Journal of Cybersecurity and Privacy

MDPI

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

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 exisiting 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.

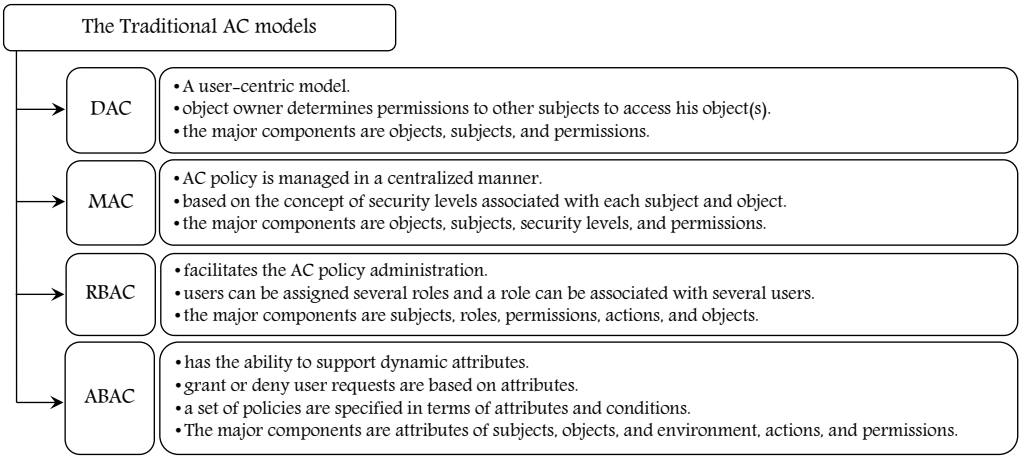

Figure 1. The common AC models.

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.

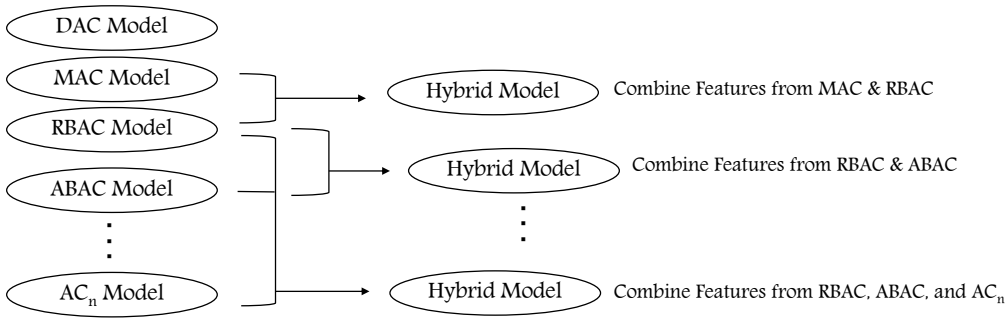

Figure 2. Hybrid AC models.

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.

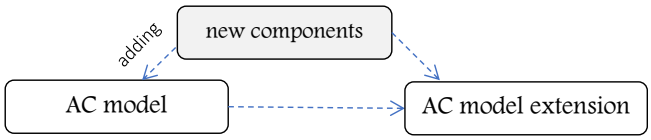

Figure 3. Illustration for AC model extension concept.

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.

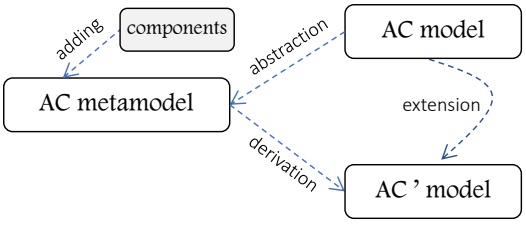

Figure 4. Illustration for model abstraction concept.

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.

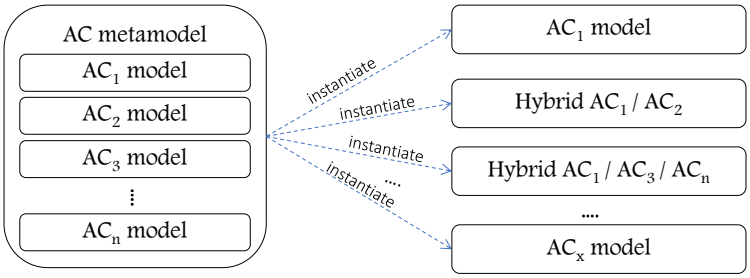

Figure 5. Illustration for AC metamodel concept.

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.

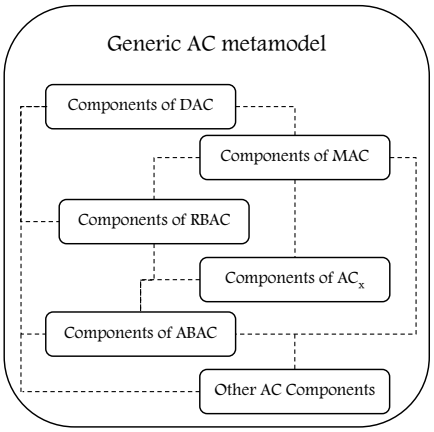

Figure 6. The concept of generic AC metamodel.

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 Trninic 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.

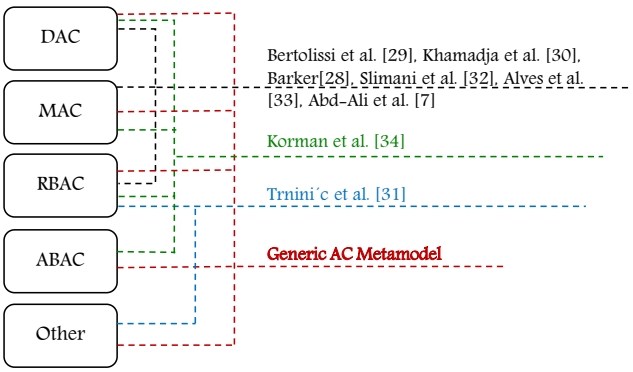

Figure 7. AC features in the core structure for each of the proposed metamodels.

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.

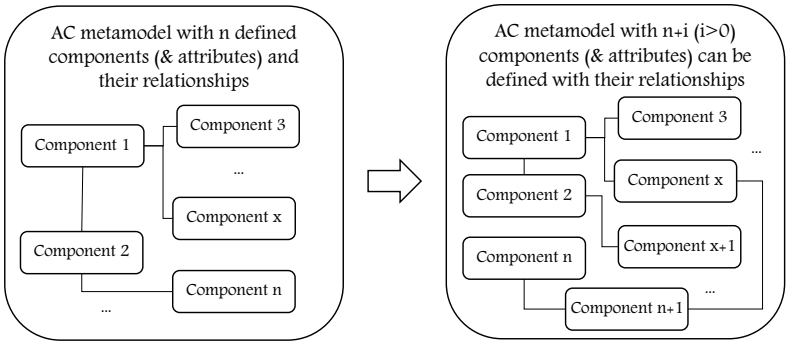

Figure 8. The concept of dynamic AC metamodel.

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.

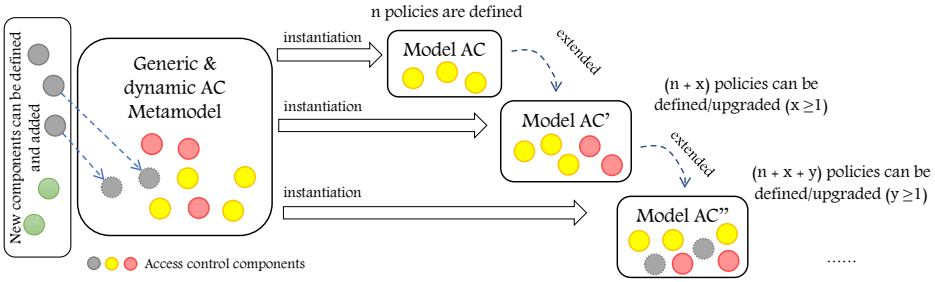

Figure 9. The concept of extensible AC metamodel.

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.

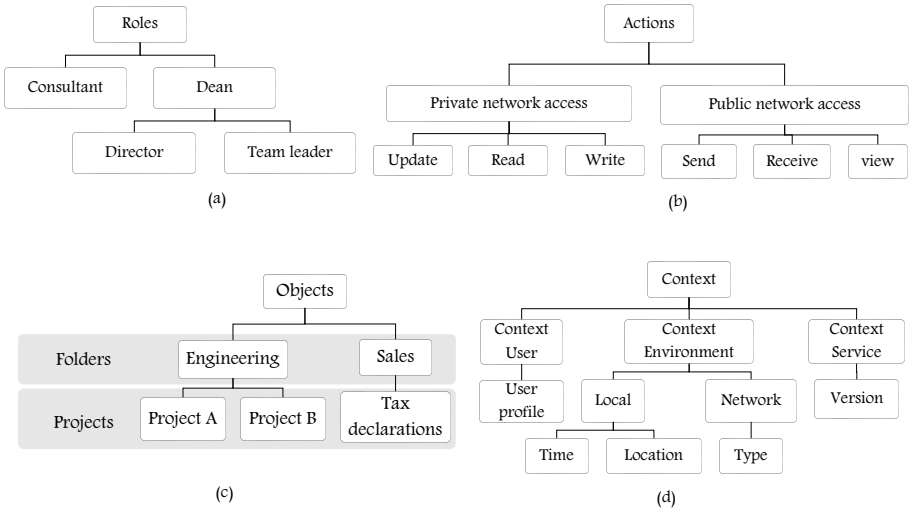

Figure 10. Examples for hierarchy of (**a**) roles, (**b**) actions, (**c**) objects, and (**d**) contexts [3].

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.

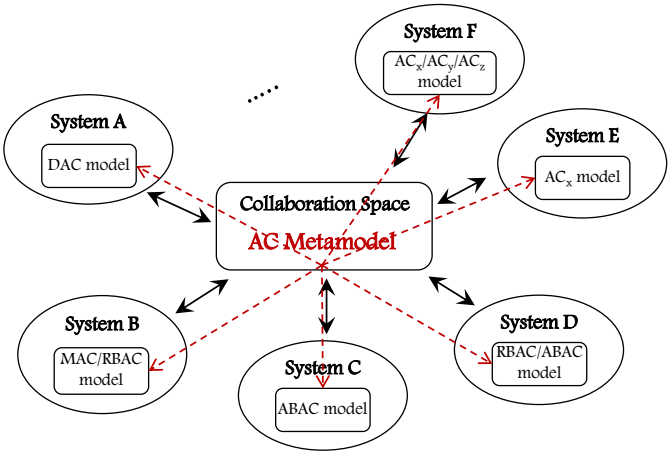

Figure 11. The concept of collaboration and interoperability of AC models.

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.

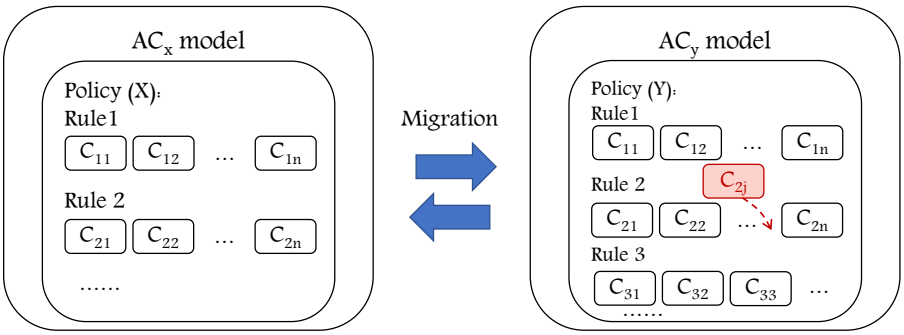

Figure 12. The concept of migration from one AC model to another.

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.

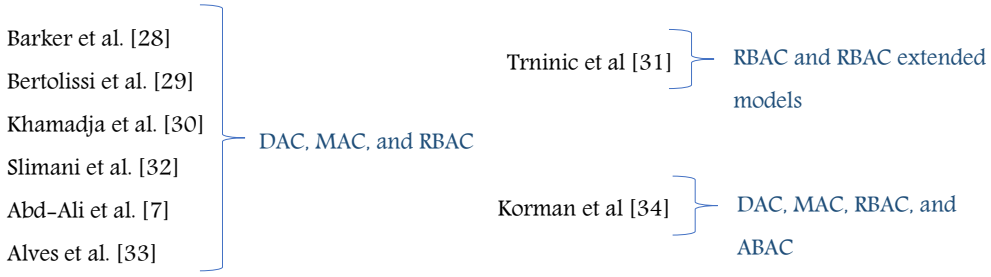

Figure 13. The components that the proposed metamodels include.

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 metamodel 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 metamodel.

5.1. Metamodel 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 metamodel, 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 metamodel, named HEAD metamodel, that addresses the limitations of the existing metamodels. 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 metamodel 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.).

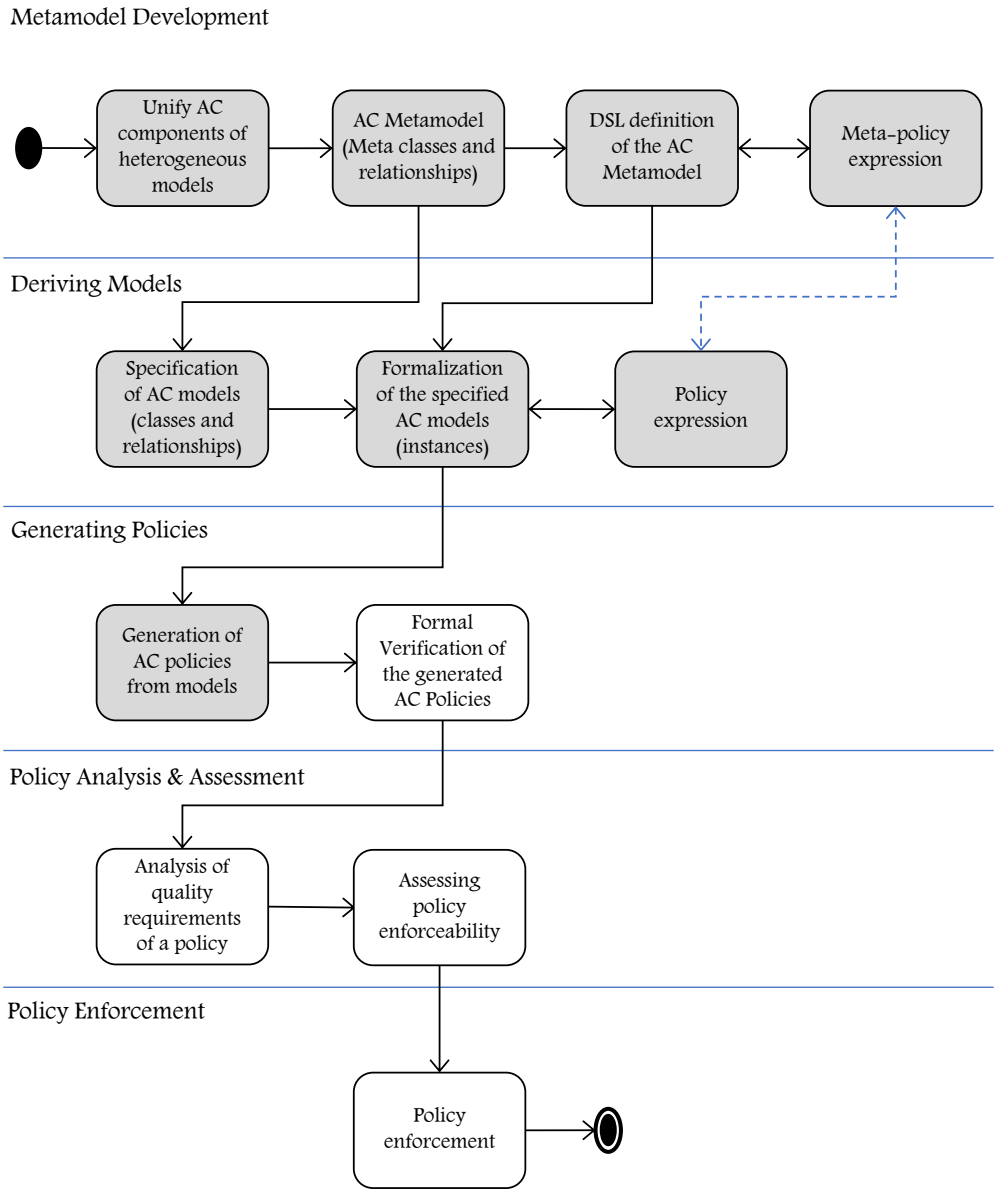

Figure 14. The development approach.

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.

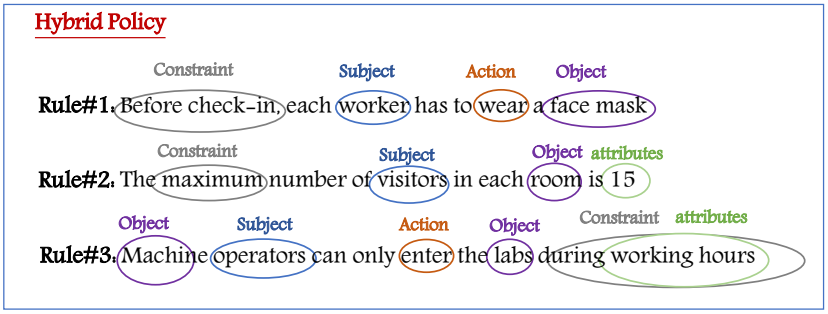

Figure 16. An example of hybrid policy.

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.

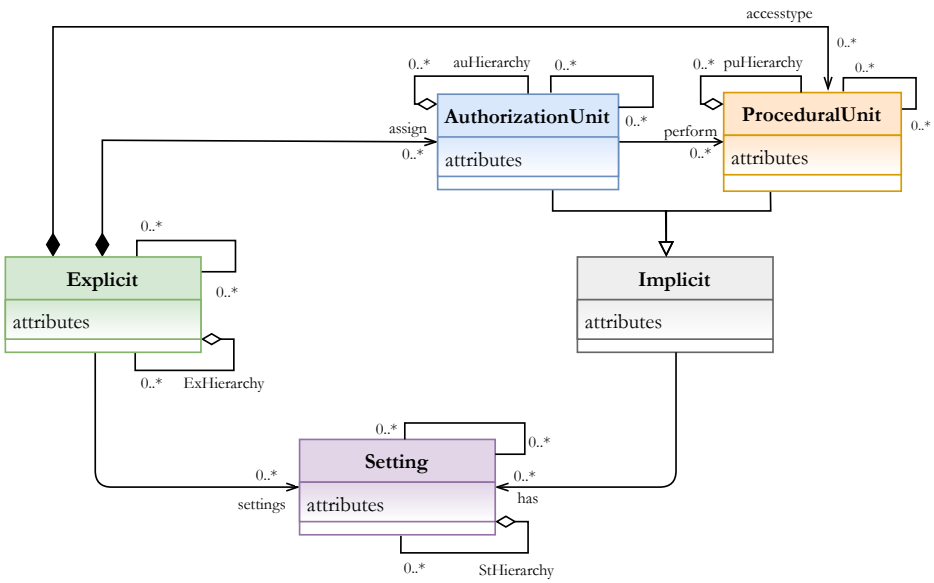

Figure 17. HEAD metamodel [3].

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:

$$Metapolicy = \langle E_x, I_m, S_t \rangle$$

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:

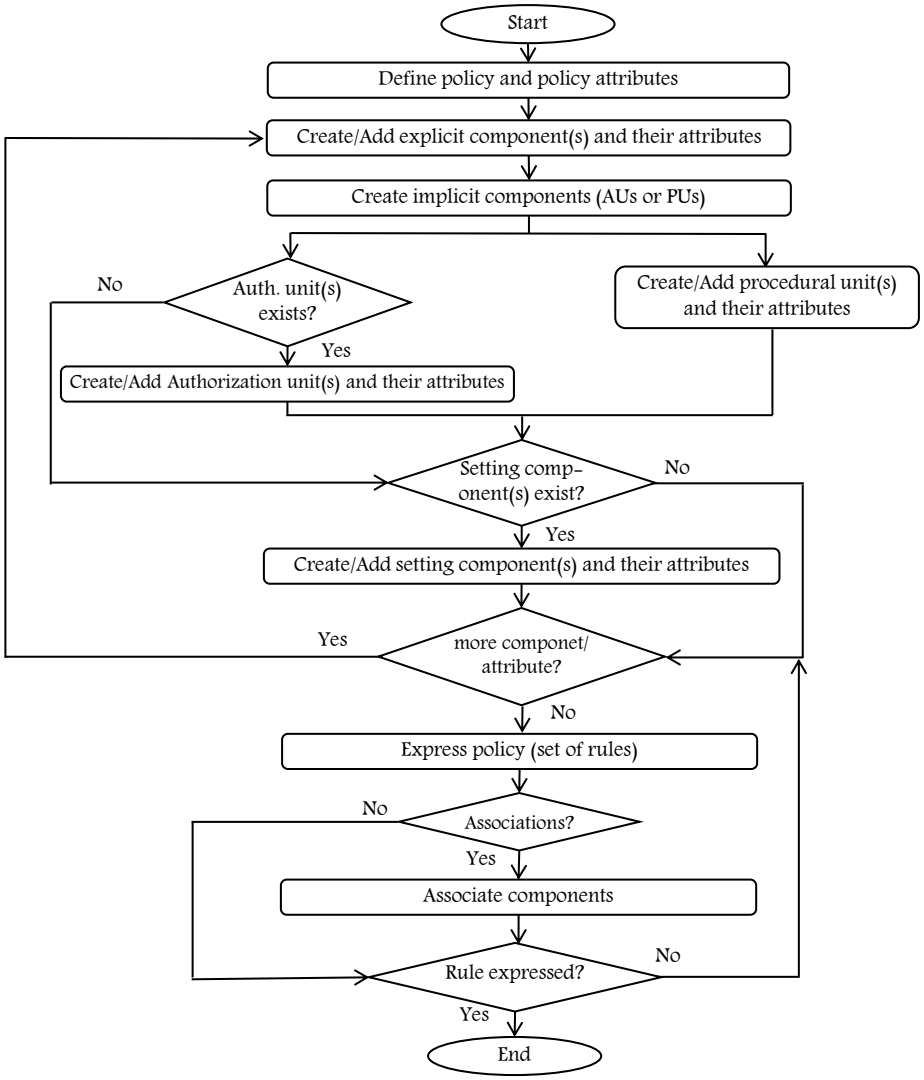

Figure 19. Instantiation of AC models using the DSL of HEAD metamodel.

- 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).

```
policy MAC(descr:String)
    explicit subject(name:String) object(type:String) end
    authorization securitylevel(level:String) end
    procedural operation(op:String) end
end

rule: (ruleid:int){
  MAC.subject(MAC.subject.name)
    [MAC.securitylevel(MAC.securitylevel.level)]{

        MAC.object(MAC.object.type)
            [MAC.securitylevel(MAC.securitylevel.level)]{
                MAC.operation(MAC.operation.op)
        }
    }
}--> 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.

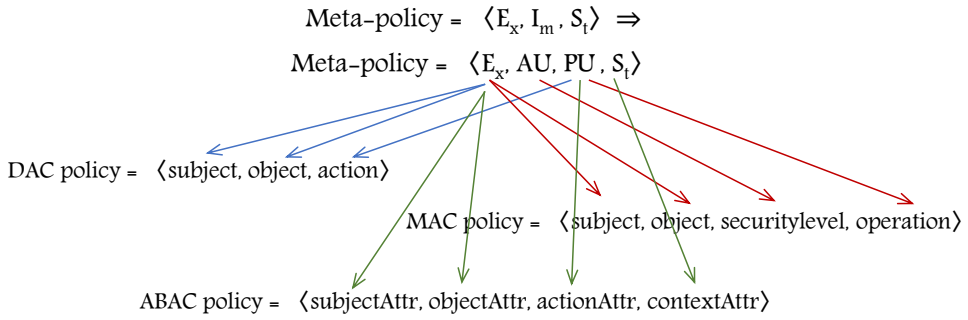

Figure 21. Examples of policy expressions using the meta-policy of HEAD metamodel.

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]