Semantic Web 8 (2017) 1–5
IOS Press

Ranking Semantic Web Authorization Systems

Riste Stojanov [a,*], Slobodanka Stojanova [b], Milos Jovanovik [a,c], Vladimir Zdraveski [a], Dimitar Trajanov [a]

[a] *Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, Macedonia*
[b] *S&T, Skopje, Macedonia*
[c] *OpenLink Software, United Kingdom*

Abstract. The semantic web technologies are widely accepted for annotation and publishing general knowledge and open data that should be accessible for anyone. However, the adoption of the semantic web in the enterprises is not that common, even though these technologies and standards solve one of the most profound challenges: data integration and alignment of multiple systems.

The main obstacle for the enterprise semantic web are the firm security requirements in the Corporations. Even though multiple initiatives work on solving this problem, there is no clear path for deciding which approach or system is most suitable for a given security requirements. This paper proposes an evaluation and context dependent ranking that is based on linear combination of the access control aspects' numerical indicators.

Keywords: Authorization ranking, Authorization aspects, Policy formalization, Access control, Semantic web

1. Introduction

Nowadays, companies use multiple systems for improvement of their internal business processes. Often these systems are implemented with different technologies and use various knowledge representation formats. Integrating these systems is a challenging task. Even though there are formats that are technology agnostic and easy to consume, such as XML and JSON, they fail to provide data abstraction formalism. The semantic web [3] is designed to overcome this problem and provides standards that can define the data formats [1] together with some abstraction definitions, such as what the class hierarchy, the property domains and ranges [2], as well as some functional properties such as which classes are disjoint and whether the property is symmetric [3]. Another common problem is data duplication in the distributed environments. The Linked Open Data initiative is trying to solve this issue and has made significant progress [4]. It provides data mapping and alignment among the integrating systems.

Each of the company's systems is intended to model some business process, which are initiated by authorized subjects that should be previously authenticated. There are multiple solutions for authentication in distributed environments, such as single-sign-on services [2], WebID [43][44] and OAuth [19], and there are frameworks that enable their integration and combination [41].

User's authorization in distributed environments is one of the challenges where improvements are needed. The authorization is usually declared with policies that are enforced by an access control module implementation. Even though there are standards for policy definition, such as XACML [17], most of the real systems have separate policy formats and enforcement modules, where the authentication is their integration point. The separate authorization definition is mainly due to

*Corresponding author. E-mail: riste.stojanov@finki.ukim.mk.
[1] RDF, N3, JsonLD
[2] RDFS
[3] OWL

the lack of integration in multi-domain scenarios. Multiple research groups are targeting this problem, but it is not trivial to determinate the most suitable solution in a given context.

Whenever it comes to deciding which authorization module is most suitable for protection, it all begins with the client requirements, which are usually in natural language form. The security personnel should transform these requirements into policies that will be used by the enforcement module for system protection. The authorization should be implemented within time and budget frame that includes requirement transformation into policies. The time and the budget are often determined by the team that will carry on the task. If the team is familiar with the policy format language, and a number of the policies that should be defined for the requirement is not large, the time and the required budgets are squeezed. In cases when the team does not possess the needed skills for policy management, additional training time is required, thus expanding the budget.

The transformation process is carried out by humans, which makes it error prone. Higher skilled team members will lower the probability for incorrect policies, but they cost more. The cost and the time can be reduced with simpler policy format, which will also lower the error probability. Possible conflicts may arise if the policy format allows different policies to include and exclude same resources. If there is no mechanism for conflict detection and resolution, there is no guarantee that the system is correctly protected.

The authorization enforcement module can also affect the correctness if it is not properly implemented and tested. The authorization enforcement modules introduce additional processing for system protection that influences the overall performances.

2. Related work

There are multiple comparisons among the security systems available in the literature [11,23,26]. [11] provides comparison based on access model, policy language, protection granularity, permission model, context awareness, conflict verification and evaluation aspects. Most of the aspects here are consequence of the policy format and the enforcement module implementation. [26] extends the previous evaluation. The authors here align the analyzed systems and prototypes against aspect that are classified in the following groups:

- Access control model
- Policy specification
  - Granularity
  - Partial Results
  - Underlaying Formalism
  - Reasoning
  - Condition Expressiveness
  - Evidences
  - Interoperability
- Administration
  - Delegation
  - Consistency & Safety
  - Usability
  - Understandability
- Enforcement
  - Negotiation
  - Explanation
  - Conflict Resolution
- Implementation
  - Effectiveness
  - Distributed Use case
  - Flexibility & Extensibility

The policy format definition is generalized in [24]:

$$< Subject, Resource, AccessRight >$$

The $Subject$[4] in the tuple defines the user or the software agent for whom the policy will be activated. The $Resource$ provides information about the data that will be affected by the policy, and the $AccessRight$ defines whether the policy will permit or deny access for the subject to the required resources through some action.

This formalization is not sufficient since it does not model the context in which the policy activation should be decided. Thus, in this paper will be used extended representation of the previous format:

$$< Context, Subject, Resource, AccessRight, Action >$$

The $AccessRight$ from the first formula is split into $Action$ and $AccessRight$ for clarity, and the $Context$ is added. Table 1 present some of the analyzed approaches in respect to the later policy format.

[4]The subject is often referred to as agent or requester.

Table 1
Policy formalization

	Subject	Context	Resources	AccessRight Action
Hollenbach et al.[20]	IRI, Class	/	Document IRI	+ Read, Write, Control
Sacco et al.[35]	SPARQL ASK	/	IRI, triples, graphs, few triple patterns	+ Read, Write, Control
Tonielli et al.[46]	Rules	Dynamic, for policy activation	/	+/- Domain actions
Kagal et al.[22]	IRI	/	/	+/- Domain actions
Muhleisen et al.[30]	IRI	/	triple, IRI, Class	+ Query, Update
Flouris et al.[15]	IRI	/	WHERE expressions	+/- Read
S. Kirrane[24]	IRI, Role	/	triples + static propagation rules	+ Query
Dietzold and Auer[13]	Role	/	SPARQL CONSTRUCT expression	+ Read,Write
Costabello et al.[11]	SPARQL ASK	Static, for policy activation	Resource or Graph IRI	+ CRUD
Franzoni et al.[16]	IRI	Not explained	Basic Graph Pattern with Filter	+ Read, Write
Abel et al.[1]	/	Static	SPARQL CONSTRUCT - like expressions	+/- Read
Chen et al.[7]	Role	/	IRI (Class, Resource, Property)	+/- Read
Oulmakhzoune et al.[31]	/	/	triple patterns	+/- Read
S. Kirrane[25]	IRI, Class	/	quad patterns and filters	+/- All SPARQL constructs

Considering the enforcement, there are tree general approaches:

- **Permitted data filtering/annotation.** This is storage level protection. It is closest to the VBAC [18] access control model, where separate view, most commonly implemented as graph [30,13] or data annotation [16]. The data annotation or filtering is expensive process in terms of processing time and storage, but it allows easy policy testing and design time validation. The policy action in this approach is one of the CRUD[5] or SPARQL constructs[34]
- **Query Rewriting.** Each request query is rewritten, adding the protection elements to it. This is a repository layer approach that eliminates the cost for in-advance data annotation, trading the implementation simplicity. Due to its complexity, this approach requires extensive correctness and performance testing, as discussed in [23]. The policy action in these approaches is some of the SPARQL constructs.
- **Per Action.** Constraining the available domain actions. It is most widely used, but its policies are not able to filter out the forbidden resources. It is a service layer approach, which can function best if it is combined with one of the previous approaches.

The protection aspects are usually consequence from the policy format and the enforcement approach. The most common aspects available in the literature are described below.

2.1. Access Control Model

This aspect is the one against almost every research is aligned. The following access controls can be find in the literature (including their combinations):

- **MAC**: Mandatory Access Control(AC) [38] defines a policy for each combination of subject and resource. The policies are stored and managed by central authorities. The work in [20,22] can be categorized as MAC.
- **DAC**: Discretionary AC [40] extends MAC with policy delegation for distributed environments [25]
- **RBAC**: Role Based AC [39] extends the previous models with introduction fixed user groups called roles [24,13,7,25]. This model simplifies the policy management, since the roles significantly lower the number of the policies.

[5] CRUD

- **ABAC**: Attribute Based AC [33] allows subject grouping based on common attribute values[35, 36] [37,46,11,9,10]. This model provides better flexibility, since the groups are dynamic and solves the dynamic separation of duty problem discussed in [14].
- **VBAC**: View Based AC [18] is most similar with RBAC, only it groups the resources in views. The approaches that enforce the policies with permitted data annotation and filtering are in this category [13,11,30,15].
- **CBAC**: Context Based AC [8] defines that the model provides contextual dependence [45,46] [10,11,1,16].

2.2. Actions

This aspect define which actions are protected by the policy, and in [11] is referred to as Permission Model. The actions in the semantic web systems are first categorized in [20] through the WAC ontology as: Read, Write and Control. These actions, or some of them, are reused in the follow up works [35,15,13,16,1,7,31]. In [30] an *Update* action is used instead of Write, and *Query* instead of Read. The Schi3ld platform [11] extends the protection for the CRUD[6] actions for the linked data platform [42].

The systems that enforce per action protection [45, 22], define policies referring to *Domain Actions*.

In [25] is presented most complete approach, where the actions are the SPARQL constructs: *SELECT, ASK, CONSTRUCT, INSERT, INSERT DATA, DELETE, DELETE DATA, COPY, CLEAR, DROP, ADD* and *MOVE*.

2.3. Access Rights

The access rights define whether the policy *permits* or *denies* access to an action to interacts with resources on behalf of a subject in a given context. When both, permit and deny policies are available, their definition is simpler, but conflicts may arise. If only one option is available, the policies will require the administrator to negate the client requirement in order to define the policy, which has higher error probability.

This aspect is defined as a condition expressiveness in [26], with additional obligation value that requires something to be executed. However, this paper focuses only on access control, and the obligation should be incorporated in the business logic.

[6]CRUD stands for Create Retrieve Update Delete

2.4. Conflicts resolution

When policies with both, permit and deny, access rights are available, conflicts may occur [26,15,11].

There are various ways to solve the conflicts, such as: default behavior [1,15], algorithms [6], meta-policies [22], priorities [27], and detection and prevention [45,46,28,30].

2.5. Granularity

The granularity aspect slightly interferes with the access control models, particularly with RBAC, ABAC and VBAC, since they define the granularity level. Granularity defines the policy's ability to protect certain peace of data. As Table 1 describes in the columns Subject and Resource, there are various ways to declare them. The granularity level is often provided through the mechanism for data or subject selection: resource (IRI), Statement (triple), resources in class, triple or quad pattern , basic graph pattern, filter, graph IRI, dataset URL, SPARQL WHERE expression, and rules.

In [26] the granularity refers only to data selection, while reasoning and underlaying formalism aspects are defined for policy propagation among the subjects. They have also aspect for Partial Results that defines how the data will be filtered. Unlike [26], this paper unifies the subject and resources selection under granularity, and the partial results aspect was already described as type of enforcement.

2.6. Context

The system Proteus [45] is one of the most flexible in terms of context modeling and using in policies. The author define real-life use cases that demonstrate the need for dynamic context definition.

A context is also used in [11,16] and [1], but these systems use predefined ontologies or attributes for context definition, failing to meet the dynamics requirement.

2.7. Correctness

The authorization correctness is only treated in [25] based on the definitions in the earlier work in the Relational Databases field [47]. The authors only test for the query rewriting implementation correctness against a permitted data annotation approach. They do not consider the possible errors in the policy design and definition processes.

2.8. Performance

This is the most common evaluation and comparison aspect. It is highly dependent on the enforcement type and the underlaying implementation. Even though the authors of most approaches provide performance evaluation [1,15,20,11,46,25], each of them uses different data set for this purpose. Besides different datasets, they use different evaluation scenarios (queries). The query execution performance depends on the constructs being used and the underlaying storage technology. This is making the performance comparison task even harder.

3. Policy formalization

In order to be more illustrative in the process of formal definition of authorization policies, we will use an example case. The assumption, in this case, is that the administrator is familiar with the semantic web technologies. He/she would also know the structure and the meaning of the data that should be protected, together with the actions that should be protected (described in subsection 3.2).

3.1. Authorization environment

An authorization system protects its data (Def. 1) that can be used in the requester [7] actions. These actions may not be allowed; thus they are known as Intents (Def. 2). When an intent is sent to an authorization enabled system, it usually provides some evidence [26]. The evidence often includes the requester information, the intended action with its arguments and other time and spatial information. It is also a common practice to include the software agent parameters as evidence, since it submits the intent on behalf of the requester.

Def. 1 ***Protecting data*** $(\mathbb{D})$ *is a set of statements and resources that the authorization system should protect.*

Def. 2 ***Intent*** $(\mathbb{I})$ *is a set of statements and resources that define the requester intent together with its environment.*

[7] The requester is often referred to as agent or subject in the literature.

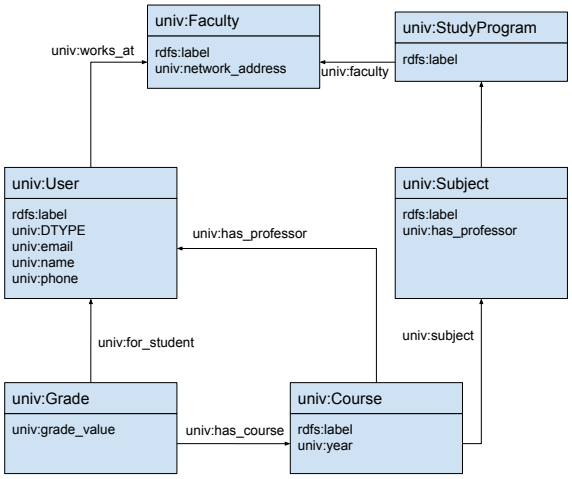

Fig. 1. University ontology

3.2. Example protecting data model

Fig. 3.2 displays the ontology from the University domain, containing relations among the following classes: faculty, study program, subject, course, grade, and user. Let us assume that the instances and the triples from this ontology are stored into multiple datasets and graphs, such that there is separate dataset for each $univ : Faculty$, and a distinct graph for each $univ : StudyProgram$.

The data from these datasets will be accessed by the following groups of requesters:

Anonymous requesters are not associated to any resource from the dataset.
Users are all subjects represented with the class *univ:User*.
Professors are users that are connected with *univ:Course* through the property *univ:has_professor*.
Students are users that are connected with *univ:Course* through the property *univ:for_student*.
Technical Staff are subset of the users with *'TechStaff'* as value of their property *univ:DTYPE*.

These groups are designed to describe the different duties described in [14]. The anonymous group is disjoint from all other groups, providing static separation of duty, while the Users group contains the Professors, Students, and Technical Staff groups, and thus presenting hierarchy. The professors and the students are not explicitly separated, giving a possibility for a subject to be professor and student at the same time, which is possible in real life (post-doctoral students can be professors).

The system that should be protected enables access and management of the previously described data. It provides the following actions:

- SPARQL 1.1 endpoint
- create new faculty
- delete faculty
- rename faculty

3.3. Requirement analysis

Let's consider the following requirement:

Req. 1 *The professors can manage their courses' grades from their faculty's network.*

Req. 2 *The professors can not manage their inactive courses' grades.*

The policy administrator's first task is to understand the requirement. Then he/she should identify the mentioned concepts and relations in the underlaying data and its model. For easier analysis of the Req. 1 and Req. 2 in Fig. 2 is shown a NLP static dependency parse [12] obtained using [29] [8] for a combination of the two sentences. The elements annotated with *NN* or *NNS* POS tags[12] represent common nouns that refer to general concepts. The terms *courses[NNS]*[9], *faculty[NN]* and *grades[NNS]* have corresponding classes in the university ontology that models the protecting data, and their mapping to the resources is a trivial task. Using the requesters' group definitions in §3.2, the administrator can deduce that the term *professors[NNS]* refer to the instances of the class *univ:User* that have a property *univ:has_professor* pointing toward them. The link *network* $\overset{poss}{\longmapsto}$ *faculty* from Fig. 2 defines that the network is possessed by the faculty, and it unambiguously identifies the property *univ:internal_network_address*.

The term *mange[VB]* denotes the intent's action that should be protected. Since the administrator is a human being capable for reasoning, and by assumption is familiar with the system's organization and technologies, he/she can infer that this requirement can only protect the SPARQL endpoint action, and it allows all query forms. Fig. 2 shows that there are tree main dependency branches originating from the action: (1) $\overset{aux}{\longmapsto}$*can*, (2) $\overset{nsubj}{\longmapsto}$*The professors*, (3) $\overset{dobj}{\longmapsto}$*their courses' grades*, and (4) $\overset{prep}{\longmapsto}$*from their faculty's network*. The branch (1) provides that the manage action can be executed over the permitted data; (2) defines a condition that should be satisfied by the requester; (3) defines which resources and triples are permitted [10], and (4) defines the intent's agent address ie. the context in which the policy will be valid.

The term *their[PRP$]* also should be taken into account in the policy design. It is annotated with the POS tag *PRP$* that describes that it is a possessive pronoun. In his requirement, this means that the *courses[NNS]* and the *faculty[NN]* are possessed by some *professors[NNS]*, which is not modeled in the data. However, since there is only one relation between the *univ:User* and the other two classes, it is most likely that the possessiveness refers to the *univ:works_at* for the *univ:Faculty*, and to the *univ:has_professor* for the *univ:Course* possessiveness.

In Req. 2 is almost identical with Req. 1. The difference is that it does not impose condition about the intent's address, but it introduces the adjective *inactive*. This term from this requirement describes a property of the courses that is not present in the data definition model. In order to model the policy for the Req. 2, the administrator must be familiar with, or otherwise to obtain the definition for *active course*. Let assume that he/she already knows that *the course is active its univ:year's 01 September until the next year's 01 September*. This shows that the new relation depends on the *intent's invocation moment* i.e.,. there is a link between $\mathbb{D}$ and $\mathbb{I}$ that should be modeled in the policy in order to define which data is modeled by the requirement.

When Req. 1 and Req. 2 are aligned against each other, there is obvious conflict between them. Req. 2 is valid for the professors no matter where they are, denying access to their inactive courses' grades. On the other hand, Req. 1 is a bit more restrictive when it comes to its activation, and it is valid only when the professors access from their faculty, but it is more general with respect to the protected data, meaning that it protect all requester's courses. The conflict here is that Req. 2 forbids management of inactive courses, while Req. 1 allows the action only when it is requested from the faculty. The challenge imposed here is how the administrator can detect this kind of conflicts and how to resolve them.

[8] http://demo.ark.cs.cmu.edu/parse

[9] The notation term[POS tag] will be used from now on when referring terms from the dependency parse

[10] If the policy was denying the action, then these resources and triples were going to be prohibited

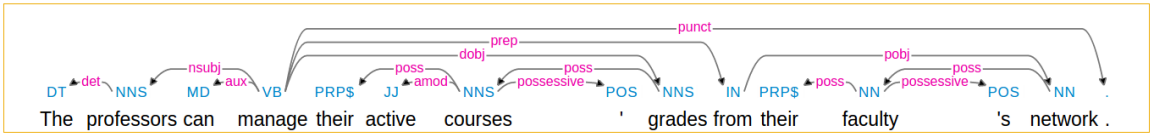

Fig. 2. Requirement dependency parse

3.4. Formal definitions

For complex domain models, it is not possible to define all the authorization rules in one statement (one requirement). Thus, the authorization systems have to support modeling of multiple requirements that define the protection for various scenarios and data parts. This implies that the authorization system should operate using multiple policies that can be activated depending on the scenario, which is represented by the incoming intent and the system's overall state. The activated policies should define the enforcement method that describes whether the action or partial data is permitted or prohibited. The policies should also define their target i.e. the data they protect. Additionally, when there are multiple activated policies for the same scenario, there should be a way to combine them.

The administrator should enable the policy format to model an arbitrary authorization request. Considering the discussion above and the examples from Section 3, there are four generalized parts that the policy should model, as defined in Def. 3.

Def. 3 ***Policy*** *is a tuple of* $\langle \alpha, \varphi, \epsilon, \rho \rangle$ *that defines the condition* $\alpha(\mathbb{I} \cup \mathbb{D})$ *that an intent* $\mathbb{I}$ *should satisfy in relation to the protected data* $\mathbb{D}$*, so that interaction* $\epsilon \in \{reject, allow, include, exclude\}$ *will be enforced with the result data* $\mathbb{R} = \varphi(\mathbb{I} \cup \mathbb{D})$*. The element* ρ *is a comparator that is used for conflict resolution,* α *stands for policy activation condition and* φ *represents a partial data filtering function.*

3.5. Policy activation

The first part is the activation condition α. In the best case scenario, this condition should be executed against all available facts at the execution moment $(\mathbb{I} \cup \mathbb{D})$. The enforcement process executes α first in order to determine whether the policy should be considered in the next enforcement phases. The activation function is a required condition provided as a natural language rule combination. Thus, it can be formally observed as a set of rules combined with a boolean expression using the operators $\neg, \vee, \wedge$. Each rule may contain variables, the existence quantifier $(\exists)$, and predicates. For convenience, this paper uses the Semantic Web Rule Language (SWRL) [21] predicates. The predicates in the form of: *predicate(subject,object)*, where *predicate* denotes a concrete property that connects the subject and object resources passed as arguments. The *subject* and *object* parts can be described with any value blank element *_(lowerdash)*, an IRI that represents concrete resource and variables starting with question mark *(?)*. The build-in functions will be used in the form *buildin(fn, r1,..rn)*, where *fn* is the name of the build-in function, and the *r1,...rn* are the resources or variables used as its arguments. The Ex. 1 shows the definition of the activation function α for the Req. 1. The *belongs_in* predicate is used for the compact representation of IP address network membership check, as a build-in function.

Ex. 1

$$
\begin{aligned}
\alpha(\mathbb{I} \cup \mathbb{D}) \leftarrow & \exists \langle ?r, ?ip, ?c, ?f, ?na \rangle, \\
& ?r, ?ip \in \mathbb{I} \\
& \wedge ?c, ?f, ?na \in \mathbb{D} \\
& \wedge requester(\_, ?r) \\
& \wedge requested\_from\_ip(\_, ?ip) \\
& \wedge has\_professor(?c, ?r) \\
& \wedge works\_at(?r, ?f) \\
& \wedge network\_address(?f, ?ip) \\
& \wedge buildin(belongs\_in, ?ip, ?na)
\end{aligned}
$$

3.6. Partial data filtering

The partial data filtering function φ is executed over the protecting data $\mathbb{D}$ combined with the intent's data $\mathbb{I}$ when possible. The requirements from Section 3 were additionally composed of rules that should be satisfied by the resources and their relationships in order to be protected by the policy. Thus, φ can be represented in a similar way to the activation function, with a predicate

combination. The difference from α's representation is that φ infers new triples, instead of testing variable mapping existence.

Ex. 2

$$\begin{aligned}\varphi(\mathbb{I} \cup \mathbb{D}) \equiv \\ & ?r, ?time \in \mathbb{I} \wedge ?g, ?p, ?o, ?c, ?y \in \mathbb{D} \\ & \wedge requester(\_, ?r) \\ & \wedge invocation\_time(\_, ?time) \\ & \wedge has\_professor(?c, ?r) \\ & \wedge has\_course(?g, ?c) \\ & \wedge year(?c, ?y) \\ & \wedge ?from \leftarrow buildin(date, ?y, 9, 1) \\ & \wedge ?to \leftarrow buildin(date, ?y + 1, 9, 1) \\ & \wedge (\neg buildin(after, ?time, ?from) \\ & \vee \neg buildin(before, ?time, ?to)) \\ & \Rightarrow triple(?g, ?p, ?o)\end{aligned}$$

Ex. 2 shows the representation of the partial data filter function for Req. 2. The construct $triple(?g, ?p, ?o)$ is used to select statements that have a variable as a predicate, while the implication is used to denote that when the precondition from its left side is true, the consequence on the right side selects the data that will be protected by the policy.

3.7. Policy enforcement methods

The policy enforcement methods define how will the system process the policy when it is activated. According to the literature analysis presented in Secition 2 the following enforcement methods can be identified: (1) allowing or denying actions, (2) include or exclude parts of data and (3) positive and negative obligations. The obligations are treated as a part of the business logic in this paper, and will not be considered in the authorization process.

The Def. 3 has identified the following possible values $\{reject, allow, include, exclude\}$, from now on referred to as $\{\epsilon_\times, \epsilon_\checkmark, \epsilon_+, \epsilon_-\}$. When a policy with $\epsilon_\times$ enforcement method is activated, the system rejects the intent as a whole, and the intent's processing stops at the same point. This enforcement method has empty partial data filter function φ, since it is never invoked. When the enforcement method is $\epsilon_\checkmark$, then the action is allowed without additional partial data filtering.

ϵ_+ and ϵ_- enforcement methods are used to permit or prohibit interactions with the parts of the protection data $\mathbb{D}$ selected with the partial data filtering function φ.

3.8. Policy combination

The policy combination is important for two main reasons: (1) breaking down a complex authorization into simpler rules and (2) conflict resolution.

The policy combination can be observed from two different perspectives: set operations and logical operations. The first perspective is discussed in [15], and it suggests that the policies[11] should be combined with $\bigcup \varphi^+ \setminus \bigcup \varphi^-$, that is same as $\bigvee \varphi^+ \wedge \neg \bigvee \varphi^-$, since there is bijection equivalence mapping among them.

The previous combination method is not suitable for conflict resolution, since the excluding policies have a higher priority, and the administrators do not have a control of this process. As analyzed in Section 2, there are three most common approaches for a conflict resolution: meta-policies, priority, and harmonization. The last approach does not resolve the conflicts, but provides tools for their detection and requires disjoint policies. The meta-policies are rules that define the result in a conflict situation and have a higher priority than the other, which makes this approach part of the paradigm for a conflict resolution with priorities.

The priority conflict resolution defines a policy ordering, such that one policy's data protection is more important than another. This way is the most flexible because it is similar to the way the requirement defines the conflict resolution and it is easier to say which rule is more important. Even when a new rule is introduced to resolve the conflict, that rule has a higher priority than the other. This is the cause for the combination comparator ρ in Def. 3, which defines the priority of the policy. Def. 4 defines an operator for combining an ordered set of policies.

Def. 4 ***Policy result combination*** *is a non-commutative operator $\odot$ predicate operator such that:*

$$\langle \epsilon_1, \varphi_1 \rangle \odot \langle \epsilon_2, \varphi_2 \rangle = \begin{cases} \langle \epsilon_1, \varphi_1 \vee \varphi_2 \rangle, \epsilon_1 = \epsilon_2 \\ \langle \epsilon_1, \varphi_1 \wedge \neg \varphi_2 \rangle, \epsilon_1 \neq \epsilon_2 \end{cases}$$

[11] In this description the partial data filter function φ has superscript + or - if it is part of a policy with enforcement method ϵ_+ and ϵ_-, correspondingly.

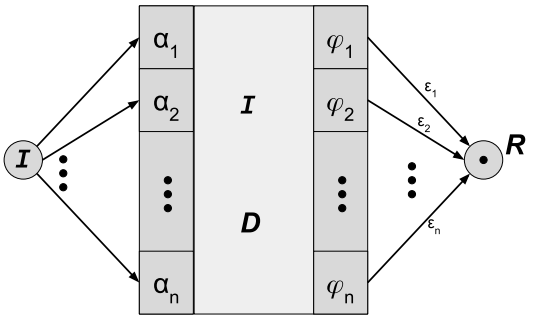

Fig. 3. Enforcement process

Ex. 3 defines the different output for different policy priority assignments. Note that the function φ is executed over a set of statements and returns a filtered set of statements. Thus the corresponding boolean expressions from Def. 4 could be described as union $\cup$ and set minus $\setminus$ relations over the resulting sets from the φ results.

Ex. 3 *There are policies that include* Φ_1, Φ_2, *and exclude* Φ_3 *part of the data, so that* $\Phi_\cap = \Phi_1 \cap \Phi_2 \cap \Phi_3$ *and* $\Phi_\cap \neq \emptyset$. *The result sets* $\Phi_i = \varphi_i(\mathbb{D} \cup \mathbb{I}), i \in [1,3]$ *are in conflict* ($\Phi_\cap \neq \emptyset$) *and the administrator can choose one of their 6 possible orderings. Here are three example resolution results:*

$$\rho_1 < \rho_2 < \rho_3 \Rightarrow \langle \epsilon_+, (\Phi_1 \cup \Phi_2) \setminus \Phi_3 \rangle$$
$$\rho_1 < \rho_3 < \rho_2 \Rightarrow \langle \epsilon_+, (\Phi_1 \cup \Phi_3) \setminus \Phi_2 \rangle$$
$$\rho_3 < \rho_2 < \rho_1 \Rightarrow \langle \epsilon_-, (\Phi_3 \setminus \Phi_2) \setminus \Phi_1 \rangle$$

3.9. Enforcement process

In practice, everything starts with a requester intent, that is sent to the system with implemented authorization enforcement. The requester can choose to provide multiple pieces of evidence in the intent, among which can be its identity, additional attributes about him/her, the environment in which it operates, the action he/she intends to invoke, and some additional action parameters.

The authorization system is the component that intercepts the intent and decides whether the requested action will be allowed or denied, and in the first case filters the data available in action. This generalized authorization system is displayed in Fig. 3. The figure shows that the intent pass through the activation function ($\alpha_i, i \in [1,n] \wedge n = |\mathbb{P}|$) of each policy from $\mathbb{P}$. Then, if $\epsilon = reject$, the action is rejected, otherwise the partial result filter ($\varphi_j, j \in [1,k] \wedge k \leq n$) is invoked, where k is the number of the activated policies for the scenario. The last part shows that the results from each activated policy are combined using the $\odot$ operator. The indexes of the policies are sorted according to the comparator ρ, used for a conflict resolution.

4. Access control aspects

The comparison aspects provided in §2 are mainly defined by the policy format and enforcement implementation. However, the analysis and the comparison available in the literature rarely connect them to the time and budget for requirements transformation into policies. Instead, most of the approaches provide characteristic values for each aspect that can not be intuitively compared. For example, Table 1 provide the values of each analyzed aspect, but it is hard to compare the systems based on this information. The same holds with the categorization presented in [26]. The comparison tables are a convenient way to eliminate the systems that do not fit the requirement, but when multiple candidates remain, it is hard to rank them based on this information.

The client authorization requirements transformation into policies is a manual process. The policy format and language directly influence the required skills and time for their management.

The client requirement defines when (*Context*) and how (*Action*) can somebody (*Requester*) interact with what resources (*Resource*). As discussed in §3.4, the resources are part of the protection data $\mathbb{D}$, that lies hidden behind the authorization. The *Context*, the *Requester* and the *Action* that should be protected are part of the intent $\mathbb{I}$, that is dynamic and different every time. In many cases, the requester is represented by a resource, and they are somehow linked to the context and other resources (as shown in Req. 1). Req. 2 defines a scenario where there is a need to connect the protection data with the context. The protection data $\mathbb{D}$ and the intent $\mathbb{I}$ are composed of a set of statements (or can be simply transformed into this form) that form a graph. From hereafter, the union from these two linked graphs will be referred to as *ProtectionContext*[12].

Every authorization system relies on some policies that model the requirements. The policy model constrains the authorization process flexibility, imposing

[12]The name is inspired from the work in [45]

limits of the possible requirement that can be protected. The policy language and formalism in the system require corresponding skills from the administrators. The model and the available tools available define the speed of requirement transformation into policies.

As discussed in the introduction of the paper, finding a suitable authentication system in a given context is not a trivial task. The reviewers should peak the system that best matches their team skills, and uses policy model that can represent all client requirements. The system should also provide fast enough requirement transformation into policy, and a way to guaranty the policy correctness. At the end, the authorization performance trade-off should be acceptable. All these aspects will be discussed in more details in the rest of this section.

4.1. Flexibility aspect

Def. 5 ***Flexibility*** *is the ability to transform an arbitrary requirement into policies.*

Since Def. 3 models the requirement transformation into policy, an authorization system flexibility from Def. 5 can be directly correlated with this definition so that it will represent which parts from Def. 3 are covered by the system's policies.

4.1.1. Rule expressiveness

The rules defined for policy activation (α) and partial data filtering (φ) are composed of multiple predicates because they are modeling the requirement's statements. As shown in Ex. 1 and Ex. 2, these predicates can be connected with $\wedge, \vee, \neg$ operators[13]. There is also a membership $\in$ introduced in those rules because the requester is always obtained from the intent's statements graph. A policy's model flexibility level depends on its ability to represent all these predicates and combination operators in the enforcement system.

The rule expressiveness should be considered separately for the activation and partial data filtering, since the authorization systems may use different ways to define them. The PPO ontology [35] and the Schi3ld platform [11] are such examples, where they both use SPARQL ASK queries for policy activation, but the partial resource protection is either with direct resource linking using IRI or with triple patterns.

The membership operator is usually incorporated in the system implementation, where the activation function can be executed using the intent as an argument ($\alpha(\mathbb{I})$), but when it comes to select, which graph or data set is allowed few approaches provide this flexibility. The dataset membership is implemented in [20] through access control lists, while graph membership is discussed in [24,25] through graph patterns, and SPARQL queries in [13,11].

The authorization systems are most diverse when it comes to the allowed predicates. The main goal of the predicates is to define a rule about the statements, which are composed of subject, property[14] and object resources. Some system allow arbitrary predicates as part of the SPARQL queries [11,13,35], some variation of SWRL rules [1,13,30] or other form of triple selection [15,16,24,25,31,32]. The approaches [7,11,20] use few fixed predicates by class or exact IRI, thus providing less flexibility for requirement transformation. The support for statement filter using these predicates is shown in Table 2, under the columns *TP*, for the activation and partial data filtering functions separately.

There are rules that filter out some of the primitive properties of the resources, such as the *_in_network* predicate from Ex. 1. The approaches that are based on SPARQL and SWRL support their built-in predicates and may provide extensions depending on the standard's implementation. The support for this type of filters is shown in Table 2 in the columns *PF*. In these columns, the support for filtering is presented, while no further analysis about the particular supported filtering functions is made.

The combination with all possible operators is supported only in the approaches that are based on a SPARQL or SWRL variant. There are approaches that support only $\wedge$ operator [15,16], or primitive filter combination [31]. There are also approaches that does not support rule combination at a policy level [7,20,25, 35].

Since the activation and partial data filtering condition are expressed with natural language, that can closely be modeled with a first order predicate logic, and it imposes infinite space of rules. The authorization system can choose to model part of this space, but it is impossible to define to which extent they can protect the requirements. Table 2 shows the extent of expressiveness in the reviewed approaches on the activation function α and partial data filter function φ. The supported expressive elements are marked with *y*.

[13]The implication operator $\Rightarrow$ is omitted since it can be represent as: $a \Rightarrow b \equiv \neg a \vee b$.

[14]The term property is used to avoid ambiguity with the logical predicates mentioned earlier.

Table 2
Activation and partial data filtering rule expressiveness

	Intent evidences and linking				Activation - α					Partial Data Filtering - φ					ϵ
	C	CR	CD	RD	TP	PF	C	Gm	DSm	TP	PF	C	Gm	DSm	
Hollenbach et al. [20]					c,i									y	+
Sacco et al. [37]					y	y	y	y		y			y		±
Toninelli et al. [46]	α,φ	α,φ	α,φ	α,φ	y	y	y			y	y	y			$\times,\pm$
Muhleisen et al. [30]				α,φ	y	y	y			y					+
Flouris et al. [15]										y	y	p			±
Kirrane [24]					r,i					y			y		±
Dietzold & Auer [13]				α,φ	y	y	y			y	y	y	y		+
Costabello et al. [11]	α	α								i			y		±
Franzoni et al. [16]					i					y	y	p			±
Abel et al. [1]	α,φ	α,φ	α,φ	α,φ	y	y	y			y	y	y			±
Chen&Stuckenschmidt [7]					r					p					±
Oulmakhzoune et al. [31]					i					y	y				±
Kirrane [25]					y	y	y	y		y			y		±
Padia et al. [32]				φ	y	y	y			y					±

C - context evidence; CR - context requester link; CD - context data link; RD - requester data link; TP - triple pattern; PF - primitive filter; Gm - graph membership; DSm - dataset membership; y-yes; i-resource IRI; c-Class; r-Role; p-partially; &-conjunction only

4.1.2. Intent evidences and linking

There are three essential pieces of evidence that the intent should provide: the action with its attributes, the requester, and the requester's context. If these attributes are available, the question is whether the policies can model these data.

Most of the systems usually provide the intended action and the requester is through its IRI, Class or Role [7,16,20,24,25], while the authors of [11,37] are using WebID [43] for adding additional requester's attributes into the intent. Additional contextual attributes are provided in [1,11,46].

Another issue is whether the intent's evidence can be connected with the protection data. Linking of all intent's evidence with the protection data in both, activation and partial data filter rules is provided only in [1,46], while in [13] this is achieved without the presence of context information.

The evidence and the linking of the intent with the underlying data can be differently utilized in the activation and data filtering functions. Table 2 shows whether the context evidence can be modeled in the policies in each of the approaches, as well as whether these evidence are considered together with the requester and the data in the α and φ functions.

4.1.3. Enforcement methods coverage

All enforcement methods denoted with ϵ in Def. 3 are covered in [46]. All other policy system either protect only action or only the data.

Table 2 summarizes the approaches regarding the enforcement methods in its column ϵ.

4.1.4. Conflict resolution ability

The conflict resolution is rarely treated in the reviewed literature, even though it limits the ability to put all requirements in a single context for system protection. The system that does not provide conflict resolution is difficult for maintenance since they require from the administrators to split up the client requirements into disjoint policies that fit in the overall protection. Even in the cases when this is possible, it is error prone and affects to the system correctness. Some systems are providing conflict detection to simplify the maintenance [46].

Most of reviewed approaches are using default optimistic or pessimistic protection [1,11,15]. Some approaches use meta-policies [22,24], but they do not discuss the case of conflicts between them. Policy priority assignment is discussed in §3.8, and the potential of this approach is highlighted there.

4.2. Maintainability aspect

Maintainability is determined by the time required to transform the policies from the requirement, and

to validate their correctness. The transformation process can output a number of policies per requirement, which mainly define the required time. Different client requirements may have different complexity, and thus require a different number of policies, which makes it difficult to compare the systems' maintenance efforts, due to this variance. Thus, the policy complexity also plays an important role, but it diminishes with respect to the policy number cardinality.

The policy number mainly depends on the way activation, and partial data filtering functions are defined. Here, the policy expressiveness plays the most important role.

In cases when the resources are provided directly [11, 20,24], a number of policies are equal to the cross-product of the involved requesters, actions, and resources. The definition of attribute value significantly lowers the policy number, but the number depends on the separation of the resources and subjects based on sharing attributes. The basic graph patterns with filters are indeed query where expressions, that provide fine-grained resource and subject selection with one query. The rules have the same flexibility as the queries.

The policy validation process depends on the tools available, and it is not dependent on the policy format. This process will be modeled in the correctness aspect since it serves to assure that the transformation is error free.

4.3. Understandability aspect

Understandability can be observed as the time required to master the policy format and language. As a general rule, the time required to master a skill is proportional to the price paid for a person that already possess the skill.

The reasoning and formalism aspects are discussed under the policy specification category in [26], and the policy language in [11]. These policy features directly define the required skills for policy management. The following knowledge requirements are mentioned through their work:

- Description Logic (RDF, RDFS, OWL)
- Deontic logic (permission, obligation)
- Logic Programming (Rules, SWRL)
- DATALOG
- Sorted first order logic
- Deduction
- Abduction
- RDFS entailment
- Subsumption
- Query languages (SPARQL, SeRQL)

A ranking of these skills depends on the evaluator's standpoint and is highly subjective. Thus, this paper only identifies the most common skills for authorization management but does not assign concrete rankings for them.

4.4. Correctness aspect

The correctness is a key aspect of all systems. It defines whether the system can guaranty that the resources will be accessed according to the requirement. Conflict resolution and prevention techniques can significantly improve the overall system correctness, but tools for policy results examination are necessary to evaluate the policies against the requirements.

During the policy design and enforcement processes, the following mistakes stand out:

- Policy mismatches requirement
- Overseen conflict
- Unresolved conflict
- Implementation bugs

The first tree mistakes occur during policy design phase due to human imperfection. The only way to overcome them is with tools for policy testing, conflict detection and ability for conflict resolution.

The implementation bugs mainly affect the systems with query rewriting and per action enforcement. The per action protected systems should be tested with unit tests for all subject types in order to guaranty their correctness, while the query rewriting enforcement can be assured with result comparison against the original query run over a previously annotated permitted data [25].

5. Ranking guideline

The comparison tables presented in [11,26] provides enumerated values assignment for the presented aspects. These values are useful for system categorization and elimination if some required feature is absent. When there are multiple suitable systems, it is hard to tell which one is better only from the table values, since the enumerations are not intended for this purpose. Another disadvantage of the table preview is that it may be subjective, and it is hard to prove its accuracy. As discussed in §4, the tables can approximate

the real systems in order to be easier for understanding. Such was the case with Table 2, which do not show whether the elements from different columns can coexist in a same policy or system. Additionally, filling the table's values requires high analytical skills and system understanding, and is a time-consuming process.

This paper tries to overcome these disadvantages, as well as to simplify and speed up the system review process, by introducing the requirements from Table 3. These requirements are for the university domain ontology and data introduced in §3. These requirements are easy to understand, and their modeling in the authorization system is unambiguous, meaning that they can be either transformed to policies or not. Thus, they can be used for objective flexibility determination, not only as a numeric indicator but also can clearly shape the limits of the authorization mechanism.

It is important to note that these requirements are far from sufficient for flexibility determination, compared to the infinite space of possible requirements discussed in §3.4. However, this set of requirements is intended to be used as a starting point for the authorization system review process. The evaluators can easily add new requirements that fit their contextual needs, or remove the one that is irrelevant, and thus shape the process to their needs. These initial requirements are used to demonstrate the ranking guideline for authorization systems, presented in Eq. 1.

$$R = C_f * I_f + C_m * I_m + C_u * I_u + C_c * I_c + C_p * I_p \quad (1)$$

Eq. 1 defines the ranking of the system as a weighted linear combination of the aspects' indicators (the capital I). The capital C are contextual weights for each aspect. They are intended for tuning the importance of the corresponding indicator in the reviewing context. The values for each indicator are described in the following subsections. Before going into details about them, let's first define the ranking steps:

1. Define a suitable set of requirements relevant in the evaluation context.
2. Set each aspect's relevance weight (C_f, C_m, C_u, C_c, and C_p) and mapping values μ_u for each required skill
3. Identify which of them can be transformed into policy.
4. If some key requirements can not be modeled, reject the system.
5. Transform the requirements into policies (the one that can be modeled).
6. Calculate the numerical indicators I_f, I_m, I_u, I_c, and I_p.
7. Calculate the ranking value based on Eq. 1.

The steps 1 and two are executed only at the beginning, while all other steps are executed for each reviewed authorization system. Step 4 rejects the systems that are not able to protect the most important requirements and thus shortens the evaluation time. In the next subsections, each indicator is explained in more details.

5.1. Flexibility indicator I_f

The flexibility aspect was defined in §4.1, and Eq. 2 presents a numerical indicator for the flexibility aspect, according to Def. 5.

$$I_f = \frac{\#supported\ requirements}{\#all\ requirements} \quad (2)$$

The different factors that influence flexibility were identified and were presented in Table 2. The intent expressiveness and its linking with the data is one dimension that limits the flexibility, while the rule expressiveness is the other. The requirements from Table 3 are trying to cover these dimensions. *A1* does not require an activation function and has simple statement conditions for partial data filtering without primitive filters and combinations. The graph membership is used in *TS1*, and dataset membership in *SU1*.

The column ϵ defines whether the policy denies the action, or includes/excludes a partial data. The column evidence describes which evidences are used for activation or partial data filtering. The Greek letter α denotes the evidence required for activation, while φ describes what is used for partial data filtering. The subscript *r* describes that only the requester is used in the corresponding rule, while *cd* and *rd* denote that the data is combined with the context or requester evidence, correspondingly.

The columns "α expr." and "φ expr." describe the expressiveness values required for modeling of the activation and partial data filtering rules in the corresponding requirement. The values in these two columns are described in Table 2.

The requirements from Table 3 also cover a conflict resolution cases. There are four conflicts in these requirements: *A2-U1* and *A2-S1* for mobile phone access, *A2-U2* for mobile phone modification and *P1-P2* for course's grade modification. The column ρ de-

Table 3
Protection requirements

#	Requirement	ρ	ϵ	Evidences	α expr.	φ expr.
A1	The faculty's, study program's and subject's properties are publicly available for everyone.	1	+	/	/	TP
A2	The users' mobile phone number is private.	3	-	α_r	TP	TP, PF
U1	The users can view their own properties and the direct properties of the resources connected with them.	4	+	α_r, φ_{rd}	TP	TP, C
U2	The users can manage their name, phone or email.	5	+	α_r, φ_{rd}	TP	TP
S1	The students can see everything about the professors.	2	+	α_{rd}, φ_{rd}	TP, C	TP, C
S2	The students can see their course colleagues during the semester.	6	+	α_{rd}, $\varphi_{cd,rd}$	TP, C	TP, C
P1	The professors can manage their courses' grades from their faculty's network.	7	+	$\alpha_{rc,rd}$, φ_{rd}	TP, C, PF	TP, C
P2	The professors can not manage their inactive courses' grades.	8	+	$\alpha_{r,rd}$, $\varphi_{rd,cd}$	TP, C	TP, C, PF
P3	The professors can see the average grade for every student from their faculty.	9	+	α_{rd}, φ_{rd}	TP, C	TP, C, DSm
TS1	Technical Staff can manage study programs only for his/hers faculty.	10	+	α_{rd}, φ_{rd}	TP, PF	TP, C, Gm
TS2	Technical Staff can manage everything for all students and professors at their faculty, including the grades for the subjects from the faculty.	11	+	α_{rd}, φ_{rd}	TP, PF	TP, C, DSm
SU1	Only super users can manage the faculties.	12	✓	α_r, φ_d	TP, PF	DSm
MP1	Policy A2 has lower priority from U1					
MP2	Policy A2 has lower priority from U2					
MP3	Policy S1 has lower priority from A2					
MP4	Policy P1 has lower priority from P2					

fines each policy priority and is used to infer the meta-policies with prefix *MP*. These conflict resolution requirements are added to the total number so that the indicator will cover the conflict resolution ability of the reviewed system.

It is clear that these requirements do not cover all possible flexibility options, but the review process always starts with an execution context as a precondition, and these requirements are good starting point for shaping the evaluation toward this context.

5.2. Maintainability indicator I_m

As described in section 4.2, this paper defines a maintainability indicator as reciprocal of the average number of policies required to protect the requirements defined in Table 3. The reciprocity is introduced in order to provide higher value for systems that are easier to maintain with less policies. This is shown in Eq. 3.

$$I_m = \frac{\#all\ implementable\ requirements}{\#policies} \tag{3}$$

5.3. Understandability indicator I_u

$$I_u = \sum^{s_i \in \mathbb{S}_{as}} \mu_u(s_i) \tag{4}$$

The understandability depends heavily on the evaluation context, the administrator's skills in particular. In section 4.3 were presented some possible values for this aspect. A numerical ranking of these aspects is a subjective process. Thus, the indicator this paper propose for this aspect is the sum of the evaluator assigned weights for each of the languages and formalisms. This way the evaluator can decide which of those values are closer to him/her, and which would require a longer time to learn.

The Eq. 4 is describing this relationship where μ_u is the mapping function that assigns a numerical value for understandability of the skill, and $\mathbb{S}_{as}$ is the set of the required skill for the evaluating authorization system (as).

This way the evaluator can account each skill required by the authorization system regarding whether some of the team members possess it; how long would it take to master it; or how much will it cost to outsource it or to hire a person with that skill.

5.4. Policy correctness indicator I_c

The policy definitions correctness depends on the ability for their validation, conflict detection and resolution. In this process, only human can validate the correctness of the requirement.

Since conflict resolution was discussed as a part of the flexibility aspect, the correctness will only depend on the mechanisms and tools available for policy testing and conflict resolution. Since all systems should be tested for implementation correctness by their authors, this type of correctness is not considered as a part of the correctness indicator.

The following values can be made:

- no testing nor conflict detection is available $\Rightarrow$ $I_c \leftarrow 0$,
- only conflict detection is available $\Rightarrow$ $I_c \leftarrow 1$, and
- only testing tool is available $\Rightarrow$ $I_c \leftarrow 2$,
- testing with conflict detection and resolution $\Rightarrow$ $I_c \leftarrow 3$.

The reason for value 1 assignment when there is only conflict detection available is because the conflict detection is a subset of whole testing ability that is required.

5.5. Performance indicator I_p

The performances are the aspect that depends on from the enforcement implementation. As discussed in §2, the authors of different authorization systems are using different data set and evaluation scenarios. Some common practices are to compare the performances of the test queries against the unprotected system, measuring the correlation between the processing time and dataset size, and the correlation between the number of the policies (or their constructs) with the performances. In [25] the authors additionally evaluate the query against different SPARQL query types, accounting different way of processing query constructs.

The benchmark in [5] shows that the query construct combination and the processing engines can influence different query processing and evaluation times. This is the reason for introducing the queries from Table 4 that are covering most of the different SPQRQL constructs.

The goal of the performance evaluation of an authorization system is to measure its impact depending on the imposed access control factors. The generalized model from Fig. 3 shows that for each policy the activation function should be evaluated. Thus, the number of policy and the complexity of the activation function are one factor that imposes additional processing time. Also, the complexity of the partial data filtering function introduces extra time. The activation and data filtering functions are executed over the stored data, which size influences the query execution. Depending on the system's enforcement type, the performances may depend on the amount of the protected data. This is the case in the system that uses enforcement by permitted data annotation and filtering.

For evaluation of the correlation with the dataset size, there are 3 pre-generated datasets (DS1-DS3)[15] with incremental size are available, modeling the instances from the university ontology discussed in §3. Different size of the protected data can be obtained when different requester is used in the intent. Except for Q1, all the other queries return least data for the regular user, then for the students obtain only their grades and data, while the professors retrieve the data

Table 4
Test Case Queries

#	Question	Construct	Policies
Q1	Describe subject	Describe	A1
Q2	Show all students that passed a given course with at least 8 (C)	SELECT	U1, P1, P2, TS2
Q3	Check if a certain student have passed a course	ASK	U1, P1, P2, TS2
Q4	Display the average grade by professor	Aggregation	U1, P1, P2, TS2
Q5	Select the students that have average grade below the course's average	Sub-query	U1, P1, P2, P3, TS2
Q6	Select all students, despite those without phone number	Exist op.	A2, U1, U2, TS2
Q7	Select everything about the users and the subjects	Set op.	A1, A2, U1, U2
Q8	Display a list of dependent subjects for a given subject	Path expression	A1
Q9	Insert grade for student	Insert	P1, P2, TS2
Q10	Remove user's phone number	Delete	A2, U1, U2, TS2
Q11	Create study program	Add graph	TS1
Q12	Remove study program	Delete graph	TS1
Q13	Create faculty	Create dataset	SU1
Q14	Delete faculty	Delete dataset	SU1

[15] https://github.com/ristes/univ-datasets

for their students, and the technical staff retrieves all data from the faculty. The impact of the policies can be observed by incrementing the number of relevant policies in each test run by their copying. The evaluation process should be repeated with each query ($Q_i, i \in [1, 14]$) from Table 4, with the following steps:

- $f_{ds}(Q_i)$: The authorization system with the standard policy setup will be evaluated against the datasets DS1-DS3 with the technical staff user as a requester
- $f_{pd}(Q_i)$: The authorization system with the standard policy setup will be evaluated against the dataset DS3 with a random requester from each of the groups: user, student, professor and technical staff.
- $f_p(Q_i)$: The authorization system will be evaluated using the DS3 for a technical staff user with different setups. First without policies, then with the standard setup, and each next run all policies will be copied one more time, so that their processing impact can be determined.

$$I_p = \sum_{i=1}^{i \leq 14} \mu_p(f_{ds}(Q_i)) + \mu_p(f_{pd}(Q_i)) + \mu_p(f_p(Q_i)) \tag{5}$$

Another challenge is combining all these values together, since they can differ significantly. This paper proposes the performance indicator presented in Eq. 5, based on the categorization of the dependencies through the following function μ_p:

- $\mu_p(speedup) \leftarrow 1$
- $\mu_p(insignificant\ slowing) \leftarrow -0.5$
- $\mu_p(linear\ slowing) \leftarrow -1$
- $\mu_p(significant\ slowing) \leftarrow -2$

The decision of what is insignificant and what is significant slowing is context dependent and left out for the evaluator to decide. As a guideline, linear slowing is a dependency where the function changes linearly with coefficient around 1. The speedup scenario is possible due to the reduction of the returned results with the authorization. Negative values for the slowing categorizations are introduced in order to lower the rank of the authorization system when the slowing is more significant, and to increase the rank when speedup occurs.

5.6. Discussion

For better explanation of the rating guideline, let's put the evaluation into a context where the requirements from Table 3 are sufficient, and there is a crucial need for connecting the intent's requester with the data through the evaluation process.

The second step from the guideline requires to set up the relevance context weights for each aspect. Let's assume that all the factors are equally important, so that $C_f = C_m = C_u = C_c = 1$. Since there is no implementation available for testing of most of the reviewed approaches, this explanation will exclude the performance indicator and will set it contextual weight to 0 ($C_p = 0$). Another assumption used here is that the reviewer is a part of a team that knows SPARQL, is familiar with SWRL and is conservative and careful when new languages or formalisms are introduced. Thus, the understandability mapping is assigned as follows: $\mu_u(SPARQL) = -0.5, \mu_u(SWRL - like) = -1$ and $\mu_u(everythingelse) = -2$. The negative values are introduced to each mapping due to the complexity of each language for policy modeling, meaning that SPARQL is penalized due to its complexity, even though the team knows it.

The last condition for connecting the intent's requester with the data can be used to filter out the approaches that are not suitable, in order to shorten the review process. According to Table 2, the work in [46,30,13,1] matches this condition, and they will be the only one that will be modeled in the following ranking steps.

During step four, [46] is able to model all requirements from Table 3, except TS1 and TS2, since it policies do not have ability to express graph membership, giving flexibility indicator $I_f = 14/16$. The approach in [30] has flexibility indicator $I_f = 5/16$, covering the requirements A1, A2, U1, U2 and S1, [13] has $I_f = 7/16$ (A1, A2, U1, U2, S1, TS1, TS2), and [1] has $I_f = 8/16$ (A1, A2, U1, U2, S1, S2, P1, P2).

In [46] a descriptive logic programming (LP) is combined with decision logic (DL), leading to one policy per modeled requirements ($I_m = 1$) and $I_u = \mu_u(LP) + \mu_u(DL) = -4$). [30] uses SWRL based policies ($I_u = -1$), [1] uses format similar to SWRL ($I_u = -1$) and [13] combines SWRL with SPARQL construct expressions ($I_u = -1.5$). Each of them can model the requirements with one policy, leading to $I_m = 1$ for all of them.

In respect to the correctness, [30] and [13] are using enforcement with permitted dataset filtering which

allows policy result testing, but they do not provide conflict resolution, leading to $I_c = 2$. In [1], a query rewriting approach is presented, but there is no policy testing nor conflict detection tool or mechanism available ($I_c = 0$). The Proteus system [46] utilizes DL and LP for policy definition, which enables conflict detection and policy testing, giving $I_c = 3$.

Once the indicators are computed, the ranking can be computed using the Eq. 1, which gives highest value $R = 2.3125$ to [30], followed by [13] with $R = 1.9375$, [46] with $R = 0.875$, and [1] with $R = 0.5$. This shows that the approach presented in [30] is most suitable if all aspects are equally important, and the performances are not considered. Once the indicators are calculated, it is easy to obtain different rankings by tunning the contextual weights.

6. Conclusion

This paper provides thorough analysis about the authorization systems from the implementers standing point. It identifies that the process starts with requirement transformation into policies that are used for access control enforcement. The flexibility aspect is identified as the ability for requirement transformation into policies, the maintainability represents the time required for this process, and the understandability is determined by the required skills to carry out this process. The correctness of the transformation process is also observed, together with the performance implications that may be acceptable or not.

Using the conclusions from the analyzed literature, the requirements are formalized in order to define the authorization aspects better. In §4 are presented multiple perspectives involved in each aspect, identifying yet another tabular representation of their values.

In §5 notes that the enumerated values are not suitable for ranking purposes. The proposed ranking guideline overcomes this problem with a comprehensive and extensible approach, using multiple requirements that cover a significant amount of the flexibility space. A numerical indicator for each aspect is defined, which enabled the ranking formula definition as an evaluation context dependent linear combination from these indicators.

References

[1] Fabian Abel, Juri Luca De Coi, Nicola Henze, Arne Wolf Koesling, Daniel Krause, and Daniel Olmedilla. Enabling advanced and context-dependent access control in rdf stores. In *The Semantic Web*, pages 1–14. Springer, 2007.

[2] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra. Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, pages 1–10. ACM, 2008.

[3] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

[5] Christian Bizer and Andreas Schultz. The berlin sparql benchmark, 2009.

[6] Piero Bonatti and Daniel Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, pages 14–23. IEEE, 2005.

[7] Willy Chen and Heiner Stuckenschmidt. A model-driven approach to enable access control for ontologies. In *Wirtschaftsinformatik (1)*, pages 663–672, 2009.

[8] Antonio Corrad, Rebecca Montanari, and Daniela Tibaldi. Context-based access control management in ubiquitous environments. In *Network Computing and Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on*, pages 253–260. IEEE, 2004.

[9] Luca Costabello, Serena Villata, Nicolas Delaforge, and Fabien Gandon. Linked data access goes mobile: Context-aware authorization for graph stores. In *LDOW-5th WWW Workshop on Linked Data on the Web-2012*, 2012.

[10] Luca Costabello, Serena Villata, and Fabien Gandon. Context-aware access control for rdf graph stores. In *ECAI*, volume 242, pages 282–287, 2012.

[11] Luca Costabello, Serena Villata, Oscar Rodriguez Rocha, and Fabien Gandon. Access control for http operations on linked data. In *Extended Semantic Web Conference*, pages 185–199. Springer, 2013.

[12] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa, 2006.

[13] Sebastian Dietzold and Sören Auer. Access control on rdf triple stores from a semantic wiki perspective. In *ESWC Workshop on Scripting for the Semantic Web*. Citeseer, 2006.

[14] Tim Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi Sandhu, William Winsborough, and Bhavani Thuraisingham. R owl bac: representing role based access control in owl. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 73–82. ACM, 2008.

[15] Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. Controlling access to rdf graphs. In *Future Internet Symposium*, pages 107–117. Springer, 2010.

[16] Stefano Franzoni, Pietro Mazzoleni, Stefano Valtolina, and Elisa Bertino. Towards a fine-grained access control model and mechanisms for semantic databases. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 993–1000. IEEE, 2007.

[17] Simon Godik, Anne Anderson, Bill Parducci, P Humenn, and S Vajjhala. Oasis extensible access control 2 markup language (xacml) 3. Technical report, Tech. rep., OASIS, 2002.

[18] Patricia P Griffiths and Bradford W Wade. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems (TODS)*, 1(3):242–255, 1976.

[19] Dick Hardt. The oauth 2.0 authorization framework. 2012.

[20] James Hollenbach, Joe Presbrey, and Tim Berners-Lee. Using rdf metadata to enable access control on the social semantic web. In *Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2009)*, volume 514, 2009.

[21] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79, 2004.

[22] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 63–74. IEEE, 2003.

[23] Sabrina Kirrane. *Linked data with access control*. PhD thesis, 2015.

[24] Sabrina Kirrane. *Linked data with access control*. PhD thesis, 2015.

[25] Sabrina Kirrane. *Linked data with access control*. PhD thesis, 2015.

[26] Sabrina Kirrane, Alessandra Mileo, and Stefan Decker. Access control and the resource description framework: A survey. *Semantic Web*, 8(2):311–352, 2017.

[27] Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *Proceedings of the 16th international conference on World Wide Web*, pages 677–686. ACM, 2007.

[28] Jian Li and William K Cheung. Query rewriting for access control on semantic web. In *Workshop on Secure Data Management*, pages 151–168. Springer, 2008.

[29] André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics, 2010.

[30] Hannes Muhleisen, Martin Kost, and Johann-Christoph Freytag. Swrl-based access policies for linked data. *Procs of SPOT*, 80, 2010.

[31] Said Oulmakhzoune, Nora Cuppens-Boulahia, Frédéric Cuppens, and Stephane Morucci. fquery: Sparql query rewriting to enforce data confidentiality. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 146–161. Springer, 2010.

[32] Ankur Padia, Tim Finin, and Anupam Joshi. Attribute-based fine grained access control for triple stores. In *3rd Society, Privacy and the Semantic Web-Policy and Technology workshop, 14th International Semantic Web Conference*, 2015.

[33] Torsten Priebe, Wolfgang Dobmeier, and Nora Kamprath. Supporting attribute-based access control with ontologies. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8–pp. IEEE, 2006.

[34] Eric PrudâĂŹHommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[35] Owen Sacco and John G Breslin. Ppo & ppm 2.0: Extending the privacy preference framework to provide finer-grained access control for the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 80–87. ACM, 2012.

[36] Owen Sacco, Matteo Collina, Gregor Schiele, Giovanni Emanuele Corazza, John G Breslin, and Manfred Hauswirth. Fine-grained access control for rdf data on mobile devices. In *International Conference on Web Information Systems Engineering*, pages 478–487. Springer, 2013.

[37] Owen Sacco and Alexandre Passant. A privacy preference manager for the social semantic web. In *SPIM*, pages 42–53, 2011.

[38] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.

[39] Ravi S Sandhu, Edward J Coynek, Hal L Feinsteink, and Charles E Youmank. Role-based access control models yz. *IEEE computer*, 29(2):38–47, 1996.

[40] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.

[41] Carlo Scarioni. *Pro Spring Security*. Apress, 2013.

[42] Steve Speicher, John Arwe, and Ashok Malhotra. Linked data platform 1.0. *W3C Recommendation, February*, 26, 2015.

[43] Manu Sporny, Toby Inkster, Henry Story, Bruno Harbulot, and Reto Bachmann-Gmür. Webid 1.0: Web identification and discovery. *Editor's draft, W3C*, 2011.

[44] Henry Story, Bruno Harbulot, Ian Jacobi, and Mike Jones. Foaf+ ssl: Restful authentication for the social web. In *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, 2009.

[45] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In *International semantic web conference*, pages 473–486. Springer, 2006.

[46] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. Proteus: A semantic context-aware adaptive policy model. In *Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, pages 129–140. IEEE, 2007.

[47] Qihua Wang, Ting Yu, Ninghui Li, Jorge Lobo, Elisa Bertino, Keith Irwin, and Ji-Won Byun. On the correctness criteria of fine-grained access control in relational databases. In *Proceedings of the 33rd international conference on Very large data bases*, pages 555–566. VLDB Endowment, 2007.