and principles are deployed to protect the IS environment. Their function is to control which object have access to which subject in the system (files to read, programs to execute, data to share, etc.). Fig. 1 shows AC at different levels in a system. Applications may be written on top of middleware, such as a database management system. The middleware use services provided by the underlying operating system. Also, the operating system of ACs usually relies on hardware features provided by the processor or by associated memory management hardware [13].

The IT security requirements are widely explained in literature. Some examples of these requirements are: protection from improper access, user authentication, data integrity, etc. In the following sections we will present AC usage in operating systems, databases, networks, cloud computing, and Internet of Things (IoT).

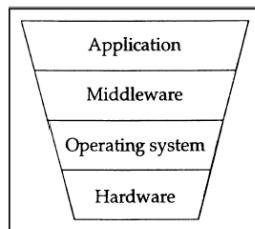


Fig. 1. Access control levels in a system [13].

3.1 Access Control in Operating Systems

Access control mechanisms are provided with Operating Systems (OSs) to authenticate system administrators and users using some procedures, for example, passwords. After authentication, AC play an important role in allowing users to access files, communications ports, and other system resources. User permissions are modeled as a

matrix of access permissions, where columns represent files/folders and rows represent users. In this context, a file owner determines the permissions that are assigned to other users requiring access to the file. However, only these users may have some permissions (privileges), on the file, to read, write, execute, etc. An ACM is used to show the users' access rights on a system and the files in it [4]. Although ACMs can be used to implement protection mechanisms, they don't scale well with many users. For example, in an institution with a large number of users and applications, a plenty of entries will occur and this will cause a performance problem and administrative mistakes. Two practical ways to overcome those issues are presented in [13]. Firstly, by using groups or roles to manage the privileges of large sets of users simultaneously. Secondly, by storing the ACM either by columns (ACL) or rows (CLs).

3.2 Access Control in Databases

Today, it is common for institutions to have databases (DBs) with critical data, such as: bank accounts, employment records, hospital reports, etc. Thus, the security needs become very critical, especially for online users. DB management systems (Oracle, SQL...) have their control mechanisms and users should have different types of permissions based on their job functions. In this context, OS play an important role to separate the DB from other applications running on the same computer by identifying users. Besides, ACLs and CLs are mixed to provide the needed mechanisms for DBs [13]. Database users should have different types of permissions based on their job functions. Each user should only have the permissions which are granted for him after his login to the system, which is known as authorization. These permissions are assigned to determine the user actions within the DB.

Different DB tasks and maintenance procedures must be occurred, these tasks must be implemented by DB administrators. These tasks include creating DBs, removing unneeded DBs, managing disk space allocation, monitoring performance, and performing backup and recovery operations. DB platforms allow default system administrators to perform such tasks and delegate permissions to other users [14].

3.3 Access Control in Networks

Network (NW) security is also an essential part of IS. In this domain, different issues must be taken into consideration, which are: the NW design, NW device security, firewalls, virtual private NWs, Intrusion Detection and Prevention Systems (IDPS), etc. The section below describes and summarizes these considerations [15].

The requirements of NW security and budgets are specified based on NW design. For this, different aspects must be taken into consideration in designing NWs, such as: availability, cost, performance, number of users, etc. In security, it is important to enable effective and secure links to other NWs, provide a platform that is helpful for securing sensitive NW assets, and identify critical security controls and understand the consequences of a failure of these controls. Also, NW device security concern is how to use routers and switches to increase the security of the NW. The internetworking protocol in use today is known as Transmission Control Protocol/Internet Protocol (TCP/IP). It is a suite of protocols and applications that have discrete functions that map to the Open Systems Interconnection (OSI) model.

Each connected device on a NW has two NW addresses: The Media Access Control (MAC) address, and the IP address. However, there are several configuration steps to configure the device (router, switch...) for increased security. The various steps are: switch security practices, ACLs, administrative practices, Internet Control Message Protocol (ICMP), logging to routers, etc. Furthermore, firewalls are defined as the first line of defense between the internal NW and untrusted NWs like the Internet. They play an important role in controlling application communication and other functions, such as: Network Address Translation (NAT), antivirus, e-mail (spam) filtering, IDPS, etc. Virtual Private Networks (VPNs) are created by establishing a virtual connection using dedicated connections is to provide a secured communication channel over public NWs. To secure VPNs, different issues must be addressed, e.g. the authentication process, client configuration, etc. A security administrator always checks the system and security log files looking for something abnormal. Audit tools are used by administrators to detect a wide range of rogue events, such as: unauthorized registry changes, protocol attacks, Denial of service (DoS) attacks, etc. Also, for website security different methods are handled, e.g. prevention of SQL injection, using complex passwords, management of cookies, and others.

The AC models are developed to match the security needs in all aspects of IT. In the presence of new technologies, such as: Cloud Computing and IoT, it is worth presenting some AC mechanisms in such fields.

3.4 Access Control in Cloud Computing

Cloud Computing (CC) is an emerging technology. Due to the huge amount of data which are generated from different end users' applications and information systems, CC is considered as an efficient solution for easier and faster storage retrieval of data. In CC, users can access computer services via the internet and this makes their data vulnerable to attacks. For this reason, different AC methods for CC are presented in literature. This section presents some of AC in CC methods.

Onankunju, in [16], introduces a method for providing secure AC in CC after presenting the possible CC attacks, e.g. DoS and authentication attacks. Hence, a hierarchical structure using a clock is presented to upload, download, and delete files to and from the cloud. The root of the hierarchical structure is the trusted authority which authorizes the top-level domain authorities. Also, domain authorities authorize cloud users. The system is composed of 4 parts: cloud owner, untrusted cloud, clock and cloud users. The user, with a key, encrypts his data before uploading it to the untrusted cloud. For the user, to access his data, he should send a request to the cloud owner which in turn sends him back a key. The key remains available for a certain period, then it becomes invalid after the clock stops counting. The user should access his data within the time limit. Another method is proposed in [17]. The method avoids using static passwords, it uses a one-time password and one day password. Whereas, the first password expires in two minutes, and the second one after twenty-four hours. The user receives passwords with encryption via e-mail for each login session.

3.5 Access Control in IoT

In [18], IoT is defined as “a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols”. Which means a huge number of heterogeneous objects, with different technologies and platforms, are communicating together via the internet. Hence, all devices connected to the internet are vulnerable to attacks, and this is the case for IoT devices. In this context, various authentication and AC methods in IoT are also presented in literature, to integrate security issues with this technology.

Liu et al. in [19] propose a model to find a secure communication between things by a certain procedure. The main idea of this procedure is based on verifying identities between two IoT devices. The method is based on implementing authentication protocol in the presentation layer, where identification key establishment occurs. They adopt the concept of authorization in RBAC model, and for secure key establishment they implement Elliptic Curve Cryptosystem (ECC). Moreover, authors in [20] propose a “smart contract-based framework to implement distributed and trustworthy access control”. Their aim is “to apply the smart contract-enabled blockchain technology to achieve distributed and trustworthy AC for the IoT”. This framework contains multiple Access Control Contracts (ACCs), one Judge Contract (JC), and one Register Contract (RC). ACCs are implemented between subjects and objects for AC. JC is used for judging the unpleasant behavior of the subject during AC. RC is used to manage the ACCs and JC. To demonstrate the framework feasibility, case studies are addressed.

Different other AC proposals are presented in this field, which reflects the evolutionary stage of CC, IoT, and security concerns due to the presence of attacks.

4 Access Control Metamodels

The need to use AC methods in different system levels and technologies, imposes the necessity of finding AC models with combined features from two or more models. Due to the continuous increase and upgrade of information technology features, the presence of security threats also increases. The technological environment which is open to all types of users is a crucial concern, because it is also open to various types of attacks. This makes security enforcement, through AC models, an urgent need. So, many AC models are presented in literature with combined features from two or more AC models based on research motivations and needs. For example, we can find many models with combined features from both RBAC and ABAC. Some hybrid RBAC and ABAC models and others are presented for this purpose.

In [11] and due to RBAC’s difficulty to set up an initial role structure in rapidly changing environments, and because RBAC does not support dynamic attributes, the idea of adding attributes to RBAC is addressed. The aim is to find a model that supports dynamic attributes specially in organizations. These features are presented to handle relationship between roles and attributes to provide better AC features in dynamic environments. Also, authors in [21] target the idea of enhancing features from both RBAC and ABAC, because both have complimentary features to each

other. Hence, Attribute Enhanced RBAC model (AERBAC) is presented. The model “retains the flexibility offered by ABAC, yet it maintains RBAC’s advantages of easier administration, policy analysis and review of user permissions [21]”.

However, AC models must consider the continuous developments and changes. The new technologies (CC, IoT...), the variety of platforms and applications, users’ types, etc. comprise a complex fact in controlling secure and private accesses to the needed resources in different areas. All this makes AC models and even combining some features of them are insufficient to handle the needed target. This fact forces the need to find models with higher level of abstraction, which is called AC metamodels. The aim of AC metamodels is to serve as a unifying framework for specifying and enforcing any AC policy. For this purpose, different research works are present and still conducting for finding an AC metamodels. The following sections present some existing AC metamodels.

4.1 Metamodel Definition

Before exploring the existing AC metamodels, it is worth mentioning some essential definitions and ideas about metamodeling. Metamodel in [22] is defined as a textual, graphical/visual, or formal representation of concepts and how they are linked together. In other words, it is a structure of a collection of concepts in a certain domain. These concepts might be terms, rules, guidelines, etc. for an institution or organization. In [23], metamodeling is defined as modeling of a model, where they should describe the permitted structure to which models must adhere. Furthermore, models and metamodels need adaptable supporting tools due to changing requirements and policies. As mentioned earlier, metamodels can be illustrated using textual, graphical or formal representations. Though, different metamodeling tools and languages are presented in [22, 23] such as: Unified Modeling Language (UML), Meta-Object Facility (MOF), Eclipse Modeling Framework (EMF), MetaEdit, Conceptbase, and other tools. Fig. 2 shows the metamodel abstraction levels. M_3 is an instance of a model, it defines a specific information domain. M_2 is an instance of metamodel, it defines a language to describe an information domain. M_1 is an instance of a Meta-metamodel, it defines the language for specifying a model. M_0 is the infrastructure for a metamodeling architecture, it defines the language for specifying metamodels [22].

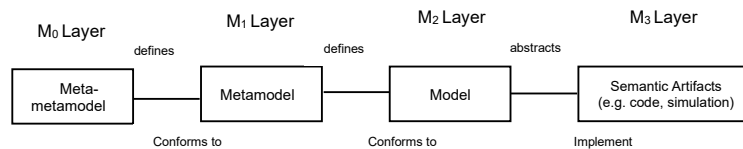


Fig. 2. The Four-layer metamodeling Architecture [23].

4.2 State of Art

Korman et al. in [24] propose a unified metamodel designed using Enterprise Architecture (EA) modeling language, the ArchiMate, and explain its use on many

scenarios and two business cases. Their aim is to combine different AC models in a single EA model, and to propose an extension to an established EA modeling language. Their model targets the challenge of flexibly modeling policies of authorization according to the most well-known AC models: DAC, MAC (BPL, BIBA, and Chinese Wall), RBAC, and ABAC, in terms of EA. The authors summarize some of the existing AC models, then define the vocabulary of these models, such as: subjects, objects, sessions, etc. to use them later in their presented metamodel. Then, they represent the conceptual models for the most basic common terms of AC, which are: subject, object, and access mode for the same purpose of later use. Also, the configurations of DAC, BLP, BIBA, Chinese Wall (CW), RBAC_{0,1,2,3}, and ABAC are represented as metamodels, e.g. in Fig. 3 they represent the metamodel for ABAC. Finally, these predefined steps are used as an introduction before presenting their unified metamodel in Fig. 4. The model mostly builds on the conceptual model of ABAC for its ability to emulate most functions of the other AC models.

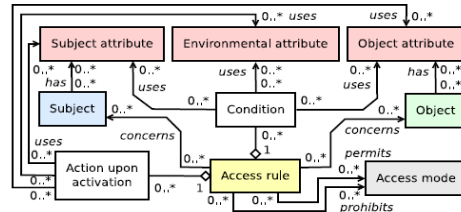


Fig. 3. Metamodel for expressing configurations of ABAC [24]

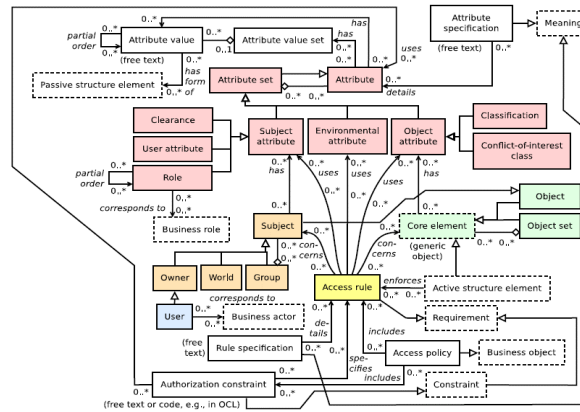


Fig. 4. Unified metamodel for modeling authorization [24]

Authors in [25] propose an AC metamodel to concurrently handle multiple AC models. Their metamodel consider four models, which are: CW, BLP, BIBA, and RBAC models. They start from the general concepts of metamodels; the object, subject, access mode, role, and the instances of associations between metamodeling elements (subject and role). They present the decision concept which relies behind

applying logical rules that are expressed in terms of AC metamodel elements. As proposed, each AC metamodel has a special element called Decision Handler. Fig. 5 illustrates the initial concept of their metamodel. Then, kernel AC elements are presented to later illustrate the associations between metamodel elements. These elements are: Object, Objects Group, Subject, Subjects Group, Access Mode, Additional Attribute, Query, Environmental Attribute, and Decision. Objects Group and Subject Group respectively represent sets of objects and subjects sharing some properties. The AccessMode represents, for example, some actions like: read, write, execute, etc. Query is the access request on an object by a subject. Environmental Attribute is used to hold information related to access request events such as time, place, temperature, etc. Additional Attribute represents a construct associated to a metamodel element to support the specification of some property of that element. Decision (e.g. permit, deny, indeterminate, NotApplicable...) is where the response is issued by the AC request. The metamodel is implemented with the four AC models (CW, BLP, BIBA, and RBAC) as shown in Fig. 5. ACmetaModelElement is a generalization of any element of the AC metamodels other than DecisionHandler. The proposed AC metamodels relies on a subset of UML but without determining how an instance of the metamodel returns an AC decision. To do so, First Order Logic (FOL) mapping is used for relating entities to their types, specifying relationships between entities, and for expressing a decision logic based on relations. To specify the integration of several AC metamodels of a hybrid AC policy, they presented an example of Integration Metamodel (IM) based on Ascending Decisions Tree (ADT). Hybrid AC policies mean applying multiple AC specifications and policies. Fig. 6 shows the ADT metamodel example which concludes with only one AC decision as output, in response to a set of multiple AC decisions as input. ADT nodes are, DecisionHandler instances and nodes applying Combining Algorithms (ComAls). DecisionHandler instance issues its AC decision based on its metamodel decision logic (explained in [25]). ComAlNode issues its decision by applying its ComAl on the decisions of its direct children nodes. It has a unique root which returns the decision of the whole tree with the hybrid AC policy. Finally, the proposed IM metamodel is illustrated in Fig. 7, which encompasses the whole above concepts.

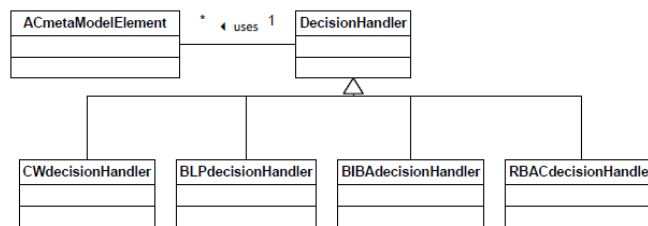


Fig. 5. DecisionHandler specializations in access control metamodels [25]

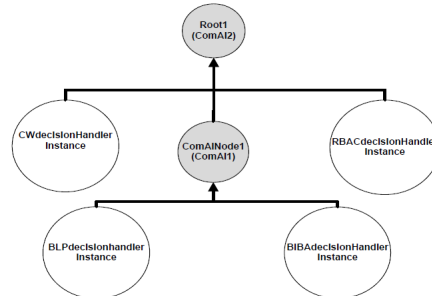


Fig. 6. An ADT integrating multiple AC metamodels instances [25]

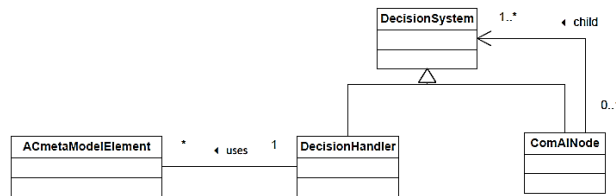


Fig. 7. The main view of IM metamodel [25]

Besides, designing AC metamodels for distributed environments (consisting of several sites) are also taken into account. This is due to the importance of finding dynamic or collaborative policies, to consider system changes or collaborate with other policies of several sites. This concept is covered in [26], where authors present the importance of finding a formal specification language to define AC models and policies in such environments. For this purpose, the concept of rewriting techniques (for security policies and protocols) is suggested to provide semantics for distributed AC mechanisms. Also, they mention some several available rewrite-based programming languages for fast prototyping, such as, Muade. Their proposed rewriting techniques are defined as an instance of a metamodel based on Distributed Event Based AC model (DEBAC). However, the authors first explain the advantages of rewriting systems, then explore the existing AC models to introduce the concept of a unifying metamodel for AC. Also, they describe the main features of the AC metamodel and define the extension of the metamodel for distributed environments. Second, they propose a formal specification of the distributed metamodel in a rewrite-based language, constructed on core concepts of AC models, and focus on the modular properties of the system. In this context, the federation notion is considered, like database systems where several systems are integrated by a federated system and each system preserves its autonomy. Third, general policy combining operators are well-defined to define combinations of policies. Algebraic terms are used in all the steps to define and rewrite policies, properties, operational semantics of the distributed metamodel, and integrating combination operators in the distributed metamodel. In [26], detailed explanations about the used expressions are also provided. Similarly, a metamodel extension mechanism is proposed as a solution in

the context of MoNoGe French collaborative project [27]. This project is based on a textual Domain Specific Language (DSL). The aim is to face up current limitations and the lack of standard solutions in the existing project. In addition to building a generic lightweight metamodel extension approach for the industrial environment where rapid and efficient adaptations of the used modeling tools are needed. Authors begin by defining the concept of modeling in real industrial projects, which deal with different models and metamodels, in addition to supporting tools. Hence, they present the main industrial use case of MoNoGe, which comes from DCNS (a world-leading company in naval defense and energy that especially develops Combat Management Systems, CMS, for ships). DCNS use two separate modeling tools: DoDAF (U.S. Department of Defense Architecture Framework) standard, and Modelio supporting software design and development. Then, a metamodel extension operators and a DSL are introduced to easily use them. Also, two different implementations of their proposed extension mechanism, based on Eclipse/EMF and the Modelio modeling environment, are also described. Fig. 8 shows a sample of the grammar of their proposed metamodel extension textual DSL.

```

Model: 'define' extensionName=ID 'extending' metamodel+=Metamodel ':'
      prefix+=Prefix ("," metamodel+=Metamodel ':' prefix+=Prefix)*
      '{' extensions += Extension* '}';
Extension: Create | Refine | Generalize | ModifyClass | FilterClass;
Metamodel: name=ID;
Prefix: name=ID;
Create: 'add class' class=ID;
Refine: 'add class' classNew=ID 'specializing' prefix=[Prefix] '.'
      classOriginal=ID;
Generalize: 'add class' classNew=ID 'supertyping' prefix+=[Prefix]
          '.' class+=ID ("," prefix+=[Prefix] '.' class+=ID)*;
ModifyClass:
    'modify class' prefix=[Prefix] '.' class=ID '{'
    modifyOperators += ModifyOperator*
    '}';
ModifyOperator: AddProperty | ModifyProperty | FilterProperty |
    AddConstraint | FilterConstraint;
AddProperty: 'add property' property=ID 'type' type=ID;
ModifyProperty: 'modify property' property=ID value+=ValueAssignment
    ("," value+=ValueAssignment)*;
ValueAssignment: attribute=ID '=' value=EString;
FilterProperty: 'filter property' property=ID;
FilterClass: 'filter class' prefix=[Prefix] '.' class=ID;
AddConstraint: 'add constraint' constraint=ID value=EString;
FilterConstraint: 'filter constraint' constraint=EString;

```

Fig. 8. The Grammar metamodel extension textual DSL [27]

Furthermore, AC metamodels also consider the Web Content Management Systems (WCMSs). WCMSs are frameworks that are widely used for web applications development for enterprises, e.g. Drupal, Wordpress, and Joomla. Users with little technical knowledge can fully develop technical systems due to their integrated environment. This environment provides design definition, layout, content management and organization of the application. In this context, Martínez et al. in [28] highlight the importance of security requirements, since WCMSs may contain sensitive information. Authors propose a metamodel to the representation of WCMS AC policies, to ease the analysis and manipulation of security requirements by abstracting them from vendor-specific details. They focus on the idea of facilitating WCMS configuration, to minimize the possibility of vulnerabilities because users often lack depth technical and security knowledge. Besides, authors enumerate some law-level security aspects, for example, management of cookies, and prevention of

SQL injection vulnerabilities. Although AC techniques are integrated in most WCMS systems, some limitations still exist in such systems, e.g. knowing the level of protection of the implemented access policy in a WCMSs is complex and error-prone task. For this purpose, authors propose to raise the level of abstraction of the AC implementation, to be represented according to a vendor-independent metamodel. Fig. 9 represents the proposed WCMS metamodel sample, which is inspired by RBAC.

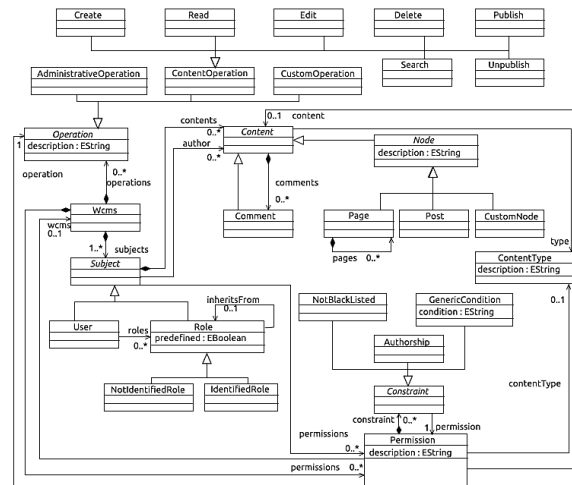


Fig. 9. WCMS metamodel sample [28]

The four metamodel basic elements functionalities explained in [28] are: content, actions, permissions and subjects. The idea of their process is to automatically extract the AC information in the domain of WCMSs. Moreover, they mention that even their metamodel could be manually filled by investigating AC information using WCMS administration tools, it should also be filled by an automatic reverse engineering approach. Thus, they present an automatic process for Drupal in Fig. 10, where Drupal contents with AC information are stored in the backend database. SQL queries over the database are injected to obtain a model that conform their proposed WCMS metamodel. This process allows the possibility of defining extra AC rules or modifying them programmatically. The abstract representation of the WCMS AC model is developed using Model Driven Engineering (MDE) where the relation between metamodel elements can be easily realized. Concerning WCMS migration, they present using their metamodel as a pivot representation. This illustration is to represent the AC information of the old WCMS in a way corresponding to their metamodel, to facilitate its analysis. Another web service metamodel is proposed in [29] to handle the verification of authorization in Web Service Oriented Architecture (WSOA). The aim of this metamodel is to improve the existing AC models for better features to match the requirements of WSOA. The proposed metamodel is depicted in Fig. 11. The metamodel is an enhancement for hierarchical RBAC and ABAC. Conceptual UML modeling is used to present the metamodel and define the sets and

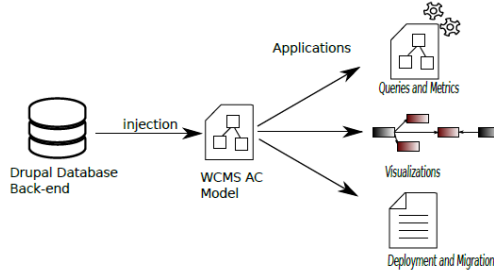


Fig. 10. Drupal access control extraction [28]

relations. In this metamodel the commonly used type of operations, e.g. read, write, which are carried between permission and object elements are removed. Instead, the focus is on placing the input parameters of web service operation. As shown in Fig. 11, there are two relations from permission to an object: 1) indirect relation via policy, 2) direct relation to the input parameter, where “a parameter does not need to know if its value is evaluated for access control [29]”. Web services Composition “consists of multiple invocations of other Web Service Operations in a specific order [29]”. This element plays an important role to stop execution in case of missing authorization at an early stage. The proposed metamodel is mapped to an authorization verification service, which is part of Identity Management (IdM) architecture. Then it is linked with the core concern of WSOA, and the feasibility of this approach is illustrated in a case study. Also, Web Services Description Language (WSDL) is used for service interface definition. Likewise, addressing network security is also a critical concern. Martínez et al. in [30] propose a model driven approach to extract network AC policies enforced by firewalls within a network system. Their concept tackles the problem of filtering the traffic of a network with the presence of a number of filtering rules. based on their analysis, “the network topology, that may include several firewalls, may impose the necessity of splitting the enforcement of the global security policy among several elements. [30]”. In this context, their aim also is “to raise the

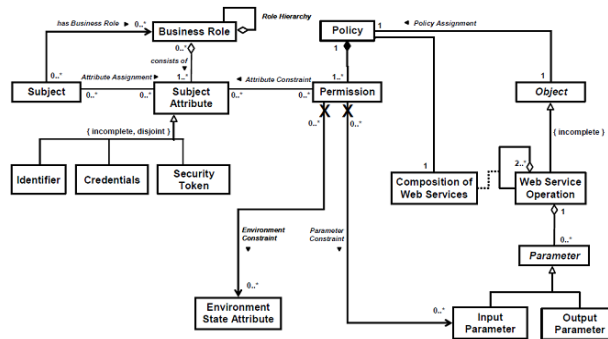


Fig. 11. Metamodel for AC in WSOA [29]

level of abstraction of the information contained in the firewall configurations files so that the AC policy they implement is easier to understand, analyze, and manipulate [30]”. However, Eclipse tool (Xtext) is used to extract AC information out of the net-filter iptables language. The features of RBAC and OrBAC AC models are implemented in this approach. Fig. 12 shows the network connection metamodel. The metamodel consists of two entities, host and connection. The former represents a network host, e.g. IP address. The latter represents connections between hosts, where the port and the protocol are specified to establish connections and specify if the connection is allowed or denied.

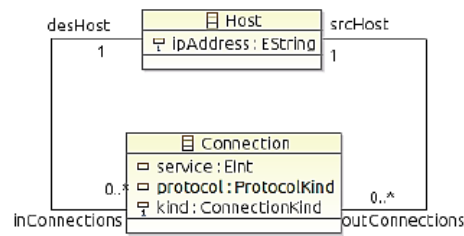


Fig 12. Network connection metamodel [30]

5 Potential Research Issues

Developing AC metamodel, that covers the features of all other AC models, is a challenging issue especially if the aim is to find a metamodel that encompasses all existing AC models. The dynamic requirement for enforcing security issues and the rapid propagation of technology makes it an urgent need.

The presented metamodels come with some advantages, and many case studies are addressed to handle each metamodel design. But we notice that each metamodel is itself a case and does not encompass a general base concept. Table I summarizes the presented AC metamodels, which depicts different metamodels in IT systems. Also, it indicates that metamodeling is a recent research field, especially in the last few years. As we notice, all the presented metamodels are designed for dedicated case scenarios or projects based on some features of AC models. Furthermore, in addition to the achieved progressions, there are still issues to be addressed. In fact, despite the advantages of the presented metamodel in [24], authors present some of its limitations. For example, it misses the concept of logging, in addition to the difficulty for a potential implementation of automated analytical capabilities of the unified metamodel. So is the case for the other metamodels, they are not generic enough to include all AC models features. As we can see some combined features from some models to cure some existing deficiencies in some projects or enhancing some service features. In addition to the concept of applying the same complex process in assigning relationships between model elements in some metamodels.

Table 1. Summary of Presented Access Control Metamodels.

Ref.	Publication	Metamodel Features			
		Designed for	Type	Used Models	Modeling tools
[24]	2016	Enterprise architecture	Unified	DAC, MAC RBAC, ABAC.	ArchiMate
[25]	2015	Enterprise	Hybrid	CW, BLP, BIBA, RBAC	UML, FOL
[26]	2014	Distributed Environment	Metamodel Extension	DEBAC	Rewrite semantics
[27]	2015	Industrial project	Metamodel Extension	MoNoGe project	DSL, EMF, Modelio
[28]	2013	WCMSs	Metamodel Extraction	RBAC	MDE
[29]	2007	Web Service	Metamodel Integration	RBAC ₁ , ABAC	UML
[30]	2012	Network firewalls	Metamodel Extraction	RBAC, OrBAC	Xtext

In this paper, we try to spot on the idea of metamodels, the existing metamodels in literature, and to look into future with some raised questions concerning this matter. Subsequently, in addition to the existing concerns and metamodels, many questions are raised, such as: is it possible to find a more general concept of metamodels? Is it possible to implement easier and general unified structures of metamodels, and visualize their elements more readily? Are the existing metamodels handle the feature of flexibility for any new extensions or transformations? Or, are the current metamodeling frameworks flexible and dynamic enough for any changes? If so, what are the possible ways, steps or strategies to merge metamodel elements? Although there are many metamodels are built, based on many AC models, do they overcome the existing limitations of these AC models? Last but not least, is the existing metamodeling tools and languages enough to answer all the above questions or some of them?

Additionally, as presented in this survey we can see that metamodels are implemented for different scenarios: AC models, WCMSs, and distributed environments. Thus, is there any opportunity to find a metamodel design or plan that encompasses the different scenarios? Or is it more efficient to find a unified metamodel for each scenario? As a result, currently we may not have answers to the above inquires, but at least we know that metamodels introduce a new era of enforcing policies and controlling access in IT world.

Another interesting feature, that is missing in current AC metamodels, is the ease of the migration from an AC model to another. In fact, having a metamodel, should make it possible to translate an existing AC policy between the different AC models covered by the metamodel.

6 Conclusion

We covered in this survey existing AC metamodels, and the AC models they generally cover (DAC, MAC, RBAC...). We also presented a brief explanation about how these AC models are used in different fields e.g. Databases, operating systems, IoT, etc. Moreover, Metamodels are proposed in literature to concurrently handle multiple AC models, also in distributed environments and web systems. The main goal is to develop a metamodel that is general enough to instantiate all existing AC models and that may also help organizations to easily migrate from an AC model to another.

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number 06351].

References

1. Matt, B., *Introduction to computer security*. 2006: Pearson Education India.
2. De Capitani di Vimercati, S., S. Paraboschi, and P. Samarati, *Access control: principles and solutions*. Software: Practice and Experience, 2003. **33**(5): p. 397-421.
3. Hu, V.C., D.R. Kuhn, and D.F. Ferraiolo, *Attribute-Based Access Control*. 2018, Norwood: Artech Hous
4. Kayem, A.V., S.G. Akl, and P. Martin, A presentation of access control methods, in *Adaptive Cryptographic Access Control*. 2010, Springer. p. 11-40.
5. Ennahbaoui, M. and S. ELHAJJI. Study of access control models. in *Proceedings of the World Congress on Engineering*. 2013
6. Ausanka-Cruess, R., *Methods for access control: advances and limitations*. Harvey Mudd College, 2001. 301: p. 20.
7. Sandhu, R., D. Ferraiolo, and R. Kuhn. The NIST model for role-based access control: towards a unified standard. in *ACM workshop on Role-based access control*. 2000
8. Crampton, J. On permissions, inheritance and role hierarchies. in *Proceedings of the 10th ACM conference on Computer and communications security*. 2003. ACM
9. Belokosztolszki, A., *Role-based access control policy administration*. 2004, University of Cambridge, Computer Laboratory .
10. Zhang, C.N. and C. Yang, Designing a complete model of role-based access control system for distributed networks. *J. Inf. Sci. Eng.*, 2002. **18**(6): p. 871-889.
11. Kuhn, D.R., E.J. Coyne, and T.R. Weil, Adding attributes to role-based access control. *Computer*, 2010. **43**(6): p. 79-81.
12. OrBAC: Organization Based Access Control. 2010; Available from: http://orbac.org/?page_id=21 .
13. Anderson, R., *Security engineering*. 2008: John Wiley & Sons
14. Rhodes-Ousley, M., *Information security: the complete reference*. 2013: McGraw Hill Education
15. Rajpoot, Q.M., C.D. Jensen, and R. Krishnan. Attributes enhanced role-based access control model. in *International Conference on Trust and Privacy in Digital Business*. 2015. Springer.

16. Onankunju, B.K., *Access control in cloud computing*. International Journal of Scientific and Research Publications, 2013. **3**(9): p. 1.
17. Hussain, S., *Access Control in Cloud Computing Environment*. International Journal of Advanced Networking and Applications, 2014. **5**(4): p. 2011.
18. Atzori, L., A. Iera, and G. Morabito, *The internet of things: A survey*. Computer networks, 2010. **54**(15): p. 2787-2805.
19. Liu, J., Y. Xiao, and C.P. Chen. Authentication and access control in the internet of things. in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. 2012. IEEE.
20. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X. and Wan, J., Smart Contract-Based Access Control for the Internet of Things. arXiv preprint arXiv:1802.04410, 2018.
21. Rajpoot, Q.M., C.D. Jensen, and R. Krishnan. Integrating attributes into role-based access control. in *IFIP Annual Conference on Data and Applications Security and Privacy*. 2015. Springer.
22. Assar, S., Meta-modeling: concepts, tools and applications, in *IEEE 9th International Conference on Research Challenges in Information Science, IEEE RCIS 2015*. 2015: Athens, Greece; Available from: <https://www.computer.org/cms/ComputingNow/education/said-assar-metamodeling-tutorial.pdf>
23. Sprinkle, J., Rumpe, B., Vangheluwe, H. and Karsai, G., 3 Metamodelling, in *Model-Based Engineering of Embedded Real-Time Systems*. 2010, Springer. p. 57-76.
24. Korman, M., R. Lagerström, and M. Ekstedt, Modeling enterprise authorization: a unified metamodel and initial validation. *Complex Systems Informatics and Modeling Quarterly*, 2016(7): p. 1-24.
25. Abd-Ali, J., K. El Guemhioui, and L. Logrippo, *A Metamodel for Hybrid Access Control Policies*. JSW, 2015. **10**(7): p. 784-797.
26. Bertolissi, C. and M. Fernández, A metamodel of access control for distributed environments: Applications and properties. *Information and Computation*, 2014. **238**: p. 187-207.
27. Bruneliere, H., Garcia, J., Desfray, P., Khelladi, D.E., Hebig, R., Bendraou, R. and Cabot, J. On lightweight metamodel extension to support modeling tools agility. in *European Conference on Modelling Foundations and Applications*. 2015. Springer.
28. Martínez, S., Garcia-Alfaro, J., Cuppens, F., Cuppens-Bouahia, N. and Cabot, J. Towards an access-control metamodel for web content management systems. in *International Conference on Web Engineering*. 2013. Springer.
29. Emig, C., Brandt, F., Abeck, S., Biermann, J. and Klarl, H., An access control metamodel for web service-oriented architecture. 2007.
30. Martínez, S., Cabot, J., Garcia-Alfaro, J., Cuppens, F., & Cuppens-Bouahia, N. A model-driven approach for the extraction of network access-control policies. in *Proceedings of the Workshop on Model-Driven Security*. 2012. ACM

APPENDIX II

The 12th International Conference on Ambient Systems, Networks and Technologies (ANT), March
23 - 26, 2021, Warsaw, Poland
Procedia Computer Science, Volume 184, 2021, Pages 887-892,
<https://doi.org/10.1016/j.procs.2021.03.056>



The 12th International Conference on Ambient Systems, Networks and Technologies (ANT)
March 23 - 26, 2021, Warsaw, Poland

A Review of Access Control Metamodels

Nadine Kashmar^{a,c,*}, Mehdi Adda^a, Mirna Atieh^b, Hussein Ibrahim^c

^a*Département de mathématiques, informatique et génie, Université du Québec à Rimouski, 300 Allée des Ursulines, QC G5L 3A1, Canada*

^b*Business Computer Department, Faculty of Economic Sciences and Administration, Lebanese University, Hadat, Lebanon*

^c*Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC G4R 5B7, Canada*

Abstract

The emergence of ubiquitous computing, especially with the Internet of Things (IoT), releases new prospects to traditional information systems by merging new technologies and services for seamless access to information sources at anytime and anywhere. Concurrently, this emergence opens new threats to information security and new challenges to control access to the resources. To ensure security, several techniques have been employed, and access control (AC) is one of the essential security requirements for IoT and non-IoT systems. Various authentication and AC methods are proposed to enforce AC policy and to prevent any unauthorized access to logical/physical assets. The continuous technology upgrades and the diversity of AC models force the need to find AC metamodels with higher level of abstraction that serve as a unifying framework for specifying any AC policy. AC metamodels are proposed to encompass AC features and are used to derive various instances of AC models and methods. In this paper we review the proposed AC metamodels and their implementation scenarios, we analyze them, their objectives, their limitations, and present open research questions and issues that still need to be addressed.

© 2021 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Access control; metamodels; IoT; Industry 4.0; security and privacy; security policy;

1. Introduction

The importance of security and privacy requirements increases with the increase of cybercriminals and cyberattacks due to the massive presence and integration of new paradigms and technologies, such as the Cloud Computing and the Internet of Things (IoT). Also, with the deployment of digital and intelligent solutions based on the smart industry concept. To mitigate their impacts, several techniques have been employed, and access control (AC) is one of the essential solutions for privacy settings to measure and optimize IT security [1] in IoT [2], cloud computing [3], social networks [4], and other fields. AC methods are implemented to control what users can access, when and how by enforcing AC policy to prevent any unauthorized access for logical or physical assets.

* Corresponding author. Tel.: +14188338800; fax: +141883311.

E-mail address: nadine.kashmar@uqar.ca

Various AC models are developed to enforce AC policies such as Role-Based AC (RBAC), Organization-Based AC (OrBAC), and Attribute-Based AC (ABAC) [1, 5]. To enhance AC methods, various hybrid models are implemented by combining features of two or more AC models [1, 6]. Additionally, with the emergence of industry 4.0 and IoT applications, it is realized that the existing AC models have reached their limits, and they no longer meet the increasing demand for privacy and security standards [1, 7]. This reality urge the need to find more advanced AC methods and develop AC metamodels with advanced features for specifying and enforcing different AC policies [7, 8, 9]. AC metamodels are used to derive various instances for the common AC models, hybrid models, and other AC methods. Note that, in [1] we present a preliminary survey for the common used AC models with some of the proposed AC metamodels, then raise some questions in this domain.

The objective of this paper is to present a detailed review, analyze and criticize the proposed AC metamodels, find out their limitations in the presence of new technologies, and determine various research issues in this domain, then raise some essential research questions. Hence, it can be considered as a pillar towards developing a new generic and dynamic AC metamodel with advanced features for IoT and non-IoT systems. To the best of the authors' knowledge, this is the first paper that provides a literature review of the existing AC metamodels with discussion and critical analysis, and various research issues in the domain. The remaining of this paper is organized as follows: section 2 summarizes the existing AC models. Section 3 presents the state-of-the-art of the proposed metamodels, their objectives, and limitations. Discussion and critical analysis, and common limitations for the proposed metamodels are presented in section 4. Issues and open research questions are proposed in section 5. Section 6 concludes this paper with future perspectives.

2. The Common Access Control Models

Access control is the process of restricting access to a place or resource based on defined set of security policies. Security policies are the definition of rules that must be regulated in an organization, and they are usually defined by managers and system administrators. An AC model is a framework for making authorization decisions based on the defined AC policies, and AC mechanism is the processes of enforcing AC policy and translating user's access request [1, 5]. Despite the presence of several papers reviewing the state-of-the-art of the common AC models [1, 10], in this paper we summarize them since they are used in the core for constructing different AC metamodels.

- *Discretionary Access Control (DAC)*: is a user-centric AC model with three major components: objects, subjects, and permissions. DAC allows subjects to control access permissions to their objects, an AC matrix (ACM) is an example of how AC rights of subject(s) over object(s) can be specified. Lampson and Harrison Ruzzo Ullman (HRU) are two variants of DAC model. [1, 5].
- *Mandatory Access Control (MAC)*: is based on the concept of security levels (top-secret, secret, confidential) that are associated with subjects (as clearance levels) and objects (as classification levels) where permissions and actions are derived. In MAC, AC policy is managed in a centralized manner, it has four key components: a set of objects, a set of subjects, permissions, and security levels. Bell and LaPadula (BLP) and BIBA (developed by Kenneth J. Biba) are two MAC variants [1, 5].
- *Role-Based Access Control (RBAC)*: is based on several entities: subjects, roles, permissions, actions, operations, and objects. A role means a group of permissions to use object(s) and perform some action(s), it can be assigned to several subjects, and subjects can be assigned to several roles such as, doctor, nurse, etc. [1, 5]. RBAC example, is a hospital system where there exists a variety of relations between doctors, nurses, etc. Only the system administrator has the right to control the system security and assign roles to users [11].
- *Organization-Based Access Control (OrBAC)*: is presented to find more abstract AC policy. OrBAC key components are subject, action, object at concrete level; role, activity, view at abstract level; and context lies between the two levels to express dynamic rules for relations between entities (e.g. permission, prohibition, etc.) [1].
- *Attribute-Based Access Control (ABAC)*: has some advantages over RBAC because of its ability to support dynamic attributes. It has three types of attributes: object, subject and environmental (e.g. current time, location, etc.) attributes. It allows/denies user requests based on some attributes for subjects, objects and environment, and a set of conditions, and it is dynamic since it uses attributes to determine access decision [1, 5].

Despite the advantages of AC models, they also have some limitations. For example, (1) in DAC the granted user who could access a file can allow other users to read it without asking the owner; (2) the assignment of security levels by a system limits user's actions which prevents dynamic modification of original policies in MAC; (3) in large systems role inheritance of RBAC make administration potentially heavy; (4) poor support for dynamic attributes (e.g. time of day) in OrBAC; and (5) ABAC implementations require significant time to run [1, 5].

Moreover, the evolution of technology trends urges the need to enhance AC methods, hence various hybrid AC models with combined features from two or more AC models are proposed. For example, Kuhn et al. in [12] address the feature of adding attributes to RBAC to find a model that supports dynamic attributes in organizations and provide better AC features in dynamic environments. As well, Rajpoot et al. in [13] propose Attribute Enhanced RBAC (AERBAC) model to enhance both RBAC and ABAC because they have complimentary features to each other. In [14], authors purpose they propose more fine-grained, flexible, and efficient RABAC (RBAC/ABAC) model. To increase the flexibility of RBAC, an Emergency RBAC (E-RBAC) approach is proposed in [15], and other approaches.

3. Access Control Metamodels: the state-of-the-art

The rapid evolution of technology and the limitations of AC models, force the need to find AC metamodels to implement advanced AC features. In general, a metamodel is defined as textual, visual, or formal representation of concepts and how they are linked together, these concepts might be rules or guidelines for an organization [1]. Our concern in this paper is to review the proposed AC metamodels in the last decade, analyze them to find if they are effective to follow technology upgrades. Fig. 1 illustrates a classification for the proposed AC metamodels for centralized and distributed environments.

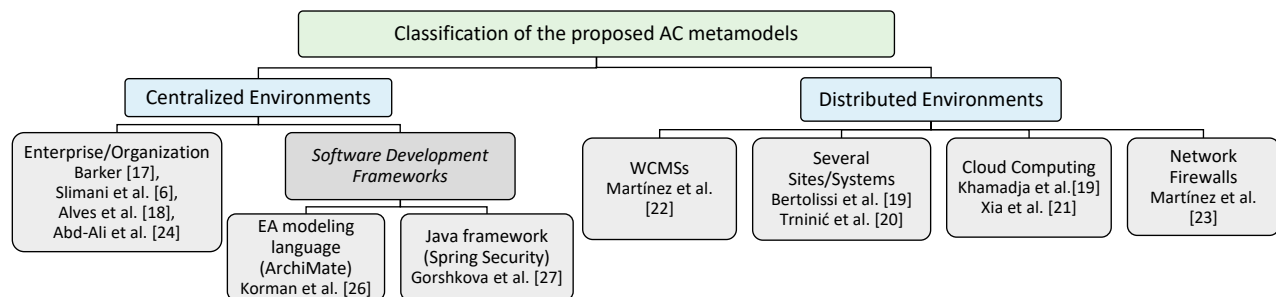


Fig. 1. Classification for the proposed AC Metamodels

Ferraiolo et al. [16] revise some concerns and raise some issues related to AC policy enforcement and focuses on the important role that a metamodel might play if it is achieved. To address them, a paper published by Barker [17] demonstrates that multiple AC models can be derived as special cases from a defined AC metamodel called Category Based Access Control (CBAC) metamodel. A category is interpreted as a synonym for, a role, a class, a group, etc. where entities (e.g. subjects) may be assigned. CBAC includes features of MAC, DAC, and RBAC where a wider range of constraints may be expressed based on it. In [6], Slimani et al. extend Barker's metamodel to support resource and action hierarchies. They propose a Unified Access Control Modeling Language (UACML) for finding hybrid AC policies by allowing categories to be associated with other categories and finding hierarchical relationships between them. A category is identified as a central component to abstract the key components of AC models such as roles, security levels, etc. A CBAC metamodel extension is proposed by Alves et al. in [18] to expand a general notion of obligation for the existing AC models and study the interaction between obligations and permissions.

Another CBAC metamodel approach is proposed in [19], for distributed environments of several sites and several policies at each site. In their distributed metamodel the request can be passed to other sites and evaluated in a distributed manner, they also show how a distributed, dynamic, event-based access control model (DEBAC) can be defined as an instance of the metamodel. In the context of cloud computing, saving data on cloud servers raise security challenges to protect sensitive data. In [20] authors state that, the classical AC models (DAC, MAC ...) are not adequately expressive for highly flexible and dynamic environments. For this purpose, they present a metamodel approach for cloud computing services called Category Based Access Control (CatBAC) framework which allows

security administrators in the various company sites to find a concrete model with the constraints and specificities of each site. Xia et al. in [21] propose another metamodel approach to handle security and privacy in cloud service, called the Cloud Security and Privacy Metamodel (CSPM). CSPM integrates and extends the existing metamodels of cloud security together with newly added concepts. Moreover, an approach is presented for web services by Martínez et al. in [22]. They propose a metamodel for the representation of Web Content Management System (WCMS) AC policies to ease the analysis and manipulation of security requirements by abstracting them from vendor-specific details. Although AC methods are integrated in most WCMS systems (e.g. Wordpress, Drupal . . .), some limitations still exist. For this purpose, authors aim to raise the level of abstraction of the AC implementation to be represented according to a vendor-independent metamodel. Also, authors in [23] propose a model-driven approach to extract network AC policies enforced by firewalls within a network system. They suggest raising the level of abstraction of the information contained in the firewall configuration files, hence AC policy would be easier to analyze and manipulate.

Furthermore, Abd-Ali et al. in [24] propose an integration metamodel for hybrid AC policies to concurrently handle multiple AC models. Their idea is based on the concept of abstracting each AC model (e.g. RBAC metamodel), and each AC metamodel has a special element named DecisionHandler. The AC decision depends on integrating several AC metamodels, and the AC decision is issued by clustering the DecisionHandler instances of a hybrid AC policy, then apply them to combining algorithms to find one AC decision as output. Trinić et al. in [25] present PolicyDSL as a generic AC management infrastructure for a broad set of systems to provide a general method for specifying AC rules for different AC models. PolicyDSL is used to specify concrete AC policies in a system where a security expert would be able to express AC policies for a given AC model using the generated DSL.

Moreover, due to the lack of security features in software development frameworks, some metamodel extensions are proposed in the literature. Korman et al. in [26] propose a unified metamodel as a prospective extension for ArchiMate, the common Enterprise Architecture (EA) modeling language, to support the development of enterprises by extending their abilities to model authorization and AC in their architectures. They propose a metamodel extension based on the conceptual model of ABAC because of its ability to include most of other AC models, then mapped to ArchiMate to enrich its existing models. Also, Gorshkova et al. in [27] introduce a fine-grained AC model and provide a metamodel extension for Spring Security framework to meet modern security requirements. Spring Security is one of the open source security frameworks for Java.

Subsequently, the proposed AC metamodels especially in recent years show the concern for finding advanced AC methods. This reflects the importance of constructing more robust AC models in all computing environments, especially with the presence of heterogenous network technologies and platforms [28]. Table 1 summarizes the proposed AC metamodels, the core metamodel features, their types, the derived AC models instances, and the modeling tools.

4. Discussion and Critical Analysis

As shown in Table 1, AC metamodels are constructed based on features of AC models where various models (also hybrid models) instances can be derived from them. Some metamodels are proposed as generic, unifying, hybrid, and metamodel extension for different distributed and centralized environments. Hence:

- 1- Some AC metamodels are constructed based on features of some AC models, and the only AC model(s) (also hybrid) instance(s) that can be derived are the one(s) that are employed in the core structure, for example [6] and [24]. These metamodels are proposed as *Hybrid Metamodels*.
- 2- Some frameworks (for example, Drupal, ArchiMate, Spring Security...) are extended to support AC features of one or more AC model, and the extracted AC policies belong to the model(s) that are used to extend the main framework, for example [22, 23, 26, 27]. These metamodels are proposed as *Metamodel Extensions*.
- 3- Some AC metamodels are constructed based on a general notion which encompasses some AC features for some models (for example, CBAC). Based on this metamodel, AC model instance(s) can be derived, for example [17, 19, 20, 25]. These metamodels are proposed as *Generic Metamodels*.
- 4- Some of the existing AC metamodels are augmented with additional features to reflect a larger and more definitive set of possible rules to express AC policies, for example [18] and [21]. This type of metamodels is proposed as *Metamodel Extensions*.

Hence, the proposed works of AC metamodels in the literature can be classified into two concepts:

Table 1. Summary of the Proposed Access Control Metamodels

ref.	Author	Year	Proposed for	Metamodel	Visual rep. Y/N	Tool	Type	Based on	instance(s)	Modeling lang.
Proposed AC metamodels for Distributed Environments										
[19]	Bertolissi et al.	2014	Distributed system of several sites	Distributed Metamodel	No	n/a	Generic Metamodel	CBAC	CBAC	rewrite-based operational semantics
[20]	Khamadja et al.	2013	Cloud Computing	CatBAC metamodel	Yes	UML	Generic Metamodel	CBAC	Hybrid models	First-order logic
[21]	Xia et al.	2018	Cloud services	cloud security & privacy (CSPM)	Yes	UML	Metamodel Extension	n/a	n/a	UML
[22]	Martinez et al.	2013	WCMSs	WCMS Metamodel	Yes	MDE	Metamodel Extension	RBAC	RBAC	Drupal
[23]	Martinez et al.	2012	Network Firewalls	Network Connection	Yes	Eclipse	Metamodel Extension	Network Firewalls	RBAC, Or-BAC	Xtext
[20]	Trninić et al.	2013	Set of systems	PolisyDSL	Yes	UML	Generic Metamodel	n/a	RBAC	Textual DSL
Proposed AC metamodels for centralized Environments										
[17]	Barker	2009	Enterprise	Barker's Metamodel	No	n/a	Unifying Metamodel	CBAC	RBAC,MAC	Rule/Logic Language
[6]	Slimani et al.	2011	Enterprise	UACML Metamodel	Yes	UML	Hybrid Metamodel	CBAC and Hybrid models	Group based, MAC, RBAC, hybrid model	Object constraint language (OCL)
[18]	Alves et al.	2014	Enterprise	Obligations in CBAC Metamodel	No	n/a	Metamodel Extension	CBAC	CBAC	rewrite-based operational semantics
[24]	Abd-Ali et al.	2015	Enterprise	Integration metamodel	Yes	UML	Hybrid Metamodel	CW,BLP,BIBA, RBAC	Hybrid models	First-order logic
[26]	Korman et al.	2016	Enterprise Architecture framework	Unified Metamodel	Yes	ArchiMate	Metamodel Extension	DAC,BLP,Biba, CW, RBAC, ABAC	DAC,BLP,CW, RBAC, ABAC	ArchiMate
[27]	Gorshkova et al.	2017	Enterprise application framework	Spring security framework	Yes	Java-ORM	Metamodel Extension	RBAC	RBAC	Spring expression lang.(SpEL)

- In (1) and (3) the aim is to find a generic metamodel that encompasses most of the AC features where various AC models (and hybrid models) can be derived, Fig. 2a illustrates the idea of generic metamodels. But the proposed AC metamodels are not generic, since they have hybrid structure with some AC features rather than a generic metamodel. This hybrid structure is employed to derive some AC models where their features are already employed in the core metamodel structure.
- In (2) and (4) the aim is to enhance some of the existing frameworks/metamodels by extending them to support AC features and express more AC policies, Fig. 2b illustrates the idea of metamodel extension where AC features are added to the core metamodel/framework to allow defining (more) AC policies. Then, the extended AC metamodel/framework can be used to derive various instances of AC models based on the added features.

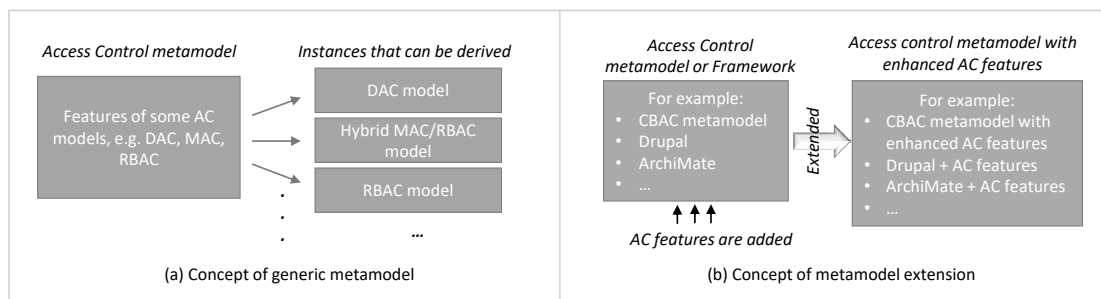


Fig. 2. Illustration for the concept of the proposed AC Metamodels

However, the presented AC metamodels come with some advantages and several combined features from AC models are implemented to enhance AC features. But they also have several limitations especially in the light of new

technologies, Table 2 summarizes the objective(s) and limitation(s) for each metamodel. Also, the existing metamodels have some common limitations and lack some essential characteristics and can be enumerated as follows:

- each metamodel address a case and does not encompass a general concept to derive all AC models instances;
- they are not dynamic enough to follow the continuous technology progress;
- they do not address the feature of collaboration between the derived AC models;
- they do not support or address the feature of migration of AC policy from one AC model to another;
- none of them tackles the issue of finding a common set for AC concepts due to the heterogeneity of AC models;
- none of them address the context of IoT.

Table 2. Objective(s) and Limitation(s) of The Proposed Access Control Metamodels

Author(s)	Objective(s)	Limitation(s)
Barker [17]	Multiple AC models can be derived as special cases from a defined CBAC metamodel.	- lacks the support of resource and action hierarchies.
Slimani et al. [6]	To provide support for hybrid AC policies by allowing categories to be associated with other categories and finding hierarchical relationships between them.	- hybrid structure to derive some AC models rather than a metamodel.
Alves et al. [18]	To allow security administrators to check the consistency of a policy combining authorizations and obligations.	- no explanation of how the approach could be dynamic in distributed contexts which are rich of events and variable attributes.
Bertolissi et al. [19]	To provide semantics for distributed AC mechanisms within distributed environments consisting of several sites.	- no real case studies are explained or implemented.
Khamadja et al. [20]	To develop a new cloud computing service named "Access Control as a Service".	- no case study or testing result; - do not explain how access can be controlled in the context of multi-cloud.
Xia et al. [21]	To handle security and privacy in cloud service development and operations.	- have not explained how access can be controlled in the context of multi-cloud.
Martinez et al. [22]	To ease the analysis and manipulation of security requirements in WCMSs.	- the notion of variable attributes is not considered. - extending Drupal framework to provide support for some AC features without explaining how these extensions can be upgraded due to updates.
Martinez et al. [23]	To extract network AC policies enforced by firewalls within a network system, then AC policy would be easier to understand, analyze and manipulate.	- extending networks firewall systems to provide support for some AC features without explaining how these extensions can be upgraded due to continuous updates.
Abd-Ali et al. [24]	To concurrently handle multiple AC models which are: CW, BLP, BIBA, and RBAC.	- hybrid structure to derive some AC models rather than a metamodel.
Trninić et al. [20]	to allow a security expert to express AC policies for a given AC model.	- does not consider dynamic constraints.
Korman et al. [26]	To provide support for architectures of enterprises by extending their abilities to model authorization and AC in their frameworks.	- difficulty for potential implementation of automated analytical capabilities. - they extend ArchiMate framework to support some AC features without explaining how these extensions can be upgraded.
Gorshkova et al. [27]	To provide a metamodel extension for Spring Security framework to meet modern security requirements.	- they extend Spring Security framework to support some AC features without explaining how these extensions can be upgraded.

Accordingly, constructing a unified or generic AC metamodel that considers the continuous technologies progressions, the variety of information systems, and the heterogeneity of AC models is not yet achieved.

5. Issues and open research questions

This is a recent research issue, and various research are still conducting for AC metamodeling approach to find a more general metamodel that can be used to dynamically define AC policies. Despite the proposed AC metamodels have some enhanced features, they lack some important characteristics that are essential to the current fact of technologies. From this review, we can find that some issues need to be addressed which are:

5.1. Generality

Finding a generic AC metamodel concept that includes all features of the common AC models, and can be oriented to extract various AC models and serves as a basis for specifying any AC policy is an essential issue in this domain. Finding this generic basis could lead to find other essential characteristics for this metamodel (e.g. collaboration between AC models).

5.2. Dynamism

The structure of the generic AC metamodel should be dynamic with the ability to define various types of attributes and entities, hence various models can be formulated for static and dynamic policy enforcements.

5.3. Extensibility

In the literature, several AC methods are implemented in different computing environments, some of them are based on (or extended from) the common AC models, while others are formulated based on the needed context. This reflects the diversity of the implemented AC models in different fields and the importance of upgrading them to follow the continuous technology progressions. Hence, a metamodel supporting the feature of defining new components in addition to the existing ones would allow describing larger set of possible rules to express policies. Hence, developing a metamodel that supports this feature is important to extend and upgrade the existing AC methods, and to formulate and implement new AC methods.

5.4. Collaboration and interoperability

In collaborative computing environments, various collection of information systems and technologies are performed to support work and cooperation between organizations, sites, etc. where users are allowed/denied to communicate via a wide range of applications such as audio/video conferencing, collaborative document sharing/editing, etc. hence, collaborative environments need to control access to their assets to increase working cooperation efficiently. Finding a general basis for a metamodel, would permit handling multiple AC models, and would in turn permit the collaboration between the obtained AC models and the interoperability between their components.

5.5. Migration

Another interesting feature, that is missing in current AC metamodels, is the ease of the migration from an AC model to another. In fact, having a metamodel should make it possible to translate an existing AC policy between the different AC models covered by the metamodel. However, a metamodel with generic, dynamic, and extendable structure can be implemented to allow migrating the AC policies from one model to another.

However, in this context we can raise the following questions:

- how a new generic and dynamic AC metamodel can be designed?
- what are the main features, components, etc. this AC metamodel can include?
- how its structure can be developed to handle collaboration/interoperability/extension/migration of AC models?
- how to construct a common set of AC concepts for the heterogeneous AC models?

6. Conclusion and Future Perspectives

In this paper, we review and analyze the proposed AC metamodels, explain their objectives, their limitations specially with current technology progressions and upgrades, and explain the need of finding AC metamodels with higher level of abstraction that serve as unifying frameworks for specifying any AC policy. To the best of our knowledge, this is the first review paper with detailed analysis and explanation for the potential research issues in this domain.

In general, recent computing environments, especially IoT, are full of resources and are open to all kinds of attacks and threats. Furthermore, access control studies gain the attention of researchers along the decades, especially with IT continuous developments. The common limitations, which can also be considered as research issues in this domain, that have not been addressed yet are important to be implemented with the current heterogeneous computing environments. As a contribution in this domain, we are currently developing a new AC metamodel with essential features that address the aforementioned research issues. As a future perspective, we aim to publish a formal representation for our AC metamodel in another paper.

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number 06351], Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT), and Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

References

- [1] N. Kashmar, M. Adda, M. Atieh, From access control models to access control metamodels: A survey, in: *Future of Information and Communication Conference*, Springer, 2019, pp. 892–911.
- [2] S. Ravidas, A. Lekidis, F. Paci, N. Zannone, Access control in internet-of-things: A survey, *Journal of Network and Computer Applications* 144 (2019) 79–101.
- [3] M. Sookhak, F. R. Yu, M. K. Khan, Y. Xiang, R. Buyya, Attribute-based data access control in mobile cloud computing: Taxonomy and open issues, *Future Generation Computer Systems* 72 (2017) 273–287.
- [4] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, *Access Control in Cybersecurity and Social Media*, Université d'Ottawa, 2020, Ch. 4.
- [5] V. Hu, D. Ferraiolo, R. Chandramouli, D. Kuhn, *Attribute-Based Access Control*, Artech House Publishers, 2017.
- [6] N. Slimani, H. Khambhammettu, K. Adi, L. Logrippo, Uacml: Unified access control modeling language, in: *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, IEEE, 2011, pp. 1–8.
- [7] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, A new dynamic smart-ac model methodology to enforce access control policy in iot layers, in: *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, IEEE, 2019, pp. 21–24.
- [8] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, Smart-ac: A new framework concept for modeling access control policy, *Procedia Computer Science* 155 (2019) 417–424.
- [9] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, Access control metamodel for policy specification and enforcement: From conception to formalization, *Procedia Computer Science* (2021).
- [10] R. Sandhu, E. Coyne, H. Feinstein, C. Y. Role-Based, Access control models, *IEEE computer* 29 (2) (2013) 38–47.
- [11] E. O. Boadu, G. K. Armah, Role-based access control (rbac) based in hospital management, *Int. J. Softw. Eng. Knowl. Eng* 3 (2014) 53–67.
- [12] D. R. Kuhn, E. J. Coyne, T. R. Weil, Adding attributes to role-based access control, *Computer* 43 (6) (2010) 79–81.
- [13] Q. M. Rajpoot, C. D. Jensen, R. Krishnan, Integrating attributes into role-based access control, in: *IFIP Annual Conference on Data and Applications Security and Privacy*, Springer, 2015, pp. 242–249.
- [14] H. Qi, X. Di, J. Li, Formal definition and analysis of access control model based on role and attribute, *Journal of information security and applications* 43 (2018) 53–60.
- [15] F. Nazerian, H. Motameni, H. Nematzadeh, Emergency role-based access control (e-rbac) and analysis of model specifications with alloy, *Journal of information security and applications* 45 (2019) 131–142.
- [16] D. Ferraiolo, V. Atluri, A meta model for access control: why is it needed and is it even possible to achieve?, in: *Proceedings of the 13th ACM symposium on Access control models and technologies*, 2008, pp. 153–154.
- [17] S. Barker, The next 700 access control models or a unifying meta-model?, in: *Proceedings of the 14th ACM symposium on Access control models and technologies*, 2009, pp. 187–196.
- [18] S. Alves, A. Degtyarev, M. Fernández, Access control and obligations in the category-based metamodel: a rewrite-based semantics, in: *International Symposium on Logic-Based Program Synthesis and Transformation*, Springer, 2014, pp. 148–163.
- [19] C. Bertolissi, M. Fernández, A metamodel of access control for distributed environments: Applications and properties, *Information and Computation* 238 (2014) 187–207.
- [20] S. Khamadja, K. Adi, L. Logrippo, Designing flexible access control models for the cloud, in: *Proceedings of the 6th International Conference on Security of Information and Networks*, 2013, pp. 225–232.
- [21] T. Xia, H. Washizaki, T. Kato, H. Kaiya, S. Ogata, E. B. Fernandez, H. Kanuka, M. Yoshino, D. Yamamoto, T. Okubo, et al., Cloud security and privacy metamodel, in: *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, SCITEPRESS-Science and Technology Publications, Lda, 2018, pp. 379–386.
- [22] S. Martínez, J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Bouahia, J. Cabot, Towards an access-control metamodel for web content management systems, in: *International Conference on Web Engineering*, Springer, 2013, pp. 148–155.
- [23] S. Martínez, J. Cabot, J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Bouahia, A model-driven approach for the extraction of network access-control policies, in: *Proceedings of the Workshop on Model-Driven Security*, 2012, pp. 1–6.
- [24] J. Abd-Ali, K. El Guemhioui, L. Logrippo, A metamodel for hybrid access control policies., *JSW* 10 (7) (2015) 784–797.
- [25] B. Trninić, G. Sladić, G. Milosavljević, B. Milosavljević, Z. Konjović, Policydsl: Towards generic access control management based on a policy metamodel, in: *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, IEEE, 2013.
- [26] M. Korman, R. Lagerström, M. Ekstedt, Modeling enterprise authorization: a unified metamodel and initial validation, *Complex Systems Informatics and Modeling Quarterly* (7) (2016) 1–24.
- [27] E. Gorshkova, B. Novikov, M. K. Shukla, A fine-grained access control model and implementation, in: *Proceedings of the 18th International Conference on Computer Systems and Technologies*, 2017, pp. 187–194.
- [28] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, Deriving access control models based on generic and dynamic metamodel architecture: Industrial use case, *Procedia Computer Science* 177 (2020) 162–169.

APPENDIX III

IEEE/ACM International Workshop on Software Engineering Research & Practices for the Internet
of Things (SERP4IoT), 27-27 May 2019, Montreal, Canada
Pages: 21 - 24, <https://doi.org/10.1109/SERP4IoT.2019.00011>

A New Dynamic Smart-AC Model Methodology to Enforce Access Control Policy in IoT layers

Nadine Kashmar*, Mehdi Adda
Département de Mathématiques,
Informatique et Génie
Université du Québec à Rimouski
Rimouski, Québec
kasn0002@uqar.ca*,
mehdi_adda@uqar.ca

Mirna Atieh
Business Computer Department,
Faculty of Economic Sciences and
Administration
Lebanese University
Hadat, Lebanon
matieh@ul.edu.lb

Hussein Ibrahim
Institut Technologique de Maintenance
Industrielle (ITMI)
Sept-Îles, Québec
hussein.ibrahim@itmi.ca

Abstract—Internet of Things (IoT) is the conversion of everyday tangible devices or machines to smart objects. This means that these objects would be able to think, sense and feel. For example, your home devices will be able to detect and feel your absence to turn off the lights of empty rooms, close doors, lock the gates, and other tasks. Thus, would it be acceptable to find intruders who might mess up your daily life style or control your home appliances? Absolutely not! The same idea for factories, they definitely reject to detect any unacceptable access from any foreigner to their logical/physical assets or machines who might be able to locally or remotely control, for example, any machine operation. This would cause a significant loss for their reputation or investments, since any vulnerability or attack can produce, for example, fault products. So far, IoT is considered as one of the most essential areas of future technologies, especially for the industries. Hence, finding an environment full of smart devices needs a smart security methodology to prevent any illegal access. In this domain, various researches are conducted to find Access Control (AC) models to enforce security policies that prevent any unauthorized detection of sensitive data and enable secure access of information. For this purpose, we present a new dynamic Smart-AC model methodology to enforce security policy in IoT layers.

Keywords—IoT; access control; smart; security; model; policy

I. INTRODUCTION

In literature various Internet of Things (IoT) definitions are provided due to the integration of different technologies. The significant amount of IoT definitions can be summarized as a huge number of objects and devices with different technologies and platforms that are connected to the internet via heterogeneous networks (3G, LTE, WiFi, ZigBee ...). Figure 1 illustrates this definition. It is the integration of several technologies, such as wired and wireless sensor networks, identification and tracking technologies, communication protocols shared with the next generation internet, and distributed intelligence for smart objects [1].

The big question in IoT is how to allow a variety of objects, such as PCs, mobile phones, sensors, etc. to interact and cooperate with each other transparently and securely to attain certain tasks related to consumers, companies or

industry sectors. In this domain, the main concern for them is to keep their zone of interconnected devices and which are connected to the internet, secure, private and controlled only by them. Thus, finding an environment full of smart devices needs a smart security methodology to prevent any illegal access and enforce policy and security requirements. In this paper, we tackle IoT security and Access Control (AC) related issues and present a new methodology to enforce AC policy in different IoT layers.

The paper is organized as follows: section II briefly presents IoT limitations and challenges. Section III summarizes the existing AC models for IoT. The architecture of IoT layers and our methodology of integrating Smart-AC model in IoT are explained in section IV. Section V concludes this paper and presents future perspectives.

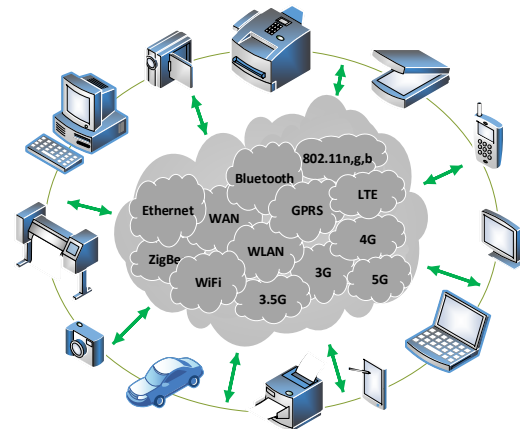


Figure 1. Heterogeneous IoT platforms, devices and internet networks

II. IOT LIMITATIONS AND CHALLENGES

The IoT is penetrating a wide range of domains including cities, homes, industry, healthcare, appliances, and much more, to make everything smart, adaptable and easy. In each area there are various opportunities and challenges. For this purpose, different plans are conducted, various applications are developed, and different software and hardware technologies are cooperatively used. The core

player in IoT technologies are the wireless technologies, where sensor networks play a critical role for linking the physical world with the digital world. For example, e-health applications, environmental monitoring (temperature, plants, weather), intelligent transportation systems, etc.

Although IoT is a popular topic, many challenging problems still need to be addressed, specially the technological and social aspects, before being the IoT idea widely accepted. The IoT challenges can be summarized under the following categories: free internet connectivity, security and all related issues, acceptability among the society, storage and computational ability, scalability, and power consumption [2, 3]. Among other challenges that are also mentioned in [4] we have: data management challenge, data mining challenge, privacy and security challenge, and chaos challenge. All these challenges and limitations open wide research issues, suggestions, methodologies, architectures, and others to address each of them. Since all IoT devices are connected to the internet, they are vulnerable to attacks and security threats. Hence, the core concern is the importance of finding methods to enforce AC policies to prevent any untrusted access and control access to the resources. Accordingly, the acceptance or rejection of this technology is determined by many factors, where security and privacy are considered the main of them.

III. RELATED WORK

The challenging IoT heterogeneous environments of interconnected networks and distributed systems, the heterogeneity of platforms and applications, and the diversity of users, force the necessity to design a well coherent AC architecture to enforce AC policy and administer security features in IoT. In this context, several IoT authentication and AC methods are offered by researchers to integrate security features with this technology from two or more AC models. The most famous AC models that are presented comprehensively and reviewed in literature are: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC), Organization Based Access Control (OrBAC), and Attribute Based Access Control (ABAC) [5-7].

However, Ouaddah et. al in [8] propose SmartOrBAC AC framework for IoT environment. It is based on OrBAC, but due to some OrBAC limitations, it is coupled with the RESTFUL web services mechanisms due to its preferability for the low power constrained environment. The result is the use of web service technologies to implement secure collaboration between organizations. The basic AC requirements in IoT are analyzed by Hussein et. al in [9]. The summarized AC requirements are thin clients and server architecture, autonomous and self-contained AC, infrastructure integration, and attributes centric AC. However, authors adopt the community-based AC architecture (COBAC) to administer the AC in IoT, as a solution for these requirements. IoTCollab framework is developed by Adda et. al in [10], it is an extended study of RBAC and ABAC models. Its aim is to ease the collaboration and data sharing in IoT. In [11], a collaborative

RBAC (CollRBAC) and collaborative ABAC (CollABAC) models are described and compared in the light of IoT and IoTCollab requirements. Moreover, a model to find a secure communication between things is proposed by Liu et al. [12]. The main idea is to verify identities between two IoT devices by implementing authentication protocol in the presentation layer where identification key establishment occurs. For Authorization, authors adopt RBAC's authorization concept, implement Elliptic Curve Cryptosystem (ECC) for secure key establishment. A smart contract-based framework is proposed in [13] to implement distributed and trustworthy AC for the IoT by applying smart contract-enabled blockchain technology. It contains: multiple Access Control Contracts (ACCs), one Judge Contract (JC), and one Register Contract (RC). For AC between subjects and objects, ACCs are implemented. To judge the unpleasant behavior of a subject during the AC, JC is used. To manage the ACCs and JC, RC is used. In this context authors address different case studies to demonstrate the feasibility of the framework. As well, some researches address the different layers of IoT architecture, then propose an AC model. Authors in [14] mention that, even there is no consensus on a wide IoT architectures, they are generally comprised of three main components: an object layer, a middle layer(s), and an application layer. The difference between these architectures relies in the middle layer(s). Hence, authors propose a cloud-enabled IoT with four-layer AC Oriented (ACO) architecture which are: an object, a virtual object layer, a cloud service layer, and an application layer. Their purpose is to establish a framework to find AC models for cloud enabled IoT.

However, none of the proposed AC models encompass a general structure or methodology. Each model addresses certain case and implemented based on some features of different models (RBAC, OrBAC ...), knowing that these models have limitations and deficiencies [5]. Also, none of them consider the continuous technological changes, and it is built for specific IoT architecture or structure. Thus, two key points inspire us to develop a smart AC methodology which can be implemented to find a Smart-AC model:

- The first is the word "SMART". While we think about IoT, we think about the smart objects which are able to sense and communicate within the IoT environment. Hence, we come up with the idea of a "Smart Access Control model".
- The second is the term "HETEROGENEOUS". As we know, IoT world is heterogeneous, starting from the devices, types of networks, platforms, reaching to the applications. As presented in literature, there are various AC models implemented from two or more AC models, for various cases and studies, and they are also heterogeneous. Thus, finding a Smart-AC model that is capable to include all AC features and can be dynamic enough to be implemented in any IoT environment becomes our objective.

The aim is to find a general and dynamic AC model structure which allow building other AC models to enforce AC policies in IoT, regardless of its architecture and type of application (smart home, smart industry ...). For example, in the field of smart industry or industrial IoT, the new factories

of electricity rely on IoT applications. Any intrusion for such applications can cause hazardous consequences, such as cutting off the power for hospitals, ministries and even cities.

IV. IOT ARCHITECTURE AND THE PROPOSED SMART-AC MODEL METHODOLOGY

A. IoT Layers

In [13], authors mention that there is no consensus on a wide IoT architecture. In this context, [3, 14-16] state that IoT architectures, are generally comprised of three components: 1) the object or the perception layer, 2) the middle or network layer(s), and 3) the application or presentation layer (Figure 2). The object layer contains Internet-enabled devices (cameras, sensors, ...) to gather and exchange information with other devices through the Internet. The middle layer works as agent to transfer the collected data from an object layer to a specific destination in the application layer. In this layer, different network communication technologies are used for this purpose, such as Bluetooth, ZigBee, WiFi, 4G, etc. The application layer is where information is received and processed. Also, it is proposed that in some IoT architectures, the middle layer consists of two layers: the network and the service layers. Similarly, in some other researches it is proposed that the middle layer consists of three layers: object abstraction, service management, and service composition layers.

B. Smart-Access Control Model features and methodology

Various AC models are presented in literature, MAC, DAC, RBAC, etc. each model is developed either to overcome some limitations found in preceding models or as a solution for a specific use case and application. Moreover, some other AC models have some combined features from some models to enhance some service features. Hence, our aim is to find a Smart-AC model with the following features:

- Generic enough to include all features offered by the existing AC models.

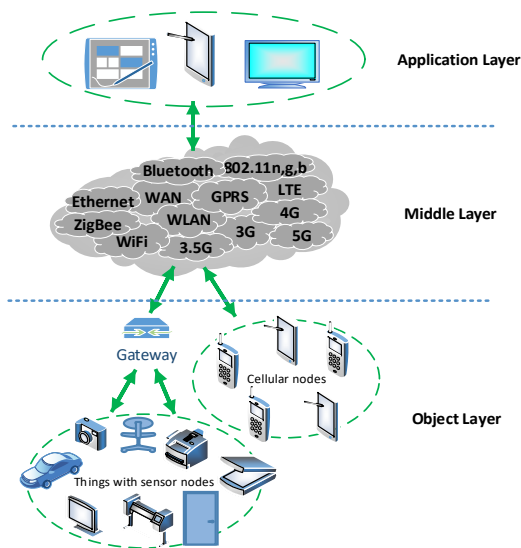


Figure 2. General IoT Layers

- Serves as a basis for specifying any AC policy.
- Eases the migration from an AC model to another.
- Handles multiple AC models and find advanced security features and operations.
- Works as a guard to restrict accesses starting from the physical locations reaching to the end user.
- Dynamic enough to handle the diverse needs, use cases, and applications for AC especially with the rapid propagation and evolution of information technologies (cloud computing, IoT ...), and others.

The features of each AC model are summarized in [5]. DAC model includes three key components: a set of objects, a set of subjects, and a matrix. AC rights of subject(s) over object(s) are specified and represented as Capability Lists (CLs) and AC Lists (ACLs), which are represented as a matrix. In MAC, AC policy is managed in a centralized way, where security levels are associated with each subject and object, then permissions and actions are derived. As an alternative for DAC and MAC, RBAC model is developed, where users can be assigned some roles and a role can be associated to many users, and a role is a group of rights to use some object(s). OrBAC model is implemented to overcome some of the limitations in DAC, MAC and RBAC, and to find a more abstract control policy.

However, Figure 3 shows our vision for a Smart-AC model with the above mentioned, and AC models features. It illustrates the features and parameters for all models (subjects, objects ...) with the ability to define new ones (e.g. X, Y ...) and find the needed mappings between each model entities. Thus, any AC model can be developed by combining features from the existing models, in addition to the ability to add or define new ones. Hence, various models can be implemented, migrated, and used dynamically to enforce AC policy based on the needed security requirements. In IoT, this approach is practical, due to its dynamicity and ability to be upgraded based on the continuous technological changes. Hence, the dynamic and generic properties of such model, if implemented, 1) will help constructing new AC models based on a general AC model concept. 2) Migrating AC policies from one model to another will also help companies

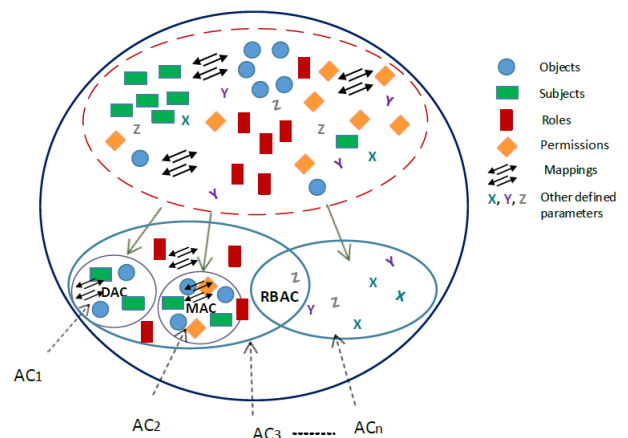


Figure 3. The vision of Smart-AC model

or industry sectors reduce the complexity and cost in this domain. 3) Different AC models can cooperate within the same company and this would be effective for IoT applications.

Each IoT layer needs the integration of AC model(s) to enforce AC policy and find secure communication environment. Various access types might exist in each IoT layer, based on the existing objects and subjects, and the needed security requirement to deny any illegal access and determine who can access what and when. Based on general architecture of IoT layers and our illustrated vision for the Smart-AC model, Figure 4 shows how any AC model can be combined with IoT layers (AC1, AC2, ... ACn...), which is derived/defined from the smart-AC model.

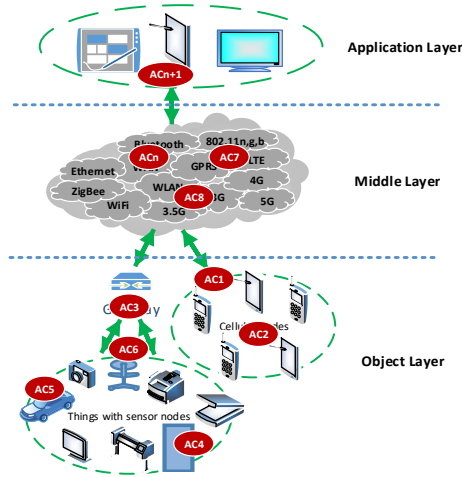


Figure 4. The vision of Smart-AC model

V. CONCLUSION AND FUTURE PERSPECTIVES

IoT is an emerging phenomenon, since various technologies and applications are combined to find a real intelligent world. Finding a secure IoT environment is a challenging issue. Various AC models are presented in literature to enforce AC policy in IoT, but they lack the idea of being upgradable or dynamic to follow technology progressions and changes. For this purpose, we present the headlines of new methodology to define a generic Smart-AC model, its concept is dynamic enough to define other AC models based on the needed security requirements. As future perspective, our aim is to implement this methodology after defining the presented headlines, of this paper, as a formal steps or guidelines and develop a general structure of the proposed methodology. Also, we will consider Industrial IoT (IIoT) or Industry 4.0 as an example to implement our methodology.

ACKNOWLEDGMENT

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT).

REFERENCES

- [1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787-2805, 2010.
- [2] S. C. Mukhopadhyay and N. K. Suryadevara, "Internet of things: Challenges and opportunities," in *Internet of Things*: Springer, 2014, pp. 1-17.
- [3] K. Ahmad, O. Mohammad, M. Atieh, and H. Ramadan, "IoT: Architecture, Challenges, and Solutions using Fog Network and Application Classification," presented at the 19th International Arab Conference on Information Technology (ACIT 2018), Lebanon, Nov. 2018.
- [4] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431-440, 2015.
- [5] N. Kashmar, M. Adda, and M. Atieh, "From Access Control Models to Access Control Metamodels: A Survey," the Future of Information and Communication Conference (FICC) 2019, US, San Francisco, March 14-15, 2019, accepted
- [6] V. C. Hu, D. F. Ferraiolo, R. Chandramouli, and D. R. Kuhn, *Attribute-Based Access Control*. London: Artech Hous, 2018.
- [7] M. Ennahbaoui and S. Elhajji, "Study of access control models," in *Proceedings of the World Congress on Engineering*, 2013, vol. 2, pp. 3-5.
- [8] A. Ouaddah, I. Bouij-Pasquier, A. A. Elkalam, and A. A. Ouahman, "Security analysis and proposal of new access control model in the Internet of Thing," in 2015 international conference on electrical and information technologies (ICEIT), 2015, pp. 30-35: IEEE.
- [9] D. Hussein, E. Bertin, and V. Frey, "Access control in IoT: From requirements to a candidate vision," in 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), 2017, pp. 328-330: IEEE.
- [10] M. Adda and R. Saad, "A data sharing strategy and a DSL for service discovery, selection and consumption for the IoT," *Procedia Computer Science*, vol. 37, pp. 92-100, 2014.
- [11] M. Adda, J. Abdelaziz, H. Mcheick, and R. Saad, "Toward an access control model for IOTCollab," *Procedia Computer Science*, vol. 52, pp. 428-435, 2015.
- [12] J. Liu, Y. Xiao, and C. P. Chen, "Authentication and access control in the internet of things," in 2012 32nd International Conference on Distributed Computing Systems Workshops, 2012, pp. 588-592: IEEE.
- [13] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," *IEEE Internet of Things Journal*, 2018.
- [14] A. Alshehri and R. Sandhu, "Access control models for cloud-enabled internet of things: A proposed architecture and research agenda," in 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC), 2016, pp. 530-538: IEEE.
- [15] S. Talari, M. Shafie-Khah, P. Siano, V. Loia, A. Tommasetti, and J. Catalão, "A review of smart cities based on the internet of things concept," *Energies*, vol. 10, no. 4, p. 421, 2017.
- [16] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

APPENDIX IV

The 14th International Conference on Future Networks and Communications (FNC) August 19-21,
2019, Halifax, Canada
Procedia Computer Science, Volume 155, 2019, Pages 417-424,
<https://doi.org/10.1016/j.procs.2019.08.058>



The 14th International Conference on Future Networks and Communications (FNC)
August 19-21, 2019, Halifax, Canada

Smart-AC: A New Framework Concept for Modeling Access Control Policy

Nadine Kashmar^{a,c,*}, Mehdi Adda^a, Mirna Atieh^b, Hussein Ibrahim^c

^a*Département de mathématiques, informatique et génie, Université du Québec à Rimouski, 300 Allée des Ursulines, Rimouski, QC, Canada*

^b*Business Computer Department, Faculty of Economic Sciences and Administration, Lebanese University, Hadat, Lebanon*

^c*Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC, G4R 5B7, Canada*

Abstract

As new technologies grow such as Internet of Things (IoT) and cloud computing, the way how people interact with devices change. The current world of interconnectivity, the heterogeneity of networks, platforms, applications, and the diversity of users make the modernization of security methods inevitable fact. Access Control (AC) is one of these essential security requirements in this domain. The continuous technology propagation forces the need to enhance AC methods, which are presented in the literature by combining features from two or more models based on a given case or scenario. The aim is to enforce AC policy and create secure communication environments. In this paper, we summarize some of the proposed methods with combined features from various AC models in the light of new technologies. Also, we present a deeper look for our idea of finding a methodology for a new dynamic Smart-AC model and a use case, in addition to the challenges to implement it. Our aim is to find a general AC framework to overcome the limitations of the presented AC methods, since they are not generic enough and do not encompass a general concept to tackle the various IT cases. The concept of our Smart-AC method is that it can be oriented to include (or exclude) all (or some) AC features for a given scenario or project, and work as a generic basis to encompass the heterogeneity of all AC models. Our proposed model aims to follow up AC requirements along with technology propagations and developments.

© 2019 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Access Control; Models; Smart; Policy, IoT; Cloud computing; Industry 4.0;

1. Introduction

The current world of interconnectivity is composed of heterogeneous networks, devices, platforms, applications, etc., such environment is vulnerable to several kinds of attacks, and this impose the need of finding heterogeneous

* Corresponding author. Tel.: +14188338800; fax: +141883311

E-mail address: nadine.kashmar@uqar.ca

Access Control (AC) models to prevent any illegal access. The idea of AC starts by implementing the principles of information security (confidentiality, authorization, authentication...), then finding different AC models such as: Discretionary Access Control (DAC) [1], Mandatory Access Control (MAC) [1, 2], Role Based Access Control (RBAC) [1, 3, 4], Organization Based Access Control (OrBAC) [3], and Attribute Based Access Control (ABAC) [1, 5]. But, the limitations of the existing AC models [6] and the continuous upgrade of IT technologies, force the need to find other AC methods with combined features to enforce AC policy and administer security requirements. However, various AC methods are presented in research field in different IT domains such as, cloud computing, IoT, etc. Moreover, recent researches are still conducting to find the needed framework with the needed characteristics.

The main concern of this paper is to summarize and analyze the reviewed AC methods in this domain [6] and show their limitations in the current IT revolution. Then, propose a deeper look for our idea of fining a methodology for a new dynamic Smart-AC model [7], and present a use case to show its flexibly for modeling various AC models (and hybrid models) in addition to the challenges for implementing this idea.

In this paper the existing AC methods with combined features and their current limitations are summarized in section 2. In section 3, our idea for dynamic Smart-AC model methodology with a deeper look is explained. A use case of how Smart-AC model is able to derive other AC models is explained in section 4. The challenges to implement this methodology are discussed in section 5. Section 6 concludes this paper with future perspectives.

2. Related Work

In a computing environment controlling access is an essential security requirement. For this purpose, various AC models are presented [1-5]. In recent years and due to the significant increase in the number of computer users, especially in the presence of new technologies, the need for upgrading AC methods becomes an important issue. Thus, different AC methods with combined features from various AC models are proposed to find a model with advanced AC features, to secure the information from unauthorized access and enforce any AC policy. For example, several model approaches with combined features from both RBAC and ABAC (called hybrid) are presented.

RBAC model does not support dynamic attributes, for this purpose Kuhn et al. in [5] address the idea of adding attributes to RBAC to find a model that supports dynamic attributes. Their aim is to handle relationships between roles and attributes to provide better AC features in dynamic environments. RBAC and ABAC models have complimentary features to each other, Rajpoot et al. in [8] present the notion of enhancing features from both RBAC and ABAC. Thus, the model Attribute Enhanced RBAC model (AERBAC) is presented, which combines the flexibility offered by ABAC and RBAC's advantages of easier administration and user permissions.

Various AC models are proposed for distributed environments and are widely implemented in cloud computing and IoT, also recent researches are still conducting in such domains. In [9], the author discusses the appropriate ABAC model features in cloud computing. An analysis of different AC mechanisms in cloud are presented in [10], which are: DAC, MAC, RBAC, ABAC, distributed RBAC (dRBAC), and cloud optimized RBAC (coRBAC). A survey on AC models in cloud computing is presented and analyzed in [11]. Furthermore, a method for providing secure AC in cloud computing is introduced in [12]. The author presents a hierarchical structure, its root is the trusted authority which authorizes the top-level domain authorities. This structure uses a clock to upload, download, and delete files to and from the cloud. It is composed of four parts: cloud owner, untrusted cloud, clock and cloud users. Another AC method is proposed in [13], the method uses a one-time password which expires in two minutes and one day password expires after twenty-four hours. For each login session the user receives passwords with encryption via e-mail. Moreover, a SmartOrBAC AC framework for IoT environment is proposed in [14]. It is based on OrBAC model coupled with the RESTFUL web services mechanisms for its preferability for the low power constrained environment. The proposed method allows the use of web service technologies to impose secure collaboration between organizations. In [15], authors propose a four-layer AC Oriented (ACO) architecture: an object layer, a virtual object layer, a cloud service layer, and an application layer. The aim of the framework is to build AC models for cloud enabled IoT. As well, a comprehensive review about AC in the IoT with the challenges and opportunities are presented in [16]. Different other AC methods are proposed in this field, which reflects the importance of finding proper AC methods in the evolutionary stage of information technologies.

The communication environment in recent technologies force the importance to find dynamic and upgradable AC methods, due to the huge number of users and the diversity of devices and platforms. The aim is to restrict and even

prevent any illegal access to any resources in different fields, e.g. industry sectors, where their current performance is widely dependent on IT technologies. After reviewing and analyzing the presented works in this domain we find that some of them analyze and discuss the appropriate AC model features, while others present AC methods to enforce AC policy. Thus, we find that these methods:

- Do not address the idea of finding solutions to existing AC models limitations.
- Designed based on some features of AC models (e.g. RBAC and ABAC).
- Are not dynamic enough to follow the rapid propagation of technologies.
- Do not encompass a general concept to handle the flexibility feature for any new extensions or transformations.

All of the above reasons and the heterogeneity of the presented AC methods for the various cases and studies inspired us with an idea for a methodology to find a new dynamic Smart-AC model, which is presented in [7]. Its aim is to find a generic concept to include all/some features of AC models and to consider the feature of dynamicity to handle diverse use cases and applications for AC, especially with the rapid propagation of IT. In this paper we explain this idea with a deeper look, describe how this idea can include all other AC models features, and how each of them can be derived from this concept.

3. The Dynamic Smart-AC Model

The existence of smart objects with the current heterogeneous communication environments (networks, applications, platforms...), inspired us to introduce a new idea for AC methodology in [7]. The idea of dynamic Smart-AC model can be used to describe access policies, where any of the AC models or their combinations might represent. In this paper we are presenting a deeper look for our model, Figure 1 illustrates the structure of our dynamic Smart-AC model. It is composed of different layers:

- **Definition of basic terms and parameters:** each AC model has its basic terms and features (e.g. subjects, objects, roles, permissions, mappings) which can be defined in this layer, with the ability to define new ones (e.g. X_i =security levels, Y_j =actions . . .)
- **Authorization Engine:** this layer is mainly composed of two parts. The first is the *formulation of AC* method to model the policies of authorization according to the needed AC model. The second is the *policy consistency* to check the coherence and the robustness of the obtained model policies before enforcing them.
- **Policy enforcement:** in this layer the AC decisions are enforced after determining whether a subject is allowed or denied to access (e.g. read, write ...) certain object(s).

Note that, the policy consistency in authorization engine layer is not addressed in this paper. However, the Smart-AC model concept has the following characteristics:

- The ability to find the needed mappings between model entities in *Access Control Model Formulation* after defining AC policy guidelines and describing the rules and regulations in, for example, an organization.
- A hybrid AC model can be developed by combining and modeling the defined parameters.
- Various models can be implemented (described in section 4) and integrated with various IT scenarios (organizations, distributed systems, IoT layers ...) to enforce AC policy.

The model concept is dynamic since it can be upgraded or modified to handle various use cases and applications for AC, especially with the rapid propagation and evolution of IT technologies. Also, it is smart because it can be oriented to include (or exclude) all (or some) AC features for a given scenario or project, and work as a framework to encompass the heterogeneity of all AC models. In the following sections various use cases are illustrated to show how our model concept can be oriented to implement other AC models. The research contribution in this paper is to show the flexibility of modeling authorization policies which are mapped to any of the AC models or their combinations using our dynamic Smart-AC model.

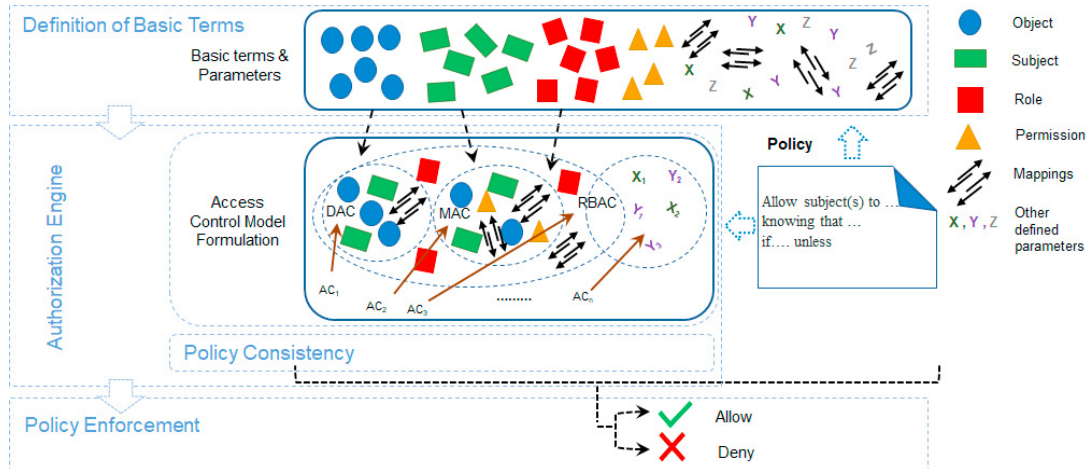


Figure 1: Smart-AC Model structure

4. Healthcare: a Use Case

This section presents a use case for the Smart-AC model concept usage, to model policies of authorization according to some of the most well-known AC models (DAC, RBAC, ABAC, and a Hybrid model). In industrial sector, AC models are widely used due to the critical need to restrict any unauthorized access and enforce AC policy. For this purpose, four AC model scenarios are illustrated in the following sections for the following policy:

Policy: *In a hospital, the receptionists (e1 and e2) in the admission office can add/write (w) and read (r) patients' records/files (f) which are identified by their Ids (fId), fname, dob, etc. In clinics, doctors (d) can write (w) and read (r) his patients' prescriptions (p). Each prescription is recorded with doctor's Id (dId), a number (pnum), date (pdate), and some details (pdetails). A nurse (n) who is helping the doctor, during her working hours within the same timezone as the hospital and have completed 30 hours of training is allowed to read this prescription. Note that, each worker in the hospital has his Id and a record with his related information (name, address...). Also, in clinics the completed training hours are calculated only for nurses.*

4.1. Discretionary Access Control (DAC)

DAC model is based on the identity of three key components: a set of objects, a set of subjects, and the AC matrix (ACM). Subjects can decide the access rules (permissions) by determining how other subjects can access their object(s) [1, 6].

DAC parameters definition for the policy:

- **Subjects:** receptionist (e1, e2), doctor (d) and nurse (n)
- **Objects:** patients' records/files (f) and patients' prescriptions (p)
- **Permissions:** read (r) and write (w)

Figure 2 illustrates the representation of Smart-AC model for DAC model configuration which is ACM. Likewise, AC rights can be depicted as Capability Lists (CLs) and AC Lists (ACLs) which can also be represented as matrices. However, the AC decisions based on DAC configuration, for policy enforcement, are: 1) The receptionists (e1 and e2) can read and write patients' files (f), 2) Doctor(s) can read and write patients' prescriptions (p), 3) The nurse(s) are allowed by doctors to read patients' prescriptions, and 4) deny any other access request.

4.2. Role Based Access Control (RBAC)

In this model subjects are given access based on their roles. A role is a group of rights to use some object(s) (e.g. accountant, director, engineer, doctor, etc.), and it can be associated to many subjects [4, 6]. Thus, in RBAC subject(s) based on their role(s) are given permissions(s) to access some object(s). The aim of RBAC is to facilitate the administration of the AC policy.

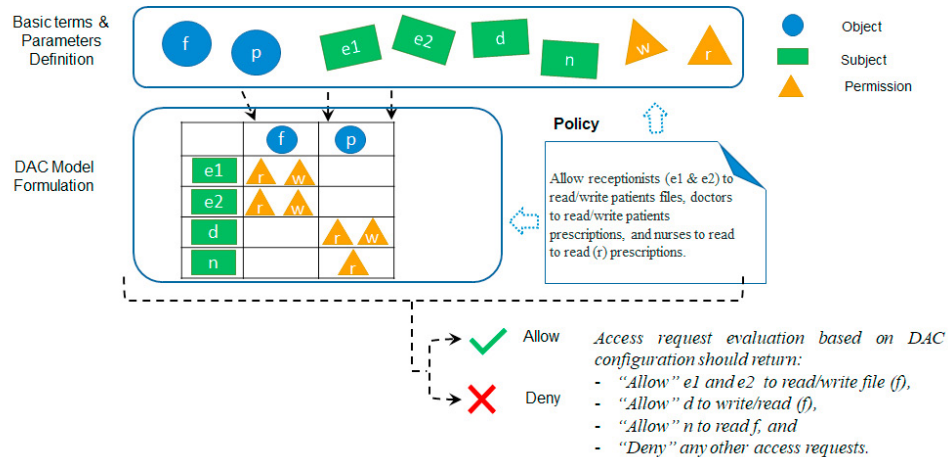


Figure 2: Smart-AC - DAC model configuration

RBAC parameters definition for the policy:

- **Subjects:** e1, e2, d, and n
- **Objects:** patients' records/files (f) and patients' prescriptions (p)
- **Roles:** receptionist (rp), doctor (dr), nurse (nr)
- **Permissions:** read (r) and write (w)

Figure 3 illustrates the representation of Smart-AC model structure for RBAC model configuration. It depicts the relations between subjects, roles and objects with the permissions for each subject based on the defined parameters for the policy. However, the AC decisions based on RBAC configuration, for policy enforcement, are: 1) allow subjects who are assigned to role *receptionist (rp)* to read and write patients' files (f), 2) allow subject(s) who are assigned to role *doctor (dr)* to read and write patients' prescriptions (p), 3) allow subject(s) who are mapped to role *nurse (nr)* to read these prescriptions, and 4) deny any other access request.

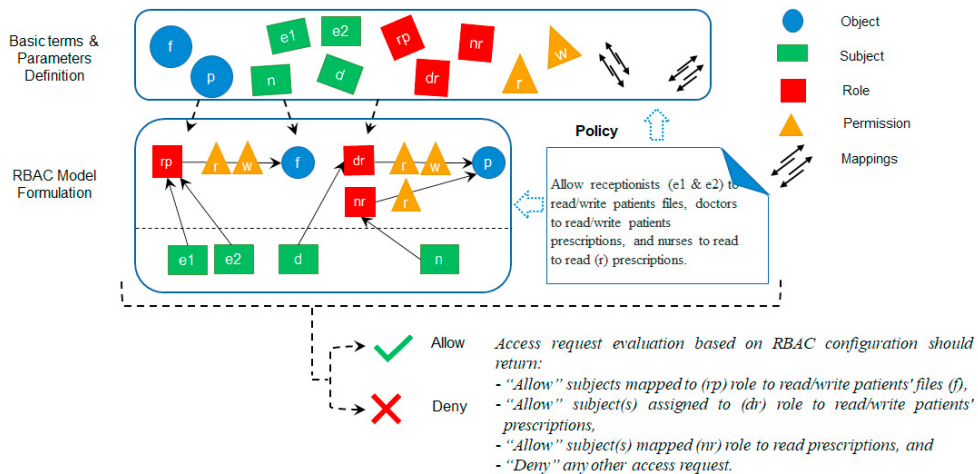


Figure 3: Smart-AC - RBAC model configuration

4.3. Attribute Based Access Control (ABAC)

The advantage of ABAC model over RBAC is that it has an ability to support dynamic attributes [1]. In this model subjects are authorized to access a wider range of objects without the need to specify individual relationships between each subject and each object. Furthermore, in ABAC there are three types of attributes: subject, object and environmental attributes. In ABAC, subjects who request to perform actions on objects are allowed or denied based on their assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions [1, 5].

ABAC is dynamic and uses subject, object and environmental attributes to determine access decision. Access control permissions are evaluated at the time of actual request is made. Consequently, to simplify representing the idea of our Smart-AC model for ABAC configuration we define subjects and objects (in addition to the attributes) to show the subjects and objects during permission evaluation at runtime.

ABAC parameters extraction for the policy:

- **Subject:** receptionist (rp); doctor (d); nurse (n)
- **Subject Attributes:** Id, name, department, ...
- **Object:** patient record/file (f), patient prescription (p)
- **Object Attributes:** fId, fname, dob...; pId, pdate, pdetails...
- **Action:** read (r) and write (w)
- **Environmental Attributes:** location (l), training hours (th), access duration (ad)

Figure 4 illustrates the Smart-AC model for ABAC configuration. The first layer depicts the defined policy attributes for subjects, objects, environment, and actions. In model formulation sublayer, subject(s) with certain attributes are allowed to access objects with some other attributes if that subject has attributes reflected in objects he wants to access. As well, subjects' actions are determined based on some conditions and constraints in the defined policy. The policy and the conditions can be implemented as follows:

- A subject working in department == "admission office" can do the action == "read" and the action == "write" for fId, fname, address ... (which identify patient record) if subject.location == object.location;
- A subject working in department == "clinic" with completed training hours="N/A" can do the action == "read" and the action == "write" for dId, pId, pdate, pdetails ... (which identify a patient prescription) if subject.location == object.location; and
- A subject working in department == "clinic" with completed training hours >= "30" and access duration time ">= 8:00, <18" can do the action == "read" for dId, pId, pdate, pdetails ... (which identify a patient prescription) if subject.location == object.location

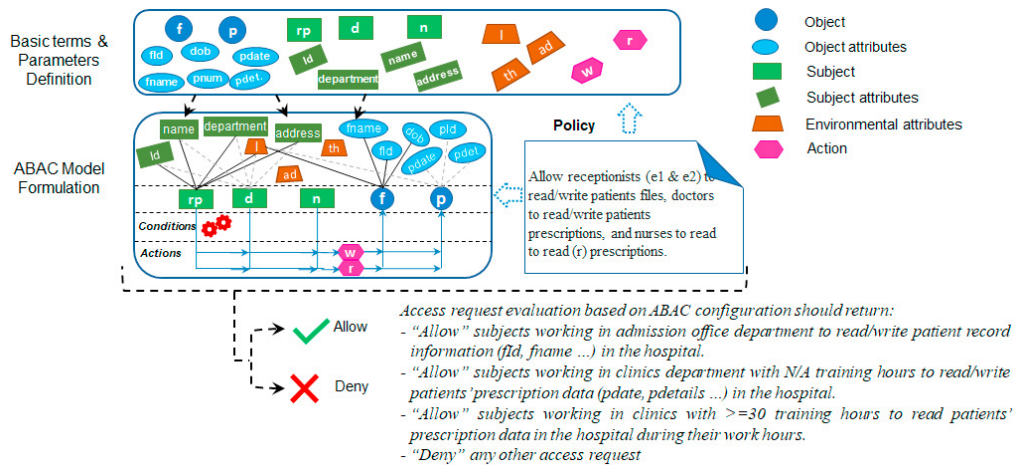


Figure 4: Smart-AC - ABAC model configuration

For the first part of the defined policy and at the time when request is made, the defined attributes for subjects (Id, name ...) with the environmental attribute (location) refer to receptionists (rp) in the admission office which have the permission to read/write patient record (f) which is identified by its attributes (fId, fname, location ...). Also, for the second part of the policy the attributes refer to a doctor with a permission to read/write prescriptions which are identified by some attributes (pdate, pdetails, location ...). Likewise, the same for the final policy part. However, the AC decisions based on ABAC configuration are: 1) allow subjects (receptionists) working in admission office department (represented as *rp* in Figure 4) to read and write patients' Ids, names, addresses, etc. in the hospital, 2) allow subjects (doctors) working in clinics to read and write patients' prescriptions details in the hospital, 3) allow subjects (nurses) working in clinics with calculated training hours >= 30 to read patients' prescriptions details and during their workinghours in the hospital, and 4) deny any other access request.

4.4. RBAC/ABAC Hybrid Model

Many AC model mechanisms with combined features from both RBAC/ABAC hybrid models are presented. For RBAC and ABAC, there are three approaches that handle the relationship between roles and attributes: dynamic role, attribute centric, and role centric. Some hybrid models use attribute centric approach where role is defined as an attribute for a subject [11], which depicts the case we consider in this part. Also, to simplify representing the idea of our Smart-AC model for hybrid configuration we define subjects and objects to indicate the permissions at the time when request is made by a subject.

RBAC/ABAC parameters extraction for the use case policy:

- **Subject:** user
- **Subject Attributes:** user-Id, username, user-role, ...
- **Object:** patient record/file (f), patient prescription (p)
- **Object Attributes:** Id, name, dob...; pId, pdate, pdetails...
- **Action:** read (r) and write (w)
- **Environmental Attributes:** location (l), training hours (th), access duration (ad)

Figure 5 illustrates the Smart-AC model for RBAC/ABAC hybrid configuration. The first layer depicts the definition of all policy attributes for subjects, objects, environment, and actions. In model formulation sublayer, subject(s) with certain role are allowed to access objects with some attributes. Also, subjects' actions are determined based on policy conditions and constraints. The hybrid model can be implemented as follows:

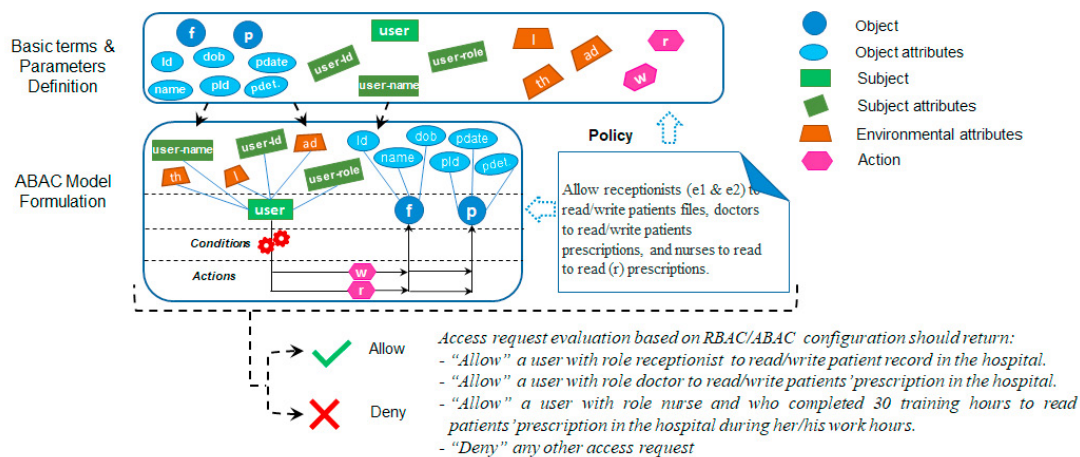


Figure 5: Smart-AC - RBAC/ABAC hybrid model configuration

- A user with role == "receptionist" can do the action == "read" and the action == "write" for fId, fname, address of patient record if user.location == object.location;
- A user with role == "doctor" can do the action == "read" and the action == "write" for dId, pId, pdate, pdetails ... of patient prescription if user.location == object.location; and
- A user with role == "nurse" can do the action == read for dId, pId, pdate, pdetails ... of patient prescription if user.traininghours == 30 and user.access duration time ">= 8:00, <18" and user.location == object.location.

All of the above models show that the Smart-AC model is dynamic to formulate other AC models. The advantage is that it can be used as a basis for this purpose.

5. Challenges

The idea of finding an AC model with hybrid features is not new. But, finding a Smart-AC model with the following features is new and challenging issue: 1) Dynamic and generic enough and can oriented to include/exclude all/some features from the existing models. 2) The ability to extract different AC models based on a general basis and finding new ones (other than the presented models in literature). 3) The obtained AC model can be extended or integrated with other AC methods to follow the IT developments. 4) The ease of migrating AC policies

from one model to another which help organizations reduce the complexity and cost. 5) The possibility to have several AC models can cooperate within the same organization. Thus, all the mentioned challenging features would be effective in a world with continuous IT developments and upgrades, and rich of heterogeneous applications, users and networks. Although these features are challenging to be included in a general framework, in this paper we start implementing the idea of our dynamic Smart-AC structure to derive other AC models.

6. Conclusion and Future Perspectives

In the current era, various technologies and applications are combined to find a real intelligent and smart world. This fact inspires us to find a new basis for finding a Smart-AC model structure which can be implemented to obtain other AC models. For this purpose we manipulate our idea for Smart-AC model methodology, which is presented in [7], with a deeper look by providing a use case of how it can be used to find other AC models. Furthermore, finding a secure communication environment is a challenging issue, for this, various AC models are presented in the literature to enforce AC policy in different IT environments, but they lack the feature of being upgradable follow technology progressions. In this paper, we explain how our idea can be dynamic and implemented to find other AC models through a simple illustration for a use case. Then, we mention the challenging features for our model to be implemented. As a future perspective, our aim is to provide detailed explanations and illustrations for our Smart-AC layers and the needed steps to formulate AC models, then implementing more of the challenging features to show its efficiency and consistency.

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT).

References

- [1] Hu, Vincent C, David F. Ferraiolo, and D Richard Kuhn. (2018) "Attribute-Based Access Control." *Norwood: Artech House*
- [2] Ge, Xiaocheng, Fiona Polack, and Régine Laleau. (2004) "Secure databases: an analysis of Clark-Wilson model in a database environment." *International Conference on Advanced Information Systems Engineering*. 234-247. Springer, Berlin, Heidelberg.
- [3] Ennahbaoui, Mohammed, and Said Elhajji. (2013) "Study of access control models." *Proceedings of the World Congress on Engineering*. 2: 3-5
- [4] Boadu, Edwin Okoampa, and Gabriel Kofi Armah. (2014) "Role-based access control (RBAC) based in hospital management." *Int. J. Softw. Eng. Knowl. Eng.* 3: 53-67.
- [5] Kuhn, D. Richard, Edward J. Coyne, and Timothy R. Weil. (2010) "Adding attributes to role-based access control." *Computer*. 43(6): 79-81.
- [6] Kashmar, Nadine, Mehdi Adda, and Mirna Atieh. (2019) "From Access Control Models to Access Control Metamodels: A Survey." In *Future of Information and Communication Conference*, 892-911. Springer, Cham.
- [7] Kashmar, Nadine, Mehdi Adda, Mirna Atieh, Hussein Ibrahim (2019) "A New Dynamic Smart-AC Model Methodology to Enforce Access Control Policy in IoT Layers." In *the 1st International Workshop on Software Engineering Research and Practices for the Internet of Things (SERP4IoT'19)*, Montreal, Canada.
- [8] Rajpoot, Qasim Mahmood, Christian Damsgaard Jensen, and Ram Krishnan. (2015) "Integrating attributes into role-based access control." In *IFIP Annual Conference on Data and Applications Security and Privacy*, 242-249. Springer, Cham.
- [9] Khan, Abdul Raouf (2012) "Access control in cloud computing environment." *ARPJ Journal of Engineering and Applied Sciences*, 7(5): 613-615.
- [10] Punithasurya, K., and S. Jeba Priya. (2012) "Analysis of different access control mechanism in cloud." *International Journal of Applied Information Systems*, 4(2): 34-39.
- [11] Aluvalu, RajaniKanth, and Lakshmi Muddana. (2015) "A survey on access control models in cloud computing." In *Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India (CSI)*, 1: 653-664. Springer, Cham.
- [12] Onankunju, Bibin K. (2013) "Access control in cloud computing." *International Journal of Scientific and Research Publications*, 3(9): 2250-3153.
- [13] Hussain, Soorat. (2014) "Access control in cloud computing environment." *International Journal of Advanced Networking and Applications*, 5(4): 2011.
- [14] Ouaddah, Aafaf, Imane Bouij-Pasquier, Anas Abou Elkalam, and Abdellah Ait Ouahman. (2015) "Security analysis and proposal of new access control model in the Internet of Thing." In *2015 international conference on electrical and information technologies (ICEIT)*, 30-35. IEEE
- [15] Alshehri, Asma, and Ravi Sandhu. (2016) "Access control models for cloud-enabled internet of things: A proposed architecture and research agenda." In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, 530-538. IEEE.
- [16] Ouaddah, Aafaf, Hajar Mousannif, Anas Abou Elkalam, and Abdellah Ait Ouahman. (2017) "Access control in the Internet of Things: Big challenges and new opportunities." *Computer Networks*, 112: 237-262.

APPENDIX V

Access Control in Cybersecurity and Social Media. Chapitre du livre Cybersécurité et médias sociaux: qui apparaitre dans Presses de l'Université Laval.

Chapitre 4 : “Access Control in Cybersecurity and Social Media” par Nadine Kashmar^a, Mehdi Adda^a, Mirna Atieh^b, Hussein Ibrahim^c

^aDépartement de Mathématiques, Informatique et Génie, Université du Québec à Rimouski, Rimouski, Québec

nadine.kashmar@uqar.ca, mehdi_adda@uqar.ca

^bBusiness Computer Department, Faculty of Economic Sciences and Administration, Lebanese University, Hadat, Lebanon, matieh@ul.edu.lb

^cInstitut Technologique de Maintenance Industrielle (ITMI), Sept-Îles, Québec

hussein.ibrahim@itmi.ca

Biographies

Nadine Kashmar is a PhD student at Université du Québec à Rimouski (UQAR), Canada. Her research project pertains to developing a new generic and enhanced access control metamodel and using it in the field of Internet of Things as a use case, specifically in Industrial IoT (IIoT). She holds her master’s degree in Science of Computer Engineering from Beirut Arab University (BAU), Lebanon in 2016. She also has work experience in the field of IT and teaching experience in Computer Science. Her research interests focus on access control, IoT, Industry 4.0, and data mining.

Mehdi Adda is a professor of Computer Science at Université du Québec à Rimouski (UQAR), Canada since June 2010. From August 2008 to May 2010, he was an invited professor at the same university. His principal research interests lie in the fields of knowledge and data engineering, IoT and security. Mehdi Adda obtained two PhD degrees in Computer Science from Université de Montréal, Canada and Université de Lille, France in 2008. He received two MSc. degrees in Computer Science from Joseph Fourier University in 2002 (Grenoble, France), and Université du Havre (Le Havre, France) in 2003 as well as an Engineering degree in Computer Science from University of Sciences and Technology Houari Boumediene (Algiers, Algeria) in 2001.

Mirna Atieh obtained her PhD in Informatics and Artificial Intelligence in February 2008 from the Institut national des Sciences Appliquées INSA de Rennes, France. She is currently an Assistant Professor and Researcher at the Lebanese University in Lebanon – Faculty of Economic Sciences and Business Administration – Department of Business Computer. Her main research interests are in the areas of Artificial Intelligence (AI), Networking and

Telecommunication, and Internet of Things (IoT). She has multiple scientific collaborations with various universities in France and Canada. She published several papers in international conferences and journals.

Hussein Ibrahim received his PhD degree in Engineering from Université du Québec à Chicoutimi (UQAC), Canada. From August 2009 to September 2016, he worked as research manager at TechnoCentre éolien in Gaspé. He has been working as research manager at Cégep de Sept-Îles in northern Quebec since September 2016, and as general manager of Institut Technologique de Maintenance Industrielle (ITMI) in Sept-Îles since September 2018. His research interests focus on renewable energy sources integration, hybrid energy power systems, storage energy, heat and mass transfer, energy efficiency, Industry 4.0 and IoT.

Abstract

Social media networks and their applications (e.g. Facebook, Twitter...) are a current phenomenon with a great impact on several aspects such as, personal, commercial, political, etc. This media is vulnerable to various forms of attacks and threats due to the heterogeneity of networks, diversity of applications and platforms, and the level of users' awareness and intentions. As network technologies and their various applications evolve, the way people interact with them changes. Thus, the main concern for all social media users is to protect their data from any type of illegal access. With network and application developments, the concept of controlling access evolves in various stages. It begins with the implementation of the principles of information security (confidentiality, authentication ...), then by finding various access control (AC) models to enforce security policy in this field. For cybersecurity and social media, various methods are developed based on conventional AC models: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC), Organization Based Access Control (OrBAC), and others.

In this chapter, we highlight the various types of cybercriminal attacks in social media networks. We then introduce the challenges faced for controlling users' access and the importance of the AC concept for cybersecurity and social media. We will also review the common AC models and the AC methods that are proposed to enhance privacy issues in social networks. Based on these methods, we will conclude our chapter by analyzing them to know their efficiency in such media and their adaptability for any future requirements.

1. Introduction

In social media, people create their own spaces to upload/share their private information (photos, videos, and audios) with family and friends using various forms of social networks, for example, Facebook, Twitter, LinkedIn, etc. Figure 1 shows the number of social network users worldwide from 2010 with a prediction up until 2020 (Rathore et al. 2017). Social networks allow users to extend their relationships and interact with strangers beyond family and friends. These networks have some interesting features (Rathore et al. 2017, Ali et al. 2018, Sayaf et al. 2014), as they:

- shorten the geographical distances between people worldwide;
- are used for entertainment, education, job searching, etc.;
- allow users to share their personal data with others in a relatively private manner;
- allow users to develop their social relationships by linking their profile with other users with similarities;
- users, companies or education sectors can create their own pages and post pertinent information in their timeline;
- used as an effective marketing strategy.

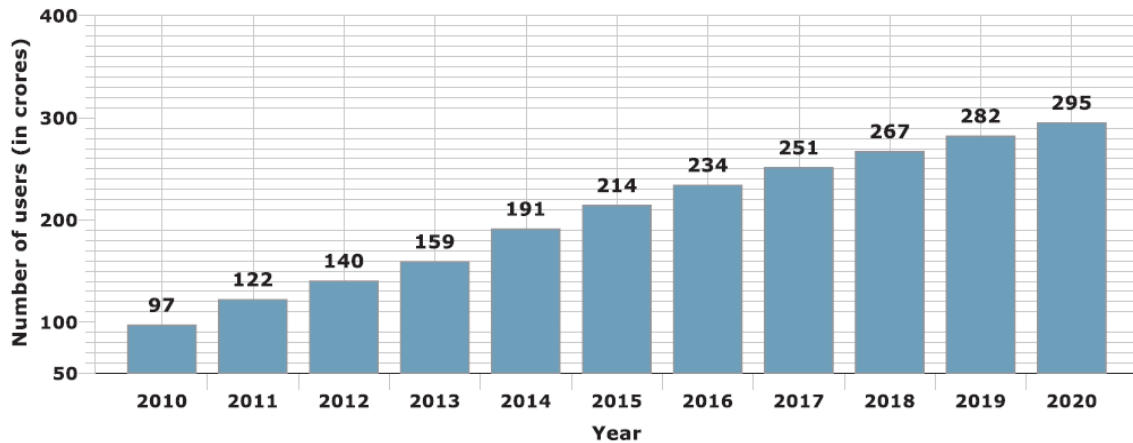


Figure 1: Number of social network users worldwide from 2010 with prediction up until 2020 (Rathore et al. 2017)

Despite these features, we cannot ignore the fact that the increase in the number of social network users and the quantity of shared and uploaded information and multimedia data, result in a tremendous increase in security threats and vulnerabilities which affect users' confidentiality, authenticity and privacy (Rathore et al. 2017, Ali et al. 2018). Furthermore, a large number of users do not grasp the risks associated with what they post. Their lack of knowledge leads to an increase in cybercrimes. Social media security threats and vulnerabilities are generally divided into three categories (Rathore et al. 2017, Fire et al. 2014, Ali et al. 2018, Patsakis et al. 2015):

- a. *Traditional threats*: have been a problem since the widespread of the internet usage, this category includes many traditional attack techniques such as malware, phishing, clickjacking, etc.
- b. *Social threats*: this category relates to the threats that intentionally target people of all ages (children, teenagers...) such as cyberbullying, cybergrooming, cyberstalking, risky behaviors, etc.
- c. *Multimedia content threats*: when social media network users generate content and upload and share it online, they could be vulnerable to several risks from malicious behaviors such as multimedia content exposure, shared ownership, tagging, unauthorized data disclosure and more.

In the literature, a variety of solutions are proposed to deal with the above-mentioned threats. AC policies are considered as one of the possible solutions for privacy settings to measure and optimize security in social networks (Rathore et al. 2017). For this purpose, different AC methods to enforce AC policies are developed to find secure communication environments and prevent any illegal access from attackers to users' information in social media networks (Sachan et al. 2011, Sayaf et al. 2014, Carminati et al. 2006). In this chapter, our concern is user security and privacy and for this purpose, we highlight the importance of AC mechanisms to mitigate security risks, then we explain some recent AC methods in this domain and how they are implemented to keep users' zones private and secure. This chapter is organized as follows. Social media network types and services are presented in section 2. The security threats and the possible vulnerabilities, also the importance of AC models for cybersecurity and social media are described in section 3. In section 4, the common AC models are summarized and the used AC methods in social media are also described. Finally, we conclude our chapter in section 5.

2. Social Media Network Types and Services

Current social media network services are web based. There exists numerous social media sites and applications with various purposes, services, and types which are developed over the years to include different types and categories. Table 1 summarizes these types of networks and provides some examples with a description for each type.

Type	Examples	Description
Social/Relationship networks	Facebook, Twitter, Whatsapp, LinkedIn, Google+	People can connect and expand their relationships.
Media sharing networks	YouTube, Instagram, Snapchat, Facebook Live, Whatsapp, Flickr	Users can share photos, videos, and other media types.

Shopping networks	Wish, GifteeHub, AliExpress	Users can shop online, send gifts, share great finds, follow brands, etc.
Discussion forums	Reddit, Quora	Based on the posted subject, people can share news and ideas.
Bookmarking networks	Pinterest, Flipboard	Users can collect content they find interesting from the Internet, save and organize it, so it may be consulted at a later date (e.g. recipes, decorating ideas, etc.).
Interest based networks	Goodreads, Soundcloud, Houzz	Allow users to connect with other users with similar interests and hobbies.
Sharing economy networks	Airbnb, Uber, Rover, Taskrabbit	These types of sites allow people to advertise, find, share, buy, sell and trade goods and services.
Consumer review networks	TripAdvisor, Booking.com, Expedia, Local.com	These types of sites allow users to find, review and share information on products, services, hotels, restaurants, etc.
Blogging networks	Wordpress, BigCommerce, Wix, Medium, Ghost	Users can build their websites and publish content online, discover and comment.
Anonymous social networks	After School, Anomo, ASKfm, NoName	These types of applications allow users to post anonymous content and share on a private message board, their feelings, accolades, to gossip or snoop.

Table 1: Social Media Network Types

Social media network services are web based and need an internet connection. Users can use web sites via various devices (computers, smart phones...) with different platforms and applications. Some sites and applications combine more than one type, for example, Facebook is a social network site where users can share media (photos and videos). Social media networks allow people to expand their relationships (personal, business, education...), shop for various items, share information, advertise products or services, and express their feelings or to gossip. As shown in Table 1, social network types allow users of all ages to

address various concerns and interests. Furthermore, it is well known that the internet world with all its related services, is vulnerable to numerous types of attacks. Hence, what are the social media security threats and vulnerabilities? Which procedures and methods are required to preserve user security and privacy? A media with several types of sites and services and a large number of users of all ages, of different cultures and intentions, opens wide the doors for such inquiries. Section 3 summarizes social media security threats, vulnerabilities and cyberattacks.

3. Social Media Networks: User Security and Privacy

Social networks let users communicate, share posts, interests, music, recommend books, movies and so on. The most serious concern for users in all aspects of their lives (e.g. personal, professional, entertainment...) is how to keep their zone secure and private. Hence, what are the possible risks when users post their private information on public networks? Before explaining the privacy requirements for social networks, it is important to present the security threats and vulnerabilities in this area.

3.1. Security Threats and Vulnerabilities

Social media users must realize that even when they use high security settings or websites, and even when they only select/accept their known friends, they are unintentionally leaking their information. Likewise, most people are unaware that the personal information they share/upload could lead to attacks against them or their friends. Hence, they must be aware and know the following:

- Once users post their information on a social network, there is no guarantee that this information is still private. As the quantity of information being posted increases, so does its vulnerability.
- It is more likely that, the more information is shared, an attacker or intruder could impersonate users and mislead their friends with actions such as download malware, share personal information, etc.
- On some social networks, information is publicly visible by default. Consequently, users should be mindful to change their privacy settings to *private* before posting information. Also, users must be aware that some other social networks change their privacy policy without their approval and that some of their information which was posted as private, may become public.
- Moreover, users should not expose themselves to several types of privacy and security issues. For example (Rathore et al. 2017, Deliri et al. 2015):
 - it is preferable not to share a large amount of personal data on social networks;

- users should not post their location (home, work...) or children's location (school, summer camping...);
- they should not provide their telephone number and credit card details, for example, for game applications;
- users should not accept friend requests from unknown people;
- most users do not read the privacy policy and terms of service for a social network before they create their accounts. They are therefore unaware of the policy nor of its updates.

Consequently, this could lead to various cybercriminal attacks. As mentioned earlier, security threats and vulnerabilities are generally divided into three categories: traditional threats, social threats, and multimedia content threats. Table 2 summarizes the various traditional attacks, such as spamming, phishing, etc. Through these types of attacks, an attacker tries to obtain personal information (password, bank account details ...) for a user to commit to some critical attacks, for example, identity theft (Rathore et al. 2017, Ali et al. 2018, Fire et al. 2014, Zhang et al. 2018, Delerue et al. 2012, Deliri et al. 2015).

Threat	Description
Malware	A malicious software, consists of Trojan horses, viruses, or worms, used by attackers to assault users by sending injected scripts to the legitimate user and when clicking a malicious URL, a malware may be installed on devices and attempt to steal personal information from the victim.
Phishing	Attackers pretend to be a legitimate entity that the victim trusts, using fake websites and emails to expose a user's sensitive private information.
Spamming	Attackers send spam or junk data in bulk to internet users which causes network congestion.
Clickjacking	A malicious mechanism used to make users click on something that is different from what they meant to click. For example, an attacker can manipulate users to post spam posts on their Facebook timeline, or use users' computer hardware (e.g. camera) to record their activities.
De-anonymization	Some users preserve their anonymity and privacy by using a false name. An attacker can find their identity by linking the information that the user has disclosed on a social network. This strategy is based on data-mining techniques where the attacker uses tracking methods, such as tracking cookies, or user group membership to expose the true identity of the user.
Identity/profile clone	An attacker can clone an existing user's profile in the same social media site or in a different one. The attacker then sends friend requests to the user's contacts and creates a trusting link with the real user's friends and collects sensitive information to carry out several types of scams (e.g. cyberbullying).

Inference attacks	An attacker can deduce a user’s private information by exploiting other information that has been published about him on social media, such as data from the user’s friends list. Then, this information is carried out using data mining techniques and can be used by the attacker to obtain internal secret information belonging to organizations.
Information leakage	Plenty of sensitive information can be inferred with great accuracy from material users share or post, or through data shared with other mutual users.
Sybil attacks & fake profiles	Attackers can create many fake identities, and by manipulating them, they can outvote the legitimate users. For example, they can boost the reputation and popularity of a user by voting him as the “best” over other legal users. They also have the capacity to corrupt information.

Table 2: Summary of traditional security threats and vulnerabilities in social media (Rathore et al. 2017, Fire et al. 2014, Ali et al. 2018, Zhang et al. 2018, Delerue et al. 2012, Deliri et al. 2015)

Table 3 summarizes the social security threats and vulnerabilities which occur in this media. Attackers can badly use the social relationship feature of social networks. This feature enables attackers to interact with different types of users and in different ways. For example, they can entice teenagers by conveying sympathy, love, or care, or by offering money or gifts. Other behaviors might include espionage, blackmail, or sharing pornographic videos and images (Fire et al. 2014, Rathore et al. 2017, Deliri et al. 2015).

Threat	Description
Cyberbullying and cybergrooming	Cyberbullying is an intentional and iterative attempt by someone online harassing or harming. Cybergrooming is when a child or a teenager is humiliated and targeted with a malicious purpose by another teenager or child via the Internet. Cyberbullying and Cybergrooming are quite dangerous as it has led teenagers to extreme acts of violence, such as committing suicide.
Cyberstalking	Some users reveal their profile’s personal information (e.g. phone number, home address, location...). This information can be tapped by malicious users for cyberstalking. For example, attackers can blackmail their victims through telephone calls or by sending instant messages using a social network site.
Risky behaviors	This may occur to teenagers or children through interactions with strangers in chat rooms, direct online communication, or giving private information and photos to an attacker. These behaviors can cause massive concerns regarding children or teenagers’ safety.
Corporate espionage	Attackers use espionage techniques for commercial or financial purposes. Such techniques can be used by a competitor posing as a worker in the target company with the intent of spying on confidential information or hacking computers.

Table 3: Summary of social security threats and vulnerabilities in social media (Rathore et al. 2017, Fire et al. 2014, Deliri et al. 2015)

Table 4 presents the different multimedia security threats and vulnerabilities content in social media. Social network is another meaning for sharing data (photos, videos, interests, and so on). Multimedia data, especially high-resolution videos and images make it easier for estimating the location, geotagging, face recognition and more via multimedia retrieval techniques. Hence, the shared multimedia data can be illegally used by an intruder to detect the user's location to find out if they are away from his home with the intention of theft (Fire et al. 2014, Rathore et al. 2017, Cutillo et al. 2010, Patsakis et al. 2015)

Threat	Description
Multimedia content exposure	Posting multimedia data (videos, images, locations...) by users can expose them to various types of attacks, as they are disclosing an enormous amount of sensitive information. For example, intruders may follow the continuous posts of users' locations and when they are not home, this leaves the door open for intruders. Also, sharing a photo or video may violate other users' privacy if it is posted without their permission.
Metadata	Metadata provides information about other data. Multimedia content is considered as metadata since it contains a huge amount of other data, e.g. users' location tags, profession, family and more. Some of this metadata may be valuable to attackers when it is revealed.
Video conference	Most social network sites support video conferencing features (e.g. Facebook) which provide more interaction between users. An intruder may restrict the broadcasting of a video stream through vulnerabilities in the underlying communication architecture, or they can access the webcam of a user by using a malware program.
Tagging	Tagging is a feature within shared multimedia data made to increase interactions between users and to facilitate search capabilities. This feature could increase privacy risks for tagged users. Some users do not like to upload pictures of themselves on social media, as this feature can represent a violation to their privacy.
Hijacking	An attacker could gain control over someone and hijack his profile if the user has a weak password. It is preferable to use strong passwords in social media accounts and to change them frequently.
Shared ownership	Multimedia content (photos or videos) may correlate to many users, e.g. two people may take a photo at an event and one person can upload this photo with their preferred privacy settings without the permission of the other.
Steganography	This is a tactic for concealing data within other media data. However, a malicious user can share malicious data by concealing it within multimedia data. For example, a picture with concealed malicious messages might be shared by a malicious user and a user may download it without knowing what it contains. This type of behavior is risky for the reputation of social network sites.
Manipulation of multimedia content	Malicious users can distort shared multimedia data by using available tools. For example, they can manipulate pictures to cause harm to others or to ridicule them.

Table 4: Summary of Multimedia Content Security Threats and Vulnerabilities in Social Media (Rathore et al. 2017, Fire et al. 2014, Patsakis et al. 2015, Cutillo et al. 2010)

After exploring and explaining the aforementioned social media services and attacks, we can recognize that social media networks are the best environments for attackers to commit cybercrimes. In this context, various researches are conducted to resolve these threats and find the best ways to mitigate or prevent them, such as spam detection (Miller et al. 2014), phishing detection (Lee et al. 2013, Gupta et al. 2018), watermarking (bin Jeffrey et al. 2017, Zigomitros et al. 2012), privacy settings (Ghazinour et al. 2016, Fiesler et al. 2017, Aldhafferi et al. 2013), authentication mechanisms (Joe et al. 2017, Ikhaliya et al. 2013, Jain et al. 2015), steganalysis (Li et al. 2015, Taleby Ahvanooy et al. 2019) and other solutions. In the light of finding solutions for social media threats and vulnerabilities, we focus on privacy which is considered as one of the fundamental security objectives in social media environments (Cutillo et al. 2010, Zhang et al. 2010, Madejski et al. 2012, Sayaf et al. 2014). Privacy solutions include (Aldhafferi et al. 2013, Rathore et al. 2017, Patsakis et al. 2015):

- Protective technologies such as strong authentication and AC mechanisms;
- Users' awareness which addresses the issue of educating users about new technologies and explaining for them the possible risks of misusing social media networks.

In this chapter, our interest is to present and analyze the used AC methods in social media networks, to find their effectiveness and how they could prevent or mitigate security threats.

3.2. The Importance of Access Control Methods for Cybersecurity and Social Media Networks

In the era of social media, a huge amount of sensitive information can be easily collected, saved or deduced. Consequently, protecting users' privacy is the main objective for the services provided by social media platforms (Cutillo et al. 2010). The above-mentioned threats (section 3.1) clearly show that there are numerous security risks with the use of social media. To minimize social media security risks, various organizations have developed a formal policy to guide users on how to use social media sites for work-related activities (Delerue et al. 2012). A formal policy is the definition of guidelines, rules or regulations for a social network site (or an organization) to determine what is an acceptable or unacceptable use of social media, what information users can or cannot share, and the consequences for not following the defined policy. Guidelines examples for an organization can be as follows (Delerue et al. 2012, Cutillo et al. 2010):

- users should use strong passwords and update them regularly;
- users should not use the same password for a social media site as the one they use for their company;
- it is not allowed for users to share their organization's information or news on social networks.

Moreover, the defined social media security policies need to be effectively implemented for social media networks. Nevertheless, some users do not comprehend or ignore the privacy policy set by social network sites due to their level of understanding or to its complexity. Thus, they are unaware of the security risks that could occur when posting information (photos, videos...). In social media, AC mechanisms allow users to define their settings of privacy via control functions such as:

- the visibility of their own information;
- allow/deny others to write on their walls;
- determining the privacy of the shared contents (public, only friends, or user-defined group);
- Share posts with friends of friends, and many other privacy settings.

Various researches mention that (Deliri et al. 2015, Cutillo et al. 2010, Aldhafferi et al. 2013), users' authentication and AC functions must be powerful so that cybercrimes from cybercriminals, hackers, or spammers can be reduced as much as possible. For this purpose, different AC models and mechanisms are developed to enforce privacy policy in social media networks. An AC framework lets users set their privacy preferences and allows application developers to create a customized plan based on users' preferences. In the following section, we first introduce the common AC models, then we review the AC models that are developed for social media networks.

4. The Access Control Models

An access control model is a formalization for policies which are defined, by an organization for instance, based on a set of principles or guidelines for its system to control and authorize access to data. The common AC models implemented to prevent illegal disclosure of sensitive data and to protect data integrity are the following: Discretionary Access Control (DAC) (Hu et al. 2017, Ennahbaoui et al. 2013, Kashmar, Adda, and Atieh 2019), Mandatory Access Control (MAC) (Hu et al. 2017, Ennahbaoui et al. 2013, Ausanka-Cruet 2001, Kashmar, Adda, and Atieh 2019), Role Based Access Control (RBAC) (Hu et al. 2017, Ennahbaoui et al. 2013, Kayem et al. 2010, Sandhu et al. 2000, Kashmar, Adda, and Atieh 2019), Organization Based Access Control (OrBAC) (Kashmar, Adda, and Atieh 2019, Ennahbaoui et al. 2013), and Attribute Based Access Control (ABAC) (Hu et al. 2017, Kashmar, Adda, and Atieh 2019, Kayem et al. 2010, Sandhu et al. 2000). Each model has its particular features and methods for making AC decisions and policy enforcement. Based on these models and due to various information technology concerns and needs in different fields, many other AC methods are extended and developed using features of two or more AC models (Kashmar, Adda et al. 2020). In the following sections, the common AC models

and their features are summarized and some recent state-of-the-art AC methods for social media networks are described.

4.1. The Common Access Control Models

Each organization has its rules, guidelines and regulations which are defined as policies to control how users can access its logical and physical assets. An AC policy defines constraints on whether a user's access request to an object should be allowed or denied. Several AC methods are implemented at different information technology (IT) infrastructure levels, they are used in operating systems, databases, networks, etc. (Kashmar, Adda, and Atieh 2019). The objectives of AC models can be summarized as follows:

- protect files and directories for organizations and information for all types of users;
- Regulate access to database objects and fields to protect application information such as payroll processing, e-health. etc.;
- Minimize the risk of unauthorized access to assets, thus minimize the risk to the business or organization.

Access right means that a subject is allowed or denied performing an operation on an object (Hu et al. 2017). AC policies might have the following form:

Allow managers to... and...

Knowing that... if... and/or...

Except...when...

Some AC policy examples can be written as follows:

- allow users A and B to read/write from/into file F for user C;
- allow technicians to read and follow the technical report instructions for machine *M*, during their working hours, if it is signed and confirmed by their technical manager;
- prevent social media users to send a friend request for user *A* if they are not friends of a friend.

Consequently, the main objective of AC models is the enforcement of the defined AC policies. In general, AC methods are defined in terms of subjects (e.g. user or program), objects (e.g. file, table or class) and access rights.

4.1.1. Discretionary Access Control (DAC)

The Discretionary Access Control (DAC) model was first introduced by Lampson (in the 1960s), a member of a curriculum design team. The three major components of this model are a set of objects, a set of domains, and a matrix (Kashmar, Adda, and Atieh 2019). Graham and Denning then extended Lampson’s work where the term *subject* was included instead of the domain. Thereafter, Harrison, Ruzzo and Ullman (HRU) extended Graham-Denning’s work to find a more flexible model with the ability to describe several AC approaches (Hu et al. 2017).

This AC model is a user-centric model where a file owner can control the permission of other users requiring access to his file. Users can control the access rights (read, write, ...) to their files with the need of a pre-specified set Matrix called Access Control Matrix (ACM). Table 5 shows how AC rights of subject(s) over object(s) are specified. The intersection of u_2 and o_2 means that u_2 can read the object o_2 . This ACM can also be implemented in two other variations, the first matrix is Capability Lists (CLs), and the second is Access Control Lists (ACLs). In CLs a user’s access rights to access objects are represented by rows, while in ACLs the access rights for various users to access an object are represented by columns.

		Objects		
		o1	o2	o3
Subjects	u1	read, write		
	u2	update	read	
	u3			delete

Table 5: Access Control Matrix (ACM)

This model is provided with operating systems to authenticate system administrators and users using passwords.

4.1.2. Mandatory Access Model (MAC)

The Mandatory Access Control (MAC) model was presented in the 1970s to include the use of a security kernel. In this model, users cannot define AC rights by themselves. MAC is based on the idea of security levels which are associated with each subject and object. These levels have hierarchical and nonhierarchical components (Ennahbaoui et al. 2013, Hu et al. 2017, Kashmar, Adda, and Atieh 2019):

- the hierarchical components include *unclassified (U)*, *confidential (C)*, *secret (S)*, and *top-secret (TS)* types where $TS \geq S \geq C \geq U$, to categorize subjects and objects into levels of

trust and sensitivity. For subjects, a security level is called *clearance level* and for objects it is called *classification level*;

- the nonhierarchical components represent a set of categories where two security properties are used as security labels to indicate security levels for classification of objects and clearance of subjects, which are *simple property* and **-property*.

There are two variants for MAC, Bell and LaPadula (BLP) and BIBA (developed by Kenneth J. Biba). The first, *simple property* indicates no read up and *star property* indicates no write down. Hence, a subject is permitted to read an object if its clearance is \geq than the object's classification, and to write if it is less than or equal (\leq). The second, *simple property* indicates no read down and *star property* indicates no write up. Consequently, a subject is permitted to read an object if its clearance is \leq than the object's classification, and to write if it is greater than or equal (\geq) (Kashmar, Adda, and Atieh 2019, Ennahbaoui et al. 2013). Moreover, MAC standards are enforced by the operating system after they are defined by a system administrator. This model is proposed to overcome the limitations of the DAC model.

4.1.3. Role-Based Access Control (RBAC)

The Role Based Access Control (RBAC) model was proposed by David Ferraiolo and Richard Kuhn in 1992, it is developed as an alternative approach to MAC and DAC (Ennahbaoui et al. 2013). The RBAC approach is based on several entities which are users, roles, permissions, actions or operations, and objects. In this model, a role means a group of permissions to use object(s) and perform some action(s), and this role can be associated to several users. Also, users can be assigned to several roles based on their qualifications and responsibilities, such as accountants, directors, engineers, etc. (Hu et al. 2017). The RBAC model was implemented to facilitate the administration of the AC policy. It administers the access of a user to objects through roles for which the user is authorized to perform.

The RBAC can be applied in distributed systems because it is based on the concept of constraints and inherence. Role hierarchy determines which roles and permissions are available to subjects based on different inheritance mechanisms (Belokosztolszki 2004, Crampton 2003).

4.1.4. OrganizationBased Access Control (OrBAC)

The Organization Based Access Control (OrBAC) was presented in 2003 and proposed to solve some limitations in the previous models (DAC, MAC and RBAC). Its aim is to find a more abstract control policy. Every organization (e.g. clinics, banks, hospitals...) is composed of a structured group of subjects having roles or entities. In OrBAC, seven entities are defined (Ennahbaoui et al. 2013):

- a. the abstract or organizational level composed of (1- Role, 2- Activity, and 3- View);
- b. The concrete level constitutes (4- Subject, 5- Action, and 6- Object), and:
- c. the seventh entity which is Context lies between the two levels to express dynamic rules for relations between entities, for example, Permission, Prohibition, Isprohibited, Recommendation, Ispermitted, Isobligatory, Isrecommended, Obligation between the elements of each level.

Thus, OrBAC exceeds the notion of granting permissions to subjects, it addresses the idea of prohibitions, obligations and recommendations. In such a way, a role may have a permission, prohibition or obligation to do some activity on some view given an associated context (Kashmar, Adda, and Atieh 2019).

4.1.5. Attribute-Based Access Control (ABAC)

The Attribute Based Access Control (ABAC) is the latest AC model development and its concepts have paralleled that of RBAC. ABAC has some advantages over RBAC, because of its ability to support dynamic attributes and its benefits in managing authorizations (Jin et al. 2012). ABAC has three types of attributes: object, subject and environmental attributes. This model allows or denies user requests based on some user attributes and on some other attributes of the object and environment. It is dynamic since it uses these attributes to determine access decisions (Hu et al. 2017), also AC permissions are evaluated at the time of the actual user's request. This offers a larger set of possible combinations of variables to reflect a larger set of possible rules to express policies (Crampton 2003). Hence, subjects are enabled to access a wider range of objects without specifying individual relationships between each subject and each object. This an ABAC advantage over RBAC. The Extensible AC Markup Language (XACML) and Next Generation AC (NGAC) are two standards that widely address the ABAC framework.

To summarize, the DAC model is proposed for the academic field, and the MAC model, for the military domain. RBAC includes features from DAC and MAC, and it is implemented to overcome some limitations of the previous models. OrBAC includes features from DAC, MAC and RBAC. It is proposed to find a more abstract control policy and to overcome the deficiencies in the previous models. RBAC has some limitations in supporting dynamic attributes such as the time of day. For this purpose, the ABAC model is proposed to support these attributes. Likewise, all the presented AC models still have some limitations (Kashmar, Adda, and Atieh 2019), this necessitates the need to find other AC methods with combined features from two or more AC models, or to integrate basic privacy requirements with the existing models. Consequently, different AC mechanisms for different computing environments are implemented. Upgrading or finding new AC mechanisms for the current challenging environments (social media networks, Internet of Things (IoT), cloud

computing, etc.) is a critical requirement to follow the continuous upgrades and to mitigate security risks by preventing any illegal access (Kashmar, Adda, et al. 2019a, b).

4.2. Access Control Models in Social Networks

In the literature, various proposals exist to address the issue of AC in social networks. In general, as social media users are vulnerable to attackers, security concerns are expanded to include their privacy, financial transactions, their families and friends, cyber-theft threats and more. In addition to these concerns, it is unacceptable to allow a hacker impersonate users and trick their friends and families. Thus, social media services, the utilized technologies, security threats and vulnerabilities reflect the fact that social media and cybersecurity incorporate different security aspects, from social media sites to user behavior. In this context, several AC mechanisms are implemented with the intent to improve and preserve user privacy. AC methods are used in social media networks to enable users to control the propagation of their own data and protect their privacy against attacks (section 3).

In the subsequent sections, we review the AC methods pertaining to this domain. Some are implemented based on features of the common AC models, while others are implemented by considering the basic privacy requirements in social networks.

4.2.1. Access Control Methods based on Features of common AC models

In conjunction with the widespread use of social media network types and services (section 2), different AC mechanisms based on features of the common AC models are implemented. This section presents some of the proposed AC methods in this field.

Tie-RBAC: RBAC application to Social Networks

Tie-RBAC is the RBAC application to Social Network Analysis (SNA) (Tapiador et al. 2012). SNA provides a comprehensive body of concepts and methods for modeling social networks, it also provides the research sector with methods for social networks analysis (O'Malley et al. 2017, Tapiador et al. 2012). Hence, Tie-RBAC indicates $RBAC + SNA = Tie-RBAC$.

The objective of Tie-RBAC is for it to be implemented in a core for building social network sites. However, social entities or actors (a user, a group, a department, an organization...) are linked by social ties, where a tie is made up of two actors and a tie type. The first and the second actors are the sender and the receiver of the tie. Tie types between actors include emotional, formal or biological relationships, transfer of material resources, messages, conversations, affiliation to same organizations, etc., for example, a tie of friendship between actor A and actor B. Hence, a relationship in the network is the set of all ties of the same type

between actors. This type of relationship is called reciprocal as in Facebook, some other relationships are non-reciprocal as in Instagram, where actor A may not follow actor B, but the opposite is true. Tie-RBAC is based on non-reciprocal ties, it is the RBAC application to social networks where:

- actors define their custom relationships (friend, partner, family etc.), which are equivalent to roles;
- each actor assigns permissions to relationships, such as post to wall, read wall, etc.;
- actors establish ties using these relationships where each tie is equivalent to the association of an actor to a role-relationship, as shown in Figure 2.

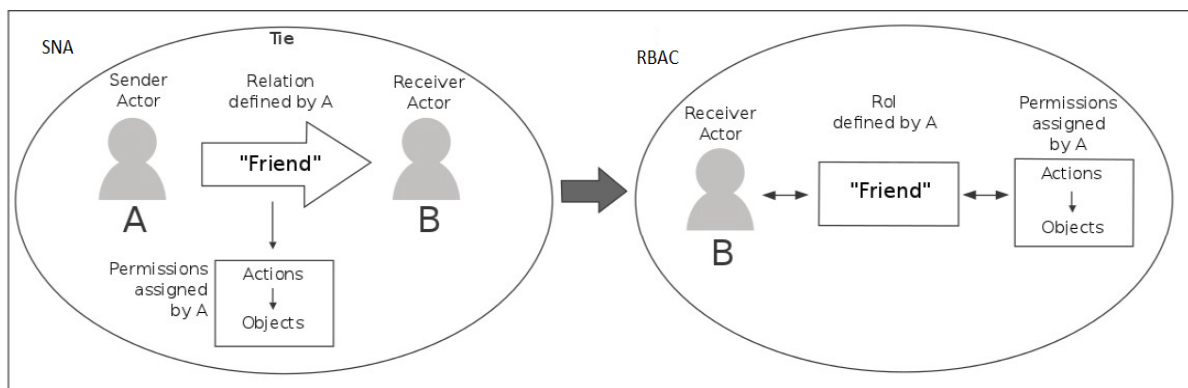


Figure 2: Tie-RBAC model: Equivalence between SNA's tie establishment and RBAC (Tapiador et al. 2012)

When establishing the tie, the sender is the entity who grants privileges to objects and the receiver is the entity assigned to the role which gains permissions on the sender's objects. In the Tie-RBAC model, the relationship which is defined by the sender is the role. In Figure 2, actor A (sender) defines the relationship "friend" and allows "friend" to read the wall and post to it. A "friend" relationship is selected when establishing the tie with actor B (receiver). Thus, actor B is authorized to read the actor's A wall and post to it.

The purpose of this model is to provide social actors and web developers with a tool to build websites with social network features, and to define their own relationships which are adapted to their field of activity. Moreover, the AC enforcement in this model is the typical RBAC (Tapiador et al. 2012).

EASiER: Encryption-Based Access Control in Social Networks with Efficient Revocation

EASiER is an architecture that supports fine-grained AC policies and dynamic group membership by using Attribute-Based Encryption (ABE) (Jahid et al. 2011). The aim of EASiER is to shift AC policy enforcement from the social network provider to the user by

means of encryption in order to mitigate the privacy risks in social networks. This case creates a key challenge in managing to support complex policies involved in social networks and dynamic groups. The key feature of this architecture, as mentioned by Jahid et al., is the possibility of removing access from a user without releasing new keys to other users or re-encrypting ciphertexts (CT). To handle this, a proxy is created to participate in the decryption process and enforce revocation restrictions. It also cannot provide access to users who are previously revoked. Figure 3 illustrates the EASiER architecture.

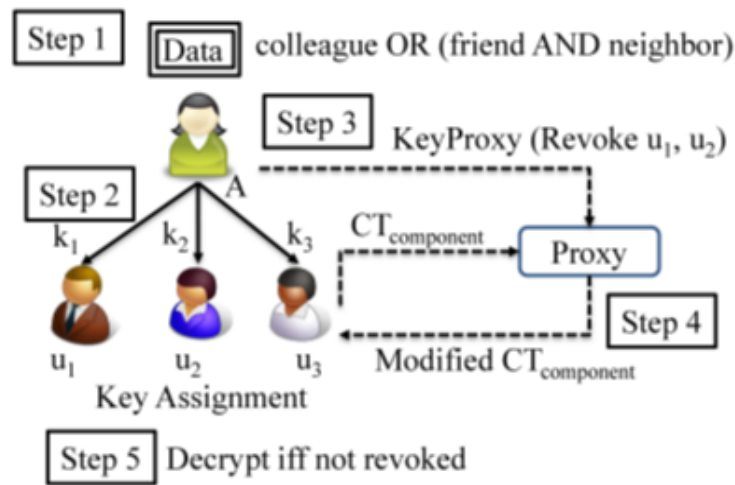


Figure 3: EASiER architecture (Jahid et al. 2011)

The primary purpose of EASiER is to protect unintentional and intentional information leakage in social networks through ABE. EASiER allows users to:

- define relationships by assigning attributes and keys to each other;
- create groups by assigning different attributes and keys to their social contacts;
- encrypt different parts of data such as profile information, wall posts, etc. with attribute policies;
- only contacts with keys having sufficient attributes that satisfy a policy can decrypt the data.

As shown in Figure 3, user or actor A can:

- define the attributes (friend, colleague, neighbor) and;
- create keys k_1 , k_2 , and k_3 for the grouping of attributes of “colleague, friend, neighbor” and for “colleague, neighbor”. Keys k_1 , k_2 , and k_3 are then assigned to u_1 , u_2 , and u_3 . User A can also encrypt his/her data with the policy “colleague or (friend and neighbor)”;

- c. user A may wish to end the relationship with u_1 and u_2 by revoking the corresponding keys which allow them to view A's data encrypted with any policy that their keys satisfactorily meet. Also, user A may wish to revoke the attribute "neighbor" from k_3 which is assigned to u_3 and do a corresponding change in access control. The proxy of each user is assigned a secret proxy key with revocation information;
- d. it then uses its key to transform CT into a form with sufficient information and
- e. that an unrevoked user can mathematically combine with his secret key, then perform decryption where a revoked user cannot do so. The proxy key allows the disclosing of the components during a decryption for unrevoked users, whereas revoked users are unable to decrypt any data since they will not get assistance from the proxy. (Jahid et al. 2011).

The EASiER mechanism does not allow the proxy to decrypt data if it does not have the attribute keys. Additionally, a new proxy key is created each time a revocation is done, hence revoked users are prohibited from conspiring against each other or the proxy to get the data. However, only particular users who have the required set of attributes can decrypt the data (Jahid et al. 2011).

Organization Based Access Control Model for Social Network

An OrBAC extension is implemented by Belbergui et al. (2016) and adapted to the Facebook context. OrBAC is an AC model based on the organization, the first-order logic is used to define relations between entities and AC policy which is defined on two levels. The first, is the abstract level (role, activity, view), and the second, is the concrete level (subject, action, object). Policy levels are adapted to the context of Facebook as follows:

- friends are defined by role (friends, friends of friends, family, etc.)
- actions are classified by activities (display, publish, etc.) and
- account owners' data is organized by views (personal information, photos, etc.).

As stated by Belbergui et al. (2016), the process of modeling the Facebook AC policy using OrBAC model is simulated based on the inventory of roles (friends, family, etc.), of activities (create, consult, etc.), of views (personal information, etc.), and of access rights (permissions). Simulation of security policy with MotOrBAC simulator is also illustrated for:

- creating organizations (Facebook, U1, etc.);
- adding abstract entities (roles, activities, views);
- adding concrete entities (subjects, actions, objects);
- adding access rights
- simulation: detection of conflicts.

Facebook is defined as an organization, users are defined as a sub-organization of Facebook, and users' accounts ($u_1, u_2 \dots$) are defined as a sub-organization of users. Roles are defined to be the usage by all users such as friends, families, studies, etc. Facebook users keep their photos, videos, etc. where other members/users are authorized/denied certain actions such as viewing pictures, writing on a wall, etc. These actions (open, view, read, search, share, etc.) can be structured into activities (create, consult, publish, block, accept, etc.). In other words, the entities *actions* define how subjects can access objects, and the structuring of these entities is called *activities*. Linking these entities is called a relation, for example, consider (org, a, a) means that the organization org considers action a, as part of activity a. Other relations exist between views and organization objects to facilitate the management of the security policy.

Based on the aforementioned concepts, Facebook-User policy and User-User policy are defined. Table 6 shows some examples for the defined policies.

Facebook-User Policy	User-User Policy
Permission (Facebook, users, create, account)	Permission (u_1 , friends, consult, publications)
Obligation (Facebook, users, compose, identifiant)	Permission (u_3 , friendOffriend, contact, account)
Permission (Facebook, users, comment, wall)	Permission (u_1 , friends, publish, wall)
Permission (Facebook, users, create, pages)	Prohibition (u_1 , public, consult, publications)
Prohibition (Facebook, friend3, publishinmywall, comment)	Prohibition (u_1 , friend1, consult, photos)
Permission (Facebook, users, accept, friend requests)	Prohibition (u_2 , public, consult, account)

TABLE 6: Examples of Facebook-User and User-User defined AC policies

For the Facebook-User policy, every user can create an account by entering his/her identity and login information such as full name, gender, age, etc. The user is then able to share messages with friends, publish photos and videos, join groups, create pages and events, etc. Users' posts and publications can be managed by account owners, consulted or commented on by other Facebook users. For a User-User policy, users are able to manage their posts and information, and can allow or deny their friends, family or public users to access their data. By using the MotOrBAC simulator, the organization (Facebook) is defined, Facebook abstract and concrete entities, subjects and their association to roles, the context and permissions at an abstract level are also defined. Figure 4 illustrates the roles and definition of Facebook. Figure 5 indicates the generation of permissions at a concrete level, which are automatically generated by MotOrBAC (using *update* tool).

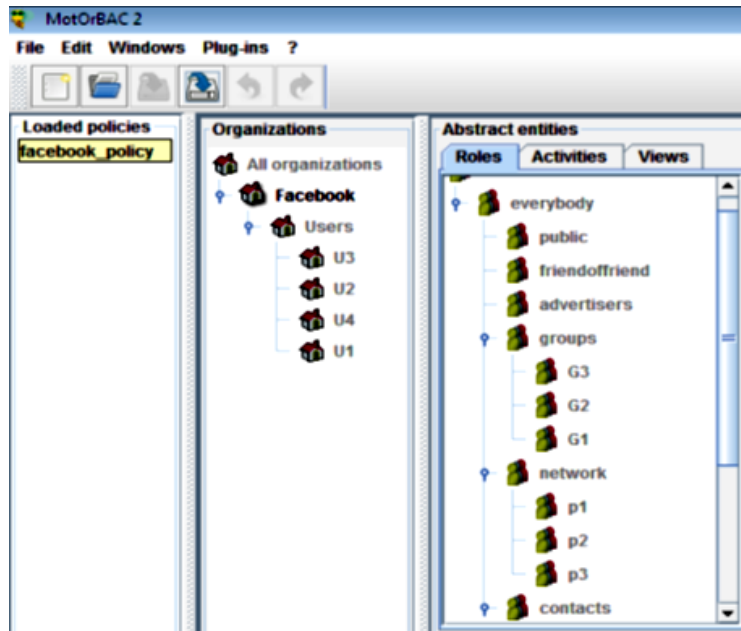


Figure 4: The roles and definition of Facebook using MotOrBAC (Belbergui et al. 2016)


Contexts	Abstract rules	Concrete rules	Conflicts	Entity definitions
Permissions	Prohibitions	Obligations		
 update	Derives from	Subject	Action	Object
	permission6	Aimee	search	statut1
	permission13	David	change	email_address
	permission6	Aimee	see	site
	permission2	Alexander	search	pictures
	permission6	Aimee	view	number
	permission13	David	change	identif_video1
	permission6	Aimee	search	March
	permission6	Aimee	read	single
	permission13	David	change	pseudo
	permission6	Aimee	read	nothing_to_report
	permission9	Emett	allow	photo

Figure 5: The generation of permissions at the concrete level (Belbergui et al. 2016)

The detection of policy coherence then follows and counts the conflicts in abstract and concrete levels, here are some examples:

- Example 1: Conflict between permissions and prohibitions defined by two users:

Prohibition (u_1 , everyone, consult, relation $u_1_u_4$)

Permission (u_4 , everyone, consult, relation $u_1_u_4$)

Users u_1 and u_4 are friends, when u_1 prohibits everyone to consult this friendship and u_4 allows it, this generates a conflict. Such conflicts show that Facebook does not suggest any solution to these types of conflicts for users.

- Example 2: Conflict between Facebook permissions and user u_1 prohibitions:

Prohibition (u_1 , advertisers, publishinmywall, publications)

Permission (Face, advertisers, publishinmywall, publications)

Although user u_1 chooses not to publish advertisements on his wall, Facebook obliges him to be contacted by advertisers.

- Example 3: Conflict between permissions and prohibitions assigned by Facebook to users.

Prohibition (Face, P3, AccessControl, profile_photo)

Permission (Face, P3, AccessControl, photos)

In this example, Facebook authorizes users to control the access to all of their photos except to their profile photo, which is always public.

Hence, OrBAC extension, which is adapted to Facebook, is used to analyze the coherence/incoherence of the Facebook security policy to enhance privacy features.

4.2.2. Other Access Control Methods

· *Multiparty Access Control Model*

Social media networks provide virtual zones or spaces for users which are identified by their profile information and contain a list of friends for each user, web pages or walls. Although these networks provide some AC mechanisms that allow users to manage access to their own information within their individual space, they do not provide control over information that exists outside their own space. For this purpose, Hu et al. (2012b) propose a Multiparty Access Control (MPAC) Model to address the following issues:

- Users can post comments on their friends' spaces (or wall), but they cannot specify which users can view them. They can tag friends by uploading photos, but the tagged friends are unable to control who can see these photos and privacy concerns may be an issue.
- Social networks provide primitive protection mechanisms for these issues, such as:

- allowing tagged users (e.g. Facebook) to remove the tags linked to their profiles;
- allowing users to report violations to social network site managers by requesting for removal of the content they refuse to share with the public.

These mechanisms have several limitations as mentioned by Hu et al., for example, a tagged user’s image is still discovered by all users who are authorized by the user who tags it even when a tag is removed from a photo. For this reason, the MPAC model is proposed as a solution to facilitate collaborative management of shared data in social networks in (Hu et al. 2012b). In MPAC, three scenarios are analyzed: profile sharing, relationship sharing, and content sharing.

- In profile sharing, social applications consume user profile attributes, such as name, birthday, activities, etc. of a user’s friends. Figure 6 depicts MPAC Pattern for profile sharing where users are allowed to select some of their profile attributes to share with the applications when their friends use these applications. The user’s friend is the owner of shared profile attributes, the application is an accessor, and the user is a disseminator. Hence, a disseminator is able to share others’ profile attributes to an accessor, and together, the owner and the disseminator can specify AC policies to restrict the sharing of profile attributes.

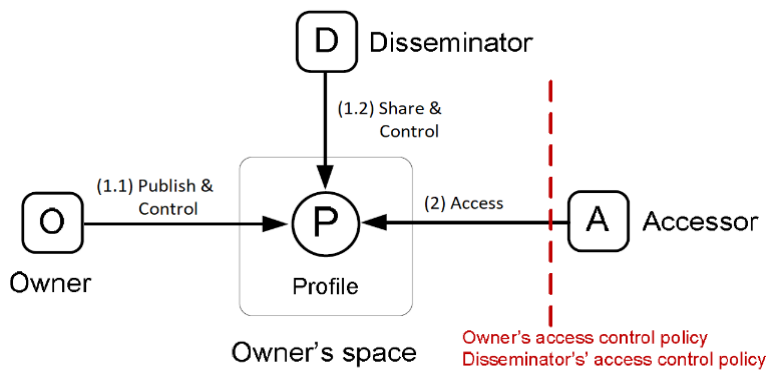


Figure 6: MPAC pattern for profile sharing (Hu et al. 2012)

- Relationship sharing is where users can share their relationships with other members and these relationships are inherently bidirectional and might carry some sensitive information. Most social networks allow users to control their friends list display, in this case a user can only manage one direction of a relationship. Figure 7 illustrates the scenario of a relationship sharing pattern. The owner which refers to the user and has a relationship with another user called stakeholder, shares the relationship with an

accessor. Hence, authorization requirements should be considered from the stakeholder and the owner, as privacy concerns for the stakeholder may be violated.

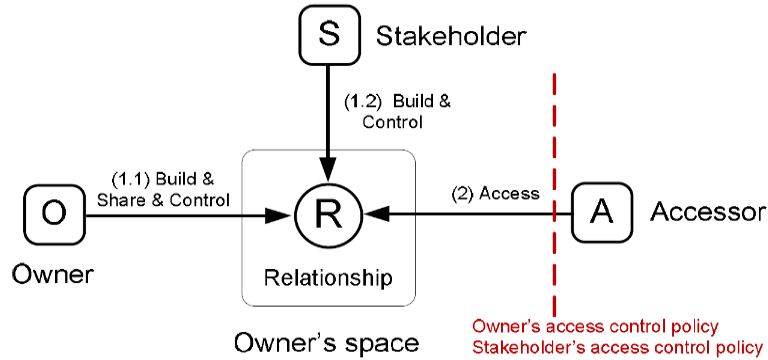


Figure 7: MPAC pattern for relationship sharing (Hu et al. 2012)

- In content sharing, users can communicate and share contents with other members. Social network users can tag or share others to the contents they upload or post on their pages. These contents may be related or connected with multiple users. In this scenario, three examples are explained to represent MAPC Pattern for content sharing and are shown in Figure 8. The first example is illustrated in Figure 8 (a), user A uploads a photo which also contains other friends B and C. User A is the owner of the photo, B and C are stakeholders of it. In this case, all users can determine AC policies to control who can see this photo, not only the owner. In other words, the content has many stakeholders who can be involved in the control of content sharing. The second is illustrated in Figure 8 (b), when user A posts a message on B's wall mentioning user C. In this case, user A is called a contributor of the message, user C is identified by a mention and considered as a stakeholder of the message, hence users A and C may want to control the disclosure of this message. As shown in Figure 8 (b), a contributor (user A) publishes a content to others' wall and this content might have many stakeholders or tagged users. In this case, all related users should be authorized to define AC policies for the posted content. The third example is demonstrated in Figure 8 (c), where users are allowed to share others' content in such a way where user A shares content (e.g. a photo) with his friends after viewing it in user B's wall. The shared photo is now in user A's space and he can determine AC policy to allow/deny his friends to see this photo. In this situation, user A is a photo disseminator and his privacy concerns might differ from that of user B. This could cause leakage of sensitive information via the data dissemination procedure. Hence, in Figure 8 (c), the owner or the contributor shares his content by uploading and publishing it, then the disseminator is able to view and share this content. In this case, to

regulate content access in the disseminator’s space, all AC policies that are defined by associated users must be enforced.

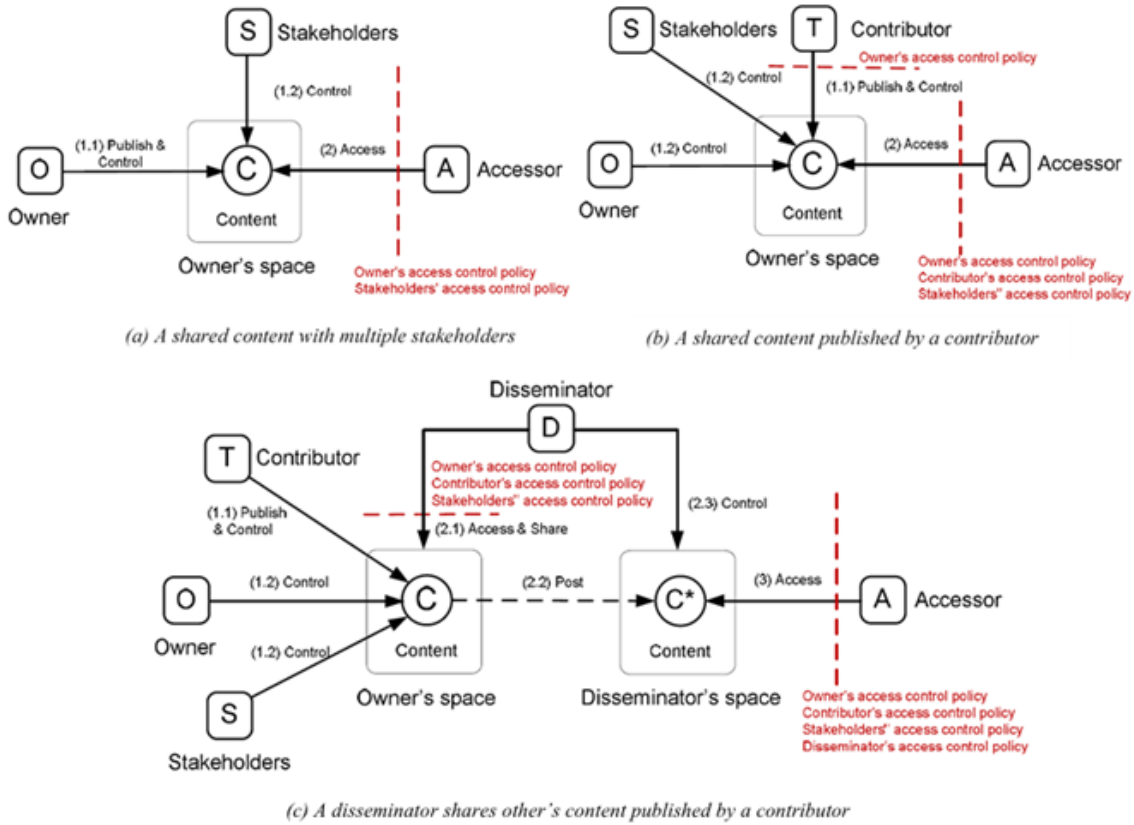


Figure 8: MPAC pattern for content sharing (Hu et al. 2012)

In a MPAC system, a group of users can collaborate together to influence the final AC decision.

PACMAN: Personal Agent for Access Control in Social Media

Personal Agent for AC in Social Media (PACMAN) is proposed by Misra et al. (2017) as a personal assistant agent that recommends personalized AC decisions on any information disclosure on social environments. This can be done by combining groups generated from the user’s network structure and using information in the user’s profile. Since social media users do plenty of interactions, an appropriate mechanism is needed to control information access by selecting the appropriate audience or friends from their lists. PACMAN is presented as a personal agent for a user to calculate accurate recommendations and minimize obtrusiveness. Figure 9 illustrates PACMAN components and inputs to produce an AC recommendation (“allow” or “deny”).

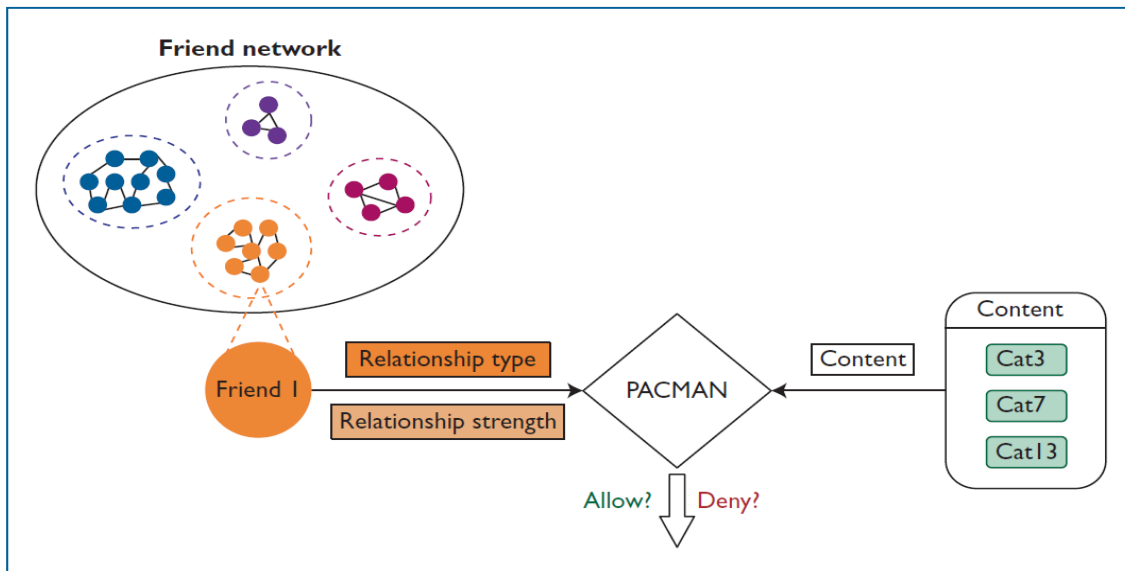


Figure 9: Components and inputs of PACMAN (Misra et al. 2017)

Relationship types are the interpersonal interactions between friends, colleagues, family, etc. and social media users. Relationship strength is the strength or closeness of interpersonal relationships between social media users. This strength is estimated by measuring similarities between users’ profiles. For this purpose, various methods are proposed for estimating the relationships tie-strength or closeness (Fogués et al. 2014, Misra et al. 2016a). Users’ total friends and mutual friends, as stated by Misra et al. (2017), are the most suitable profile attributes to support prediction of AC decisions. Moreover, the content of information that is being shared is used to enhance AC methods. To address this issue, various methods are used to generate attributes depending on the nature of the content, “for example, natural language processing techniques can be used for text, and image processing can be used for photos (Misra et al. 2017)”.

For PACMAN implementation, several building blocks are used (Figure 9), and the user’s friend network is required as an input. To represent the relationship type, one of the network-based community detection algorithms used is called Clique Percolation Method (CPM) (Misra et al. 2016b). For relationship strength, total friends and mutual friends, which are fetched from the users’ profile, are also used as input to the PACMAN mechanism. For the type of content being shared, various methods to obtain content information can be applied, Misra et al. (2017) use “manual selection of photo categories in the form of “tags” to represent the information about content”. Consequently, “allow” or “deny” AC decisions to the user which are recommended by PACMAN are of equal importance, this reflects the importance of accuracy in this mechanism. Accuracy is calculated as a percentage of the total recommendations that are correct, where:

$$\text{Accuracy} = ((F - \text{Errors})/F).$$

F is the number of total friends for a user. Errors include allow and deny errors, such errors as stated in (Misra et al. 2017) arise in the following cases:

- “An *allow* error occurs when PACMAN recommends a *deny* decision to the user when it actually should have been *allowed*. These errors are essentially *false negative (FN)* recommendations and result in a *deny* to *allow* change being made by the user.”
- “A *deny* error occurs when PACMAN recommends an *allow* decision to the user when it actually should have been *denied*. These errors are *false positive (FP)* recommendations and result in an *allow* to *deny* change by the user.”

Hence, Errors = FN + FP

For the experiment, an application similar to Facebook is created using Facebook Query Language (FQL), and a sample of 26 participants are asked to upload 10 photos (per user). The users are then asked to select categories for the photos in the form of tags to represent the content information. To calculate accuracy of prediction produced for each individual user, Weka is integrated into PACMAN to create and run the classifier applying 10-fold cross validation. Thereafter, accuracy of recommendations produced by PACMAN are shown in Figure 10.

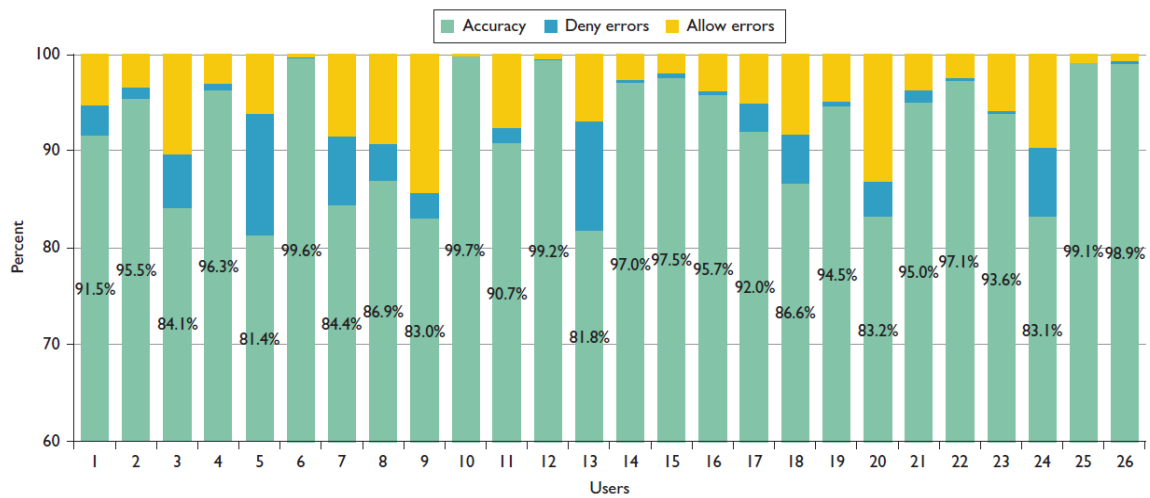


Figure 10: Accuracy and ratio of changes required to recommendations made by PACMAN for all 26 users (Misra et al. 2017)

The ratio of incorrect recommendations, *allow* and *deny* errors, shows that PACMAN produces good quality recommendations since highly accurate recommendations are demonstrated for almost all users.

5. Conclusion

Access control methods are used in computing environments to mitigate security and privacy risks of unauthorized and illegal access to data. These methods vary depending on the underlying structure of the system environment and the needed level of protection. In this chapter, a spacious overview of the main aspects of AC methods as solutions to various privacy and security related issues in social media networks is provided. First, the social media network types and services, the possible threats, and the main privacy problems in these networks are reviewed. Then, the importance of AC methods and the essential requirements for social networks are explained. Subsequently, the common AC methods that are used as a basis and implemented in different computing environments are summarized. Consequently, we present the state-of-the-art for some recent AC methods for social networks, and in the description of each presented method, we highlight the main contribution of the model with the different approaches.

Based on the aforementioned work we find that, although social media networks have a set of privacy policies, they are vulnerable to various kinds of attacks and privacy issues. The kind of personal data in these networks needs a high level of privacy protection by means of appropriate access control. Some AC methods are proposed in this domain to tackle the particular structure and the fundamental privacy issues of social networks, and some other AC methods are proposed and dedicated only for some social network sites such as Facebook (Anwar et al. 2010) and Google+ (Hu et al. 2012a). In this context, is it possible to find an AC method that works as a general basis and include all the needed features to enforce AC policy in social network sites, since all the presented methods and despite their different mechanisms focus on the same privacy issues for social media users. Furthermore, the proposed AC methods reflect that finding AC methods for social network users is a recent research issue, and research is still being conducted due to the lack of privacy features of social network sites, especially that social networks are dynamically changing environments.

References

- Aldhafferi, Nahier, Charles Watson, and AS Sajeev. 2013. "Personal information privacy settings of online social networks and their suitability for mobile internet devices." *arXiv preprint arXiv:1305.2770*.
- Ali, Shaukat, Naveed Islam, Azhar Rauf, Ikram Din, Mohsen Guizani, and Joel Rodrigues. 2018. "Privacy and Security Issues in Online Social Networks." *Future Internet* 10 (12):114.
- Anwar, Mohd, Zhen Zhao, and Philip WL Fong. 2010. An access control model for Facebook-style social network systems. University of Calgary.

- Ausanka-Cruces, Ryan. 2001. "Methods for access control: advances and limitations." *Harvey Mudd College* 301:20.
- Belbergui, Chaimaa, Najib Elkamoun, and Rachid Hilal. 2016. "Modeling Access Control Policy of a Social Network." *International Journal of Advanced Computer Science and Applications* 7 (6).
- Belokosztolszki, András. 2004. Role-based access control policy administration. University of Cambridge, Computer Laboratory.
- Bin Jeffrey, Mohd Aliff Faiz, and Hazinah Kutty Mammi. 2017. "A study on image security in social media using digital watermarking with metadata." 2017 IEEE Conference on Application, Information and Network Security (AINS).
- Carminati, Barbara, Elena Ferrari, and Andrea Perego. 2006. "Rule-based access control for social networks." OTM Confederated International Conferences" On the Move to Meaningful Internet Systems".
- Crampton, Jason. 2003. "On permissions, inheritance and role hierarchies." Proceedings of the 10th ACM conference on Computer and communications security.
- Cutillo, Leucio Antonio, Mark Manulis, and Thorsten Strufe. 2010. "Security and privacy in online social networks." In *Handbook of Social Network Technologies and Applications*, 497-522. Springer.
- Delerue, Helene, and Wu He. 2012. "A review of social media security risks and mitigation techniques." *Journal of Systems and Information Technology*.
- Deliri, Sepideh, and Massimiliano Albanese. 2015. "Security and privacy issues in social networks." In *Data Management in Pervasive Systems*, 195-209. Springer.
- Ennahbaoui, Mohammed, and Said Elhajji. 2013. "Study of access control models." Proceedings of the World Congress on Engineering.
- Fiesler, Casey, Michaelanne Dye, Jessica L Feuston, Chaya Hiruncharoenvate, Clayton J Hutto, Shannon Morrison, Parisa Khanipour Roshan, Umashanthi Pavalanathan, Amy S Bruckman, and Munmun De Choudhury. 2017. "What (or who) is public?: Privacy settings and social media content sharing." Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing.
- Fire, Michael, Roy Goldschmidt, and Yuval Elovici. 2014. "Online social networks: threats and solutions." *IEEE Communications Surveys & Tutorials* 16 (4):2019-2036.
- Fogués, Ricard L, Jose M Such, Agustin Espinosa, and Ana Garcia-Fornes. 2014. "BFF: A tool for eliciting tie strength and user communities in social networking services." *Information Systems Frontiers* 16 (2):225-237.

- Ghazinour, Kambiz, Stan Matwin, and Marina Sokolova. 2016. "YOURPRIVACYPROTECTOR, A recommender system for privacy settings in social networks." *arXiv preprint arXiv:1602.01937*.
- Gupta, Brij B, Nalin AG Arachchilage, and Kostas E Psannis. 2018. "Defending against phishing attacks: taxonomy of methods, current issues and future directions." *Telecommunication Systems* 67 (2):247-267.
- Hu, Hongxin, Gail-Joon Ahn, and Jan Jorgensen. 2012a. "Enabling collaborative data sharing in google+." 2012 IEEE Global Communications Conference (GLOBECOM).
- Hu, Hongxin, Gail-Joon Ahn, and Jan Jorgensen. 2012b. "Multiparty access control for online social networks: model and mechanisms." *IEEE Transactions on Knowledge and Data Engineering* 25 (7):1614-1627.
- Hu, Vincent C., David F. Ferraiolo, Ramaswamy Chandramouli, and D. Richard Kuhn. 2017. *Attribute-Based Access Control* Norwood: Artech House.
- Ikhaliya, E, and CO Imafidon. 2013. "The need for two factor authentication in social media." Proceedings of the International Conference on Future Trends in Computing and Communication-FTCC.
- Jahid, Sonia, Prateek Mittal, and Nikita Borisov. 2011. "EASiER: Encryption-based access control in social networks with efficient revocation." Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security.
- Jain, Sakshi, Juan Lang, Neil Zhenqiang Gong, Dawn Song, Sreya Basuroy, and Prateek Mittal. 2015. "New directions in social authentication." Proc. USEC.
- Jin, Xin, Ram Krishnan, and Ravi Sandhu. 2012. "A unified attribute-based access control model covering DAC, MAC and RBAC." IFIP Annual Conference on Data and Applications Security and Privacy.
- Joe, M Milton, and B Ramakrishnan. 2017. "Novel authentication procedures for preventing unauthorized access in social networks." *Peer-to-Peer Networking and Applications* 10 (4):833-843.
- Kashmar, Nadine, Mehdi Adda, and Mirna Atieh. 2019. "From Access Control Models to Access Control Metamodels: A Survey." Future of Information and Communication Conference.
- Kashmar, Nadine, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. 2019a. "A new dynamic smart-AC model methodology to enforce access control policy in IoT layers." Proceedings of the 1st International Workshop on Software Engineering Research & Practices for the Internet of Things.

- Kashmar, Nadine, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. 2019b. "Smart-AC: A New Framework Concept for Modeling Access Control Policy." *Procedia Computer Science* 155:417-424.
- Kashmar, Nadine, Mehdi Adda, Mirna Atieh, and Hussein Ibrahim. 2020. "Deriving Access Control Models based on Generic and Dynamic Metamodel Architecture: Industrial Use Case." The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks. (*Accepted*)
- Kayem, Anne VDM, Selim G Akl, and Patrick Martin. 2010. *Adaptive cryptographic access control*. Vol. 48: Springer Science & Business Media.
- Lee, Sangho, and Jong Kim. 2013. "Warningbird: A near real-time detection system for suspicious urls in twitter stream." *IEEE transactions on dependable and secure computing* 10 (3):183-195.
- Li, Fengyong, Kui Wu, Jingsheng Lei, Mi Wen, Zhongqin Bi, and Chunhua Gu. 2015. "Steganalysis over large-scale social networks with high-order joint features and clustering ensembles." *IEEE Transactions on Information Forensics and Security* 11 (2):344-357.
- Madejski, Michelle, Maritza Johnson, and Steven M Bellovin. 2012. "A study of privacy settings errors in an online social network." 2012 IEEE International Conference on Pervasive Computing and Communications Workshops.
- Miller, Zachary, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. 2014. "Twitter spammer detection using data stream clustering." *Information Sciences* 260:64-73.
- Misra, Gaurav, and Jose M Such. 2017. "Pacman: Personal agent for access control in social media." *IEEE Internet Computing* 21 (6):18-26.
- Misra, Gaurav, Jose M Such, and Hamed Balogun. 2016a. "IMPROVE-Identifying Minimal PROfile VEctors for similarity based access control." 2016 IEEE Trustcom/BigDataSE/ISPA.
- Misra, Gaurav, Jose M Such, and Hamed Balogun. 2016b. "Non-sharing communities? an empirical study of community detection for access control decisions." 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM).
- O'Malley, A James, and Jukka-Pekka Onnela. 2017. "Introduction to social network analysis." *Methods in Health Services Research*:1-44.
- Patsakis, Constantinos, Athanasios Zigomitros, Achilleas Papageorgiou, and Agusti Solanas. 2015. "Privacy and security for multimedia content shared on OSNs: issues and countermeasures." *The Computer Journal* 58 (4):518-535.

- Rathore, Shailendra, Pradip Kumar Sharma, Vincenzo Loia, Young-Sik Jeong, and Jong Hyuk Park. 2017. "Social network security: Issues, challenges, threats, and solutions." *Information sciences* 421:43-69.
- Sachan, Amit, and Sabu Emmanuel. 2011. "Efficient Access Control in Multimedia Social Networks." In *Social Media Modeling and Computing*, 145-165. Springer.
- Sandhu, Ravi, David Ferraiolo, and Richard Kuhn. 2000. "The NIST model for role-based access control: towards a unified standard." ACM workshop on Role-based access control.
- Sayaf, Rula, and Dave Clarke. 2014. "Access control models for online social networks." In *Digital Arts and Entertainment: Concepts, Methodologies, Tools, and Applications*, 451-484. IGI Global.
- Taleby Ahvanooy, Milad, Qianmu Li, Jun Hou, Ahmed Raza Rajput, and Chen Yini. 2019. "Modern text hiding, text steganalysis, and applications: a comparative analysis." *Entropy* 21 (4):355.
- Tapiador, Antonio, Diego Carrera, and Joaquín Salvachúa. 2012. "Tie-RBAC: an application of RBAC to Social Networks." *arXiv preprint arXiv:1205.5720*.
- Zhang, Chi, Jinyuan Sun, Xiaoyan Zhu, and Yuguang Fang. 2010. "Privacy and security for online social networks: challenges and opportunities." *IEEE network* 24 (4):13-18.
- Zhang, Zhiyong, and Brij B Gupta. 2018. "Social media security and trustworthiness: overview and new direction." *Future Generation Computer Systems* 86:914-925.
- Zigomitos, Athanasios, Achilleas Papageorgiou, and Constantinos Patsakis. 2012. "Social network content management through watermarking." 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications.

APPENDIX VI

The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks
(EUSPN 2020) November 2-5, 2020, Madeira, Portugal
Procedia Computer Science, Volume 177, 2020, Pages 162-169,
<https://doi.org/10.1016/j.procs.2020.10.024>



The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks
(EUSPN 2020)
November 2-5, 2020, Madeira, Portugal

Deriving Access Control Models based on Generic and Dynamic Metamodel Architecture: Industrial Use Case

Nadine Kashmar^{a,c,*}, Mehdi Adda^a, Mirna Atieh^b, Hussein Ibrahim^c

^a*Département de mathématiques, informatique et génie, Université du Québec à Rimouski, 300 Allée des Ursulines, Rimouski, QC, Canada*

^b*Business Computer Department, Faculty of Economic Sciences and Administration, Lebanese University, Hadat, Lebanon*

^c*Institut Technologique de Maintenance Industrielle, 175 Rue de la Vêrendrye, Sept-Îles, QC, G4R 5B7, Canada*

Abstract

With the rapid propagation of technologies and their heterogeneous structure of networks, platforms, applications, devices, etc., controlling access to physical and logical resources becomes a necessary requirement. The existing Access Control (AC) models (also hybrid models) are not sufficient to satisfy the current needs of security requirements with this diversity of technological aspects and computing environments due to fact of being vulnerable to various kinds of attacks and threats. Hence, AC metamodels are proposed to serve as a unifying framework or to work as a general basis to derive multiple AC models as special cases or instances. In the literature, several AC metamodels are presented for various scenarios for centralized (e.g. organizations, industries, ...) and decentralized (e.g. cloud computing, internet of things, ...) environments to enforce AC policies. Despite some of their advantages, they lack some essential features and have limitations specially with the current technology propagations. In this paper, we propose a dynamic and enhanced architecture for an AC metamodel, and present an industrial use case to explain how this architecture is generic to derive various AC model instances and can be used as a base for any future developments in this domain.

© 2020 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0>)

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Access control; metamodel; security and privacy; industry; IoT

* Corresponding author. Tel.: +14188338800; fax: +141883311

E-mail address: nadine.kashmar@uqar.ca

1. Introduction

Nowadays, securing logical and physical assets cannot be easily achieved by traditional AC methods due to the current fact of open computing environments, their complex architectures, and the diversity of devices and users. In general, traditional AC models focus on which subject (user) is performing what action on which object. The AC models that are presented in the literature are: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC), and Attribute Based Access Control (ABAC) models [1-3]. To enhance some features of AC methods, some of these models are combined to have hybrid models in order to define a larger set of AC policies, some hybrid AC models are presented in [4-6]. Nevertheless, hybrid models are not enough to satisfy the current security requirements and the continuous progressions in various computing domains such as Internet of Things (IoT), cloud computing, etc. [7]. Hence, AC metamodels with higher level of abstraction are proposed to serve as unifying framework to define various instances of AC models, and hybrid models [1, 3].

AC metamodels are proposed and implemented for different scenarios related to web services [8, 9], software frameworks (e.g. ArchiMate [10], Spring Security [11] ...), cloud computing [12], and others [3]. Despite the objectives of proposing them, they lack some essential features and have some limitations (explained in section 3) specially with the current fact of technology progressions [1, 2]. In this paper, we propose a generic and dynamic architecture for an enhanced AC metamodel, we use an industrial use case to show its dynamicity in deriving different AC (also hybrid) models.

For this paper some of the proposed AC metamodels in the literature are presented in section 2. In section 3, we summarize the common limitations of the existing AC metamodels and the problematic. In section 4, we propose the architecture of our AC metamodel. Food industry use case is proposed in section 5, to show how AC models can be constructed based on the proposed architecture. Section 6 concludes this paper with future perspectives.

2. Related Work

Access control models and even combining some features of them are insufficient to follow the continuous technology upgrades [13], hence this fact forces the need to find various AC models instances based on AC metamodels. Accordingly, different metamodels are proposed in the literature and implemented in different information technology scenarios [1, 3].

In [14], Barker propose a Category Based Access Control (CBAC) metamodel to develop many AC models by combining the primitives of AC models, hence a wider range of constraints can be expressed. A category is defined as a class of entities such as: roles, groups, security levels, etc. Slimani et al. [15], extend Barker's metamodel [14] and propose a Unified Access Control Modeling Language (UACML) to provide support for hybrid AC policies by allowing categories to be associated with other categories and finding hierarchical relationships between them. In [12], Category Based Access Control (CatBAC) framework for cloud computing services based on the notion of categories is presented. The framework allows security administrators in the various company sites to find a concrete model with the constraints and specificities of each site. Xia et al. in [16] propose Cloud Security and Privacy Metamodel (CSPM) to handle security and privacy in cloud service development and operations via integrating and extending the existing metamodels of cloud security together. Martínez et al. in [8] propose a metamodel to Web Content Management System (WCMS) AC policies, to automatically extract the AC information in WCMSs and ease the analysis of security requirements by abstracting them from vendor-specific details.

An integration metamodel for hybrid AC policies is proposed by Abd-Ali et al. in [17] to concurrently handle multiple AC models. Their idea is to abstract each AC model (e.g. RBAC metamodel), and each AC metamodel has a DecisionHandler element to determine the AC decision. An AC decision depends on more than one AC metamodel. DecisionHandler instances of a hybrid AC policy are applied to combining algorithms (ComAl) to conclude with only one AC decision as output in response to multiple AC decisions as input.

Some metamodel extensions are proposed due to the lack of security elements in software development frameworks, for example, ArchiMate and Spring Security. Hence, Korman et al. in [10] propose a unified metamodel as an extension for ArchiMate (common Enterprise Architecture (EA) modeling language) to support the development of enterprises by extending their abilities to model authorization and AC in their architectures. Also, in [11] Gorshkova et al. introduce a fine-grained AC model and provide a metamodel extension for Spring Security

framework (open source security framework for Java) to join modern security requirements. Their proposed framework defines a fine-grained extension of RBAC.

However, some of existing AC metamodels are proposed to enhance AC features of existing frameworks by extending them to support AC decisions and express more AC policies. Some other metamodels are proposed to find a general metamodel basis that include most/all AC features to derive various AC model instances.

3. Problematic

Due to the continuous increase and upgrade of the current technologies, the presence of security threats also increases. To ensure computer security, several techniques are employed for example encryption, authentication, access control, etc. [18, 19]. Access control is one of the essential security requirements in computing environments which are open to various kinds of attacks, and this makes security enforcement through AC models an urgent need [1]. However, it is essential to adapt the existing AC models (MAC, RBAC...) and/or create new ones (e.g. hybrid models [20], or others) to cope with the recent security threats. Unfortunately, it is realized that these models have reached their limits [1, 3, 21] and they no longer meet the increasing demand for privacy and security levels. Accordingly, several studies highlight the importance of developing AC metamodels with a higher level of abstraction which serve as a unifying framework for the definition of any AC policy [1, 7, 13]. Developing AC metamodels that are generic, dynamic, and upgradable is a challenging topic due to the following facts:

- The complex and heterogeneous structures of computing environments.
- The dynamic requirement for enforcing policies due to continues technology progressions.
- The importance of allowing collaboration between various AC models within the same IT architecture, e.g. IoT.
- The importance of migrating AC policies from one model to another.

In [3], we review the objective(s) and limitation(s) for each of the proposed AC metamodel. However, the common limitations that have not been addressed yet can be summarized as follows:

- They are not generic enough to derive instances for all common AC models. They are planned for dedicated case scenarios or projects based on some features of AC models.
- Neither the proposed AC metamodels nor the extended ones are dynamic enough to follow the IT upgrades.
- They do not support the possibility of defining new components for finding new AC models.
- None of the proposed metamodels explain how the derived AC models could collaborate within the same architecture (e.g. cloud computing, IoT...).
- None of them handle the feature of migration from one AC model to another.

Accordingly, in section 4 we propose a new AC metamodel architecture that is dynamic and generic enough to work as a base to overcome the aforementioned AC metamodels problems and limitations.

4. The proposed AC Metamodel Architecture

In this paper, our main concern is to find a dynamic and more generic AC metamodel architecture that includes all key components for common AC models with the ability to define new ones for 1) any new AC method(s), and for 2) any extension(s) or upgrades based on this architecture.

The architecture of our AC metamodel is illustrated in Fig. 1, it is mainly composed of 6 levels, and each level may have sublevels to formulate the needed AC method then enforce the policy. Hence:

- (1) **Level 1:** based on the defined policy, all the attributes ($a_1, a_2, a_3, \dots, a_n$; where $n \geq 0$) for subjects, objects, environment, context, etc. are extracted and defined.
- (2) **Level 2:** the defined attributes, in level 1, are aggregated to construct the needed classes (key components for AC methods), abstract and concrete classes for subjects, objects, actions, roles, etc.
- (3) **Level 3:** class instances are created for explicit components which are clearly stated in the policy and they represent subjects (S_1, S_2, \dots, S_n ; $n \geq 0$) and objects (O_1, O_2, \dots, O_n ; $n \geq 0$)

- (4) **Level 4:** class instances are created for implicit components which are extracted from a described event, operation, context, etc. and they represent roles (R_1, \dots, R_n), actions (A_1, \dots, A_n), permissions (P_1, \dots, P_n), security levels (L_1, \dots, L_n), Categories (C_1, \dots, C_n), etc.
- (5) **Level 5:** aggregate class instances for explicit and implicit components, of level 3 and level 4, to perform the needed assignments, for example: subjects (S_1, S_2, S_7) \in role (R_1), or object (O_n) \in security level (L_n), etc.
- (6) **Level 6:** apply the needed aggregations to set the permissions, actions or operations which can/cannot be performed by subjects over objects. This can be occurred by selecting and aggregating the needed class instances and assignments from levels 3, 4 and 5 to construct the AC decision (allow/deny access) and check the constraints before policy enforcement.

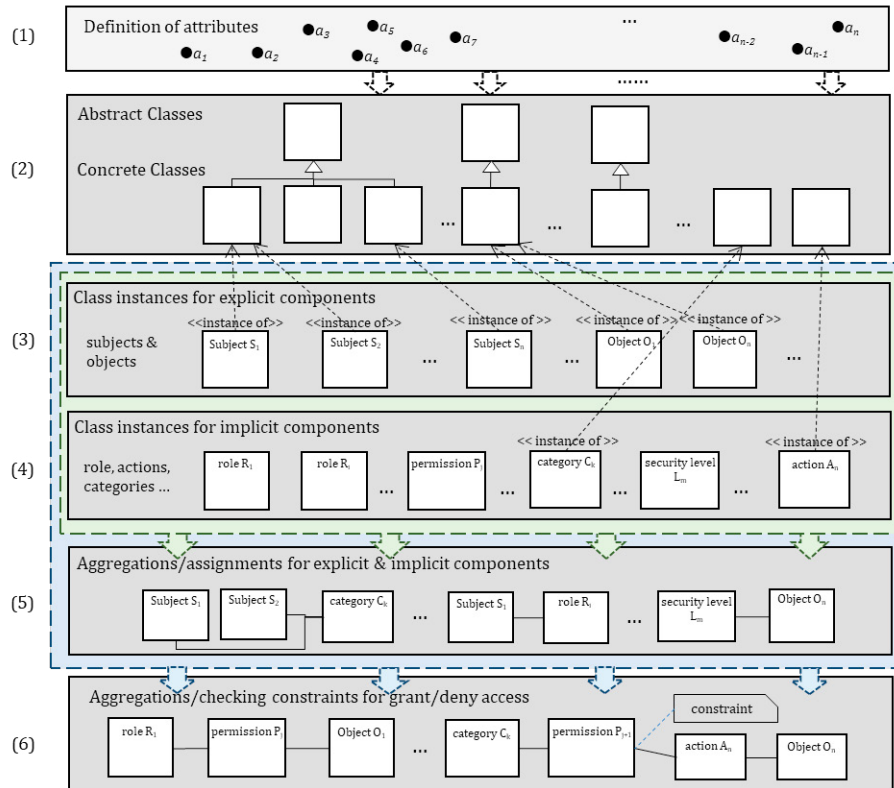


Fig. 1. The proposed AC metamodel Architecture

Constructing AC model(s) is based on the core components for each model (or hybrid models), for example, if we want to construct DAC model where its key components are subjects, objects, and permissions, the definition of attributes in layer 1 might not be considered. In section 5, we present a food industry use case with illustrations for various AC models' instances to show how they can be derived from the proposed architecture (in Fig. 1).

5. Industrial Use Case: Food Industry

In the field of industry, complex, costly, and sometimes potentially risky machines must be operated only by trained and authorized users. Hence, controlling access to machines is a significant issue to prevent any damage, to provide safe operation for machines, and to avoid any economic loss.

Food industry is one of the major industrial sectors in the world and different machines are used to handle, prepare, cook, package, and store food and food products. For this reason, companies in this domain attempt to improve their systems and machines, so their performance and productivity, by turning into information technology. Hence, all food processing aspects can be monitored and controlled by an information system. In food industry, the machines are constructed based on the demands of operation(s) and with design considerations related to machine functionality (e.g. sorting products), food safety, hygiene, integration of analytical components, and many others.

Consequently, any misuse from machine operators or maintenance engineers for them could influence their performance and efficiency, and this could lead to hazardous consequences. For example, machine-related injuries, falling objects, gas release, poisoning, etc. which in general could lead to loss of money and maybe lives.

The main goal for any food industry factory is to move the local food to consumers locally or globally with precise and defined quality. In our use case we focus on *FruitCanning* factory, its functionality, the machines, the staff, and the products, to show how some operations can be performed in such environment with the dedicated permissions for each user or group of users. Our AC metamodel architecture is used to instantiate different AC models to enforce the defined policies in the use case.

5.1. Use Case

Fruit Canning is a popular method to distribute certain fruits (or vegetables) to the marketplace. *FruitCanning* factory includes several machines for fruit processing, for example, pre-treatment machine (*M1*), fruit crusher machine (*M2*), fruit extractor machine (*M3*), sterilization machine (*M4*), and many others. Recent machines are accomplished to keep track of the settings, the daily operating hours, the identity of the operator(s), the type of the produced products, and other important data. Operating these machines can be occurred physically or via an information system, hence a well coherent access control configurations are needed. However, to ensure food quality and safety, machines must only be controlled by authorized users for example machines operators, maintenance engineers, and heads of sectors. Actions on machines include turning on/off the machine(s), monitoring machine(s) performance, modifying machine(s) settings, inserting order specifications (quantity, order deadline, product type, ...), fixing some malfunctions, and other operations.

Only maintenance engineers *Evan* and *Bob* can monitor (read) the log files and fix any disrupt. The team leader *Romy* is able to control the machines by turning them on or off, modify their settings, and insert order(s) details. The machine operators *Eve* and *Mike* can access the machines *M1*, *M2* and *M3* to operate them (on/off) and sort the produced products. Head sectors and maintenance engineers could access all the machines.

To ensure food safety and due to pandemic COVID-19, none of the users would be able to enter the factory before undergoing the necessary tests, and in case of coming back from a travel they must be quarantined for 14 days. Fig. 2 illustrates the AC schema of the use case for the users and their access rights over the machines.

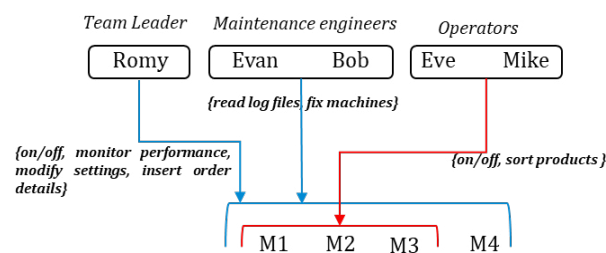


Fig. 2. Use case – AC schema

5.2. Role Based Access Control

In Role Based Access Control model (RBAC) a role can be associated to many subjects. Subjects are given permission to perform some actions based on their roles (e.g. engineers, managers ...). The key components in RBAC are subjects, objects, roles and permissions to perform actions [1, 13, 22]. In Fig. 3, we use our AC metamodel architecture to construct RBAC policy for the use case of *FruitCanning* factory. Since RBAC does not support the notion of attributes, we start our illustration from level 2 where we define the needed RBAC components (classes) and which are extracted from the above use case. We have 3 types of users (machine operators, maintenance engineers, and team leader), various actions (modify, fix, read, ...), and 3 roles (leader, engineer, and operator). At levels 3 and 4 the instances for explicit and implicit components are created, then subjects are assigned

to their roles at level 5. The association of actions to objects (machines) which can be performed by users (subjects) based on their roles are depicted in level 6.

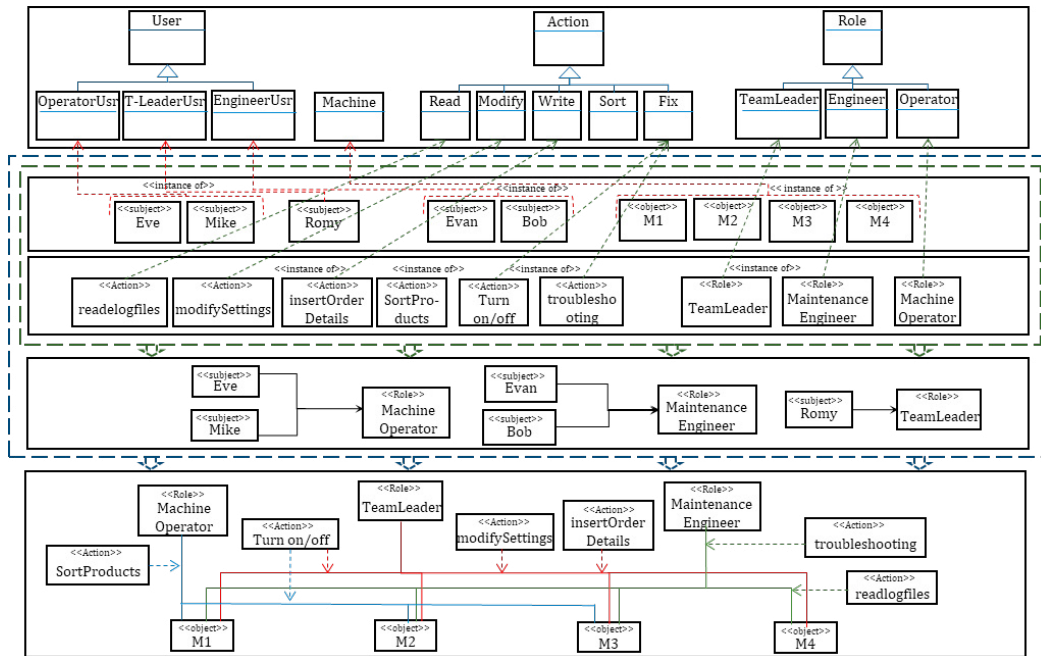


Fig. 3. Use case – RBAC

5.3. Hybrid RBAC/ABAC

Attribute Based Access Control (ABAC) model support dynamic attributes, the key components in ABAC are actions, subject attributes, object attributes, and environmental attributes. Access control decisions are assessed at run time when request is made by subject(s) to perform action(s) on object(s) [1, 22]. Hence, they are allowed/denied based on their attributes, object attributes, environment conditions, and a set of policies that are specified in terms of those attributes and conditions. In this section we illustrate a hybrid RBAC/ABAC model and use the attribute centric approach [4] where a role is defined as an attribute for a user (subject). Note that in our illustration in Fig. 4, we define instances for subjects and objects with the aggregated attributes to represent the idea, since permission evaluation occurred at runtime.

For the above use case, let us say that the maintenance engineer *Evan* has recently arrived from a vacation and his *quarantined duration* is less than 14 days. Also, the machine operator *Eve* after undergoing the COVID-19 examination, in her record the *COVID-19-Test* indicates *TRUE*. Hence, both workers are not allowed to enter the factory before having their *COVID-19-Test* = *FASLE*. Fig. 4 illustrates hybrid RBAC/ABAC model for the use case based on the AC metamodel architecture.

We use the metamodel architecture to construct hybrid RBAC/ABAC policy for the above use case. We start our illustration from level 1 to define all the attributes (for subjects, objects, and environment), in level 2 the needed components (classes) are illustrated with attribute aggregations, note that role is defined as an attribute. At levels 3 and 4 the instances for explicit and implicit components are created, then the supposed permissions for subjects are assigned at level 5. The access rights to objects, and the defined constraints are checked before policy enforcement at level 6.

Other AC methods can be constructed based on this architecture to enforce AC policies in any computing environment. As we can see from Fig. 3 and Fig. 4, we can define all the needed attributes and components no matter what feature(s) the model could have, and we can traverse its layers based on the needed model formulation.

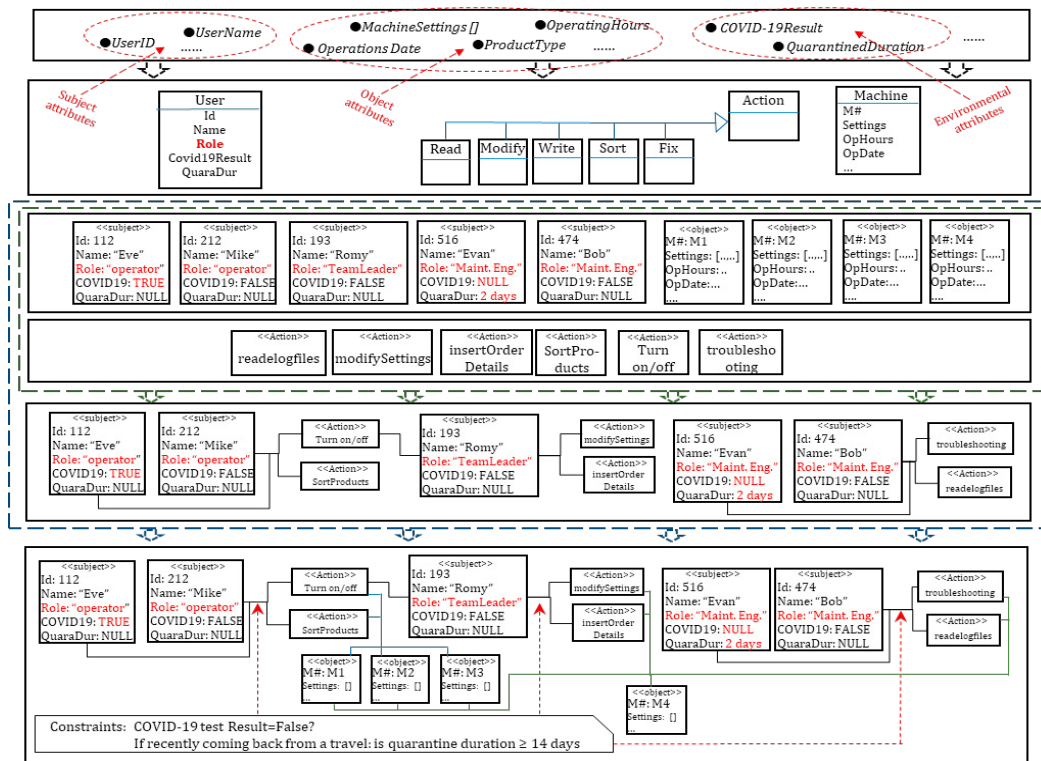


Fig. 4. Use case – hybrid RBAC/ABAC

6. Conclusion and Future Perspective

In this paper we propose a new architecture for an AC metamodel which can be developed to work as a basis to overcome the common limitations of the existing metamodels. We present a food industry use case to show its dynamicity in constructing and deriving AC models and how generic it is. Our metamodel architecture encompasses the base characteristics to follow the continuous technology upgrades since:

- 1) it has a general base concept to derive different AC models,
- 2) new AC components can be defined, in addition to the common ones, and new AC models can be formulated.
- 3) it is dynamic since various types of attributes can be defined and various models can be formulated, for static and dynamic policy enforcements.
- 4) it would facilitate the collaboration process between different AC models which are implemented within different for example IoT layers, since they are derived from the same architecture.
- 5) it is upgradable due to its dynamic architecture in its different levels, and this would allow migrating the AC policies from one model to another.

We believe that the architecture of this AC metamodel introduces a new era of controlling access in the light of new technologies (IoT, cloud computing, social networks, ...) due to the above characteristics which can be manipulated for different AC implementations. As future perspective, we aim to provide a formal representation for the design of our AC metamodel architecture with more detailed explanations and illustrations for its levels with a formal language. Also, show its challenging characteristics in the current computing environments, specially IoT, and provide a detailed industrial use case which tackles industry 4.0.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number 06351], Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT), and Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

References

- [1] N. Kashmar, M. Adda, and M. Atieh, "From Access Control Models to Access Control Metamodels: A Survey," in *Future of Information and Communication Conference*, 2019, pp. 892-911: Springer, 2019.
- [2] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, *Attribute-Based Access Control*. Norwood: Artech House, 2018.
- [3] N. Kashmar, M. Adda, M. Atieh, and H. Ibrahim, "Towards a New Generic and Enhanced Access Control Metamodel: A Complete Introduction, Review and Roadmap," *ACM Transactions on Internet Technology*, no. Special Issue on Human-Centered Security, Privacy, and Trust in the Internet of Things (Submitted), 2020.
- [4] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *Computer*, vol. 43, no. 6, pp. 79-81, 2010.
- [5] Q. M. Rajpoot, C. D. Jensen, and R. Krishnan, "Integrating attributes into role-based access control," in *IFIP Annual Conference on Data and Applications Security and Privacy*, 2015, pp. 242-249: Springer.
- [6] F. Nazerian, H. Motameni, and H. Nematzadeh, "Emergency role-based access control (E-RBAC) and analysis of model specifications with alloy," *Journal of information security and applications*, vol. 45, pp. 131-142, 2019.
- [7] N. Kashmar, M. Adda, M. Atieh, and H. Ibrahim, "A new dynamic smart-AC model methodology to enforce access control policy in IoT layers," in *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, 2019, pp. 21-24: IEEE.
- [8] S. Martínez, J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, and J. Cabot, "Towards an access-control metamodel for web content management systems," in *International Conference on Web Engineering*, 2013, pp. 148-155: Springer.
- [9] C. Bertolissi and M. Fernández, "A metamodel of access control for distributed environments: Applications and properties," *Information and Computation*, vol. 238, pp. 187-207, 2014.
- [10] M. Korman, R. Lagerström, and M. Ekstedt, "Modeling enterprise authorization: a unified metamodel and initial validation," *Complex Systems Informatics and Modeling Quarterly*, no. 7, pp. 1-24, 2016.
- [11] E. Gorshkova, B. Novikov, and M. K. Shukla, "A fine-grained access control model and implementation," in *Proceedings of the 18th International Conference on Computer Systems and Technologies*, 2017, pp. 187-194.
- [12] S. Khamadja, K. Adi, and L. Logrippo, "Designing flexible access control models for the cloud," in *Proceedings of the 6th International Conference on Security of Information and Networks*, 2013, pp. 225-232: ACM.
- [13] N. Kashmar, M. Adda, M. Atieh, and H. Ibrahim, "Smart-AC: A New Framework Concept for Modeling Access Control Policy," *Procedia Computer Science*, vol. 155, pp. 417-424, 2019.
- [14] S. Barker, "The next 700 access control models or a unifying meta-model?," in *Proceedings of the 14th ACM symposium on Access control models and technologies*, 2009, pp. 187-196.
- [15] N. Slimani, H. Khambhammettu, K. Adi, and L. Logrippo, "UACML: Unified access control modeling language," in *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, 2011, pp. 1-8: IEEE.
- [16] T. Xia et al., "Cloud Security and Privacy Metamodel," in *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, 2018, pp. 379-386: SCITEPRESS-Science and Technology Publications, Lda.
- [17] J. Abd-Ali, K. El Guemhioui, and L. Logrippo, "A Metamodel for Hybrid Access Control Policies," *JSW*, vol. 10, no. 7, pp. 784-797, 2015.
- [18] H. Pooda, "Évaluation Et Comparaison Des Modèles de Contrôle D'accès," Université de Sherbrooke, 2016.
- [19] M. A. Abakar, "Etude et mise en oeuvre d'une architecture pour l'authentification et la gestion de documents numériques certifiés: application dans le contexte des services en ligne pour le grand public," Saint-Etienne, 2012.
- [20] H. Qi, X. Di, and J. Li, "Formal definition and analysis of access control model based on role and attribute," *Journal of information security and applications*, vol. 43, pp. 53-60, 2018.
- [21] M. Ennahbaoui and S. Elhajji, "Study of access control models," in *Proceedings of the World Congress on Engineering*, 2013, vol. 2, pp. 3-5.
- [22] N. Kashmar, M. Adda, M. Atieh, and H. Ibrahim, "Access Control in Cybersecurity and Social Media," in *Cybersécurité et médias sociaux: qui sera publié par l'Université d'Ottawa*, 2019.

APPENDIX VII

The 11th International Symposium Frontiers in Ambient and Mobile Systems (FAMS), March 23 -
26, 2021, Warsaw, Poland
Procedia Computer Science, Volume 184, 2021, Pages 887-892,
<https://doi.org/10.1016/j.procs.2021.03.111>



The 11th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS)
March 23 - 26, 2021, Warsaw, Poland

Access Control Metamodel for Policy Specification and Enforcement: From Conception to Formalization

Nadine Kashmar^{a,c,*}, Mehdi Adda^a, Mirna Atieh^b, Hussein Ibrahim^c

^a*Département de mathématiques, informatique et génie, Université du Québec à Rimouski, 300 Allée des Ursulines, QC G5L 3A1, Canada*

^b*Business Computer Department, Faculty of Economic Sciences and Administration, Lebanese University, Hadat, Lebanon*

^c*Institut Technologique de Maintenance Industrielle, 175 Rue de la Vérendrye, Sept-Îles, QC G4R 5B7, Canada*

Abstract

With the widespread of data, applications, and devices in today's dynamic computing environments, controlling access to assets from multiple sources is a key challenge, especially with the presence of cybercriminals and cyberattacks. Several access control (AC) models are developed and implemented in different computing environments to control users' access to resources. But, the emergence of ubiquitous computing, especially the concept of industry 4.0 and IoT applications, releases new prospects to traditional information systems by merging new technologies and services for seamless access to information sources at anytime and anywhere. With this fact, it is realized that these AC models no longer meet the increasing demand for privacy and security standards. Hence, several AC metamodels with higher level of abstraction are developed as unifying frameworks for specifying any AC policy. Unfortunately, the proposed AC metamodels have several limitations. One of these limitations is that they are not generic enough to include all features and the heterogeneous AC models. In this paper we propose a solution for this limitation by developing a generic AC metamodel where its features can be upgraded to answer the needs and facts of the new technologies.

© 2021 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)
Peer-review under responsibility of the Conference Program Chairs.

Keywords: access control; metamodel; security and privacy; IoT; policy;

1. Introduction

Security and privacy requirements increase with the massive presence of new paradigms and technologies. To ensure cybersecurity, various security methods are applied and AC is one of the essential security requirements in this domain. Several AC methods have been implemented to control access to different resources, for example, role-based AC (RBAC) and attribute-based AC (ABAC) models [1, 2]. Information security is often considered to have three essential components; technology, operations and users. Technology is the principal aspect since it encompasses

* Corresponding author. Tel.: +14188338800; fax: +141883311.

E-mail address: nadine.kashmar@uqar.ca

various trends and concerns. Some examples of recent trends that increase the evolution of security threats are the growth in usage of mobile devices, the widespread of IoT concept and its applications and devices, and other trends.

With the current evolution of technology trends it is realized that controlling users' access and the operations they perform on information cannot be overlooked when developing approaches related to information security. As well, it is realized that AC models and hybrid models¹ have reached their limits, and they no longer meet the increasing demand for privacy and security standards with the widespread of devices and resources [1, 3]. This fact force the need develop AC metamodels that serve as unifying frameworks to include multiple AC models to define and enforce larger set of AC policies [3, 4]. Unfortunately, the proposed AC metamodels in the literature lack some key features (section 3) which are essential to follow to continuous technology progressions and upgrades [1, 3, 5].

The remaining of this paper is organized as follows. In section 2, we summarize some of the proposed AC metamodels. In section 3, we present the challenges and the common limitations of the existing metamodels. In section 4, we present our AC metamodel and its elements. In section 5, we present some examples of how AC policies can be illustrated using our metamodel concept. Section 6 concludes this paper with future perspectives.

2. Related Work

Access control metamodels are presented in the literature to concurrently handle multiple AC models. Various AC metamodels are proposed for centralized and distributed computing environments [1, 3, 5]. In this section we present them, and in section 3 we summarize their common limitations in the light of several challenges in this domain.

Barker in [6], proposes a Category Based Access Control (CBAC) metamodel where multiple AC models can be derived as special cases from it. CBAC includes features of mandatory AC (MAC), discretionary AC (DAC), and RBAC models. Since Barker's metamodel lacks the support of resource and action hierarchies, Slimani et al. in [7] extend Barker's metamodel and propose a Unified Access Control Modeling Language (UACML) to provide support for resource and action hierarchies. A CBAC metamodel extension also proposed by Alves et al. in [8] to deal with authorization and obligation assessment, to expand a general notion of obligation for the existing AC models and study the interaction between obligations and permissions. In [9], khamadja et. al present a metamodel approach for cloud computing services called Category Based Access Control (CatBAC) framework, that allows security administrators in a company with various sites to find a concrete model with the constraints and specificities of each site, since the classical AC models (DAC, MAC . . .) are not adequately expressive for highly flexible and dynamic environments. To handle security and privacy in cloud services, Xia et al. [10] propose Cloud Security and Privacy Metamodel (CSPM) which integrates and extends the existing metamodels of cloud security together with newly added concepts. For web services, Martínez et al. in [11] propose a Web Content Management System (WCMS) metamodel inspired from RBAC concept to automatically extract the AC information in the domain of WCMSs. Moreover, Abd-Ali et al. in [12] propose an integration metamodel for hybrid AC policies to concurrently handle multiple AC models where one AC decision as output can be obtained in response to multiple AC decisions as input.

The proposed AC metamodels show the concern of finding AC methods with enhanced features to enforce security policies in different computing environments. This reflects the importance of constructing more robust AC models, especially in the recent years with the presence of heterogenous network technologies and platforms.

3. Challenges and the Common Limitations of the Existing Metamodels

The definition of security policies with the current computing environments, especially IoT, involves complexities and challenges due to 1) the heterogeneity of security plans for information systems such as centralized, decentralized, or both; 2) the diversity of access control rights which might be raised from different information systems such as allow, deny, or undetermined, for different units such as subjects, roles, etc.; 3) the heterogeneity of security policies for different AC models and their extensions; 4) the heterogeneity of AC concepts of various AC models such as objects, subjects, actions, permissions, etc., and 5) the heterogeneity of networks, platforms, devices, etc. Due to these complexities and challenges, several AC metamodels are proposed including features of multiple AC models to

¹ A hybrid model combines features from two or more AC models.

derive various AC models instances. Unfortunately, they have common limitations which cannot be ignored with the emergence of ubiquitous computing and with the deployment of industry 4.0 and IoT applications. In [5], we review and analyze the proposed AC metamodels, they have some common limitations: (1) they are not generic enough and do not to include all features of AC the common models², (2) they are not dynamic and cannot follow technology upgrades, (3) no new attributes/entities can be defined, hence cannot be extended, (4) the derived AC models cannot collaborate within the same computing architecture (e.g. IoT), (5) none of them support the feature of AC policy migration from one AC model to another, and (6) none of them implemented in the context of IoT systems.

4. The Proposed Access Control Metamodel

In this paper we focus on solving the “generic” limitation by finding a common set of AC concepts for the heterogeneous AC models. Having a generic metamodel would allow 1) integrating or including new AC elements for new AC models; 2) building dynamic metamodel with the ability to define new attributes for static/dynamic policy enforcement; 3) the instances of AC models could collaborate and their entities can be interoperable in distributed systems; and 4) facilitate the idea of AC migration from one AC model to another.

4.1. The Roadmap Towards the Formal Representation

Before representing our metamodel we should mention that we have started from scratch, Fig. 1 summarizes the steps we have achieved. The first phase is reviewing and analyzing the proposed metamodels to find out their objectives and limitations [1, 13, 5]. The second phase is defining the scope of the solution and our preliminary idea for solving the limitations [2], building the metamodel concept [4], and its general architecture [3]. The third phase is representing our AC metamodel and the relationships between its components (sections 4.2 and 4.3).

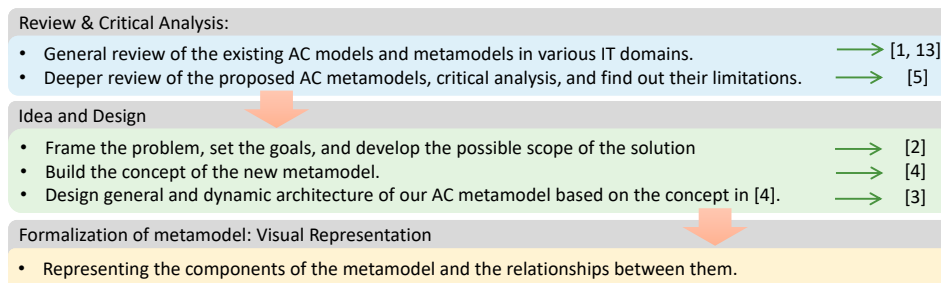


Fig. 1. The steps from review to formal representation

4.2. The Metamodel Elements: common concepts of heterogeneous AC models

Security policies define the principles and guidelines on which access is granted or denied. In general, security policies include common concepts and attributes which are common to all AC models:

- 1- a set of concepts (and attributes) to describe subjects and objects.
- 2- a set of concepts (and attributes) that describe the authorized subjects.
- 3- a set of concepts (and attributes) that explain the different access rights.
- 4- a set of concepts (and attributes) that set various constraints and conditions.
- 5- a set of concepts (and attributes) that describe the context (environmental context) to access objects.

Consequently, the above concepts are illustrated in Fig. 2 and they are interpreted as follows:

- in (1), subjects (e.g. operators, doctors ...) and objects (e.g. machines, patient files ...) are concepts that refer to something real and exists. In our illustration we name them *explicit concepts*.

² The common AC models are: discretionary AC (DAC), mandatory AC (MAC), role-based AC (RBAC), organization-based AC (OrBAC), and attribute-based AC (ABAC)

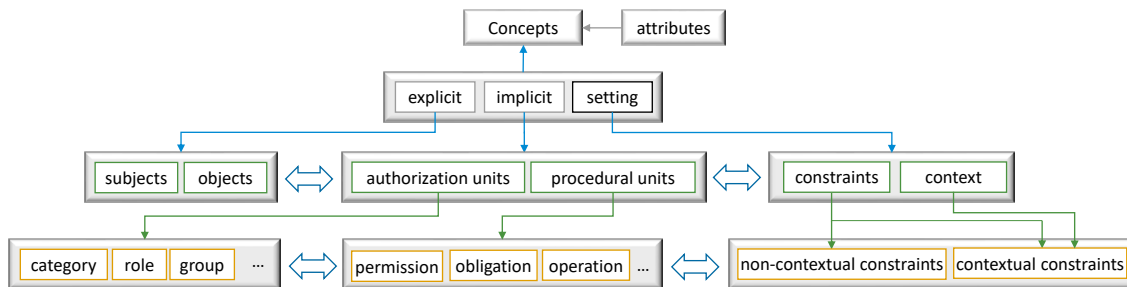


Fig. 2. The elements of our Access Control Metamodel

- in (2), the described concepts, in general, are related to how subjects are classified or assigned (for example, in RBAC subjects are assigned to roles). We name this set of concepts *authorization units*, and they include roles, categories, security levels, etc. Also, in (3) the concepts refer to processes or functions that can be performed by some authorized units on objects. We name this set of concepts *procedural units*, and they include actions, permissions, etc. In our illustration *authorization units* and *procedural units* belong to *implicit concepts*.
- in (4) and (5), the concepts refer to the settings which are planned and determined to have more accurate and regulated access to resources. The setting includes the context and constraints, where context includes contextual constraints, and constraints include contextual and non-contextual constraints.

Note that, the concepts of some AC methods (e.g. ABAC) are built using aggregation operations on the attributes.

4.3. The Formalization of Our Access Control Metamodel

In this section we use the term *entity (or class)* instead of element, and *attribute list (attlist[])* instead of attributes. In Fig. 3 we illustrate our AC metamodel and the relationships between its entities.

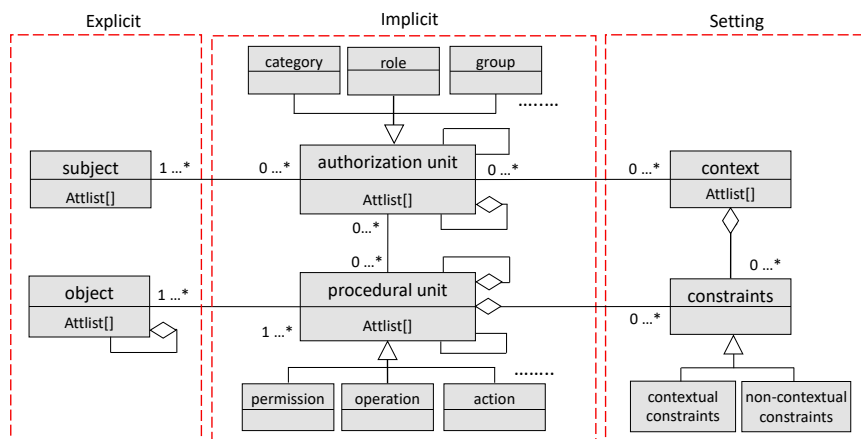


Fig. 3. A visual representation of our Access Control Metamodel

- **Subject (S):** A set of subjects $S = \{s_1, \dots, s_n\}$. The subjects represent entities within a system that are able to initiate access requests. The class *Subject* has an association with the abstract class *Authorization Unit*.
- **Authorization Unit (AU):** A set of authorization units $AU = \{au_1, \dots, au_n\}$. AU class is defined as an abstract class, hence AU must be specialized to create specific authorization units such as roles, categories, security levels, etc. to which subjects can be assigned. AU example:

$$au_1 : role(r_1, \dots, r_n) \Rightarrow au_1 = \{au_{11}, \dots, au_{1n}\}; au_2 : category(c_1, \dots, c_n) \Rightarrow au_2 = \{au_{21}, \dots, au_{2n}\}$$

$$hence, AU = \{au_1 = \{au_{11}, \dots, au_{1n}\}; au_2 = \{au_{21}, \dots, au_{2n}\}; \dots; au_n = \{au_{n1}, \dots, au_{nm}\}\}$$
- **Object (Ob):** A set of objects $Ob = \{ob_1, \dots, ob_n\}$. An object represents the files, devices, etc. The class *Object* has an association with the abstract class *Procedural Unit*.

- **Procedural Units (PU):** A set of procedural units $AU = \{pu_1, \dots, pu_n\}$. PU class is defined as an abstract class, hence PU must be specialized to create specific procedural units such as actions, permissions, etc. to which objects can be assigned. In other words, it represents operations that can be performed on objects. PU example:

$$pu_1 : action(a_1, \dots, a_n) \Rightarrow pu_1 = \{pu_{11}, \dots, pu_{1n}\}; pu_2 : operation(op_1, \dots, op_n) \Rightarrow pu_2 = \{pu_{21}, \dots, pu_{2n}\}$$

$$hence, PU = \{pu_1 = \{pu_{11}, \dots, pu_{1n}\}; pu_2 = \{pu_{21}, \dots, pu_{2n}\}; \dots; pu_n = \{pu_{n1}, \dots, pu_{nm}\}\}$$

- **Constraint (Const):** A set of contextual and/or non-contextual constraints $Const = \{const_1, \dots, const_n\}$.
- **Context (Con):** A set of contexts $Con = \{con_1, \dots, con_n\}$. The context class is to represent the dynamic context information which brings new challenges to AC models and methods. A context expression con_x can be expressed in terms of au_i, pu_j, ob_k , and $const_l$ (contextual conditions). Hence,

$$Con = \{(au_1, pu_1, ob_1, const_1), \dots, (au_m, pu_n, ob_r, const_y)\} \subseteq AU \times PU \times Ob \times Const$$

Note that in some AC models AU might be equivalent to S, for example in DAC model subjects are not assigned to roles, categories, etc. and AU is an empty set.

Furthermore, our metamodel provides support for creating hierarchy for classes of AU (e.g. role hierarchy), PU (e.g. action hierarchy), and object or resource hierarchy by aggregating AUs, PUs, and objects. These hierarchical relationships are depicted by an aggregation association. The association between S and AU, is to assign subjects to roles, groups, categories or other AUs. The association between AU & PU and PU & O, is to represent which AUs are able to perform some PUs (actions, permissions ...) and access some objects. Note that, PUs might have a set of contextual and/or non-contextual constraints before being performed on objects. Our metamodel provides support for formulating AC models and hybrid models for different policies by allowing AUs to be associated with other AUs, As shown in Fig. 3 as a self-association edge with a diamond exists on class AU. However, the rule can be represented as:

$$Rule = \langle AU, Ob, PU, Con \rangle$$

Which means, an authorization unit (or subject) can access object(s) and perform some procedures or actions based on some context and constraints. Note that, in some models we might have an empty set of AU, Con, or Const.

5. Access Control Policy: Examples and Illustrations

- *ABAC policy: In clinics department of a hospital, the doctors Mark and Joe can read/write their patients' prescriptions. The nurse Joyce who has completed 30 hours of training is allowed to read prescriptions which are identified by doctor's Id (dId), a number (pnum), date (pdate), and some details (pdetails). All workers (doctors, nurses ...) in the hospital have their Ids and a record with their related information (name, address...). ABAC allows/denies subject requests based on some attributes of subjects, objects, environment, and a set of constraints [5]. In Fig. 4a we illustrate concrete model as an ABAC policy example based on our AC metamodel. The explicit attributes for subjects are for doctors (id, name, ...), nurses (id, name, training hours, ...), and the patient prescription object (pNum, pDate, ...). The implicit attributes for PUs which represent the actions read and write (actionType). The setting refers to contextual attributes (login-location) and the constraints.*
- *Hybrid MAC/RBAC: In clinics department of a hospital, the doctors Mark and Joe have a clearance level "Top Secret" that is equal to the classification level of the object patient prescription. Hence, doctors are allowed to read/write prescriptions. Also, the nurse Joyce has clearance level "Secret" and can read patient prescription. In Fig. 4b we illustrate concrete model as an example of hybrid MAC³/RBAC policy. In the defined policy, some subjects are assigned to doctor and nurse roles, subjects are permitted to read an object if their clearance level is \leq than the object's classification level, and to write if it is greater than or equal (\geq). Note that, if for example the clearance level for doctor Joe is "secret", then he is only allowed to read patient prescription.*

6. Conclusion and Future Perspectives

In this paper, we propose an AC metamodel that addresses the "generic" limitation of the existing AC metamodels [5]. To the best of our knowledge, this is the first metamodel in the literature that solves the issue of heterogeneity of

³ In our example we use BIBA (developed by Kenneth J. Biba), a MAC variant

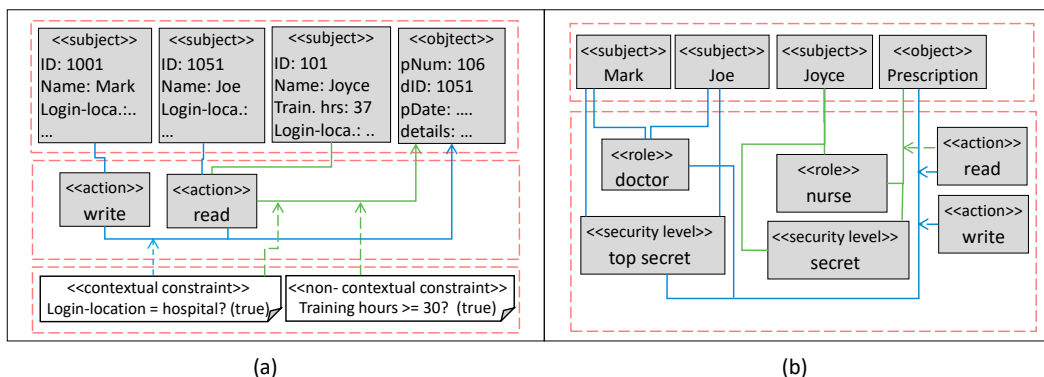


Fig. 4. (a) ABAC policy and (b) Hybrid MAC/RBAC policy examples based on our metamodel

concepts for various AC models. Unifying a common set of AC concepts allows us to develop generic AC metamodel that can be used as a base to solve other existing limitations in this domain. We present the formalization of our metamodel (as visual representation) and illustrate some AC policy examples. As future perspective in this domain, we aim to 1) illustrate more examples to show the hierarchy for classes of AU, PU, and object; 2) define the grammar of our formal language for the metamodel using xtext; 3) develop Domain Specific Language (DSL) for policy instantiation; 4) present a use case and show how all AC models can be derived (also hybrid models); 5) develop more metamodel features and show how the other limitations can be resolved; 6) apply the metamodel to IoT case study.

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number 06351], Fonds Québécois de la Recherche sur la Nature et les Technologies (FRQNT), and Centre d'Entrepreneuriat et de Valorisation des Innovations (CEVI).

References

- [1] N. Kashmar, M. Adda, M. Atieh, From access control models to access control metamodels: A survey, in: Future of Information and Communication Conference, Springer, 2019, pp. 892–911.
- [2] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, A new dynamic smart-ac model methodology to enforce access control policy in iot layers, in: 2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT), IEEE, 2019, pp. 21–24.
- [3] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, Deriving access control models based on generic and dynamic metamodel architecture: Industrial use case, *Procedia Computer Science* 177 (2020) 162–169.
- [4] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, Smart-ac: A new framework concept for modeling access control policy, *Procedia Computer Science* 155 (2019) 417–424.
- [5] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, A review of access control metamodels, *Procedia Computer Science* (2021).
- [6] S. Barker, The next 700 access control models or a unifying meta-model?, in: Proceedings of the 14th ACM symposium on Access control models and technologies, 2009, pp. 187–196.
- [7] N. Slimani, H. Khambhammettu, K. Adi, L. Logrippo, Uacml: Unified access control modeling language, in: 2011 4th IFIP International Conference on New Technologies, Mobility and Security, IEEE, 2011, pp. 1–8.
- [8] S. Alves, A. Degtyarev, M. Fernández, Access control and obligations in the category-based metamodel: a rewrite-based semantics, in: International Symposium on Logic-Based Program Synthesis and Transformation, Springer, 2014, pp. 148–163.
- [9] S. Khamadja, K. Adi, L. Logrippo, Designing flexible access control models for the cloud, in: Proceedings of the 6th International Conference on Security of Information and Networks, 2013, pp. 225–232.
- [10] T. Xia, H. Washizaki, T. Kato, H. Kaiya, S. Ogata, E. B. Fernandez, H. Kanuka, M. Yoshino, D. Yamamoto, T. Okubo, et al., Cloud security and privacy metamodel, in: Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, SCITEPRESS-Science and Technology Publications, Lda, 2018, pp. 379–386.
- [11] S. Martínez, J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, J. Cabot, Towards an access-control metamodel for web content management systems, in: International Conference on Web Engineering, Springer, 2013, pp. 148–155.
- [12] J. Abd-Ali, K. El Guemhioui, L. Logrippo, A metamodel for hybrid access control policies., *JSW* 10 (7) (2015) 784–797.
- [13] N. Kashmar, M. Adda, M. Atieh, H. Ibrahim, Access Control in Cybersecurity and Social Media, Université d'Ottawa, 2021, Ch. 4.

BIBLIOGRAPHY

- Abd-Ali, J., El Guemhioui, K., & Logrippo, L. (2015). A metamodel for hybrid access control policies. *JSW*, *10*(7), 784–797.
- Adda, M., & Aliane, L. (2020). Hobac: fundamentals, principles, and policies. *Journal of Ambient Intelligence and Humanized Computing*, *11*(12), 5927–5941. <https://doi.org/10.1007/s12652-020-02102-y>
- Al Kukhun, D. (2012). *Steps towards adaptive situation and context-aware access: a contribution to the extension of access control mechanisms within pervasive information systems* (Doctoral dissertation). Université de Toulouse, Université Toulouse III-Paul Sabatier.
- Aliane, L., & Adda, M. (2019). Hobac: toward a higher-order attribute-based access control model. *Procedia Computer Science*, *155*, 303–310.
- Alves, S., Degtyarev, A., & Fernández, M. (2014). Access control and obligations in the category-based metamodel: a rewrite-based semantics. *International Symposium on Logic-Based Program Synthesis and Transformation*, 148–163.
- Antunes, M., Maximiano, M., Gomes, R., & Pinto, D. (2021). Information security and cybersecurity management: a case study with smes in portugal. *Journal of Cybersecurity and Privacy*, *1*(2), 219–238.
- Assar, S. (2015). Meta-modeling: concepts, tools and applications. *IEEE RCIS'15: 9th International Conference on Research Challenges in Information Science*.
- Barker, S. (2009). The next 700 access control models or a unifying meta-model? *Proceedings of the 14th ACM symposium on Access control models and technologies*, 187–196.
- Bertino, E., Jabal, A. A., Calo, S., Verma, D., & Williams, C. (2018). The challenge of access control policies quality. *Journal of Data and Information Quality (JDIQ)*, *10*(2), 1–6.
- Bertolissi, C., & Fernández, M. (2014). A metamodel of access control for distributed environments: applications and properties. *Information and Computation*, *238*, 187–207.
- Cruz-Piris, L., Rivera, D., Marsa-Maestre, I., De La Hoz, E., & Velasco, J. R. (2018). Access control mechanism for iot environments based on modelling communication procedures as resources. *Sensors*, *18*(3), 917.
- Ennahbaoui, M., & Elhajji, S. (2013). Study of access control models. *Proceedings of the World Congress on Engineering*, *2*, 3–5.
- Ferraiolo, D., & Atluri, V. (2008). A meta model for access control: why is it needed and is it even possible to achieve? *Proceedings of the 13th ACM symposium on Access control models and technologies*, 153–154.
- Gorshkova, E., Novikov, B., & Shukla, M. K. (2017). A fine-grained access control model and implementation. *Proceedings of the 18th International Conference on Computer Systems and Technologies*, 187–194.

- Hasiba, B. A., Kahloul, L., & Benharzallah, S. (2017). A new hybrid access control model for multi-domain systems. *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, 0766–0771.
- Hu, V. C., Ferraiolo, D. F., Chandramouli, R., & Kuhn, D. R. (2017). *Attribute-based access control*. Artech House.
- Hu, V. C., Kuhn, D. R., Ferraiolo, D. F., & Voas, J. (2015). Attribute-based access control. *Computer*, 48(2), 85–88.
- Jaidi, F., Labbene Ayachi, F., & Bouhoula, A. (2018). A methodology and toolkit for deploying reliable security policies in critical infrastructures. *Security and Communication Networks*, 2018.
- Kaiwen, S., & Lihua, Y. (2014). Attribute-role-based hybrid access control in the internet of things. *Asia-Pacific Web Conference*, 333–343.
- Kashmar, N., Adda, M., & Atieh, M. (2019). From access control models to access control metamodels: a survey. *Future of Information and Communication Conference*, 892–911.
- Kashmar, N., Adda, M., Atieh, M., & Ibrahim, H. (2019a). A new dynamic smart-ac model methodology to enforce access control policy in iot layers. *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, 21–24.
- Kashmar, N., Adda, M., Atieh, M., & Ibrahim, H. (2019b). Smart-ac: a new framework concept for modeling access control policy. *Procedia Computer Science*, 155, 417–424.
- Kashmar, N., Adda, M., Atieh, M., & Ibrahim, H. (2020). Deriving access control models based on generic and dynamic metamodel architecture: industrial use case. *Procedia Computer Science*, 177, 162–169.
- Kashmar, N., Adda, M., Atieh, M., & Ibrahim, H. (2021a). Access control in cybersecurity and social media. *Cybersécurité et médias sociaux*.
- Kashmar, N., Adda, M., Atieh, M., & Ibrahim, H. (2021b). Access control metamodel for policy specification and enforcement: from conception to formalization. *Procedia Computer Science*, 184, 887–892.
- Kashmar, N., Adda, M., Atieh, M., & Ibrahim, H. (2021c). A review of access control metamodels. *Procedia Computer Science*, 184, 445–452.
- Kashmar, N., Adda, M., & Ibrahim, H. (2021a). Access control metamodels: review, critical analysis, and research issues. *Journal of Ubiquitous Systems and Pervasive Networks*, 3(1).
- Kashmar, N., Adda, M., & Ibrahim, H. (2021b). Head metamodel: hierarchical, extensible, advanced, and dynamic access control metamodel for dynamic and heterogeneous structures. *Sensors*, 21(19), 6507.
- Kashmar, N., Adda, M., & Ibrahim, H. (2022a). Head access control metamodel: distinct design, advanced features, and new opportunities. *Journal of Cybersecurity and Privacy*, 2(1), 42–64. <https://doi.org/10.3390/jcp2010004>

- Kashmar, N., Adda, M., & Ibrahim, H. (2022b). Head access control metamodel: distinct design, advanced features, and new opportunities. *Journal of Security and Privacy (in press)*.
- Khamadja, S., Adi, K., & Logrippo, L. (2013). Designing flexible access control models for the cloud. *Proceedings of the 6th International Conference on Security of Information and Networks*, 225–232.
- Kim, S., Kim, D.-K., Lu, L., & Song, E. (2014). Building hybrid access control by configuring rbac and mac features. *Information and Software Technology*, 56(7), 763–792.
- Klarl, H., Molitorisz, K., Emig, C., Klinger, K., & Abeck, S. (2009). Extending role-based access control for business usage. *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, 136–141.
- Korman, M., Lagerström, R., & Ekstedt, M. (2016). Modeling enterprise authorization: a unified metamodel and initial validation. *Complex Systems Informatics and Modeling Quarterly*, (7), 1–24.
- Krehling, L., & Essex, A. (2021). A security and privacy scoring system for contact tracing apps. *Journal of Cybersecurity and Privacy*, 1(4), 597–614. <https://doi.org/10.3390/jcp1040030>
- Layouni, F., & Pollet, Y. (2009). Fi-orbac: a model of access control for federated identity platform. *IADIS International Conference Information Systems Barcelona, Spain*.
- Martinez, S., Cabot, J., Garcia-Alfaro, J., Cuppens, F., & Cuppens-Boulahia, N. (2012). A model-driven approach for the extraction of network access-control policies. *Proceedings of the Workshop on Model-Driven Security*, 1–6.
- Martinez, S., Garcia-Alfaro, J., Cuppens, F., Cuppens-Boulahia, N., & Cabot, J. (2013). Towards an access-control metamodel for web content management systems. *International Conference on Web Engineering*, 148–155.
- Nguyen, P. H., Nain, G., Klein, J., Mouelhi, T., & Le Traon, Y. (2013). Model-driven adaptive delegation. *Proceedings of the 12th annual international conference on Aspect-oriented software development*, 61–72.
- Oh, S. (2007). Permission-centric hybrid access control. *Advances in web and network technologies, and information management* (pp. 694–703). Springer.
- Rajpoot, Q. M., Jensen, C. D., & Krishnan, R. (2015a). Attributes enhanced role-based access control model. *International Conference on Trust and Privacy in Digital Business*, 3–17.
- Rajpoot, Q. M., Jensen, C. D., & Krishnan, R. (2015b). Integrating attributes into role-based access control. *IFIP Annual Conference on Data and Applications Security and Privacy*, 242–249.
- Servos, D., & Osborn, S. L. (2014). Hgabac: towards a formal model of hierarchical attribute-based access control. *International Symposium on Foundations and Practice of Security*, 187–204.
- Slimani, N., Khambhammettu, H., Adi, K., & Logrippo, L. (2011). Uacml: unified access control modeling language. *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, 1–8. <https://doi.org/10.1109/NTMS.2011.5721143>

- Soltani, N., & Jalili, R. (2017). Enforcing access control policies over data stored on untrusted server. *2017 14th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, 119–124.
- Trninić, B., Sladić, G., Milosavljević, G., Milosavljević, B., & Konjović, Z. (2013). Policydsl: towards generic access control management based on a policy metamodel. *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*.
- Xia, T., Washizaki, H., Kato, T., Kaiya, H., Ogata, S., Fernandez, E. B., Kanuka, H., Yoshino, M., Yamamoto, D., Okubo, T., et al. (2018). Cloud security and privacy metamodel. *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, 379–386.