RSL-IL4Privacy: A Domain-Specific Language for the Rigorous Specification of Privacy Policies

João Caramujo, Alberto Rodrigues da Silva, Shaghayegh Monfared, André Ribeiro, Pável Calado

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

Travis Breaux

Institute for Software Research, Carnegie Mellon University, Pittsburgh, USA

Abstract: Mobile and web applications that manage users' personal information require developers to align their software design with privacy requirements commonly described in privacy policies. These policies are often the sole means to enforce accountability on that data protection. We propose the RSL-IL4Privacy, a domain-specific language for specifying privacy policies that can be simultaneously manipulated by computers and authored and analyzed by humans. In addition, RSL-IL4Privacy can be used as an intermediate language to support model-to-model transformations from and into other related languages. RSL-IL4Privacy provides policy authors with means to define a privacy policy as a set of declarative statements with explicit relationships to services, data recipients, private data types, and enforcement mechanisms. The RSL-IL4Privacy is defined with different technologies for supporting distinct levels of formality, namely support for multiple modes of presenting privacy requirements, including tabular, graphical and textual representations, to increase integration with a wider variety of authoring and analyzing practices. We apply this language to support the analysis and comparison of policies from Facebook, LinkedIn, Twitter, Dropbox and IMDb. We discuss with further detail the application of this approach to the Twitter policy by presenting several examples with multiple representations. Finally, we discuss how RSL-IL4Privacy can improve the quality of privacy policies and also identifies threats to validity.

Key Words: Privacy Policy, Privacy Requirement, Domain-Specific Language, RSL-IL4Privacy, Eddy

1. INTRODUCTION

Web and mobile applications collect data from users without ensuring traceability between privacy policies and application design decisions. A particular challenge for policy authors and application developers is the need to find a common language that supports translating important privacy policy statements into actionable requirements. European Union and United States require their organizations to publish privacy policies as "notices" to end users and, these policies are often the sole means to enforce accountability on data protection of their systems. For example, Google developed new services that ultimately re-purposed user data in ways that violated earlier versions of their privacy policy; and third-party "social-plugins" at Facebook were found to transfer Facebook user data to advertisers in violation of Facebook's platform policies. Given the pressure to post privacy policies and the pressure to keep policies honest, companies must do more to align their policies and practices. In this respect, we believe more can be accomplished by providing developers with new tools to better specify their data needs while policy authors, who are typically legal professionals, can work with those specifications to create more accurate policies or to enforce those policies in the context of developer data needs.

In general, privacy policies govern the sensitive and more private data practices between a service provider and its users. These policies include statements that convey information about several different actions concerning users' privacy, e.g., how to collect, share and disclose user data. In our point of view, a privacy policy can be defined as a technical document that states the multiple privacy-related requirements that a system should satisfy. These requirements are usually defined as ad-hoc natural language (NL) statements. Natural language is an ideal medium to express privacy policies, because it is flexible, universal, and humans are proficient at using NL to communicate. Moreover, NL has minimal adoption resistance as a requirements documentation technique [1]. Although it is the most common and preferred form of requirements representation [2], NL has intrinsic characteristics that become the root cause of quality problems, such as incorrectness, inconsistency or incompleteness [1]. On the other hand, there are some focused privacy-specific languages, such as P3P, XACML, Eddy, KAoS or Rei, as further discussed in Section 6. However, these languages tend to represent policies in a low-level XML formats or formal and compact representations, appropriated to be manipulated and processed by computers but not so by humans.

Bearing these tradeoffs in mind, we propose a domain-specific language for specifying the privacy policies that can be simultaneously manipulated by computers but also authored and analyzed by humans. RSLingo4Privacy is a complete approach for specifying and analyzing privacy policies that adopts an *intermediate language* for documenting privacy requirements, which include optative statements about the collection, use and transfer of personal information for a specified purpose by a system. By documenting these statements in a specification, we can analyze these policies to check whether a company's practices conform to international privacy standards, such as data minimization and collection and use limitation.

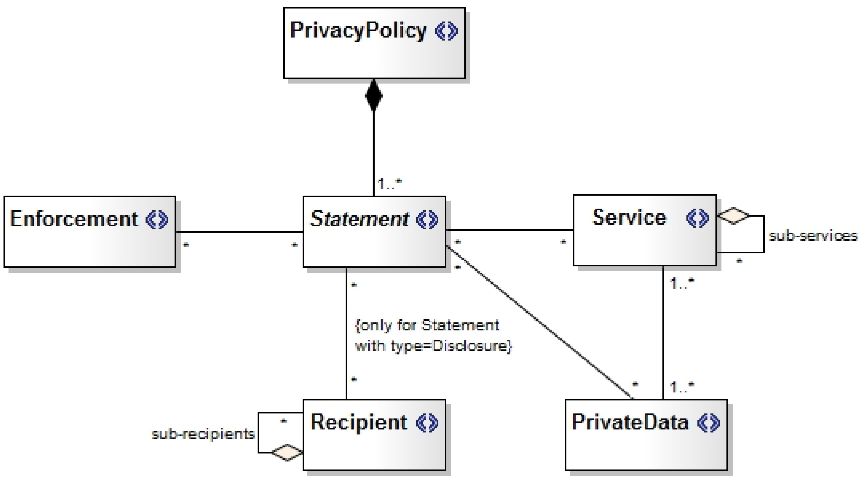


Fig.1. RSL-IL4Privacy metamodel (partial view)

This proposed RSL-IL4Privacy is a domain-specific language that was defined originally as a privacy-aware profile that identified the main concepts of current privacy policies [3]. RSL-IL4Privacy allows policy authors to specify privacy policies by providing several constructs, such as the data type, data recipient and enforcement mechanism, which are necessary to specify and document privacy-related requirements. Through the use of the RSL-IL4Privacy language, the RSLingo4Privacy approach integrates the following processes [4]: (1) automatic classification, extraction and translation of statements from privacy policies written in NL into RSL-IL4Privacy specifications; (2) visualization and authoring of specifications; (3) analysis and validation of specifications; and (4) publishing these specifications in both structured and machine-processable formats, such as Eddy [5]. The goal of the RSLingo4Privacy approach is to use the RSL-IL4Privacy formalization as the necessary mechanism to improve the specification of privacy policies while providing features for better analyzing and validating the involved privacy requirements.

In this paper, we extend the RSL-IL4Privacy language and RSLingo4Privacy approach with the following novel contributions. First, concerning the improvement of the RSL-IL4Privacy language, it is now defined in a more consistent and flexible way, for example: the multiplicities of associations between its elements are defined more consistently; elements such as services and recipients consistently defined with the possibility of having sub-elements (based on "isPart" relationships); added the deontic (i.e., "modality") property to the Statement element; eliminated the hierarchy of statement's specializations (i.e., collection, disclosure, etc.) by just keep the Statement element with the respective "statementType" property and, in addition, allowing a statement be categorized with more than just one type. Second, RSL-IL4Privacy is now implemented as a rigorous textual representation based on the Xtext framework, and its partial grammar and examples are showed. Third, we illustrate the multiple representations of this language with a complete case study based on Twitter's privacy policy. Fourth, we discuss an alignment between RSL-IL4Privacy and the Eddy language [5] and subsequent results from analyzing RSL-IL4Privacy specifications using the Eddy toolset based on five distinct privacy policies, namely Facebook, LinkedIn, Twitter, Dropbox and IMDb. Fifth, we discuss and compare these policies, detailing the differences emphasized by the RSL-IL4Privacy specifications. Sixth, we describe the main features (e.g., authoring, import, export, check quality) and transformations (e.g., from excel into Xtext representation or vice-versa) included in RSLingo4Privacy-Studio, the tool that supports the proposed RSLingo4Privacy approach. Finally, we discuss the related work and compare it with our proposal, namely by discussing related approaches taking into account the analysis of their languages and respective tool support.

This paper is organized as follows: Section 2 introduces the proposed language and overviews its abstract syntax as both a UML metamodel and a grammar defined with Xtext framework (for the formalization of such metamodel in terms of a concrete textual syntax); Section 3 overviews RSLingo4Privacy-Studio, the software tool that supports RSL-IL4Privacy and integrates the necessary features to support both the specification, analysis and publication of privacy policies. Section 4 demonstrates the flexibility of this proposal through the development of a method that can issue the necessary interoperability features between RSL-IL4Privacy and other formal languages in order to perform an in-depth analysis of a given policy. Section 5 compares five privacy policies on different levels and discusses their identified conflicts; it also provides an extensive application of our proposal based on the Twitter case study by presenting several examples with multiple representations. Section 6 presents and discusses the related work by both comparing RSL-IL4Privacy with other similar languages and technologies. Section 7 discusses how RSL-IL4Privacy can improve the analysis of privacy policies and also identifies some threats to validity. Finally, Section 8 presents the conclusion and identifies open issues for consideration in future work.

2. THE RSL-IL4PRIVACY LANGUAGE

An important activity when developing and integrating complex software systems, such as social networks or software systems with many users, is the enforcement of their privacy policies. One can argue that an incomplete and imprecise specification of privacy requirements may compromise the process of analysis and reasoning of such privacy policies. For example, a policy that describes a collection of data without any mention of how this data will be used makes it difficult to detect whether this data is being used for a different purpose than that for which it was originally collected.

RSL-IL4Privacy is a language that enables a rigorous representation and specification of privacy requirements not achievable by writing NL policies alone. RSL-IL4Privacy is particularly targeted to, on one hand, people responsible for authoring such policies (e.g., policy authors and requirements engineers), and on the other hand, people that need to read and better understand such policies (e.g., developers and even end-users).

RSL-IL4Privacy is defined as a domain-specific language (DSL) [6, 7] because it allows: the possibility of expressing solutions in the idiom and with the level of abstraction of the problem domain; increasing the solution's portability, maintainability, reliability and productivity; and the prospect of reusing domain knowledge in future solutions. In addition, the adoption of a language such as RSL-IL4Privacy allows its specifications to become simpler to read and understand which facilitates the communication between system developers and domain experts.

2.1 Overview of the Language Metamodel

Through the definition of a complete privacy-aware profile [3] it was possible to consider common elements represented in privacy policies. These elements, and multiple relationships between them, serve as the basis for the specification of the constructs within RSL-IL4Privacy. We overview this language by summarizing its core elements: *Statement*, *Recipient*, *PrivateData*, *Service* and *Enforcement*. We also highlight the refinements to the previous version of that profile. Figure 1 depicts a partial view of the RSL-IL4Privacy metamodel, a visual schema of its key constructs and associations, while the complete metamodel is illustrated in Appendix A.

2.1.1 Statement

A *Statement* describes the rules or actions specified in a policy, thus in general it can be seen as a privacy requirement. A policy comprises a set of statements and each statement may include more than one sentence (e.g., if the meaning of a sentence is dependent on the previous sentence, both sentences encompass a single statement). Depending on the data practices a statement details, it is possible to define five different categories: Collection, defines which data is collected; Disclosure, defines which data is disclosed and eventually to what entities; Retention, defines for how long data will be stored; Usage, defines the purpose for having the data; and Informative, statements with just generic information. It is noteworthy that, according to the information it conveys, a statement can be labeled with more than one type (e.g., "Collection, Disclosure"). Additionally, and for the purpose of reasoning the policies, namely in what concerns deontic logic reasoning [26], each statement is also defined in terms of its deontic modality (i.e., "Permitted", "Obligation" or "Forbidden") depending on the actions they describe. Lastly, one statement may refer to multiple services and act on different private data (*Service* and *PrivateData* elements, respectively). Aside from the Informative statements (the "generic" statements), all the remaining ones, i.e. statements of type Collection, Disclosure, Retention and Usage shall be seen as privacy requirements. These privacy requirements may then be mapped or represented as system goals, in some goal-oriented requirements engineering language (such as i*, KAOS or NFR) [52].

2.1.2 Recipient

The *Recipient* element describes the set of entities to whom is disclosed all or part of the users' personal information, i.e., a *Recipient* element is the set of entities that may receive the information shared by the service provider. The characterization of a given recipient is carried out while taking into account the description of its activity that is present in each one of the statements.

2.1.3 PrivateData

The *PrivateData* element represents the users' data that is collected and managed by the social network (or other service provider). A *PrivateData* element can be defined as personal or usage depending on the source of the information: personal, if such information clearly identifies a given user (e.g., name, email); usage, if the data is gathered based on the user's activity on the system (e.g., device specifications, current user localization).

2.1.4 Service

The *Service* element describes the multiple high-level services that are provided through the users' point of view. A service can aggregate multiple sub-services that cover more specific purposes regarding the different data flows. While it might be useful to look for major services within policies, smaller services ensure a proper conflict detection. For instance, statements like "We do not share your data in order to send promotional communications" and "We share your data to track the effectiveness of our ads" are not directly conflicting. Even though they refer to the same high-level service – Advertising Service –, the purpose for the sharing is clearly different. It is important to point out the association between *Service* and *PrivateData* elements because it makes possible to track what personal information is being employed in each service.

2.1.5 Enforcement

Lastly, the *Enforcement* element is particularly useful because it comprises the description of the mechanisms and tools that are documented in a policy and that allow one to gain insight about how it can be possible to enforce the privacy requirements of such privacy policies. On the other hand, it also encompasses rules and specific actions (e.g., check if people under 13 years old should not use the service) with regard to the use of the system that are important for the enforcement of a given privacy policy. As opposed to what happens in some real-world privacy policies, where such information is often scattered throughout the policy, this element allows policy authors to clearly identify and describe such enforcement information in just one type of object.

2.2 Multiple Representations

The RSL-IL4Privacy language is implemented with different technologies that provide multiple representations (i.e., concrete syntaxes), namely: tabular, graphical, and textual.

The tabular representation is supported by a MS-Excel template that includes some predefined sheets, defined in accordance with the RSL-IL4Privacy language. For example, a sheet with statements (for *Statement* elements), another sheet with private data (for *PrivateData* elements), etc. Additionally, it might include some analysis reports and graphics on top of that source information.

The graphical representation is based on UML CASE tools, such as Sparx Systems Enterprise Architect[1]. We implement the RSL-IL4Privacy as a UML profile and consequently we can represent any policy as a UML package, its associated elements (e.g., statements, services, and private data) as UML objects and their relations as UML links. In that way, we may represent graphically all the involved elements and respective relations.

The tabular and graphical representations provide a structured overview of the arrangements among all the constructs that exist in a policy. However, these representations do not easily support integration with other types of requirements, such as use cases or functional requirements, nor do they easily support automatic validation of such requirements. Therefore, we formalize and define the RSL-IL4Privacy language with a more rigorous textual representation. For that purpose we use the Xtext framework[2], which is an open-source framework covering all aspects of language infrastructure such as parsers, linkers, compilers, interpreters and a full-blown IDE support based on Eclipse. The Xtext framework uses an Xtext grammar (a set of BNF-like rules) for the definition of the concrete textual syntax of the language, which is able to describe elements and their relations [8][9]. A partial definition of the RSL-IL4Privacy grammar is shown in Fig. 2.

[1] Sparx System Enterprise Architect, http://www.sparxsystems.com
[2] Xtext framework, https://eclipse.org/Xtext

```
// PrivacyPolicy
‘package’ name=QualifiedName ‘{‘
   privateData+=PrivateData*
   recipients+=Recipient*
   services+=Service*
   enforcements+=Enforcement*
   statements+=Statement*
‘}’;

// PrivateData
‘privateData’ name=ID ‘:’ type=PrivateDataType ‘{‘
   ‘name’ aliasName=STRING
   (‘description’ description=STRING)?
   attributes+=Attribute*
‘}’;

// PrivateData Attribute
   ‘attribute’ name=ID ‘:’ type=AttributeType ‘{‘
   ‘name’ aliasName=STRING
   (‘description’ description=STRING)?
‘}’;

// Recipient
‘recipient’ name=ID ‘:’ type=RecipientType ‘{‘
   ‘name’ aliasName=STRING
   ‘scope’ scope=RecipientScope
   (‘partOf’ partOf=[Recipient])?
   (‘description’ description=STRING)?
‘}’;

// Service
‘service’ name=ID ‘:’ type=ServiceType ‘{‘
   ‘name’ serviceName=STRING
   (‘refersPrivateData’ (refPrivateData=RefPrivateData | refPDAll=‘All’))?
   (‘partOf’ partOf=[Service])?
   (‘description’ description=STRING)?
‘}’;

// Enforcement
   ‘enforcement’ name=ID ‘:’ type=EnforcementType ‘{‘
   ‘name’ enforcementName=STRING
   (‘description’ description=STRING)?
‘}’;

// Statement
‘statement’ name=ID ‘:’ type+=StatementType (‘,’  type+=StatementType)* ‘{‘
   ‘name’ aliasName=STRING
   ‘modality’ modality=ModalityType
   (‘description’ description=STRING)?
   (‘period’ period=STRING)?  // only for Retention Statements

   (‘recipient’ (recipients=RefRecipient | refRecipientAll=‘All’))? //only for Disclosure
Statements
   (‘privateData’ (privateData=RefPrivateData  | refPrivateDataAll=‘All’))?
   (‘services’ (services=RefService  | refServiceAll=‘All’))?
   (‘enforcement’ (enforcements=RefEnforcement  | refEnforcementAll=‘All’))?
‘}’;
```

Fig. 2. Partial Grammar of RSL-IL4Privacy

The rules of the grammar define the key elements and their relationships. Each element has a name, a type and some mandatory or optional properties. The type of a property can be defined as final data type (e.g., String, Enumeration) or as a reference to other elements. The concrete syntax of RSL-IL4Privacy has certain similarities to other widely known software languages. Each element starts with its name followed by ":" and its type. The brackets "{" and "}" encompass a new block containing its properties/attributes definition. The block ends with a ";" after the right brace. Each attribute has an identifier and their value appears afterwards. The reader should note the optional attributes such as the "period" attribute for the *Statement* element, with multiplicity "0..1", defined with the property mask "?". From this grammar definition the Xtext framework generates an authoring environment that includes various language components and features common to powerful text editors such as syntax coloring, code completion, or error checking (further information in the following Section 3).

3. RSLINGO4PRIVACY-STUDIO

RSLingo4Privacy-Studio is a tool that supports the RSLingo4Privacy approach [4], providing the technological support for specifying and analyzing privacy policies. RSLingo4Privacy-Studio is built on top of the Eclipse IDE, more specifically leveraging the Eclipse Modeling Framework (EMF). It uses the RSL-IL4Privacy language as an intermediate language to represent the privacy policies and to generate more readable representations of such policies (e.g. MS-Excel or MS-Word files), or to check their quality using either its own validators or an Eddy engine (Eddy is explained in detail in Section 4.A). As can be seen in Fig. 3, RSLingo4Privacy-Studio relies on several transformations to support the multiple privacy policies representations and the mapping between them, namely Text-to-Model (T2M), Model-to-Model (M2M) and Model-to-Text (M2T). RSLingo4Privacy-Studio provides three main features: Import, Export and Check Quality.

The Import feature allows a user to import an Excel file (transformation M2M-1) or an ad-hoc natural language text file (transformation T2M) containing the policy and generate its corresponding RSL-IL4Privacy files. M2M-1 is implemented using the Apache POI library[3], while T2M is implemented through the execution of automatic text classification and text extraction processes. The Export feature allows transforming a RSL-IL4Privacy file to other file formats, namely Word, Excel, JSON, Eddy and Text (controlled NL). Transformations M2M-2 and M2M-3 are implemented using the Apache POI library and two template files (one for Word and another for Excel). In turn, the remaining transformations for JSON (M2T-1), Text (M2T-2) and Eddy (M2M-4) are implemented using Xtend[4]. The Check Quality feature is only applicable to Eddy files and allows running the Eddy engine to check if there are any conflicts in the provided policy described in the Eddy file. The output of this process is a log file, containing the possible conflicts, and an OWL (Web Ontology Language) file with the representation of the rules used to describe the policy.

As mentioned before, Studio relies on a multi-transformation approach which includes T2M, M2M and M2T transformations. On the other hand, Studio deals with policies documented with multiple representations, namely: ad-hoc and controlled NL text, Excel, Word, JSON and Eddy. We have considered an Excel file as model, since it is a tabular and highly structured representation. In contrast, we consider that a Word file is similar to a NL text, in the sense that it contains plain text but with just low-level formatting information. However, since it is not as structured as an Excel file we considered it as text and not a model (despite being both internally organized in an archive of multiple XML files). Below we describe each transformation type and then we explain their implementation issues, which are grouped by the technology used to support them.

[3] https://poi.apache.org/

[4] http://www.eclipse.org/xtend/

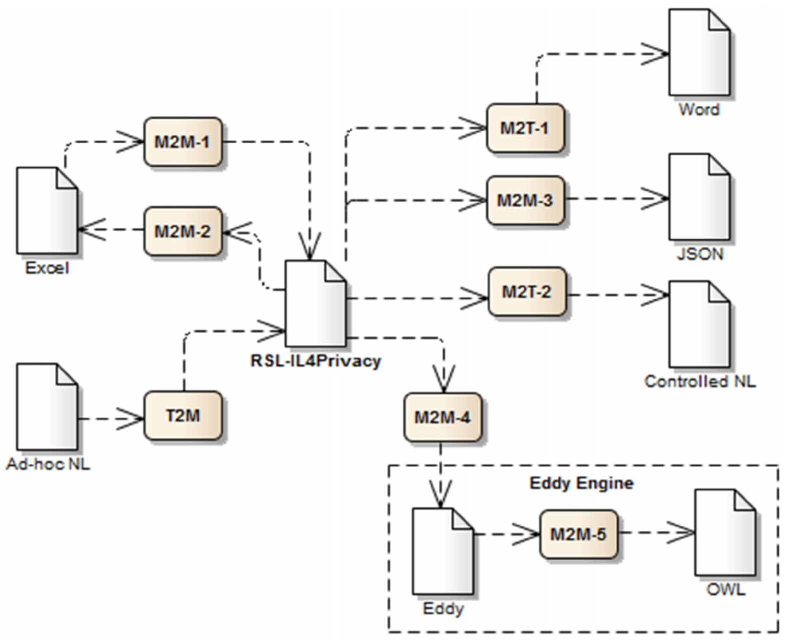


Fig 3. RSLingo4Privacy-Studio supported transformations

3.1.1 T2M Transformations

RSLingo4Privacy-Studio performs a T2M transformation during the import process of an ad-hoc NL text file. This transformation involves the execution of automatic text classification and extraction processes. The classification process identifies the set of statements in the policy provided and classifies them as “Collection”, “Disclosure”, “Retention”, “Usage” or “Informative”. The second process extracts the relevant elements from the original statements into their equivalent representation in RSL-IL4Privacy. The implementation of this transformation is a complex task that involves the integration and tuning of feature models and tools, and therefore it is still a work in progress task.

3.1.2 M2M Transformations

M2M transformations are used during the import and export of a policy in Studio. The import of an Excel file specifying a policy (transformation M2M-1) generates its corresponding RSL-IL4Privacy file(s), depending on the file structure mode the user has selected (single or multiple). M2M-1 is implemented using the Apache POI library which simplifies the processing of Microsoft Office file formats. M2M-2 performs the reverse transformation (from RSL-IL4Privacy to Excel) and is also implemented using the Apache POI library, but uses an Excel template file. The remaining transformations consist in the export of a privacy policy specified in RSL-IL4Privacy for JSON (M2M-3) and Eddy (M2M-4).

3.1.3 M2T Transformations

These transformations occur when a RSL-IL4Privacy file is exported to Word (transformation M2T-1) and controlled NL text file (transformation M2T-2). The transformations from RSL-IL4Privacy into JSON (M2M-3), Text (M2T-2) and Eddy (M2M-4) are performed using Xtend. A code generator stub written in Xtend is one of the artifacts that is automatically generated from a Xtext grammar definition and automatically integrated into the produced Eclipse plugin. Xtend simplifies the usage and maintenance of the code generator by allowing the definition of code templates, portions of code that contain dynamic parts that change according to the Xtext-based model given as input. On the other hand, the transformations from RSL-IL4Privacy into Word (M2T-1) and Excel (M2M-2) are performed using the Apache POI library and two companion template files (one for each format). We use this library instead of Xtend, because it highly abstracts the complex XML structure that underlies Microsoft Office files. Additionally, we used template files to give more flexibility for a user to customize the style and formatting of the generated files.

Both Word and Excel templates have special tag annotations that represent the dynamic part of the template and identify which property should be placed there during the generation. They are defined using the style (e.g. font type, size or color) that should be reflected in the generated file.

The Word template is a document organized in sections, one for each concept of the RSL-IL4Privacy language (e.g. *Statements*, *Services* and *PrivateData*) and contains subsections for the *Services* and *Recipients*, which can contain *Sub-Services* and *Sub-Recipients*, respectively. Each section and subsection is delimited by a start tag and an end tag. During the transformation, each section is copied as many times as the number of elements of that type that exist, and the tags are replaced by the value of the respective property of each element.

The Excel template is a workbook organized in sheets, one for each concept of the RSL-IL4Privacy language. Each sheet name identifies the set of elements that describes (e.g. *Statements*, *Services* and *Recipients*) and contains a head row identifying the content of each column and then an example row containing the tags annotations. During the transformation, the example row is copied as many times as the number of elements of that type that exist, and the tags are replaced by the value of the respective property of each element. As can be noted, this process is analogous to the one applied for the sections in the Word template.

3.2 RSL-IL4Privacy Editor

RSLingo4Privacy-Studio provides a textual editor for creating and editing RSL-IL4Privacy files. This editor was developed using the Xtext framework which allows the development of textual DSLs through their definition using Xtext's grammar. From that grammar definition it is possible to automatically generate the full language infrastructure (parser, linker, typechecker and compiler) and a fully customizable Eclipse plugin containing the DSL editor with helpful features like syntax highlighting, error checking, auto-completion or source-code navigation [8]. Xtext-based DSLs have Ecore as metamodel. Since Xtext relies on EMF, it can be combined with other popular Eclipse plugins, like Xtend, Sirius or Acceleo. The grammar of a Xtext-based language is composed of rules that describe its key elements and their relations. Each element has a name and some properties.

The RSL-IL4Privacy editor is able to deal with two file structure modes to specify a privacy policy: (1) Single file; or (2) Multiple files with one file per element. In the Single file mode, there is only one file that contains all the privacy policy concepts. This strategy is recommended when the privacy policy is small, otherwise will be hard to maintain it. In the Multiple files mode, there is a main file which is used to reference all the other files (one for each RSL-IL4Privacy concept) through "import" statements. The main file also includes metadata that describes the policy like its name, authors, version and date. This strategy can enhance the maintenance of the policy specification because the different concepts are not mixed in the same file, but instead defined in multiple and isolated files. For instance, if a user needs to add a new statement, he only needs to open the file where the statements are specified.

4. SPECIFICATION ANALYSIS WITH EDDY

One of common claims of formal languages to be used for the specification of privacy policies is their possibility of performing some complex analysis and reasoning about such requirements and policies. On the other hand, having a specification written in a language with a controlled syntax (such as RSL-IL4Privacy) that still preserves natural language to the maximum, provides a better readability and understanding of such requirements but, at the same time, makes it more difficult to automatically analyze these specifications.

One solution for this dichotomy requires combine both types of languages. Using this solution it is possible to benefit from both approaches, i.e. to present an understandable specification with complementary representations (in comparison to the original policy), while abstracting the reasoning process which is complex and a forerunner for the creation of more formal languages.

4.1 Eddy Language and Detection of Conflicts using Eddy

Eddy is a formal language defined for specifying privacy and related data flow requirements, therefore allowing one to trace data flows and check for potential requirements conflicts [5]. The specification of privacy requirements takes into account three main concepts that are related to each other and that can be structured hierarchically: datum, actor and purpose. The Eddy approach consists of mapping requirements (that include the aforementioned concepts) written in natural language text to actions (e.g., collection of data) and roles (i.e., relationships between datum, actor and purpose) in Description Logic (DL) [10]. The use of DL makes it possible to detect conflicts: within a policy, a conflict occurs when there is an intersection between permitted and forbidden actions (e.g., sharing data) carried out by an actor for some purpose. Even though an action can be a “Permission”, “Prohibition” (i.e., a forbidden action) or an “Obligation” (in terms of its modality), the detection of the abovementioned type of conflicts does not require allowed actions to be classified as “Permitted” or “Obligation”. Since our goal is just to determine if such conflicts exist, labeling an action as “Permitted” or “Obligation” has no particular relevance to the process. However, “permitted-obligation” conflicts tend to occur within privacy policies but their detection is more complex and requires an extra amount of work. Even though we allow the classification of statements (i.e., the actions they describe) as “Permitted” and “Obligation”, the eventual conflicts of this kind are out of the scope of the approach detailed in this paper, thus, their conflict detection process is not discussed. To keep the paper concise and simple, the statements analyzed in the case study which is described in Section 5, are all considered as “Permission” although the reader may be able to identify different degrees of “Permission” across statements.

4.2 Transform RSL-IL4Privacy into Eddy

A RSL-IL4Privacy to Eddy transformation was defined in the context of the Xtext framework. With this feature it is possible to generate Eddy specifications from equivalent RSL-IL4Privacy specifications. To define this generator we had to find all the common matching concepts between both RSL-IL4Privacy and Eddy grammars.

As referred above a policy specified in RSL-IL4Privacy encompasses a set of privacy elements: *Statement*, *Service*, *Recipient*, *PrivateData* and *Enforcement*. The single definition of a statement (i.e., its description, modality, etc.) encloses the various associations with the remaining elements that are, in turn, defined on the bottom of the policy in RSL-IL4Privacy. On the other hand, a policy in Eddy is represented with a specification header (“SPEC HEADER”) and the following specification body (“SPEC POLICY”). The header aggregates the prior definitions of three elements: “P” for Purpose, “A” for Actor and “D” for Datum. The statements are then described on the body. Each statement has a modality, wherein “P” indicates permission, “O” indicates obligation and “R” indicates prohibition; the action verb; the Datum; the source (following "FROM"), the target (following "TO") and the Purpose (following "FOR"). Based on the description of the different elements and keywords from both languages, it is possible to map the following concepts: the *PrivateData* can be considered as Datum, the *Service* as Purpose and the *Recipient* as Actor (target). Since the source ("FROM") refers to the service provider, there is not a direct match between concepts in the two languages. Under the scope of RSLingo4Privacy, we are only interested in the actions performed by a given single service provider, thus RSL-IL4Privacy does not need to distinguish between providers. The key mappings between both grammars are defined in Table I.

Table I. Matching keywords for RSL-IL4Privacy and Eddy grammars

Language	**Modality**	**Action (type)**	**Data**	**Source**	**Target**	**Purpose**
Eddy	P, O, R	COLLECT, TRANSFER, USE, RETAIN	D (Datum)	FROM	TO	P
RSL-IL4Privacy	Permitted, Obligation, Forbidden	Collection, Disclosure, Usage, Retention	privateData	-	recipient	service (description)

5. RESEARCH VALIDATION

The RSL-IL4Privacy language is evaluated against a set of real-world privacy policies to demonstrate how well it can be used to express and analyze policies. In addition, since this language also aims to be flexible, it is of the utmost importance to demonstrate its value by providing the tabular, graphical and textual representations for each policy. These representations are related to distinct levels of formality (being tabular the less formal and textual the most formal) and are supported by independent technologies.

To validate our approach, we selected five privacy policies from well-known companies (Facebook, LinkedIn, Twitter, Dropbox and IMDb) and then used RSL-IL4Privacy to specify such policies. A study of these privacy policies was previously completed to support a preliminary strategy on the information extraction tasks that are part of the RSLingo4Privacy approach [4]. This prior analysis drew our attention to the need of performing a revision on the original definition of the RSL-IL4Privacy language. We noted from the analysis of multiple statements that a statement did not necessarily belong to a single category but could instead be classified by more than one category, as was discussed in Section 2. This subtle refinement meant that the initial information extraction approach (e.g., the automatic statement classification part) could not be applied anymore and a new solution had to be developed. In this paper, we recognize how this update allows RSL-IL4Privacy to better express real-world privacy policies, and we leave the challenge of automatically classifying statements against multiple categories to future work.

The files containing the multiple representations for each policy that was once again analyzed under the aforementioned refinements are available on a public repository[5]. The following sub-sections discuss the analysis of these privacy policies using RSL-IL4Privacy and then we detail one of these policies validated against this proposal.

Firstly, it is important to show and discuss the results from the conflict detection process for all the policies that were analyzed. On the other hand, by specifying these five policies using RSL-IL4Privacy, one is able to determine differences and similarities across them in terms of structure but

[5] https://github.com/RSLingo/RSLingo4Privacy

also in terms of their content. Lastly, detailing one of these five policies provides better insight regarding the use of RSL-IL4Privacy as a language for specifying privacy policies. With only one case study it is easier to explain the various models and differences between representations. This case study considers Twitter's privacy policy[6], wherein a proper set of relevant statements (i.e., statements from all the different types, including conflicting statements) is selected and analyzed, thus producing different models that correspond to the various concrete representations. To make the tables easier to read, the relationships between elements are described by listing all data in the same row, correlating them by their attribute id.

5.1 Analysis of Five Popular Privacy Policies

Aside from the eventual conflicts that may arise from the reasoning process described further, it is also important to point out different aspects among policies.

5.1.1 A Broader Comparison of the Privacy Policies

We analyzed the policies individually and as independent tasks. This analysis was performed manually from the original version of the publicly available policies. For each policy, we got all its statements in plain text, then we identified and classified their elements based on the RSL-IL4Privacy constructs (statement types, private date, services, etc.), and we established the respective relationships. Finally, and based on this human-intensive analysis, we compared the results as follows, supported by both the tabular and textual representations.

Having all privacy policies documented with the same structure and style makes it easier to compare the original policies in terms of their size, lexicon, data elements and recipients. Table II depicts the number of instances of the different RSL-IL4Privacy elements that were defined for each of these five privacy policies. It also presents some general textual information, such as number of characters, words and sentences. We decided to omit the information related to the *Service* element due to the additional human effort that is needed to generalize and define each element of this kind (when compared to the remaining elements).

Table II. Comparison of the five privacy policies based on RSL-IL4Privacy

Privacy Policy	**Textual Analysis**			**RSL-IL4Privacy Analysis**			
	Characters	**Words**	**Sentences**	**Statements**	**Recipients**	**PrivateData**	**Enforcements**
Dropbox	5819	945	46	42	7	8	4
IMDb	12789	2063	95	76	9	10	10
Facebook	14000	2293	93	88	11	8	5
LinkedIn	21484	3393	139	105	10	17	14
Twitter	18929	3022	127	117	8	26	9

Despite the differences in the number of statements, there is not a significant distinction in the number of *Recipient* elements. However, all the five policies lack a clear definition of these recipients, more specifically in terms of scope and hierarchy. It is not clear what the relationship between the recipients is (if any) and it is also difficult to grasp if these recipients are internal or external to the service provider (e.g., naming recipients as "partners" is both ambiguous and unclear). This implies that the identification of Recipient elements requires a good deal of human reasoning.

[6] https://twitter.com/privacy?lang=en

Twitter's policy has the highest number of *PrivateData* elements (i.e., 26). However, many of these elements only have one attribute (e.g., name, location). At the same time, and even though Facebook's privacy policy is twice the size of Dropbox's policy, both have the same number of *PrivateData* elements (8). The major difference is about the elements that have some degree of similarity. Table III summarizes the differences between equivalent *PrivateData* elements by providing a comparison of two *PrivateData* elements ("Account Information" and "Payment Information") that convey similar information but have different definitions across all policies. Even though they ought to describe the same bit of information, these elements differ in a lot of their attributes. For example, the *PrivateData* element that represents the user account on Facebook, "Facebook Account", encompasses seven attributes, whereas "Dropbox Account" only lists five. Additionally, "Payment Info" is included as a vague attribute in the "Dropbox Account"; Facebook's policy defines "Payment Information" with details like "credit or debit card number" and "billing, shipping and contact details". Twitter also provides a very similar definition of a "Payment Information" element with roughly the same attributes. LinkedIn, however, only includes the "credit card details" as information related to payments or purchases. Dropbox and IMDb do not define this element. On the other hand, Facebook, LinkedIn and Twitter have common attributes for the "Account" (e.g., "name", "email", "mobile/phone number", "password") but Facebook also includes the "date of birth" and "gender" as account information while, for instance, the remaining policies for social networking websites (LinkedIn and Twitter) do not. Finally, almost all privacy policies provide a generic term for the *PrivateData* element that is described at a given point. However, sometimes the name of an element of this type is defined based on the context or the list of attributes that describes it. IMDb is the only one that provides the definition of some of the *PrivateData* elements through a hyperlink to an external page.

The *Enforcement* element is the one that shows the largest variation in number. Regardless, there are some similar elements between policies. Twitter and IMDb provide some rules about the use of their services by children, whereas LinkedIn and Dropbox state they will not sell user information to third-parties for purposes not described in their policies.

In addition to the differences and similarities between RSL-IL4Privacy elements, there are also observations regarding textual details. As stated in section 2.1, one statement can have more than one sentence. This property holds as the number of sentences in every analyzed privacy policy is higher than the final number of statements. At the same time, and when comparing two privacy policies (e.g., IMDb and Facebook), having a policy with more sentences does not necessarily mean a policy with more statements. Despite having a higher number of sentences (95 vs. 93), IMDb has less statements than Facebook (76 vs. 88). This is due to the fact that IMDb's privacy policy contains more sentences that are dependent on the previous or the next sentences. These sentences are then combined into a single statement. The same also happens for the case of LinkedIn and Twitter.

Finally, when considering each privacy policy in their original version and from an end-user point of view, one can easily identify differences in the way it is written and structured. For example, Facebook has a very well-organized and comprehensible policy. The writing style is concise and the discourse quite direct. Its content is divided into several sections and sub-sections (highlighted with different colors and icons) and the information regarding the major data practices (e.g., Collection, Disclosure, and Usage) is clearly identified and separated. On the contrary, LinkedIn (and also IMDb) has a more detailed privacy policy and the way it is structured makes it hard to read. Each section provides a lot of text where the information about data practices and data flows is surrounded by plenty generic content. Moreover, LinkedIn and Twitter provide an introduction about the content of their policies (e.g., overviewing the mission, explaining some basic concepts and/or listing the companies/platforms to which the policy applies), whereas the remaining companies choose not to. On the other hand, besides IMDb, all the others have a separate policy that described in more detail the information related to the Cookies.

Table III. Comparison of equivalent PrivateData elements

Privacy Policy	**PrivateData**		
	name	**type**	**attributes**
Dropbox	Dropbox Account	Personal Information	name, email, phone number, payment info, physical address
Facebook	Facebook Account	Personal Information	first name, surname, email, mobile number, password, date of birth, gender
	Payment Information	Personal Information	credit or debit card number, other card information, account and authentication information, billing, shipping and contact details
IMDb	IMDb Account	Personal Information	name, e-mail address, physical address, zip code, phone number, your age, gender, the movies and actors you like or dislike, and your general movie preferences
LinkedIn	LinkedIn Account	Personal Information	name, email, mobile number, password
	Purchases Information	Personal Information	credit card details
Twitter	Twitter Account	Personal Information	name, username, password, email address, phone number
	Payment Information	Personal Information	credit or debit card number, card expiration date, CVV code, billing address

5.1.2 Statements in Detail

The results obtained using Eddy's deontic conflict detection mechanism with equivalent RSL-IL4Privacy specifications for the policies of Dropbox, Facebook, IMDb, LinkedIn and Twitter are summarized in Table IV. These results account for the number of deontic conflicts between permitted and prohibited actions. If one statement describes a prohibition regarding the handling of some information (e.g., "We will not use your e-mail address for advertising purposes") and then other statement discusses permitted actions for the same data, a conflict occurs.

Table IV. Results using Eddy reasoner engine for conflict detection with converted RSL-IL4Privacy specifications

Privacy Policy	**Number of Deontic Conflicts Detected**
Dropbox	0
Facebook	2
IMDb	6
LinkedIn	1
Twitter	12

Twitter is the system that shows the highest number of deontic conflicts (12). Half of the conflicts occur due to the disclosure of personal information, namely Payment Information (e.g., credit card details) or the user's address. Figure 4 depicts some of the conflicting statements in RSL-IL4Privacy using its textual representation. By stating “We consider your payment information and user shipping address private and do not make such information public” (see the statement S22 in Fig. 4) one can safely argue that Twitter is concerned with the improper disclosure of information. Therefore, “not making information public” can mean not sharing any types of personal data with anyone under any circumstances. However, several other statements – in this case, S28 and S34 – describe the disclosure of the user's personal information (including the discussed types) to multiple recipients and for a variety of reasons (e.g., security, processing of transactions). On the other hand, Twitter's privacy policy also shows slightly different conflicts. Even though this social network clearly states that its “Services are not directed to persons under 13” (a business rule defined as an *Enforcement* element in RSL-IL4Privacy), the collection and retention of those users’ private information is still carried out until the provider becomes aware of such fact. Conflicts occur when this information – that should not be gathered in the first place and is still undetected – is used and processed by Twitter afterwards. More than pointing out eventual conflicts that may be then used to produce revised versions of the policy, these warnings should trigger Twitter to develop mechanisms for preventing children from using their Services or, on the upper hand, to provide a better assessment before processing the users’ information.

Facebook policy’s conflicts are also due to the inappropriate disclosure of user information. We are told that Facebook does not share personal information (i.e., information “that can by itself be used to contact or identify users”) with advertising, measurement or analytics partners, unless the user gives permission. However, its policy states that Facebook shares information (due to a vague definition of this concept, we need to assume that it includes also the users’ personal information) with a variety of recipients, such as “partners that support their business”. It is safe to say that advertising and performance measurement partners also support the business of Facebook, therefore being eligible for receiving a wide range of information – and with no explicit user consent. At the same time, Facebook also declares that it shares information internally (as one might expect) but also with external third parties for the purposes described in its policy. The unclear definition of the purposes for the disclosure of information leads to inconsistencies. These two examples show conflicts in terms of who is allowed or not to receive information (possibly including users’ private information) but also why is necessary for the company to share such information. The conflicts that occur in LinkedIn and IMDb policies are similar to the aforementioned conflicts, therefore being pointless to discuss them.

Lastly, Dropbox is the only policy that showed no conflicting statements during our analysis. Its policy is relatively small when compared to the policies of the other service providers and none of the statements explicitly states any kind of prohibition in terms of data practices (i.e., no statements with "forbidden" modality). Thus, as we pointed out, if the focus of our analysis in this paper is the detection of "permitted-forbidden" conflicts, it makes sense that no inconsistencies were found.

```
statement S22 : Disclosure {
   name "S22"
   modality Forbidden
   description "We consider your Payment Information and shipping address private and do not
make such information public."

   recipient R1-…-R8,
   privateData PD7-PD8,
   service SV1-…-SV20
};

statement S28 : Disclosure {
   name "S28"
   modality Permitted
   description "We share your Payment Information with payment services providers to process
payments.",

   recipient R5,
   privateData PD7,
   service SV6
};

Statement S34 : Disclosure {
   name "S28"
   modality Permitted
   description "If you buy goods or services through our Services, we may provide the
seller, commerce provider or marketplace with your name, email address, shipping address,
Payment Information and Transaction Data to help prevent, detect and investigate fraud or
other prohibited activities.",

   recipient R7,
   privateData PD7-PD8-PD10-PD18-PD19
   service SV10
};
```

Fig 4. Textual representation for Twitter's conflicting statements S22 with S28 and S22 with S34 (partial view)

5.2 Policy Specification with Multiple Representations

We manually selected a set of representative policy statements from the Twitter privacy policy to evaluate the application of RSL-IL4Privacy. Table V details these statements and shows the associations between the remaining elements, which were previously identified and defined in the metamodel. In this particular case, six statements have associations with *PrivateData* and *Service* elements, whereas only the disclosure-type statements are related to a certain *Recipient* as elicited on the metamodel. On the other hand, the informative statement presented in the Table V is the only one that is related to an *Enforcement* element. Even though an informative statement does not provide critical information regarding users' private data, it very often encompasses other important details that are expressed in RSL-IL4Privacy by the *Enforcement* element. The statement with id 112 (S112) illustrates such characteristic. Finally, despite the fact that the disclosure-type statements were already introduced in the previous section while discussing their conflicts, we also included them as running example for the tabular representation. These statements are easy to understand, provide important information and the tabular representation allows us to enrich the previous explanation with the complete description of the complementary elements.

Table VI comprises the remaining information that is needed to complete the analysis of the statements in this tabular representation, supported by MS-Excel. With all this information, the specification of a privacy requirement or statement is simple and almost effortless. Due to space constraints, only four *PrivateData* and two *Service* (with two sub-services) elements are described in Table VI. We apply the previous premises with an example, i.e., we describe the process of classifying just one of the statements, namely the S28. S28 is a Disclosure statement which means that it explicitly defines which information is eventually shared to other external entities or third-parties. Although not mandatory, it can also include the reason why such information is disclosed.

Firstly, by examining the description of statement S28, it specifies that Twitter shares "Payment Information" which we can consider the personal information related to processing payments or transactions and that can be used to identify a given user. Such information is defined as a *PrivateData* element, in this case, with id 7 (P7). This disclosed information is obviously named as "Payment Information" and includes the credit or debit card number and other card information. P7 is listed in Table VI. At the same time, dealing with this type of statement requires a characterization of the external entity to whom the information is disclosed.

From the statement itself it is possible to spot the entity beforehand and by taking a look at Table VI, one is able to complete such analysis. The mentioned *Recipient* element is described as "payment services providers" which means it consists of one or more organizations and, on the other hand, these third-parties are external to Twitter. Lastly, this *Recipient* element has assigned an id of 5 (R5).

Finally, since the purpose of disclosure is due to the processing of payments, purchases, etc., we define a *Service* element named "Transactions" which includes all activities related to them. This service, SV6, is a sub-service of a more generic service called "Financial" that includes not only the "Transactions" but also other related services such as "Card Services".

As discussed earlier in this paper, and in order to prove the flexibility and expressiveness of the proposed language, Fig. 5 shows an object diagram for statement S28; and Fig. 6 shows an equivalent textual representation for the same statement S28, therefore these figures show different representations for the same discussed scenario. Being the representation with the highest level of formality, its concrete syntax makes it suitable for the further processes regarding the analysis of a policy. The RSL-IL4Privacy editor (included in the RSLingo4Privacy-Studio tool) provides the syntactic validation of these textual specifications in RSL-IL4Privacy. While creating a new or editing a given RSL-IL4Privacy specification, the editor continuously validates the traceability between these core elements (e.g., *Recipient* elements must be linked to a disclosure-type *Statement* element). This syntactic validation is useful to ensure the correctness of these specifications to perform some kind of analysis afterwards.

Table V. Twitter Case Study: Set of statements and associations between different elements

id	description	type	PrivateData (id)	Service (id)	Enforcement (id)	Recipient (id)
2	You may provide us with profile information to make public on the Twitter Services, such as a short biography, your location, your website, or a picture.	Collection	2	3		
22	We consider your Payment Information and shipping address private and do not make such information public.	Disclosure	7, 8	All		All
28	We share your Payment Information with payment services providers to process payments.	Disclosure	7	6		5
34	If you buy goods or services through our Services, we may provide the seller, commerce provider or marketplace with your name, email address, shipping address, Payment Information and Transaction Data to help prevent, detect and investigate fraud or other prohibited activities.	Disclosure	7, 8, 10, 18, 19	10		7
45	We may store information about your location to improve and customize the Services.	Retention	3	20		
71	Twitter uses Log Data to make inferences, like what topics you may be interested in.	Usage	5	20		
112	Our Services are not directed to persons under 13.	Informative			3	

Table VI. Twitter Case Study: Relevant information about the remaining elements

Recipient

id	name	type	scope	PartOf (id)
4	Service Provider	Organization	Out	
5	Payment Service Provider	Organization	Out	4
7	Seller, Commerce Provider or Marketplace	Organization	Out	

PrivateData

id	name	type	attributes
2	Public Profile	Personal Information	short biography, location, website, picture
3	User Location	Personal Information	location
5	Log Data	Usage Information	IP address, browser type, operating system […]
7	Payment Information	Personal Information	credit or debit card number, card expiration date, CVV code, billing address

Service

id	name	type	description	PrivateData (id)	PartOf (id)
5	Financial	Compound	all issues related to financial reasons	All	
6	Transactions	Simple	purchases, process payments, acceptance of credit or debit cards […]	All	5
17	Development	Compound	all issues related to the services development	All	
20	Service Customization	Simple	customize the Services, customize content	2, 3, 5, 15, 16, 17, 18, 21, 25	17

Enforcement

id	name	type	description
	Children Use Recommendation	Action	People under 13 years old should not use Twitter services.

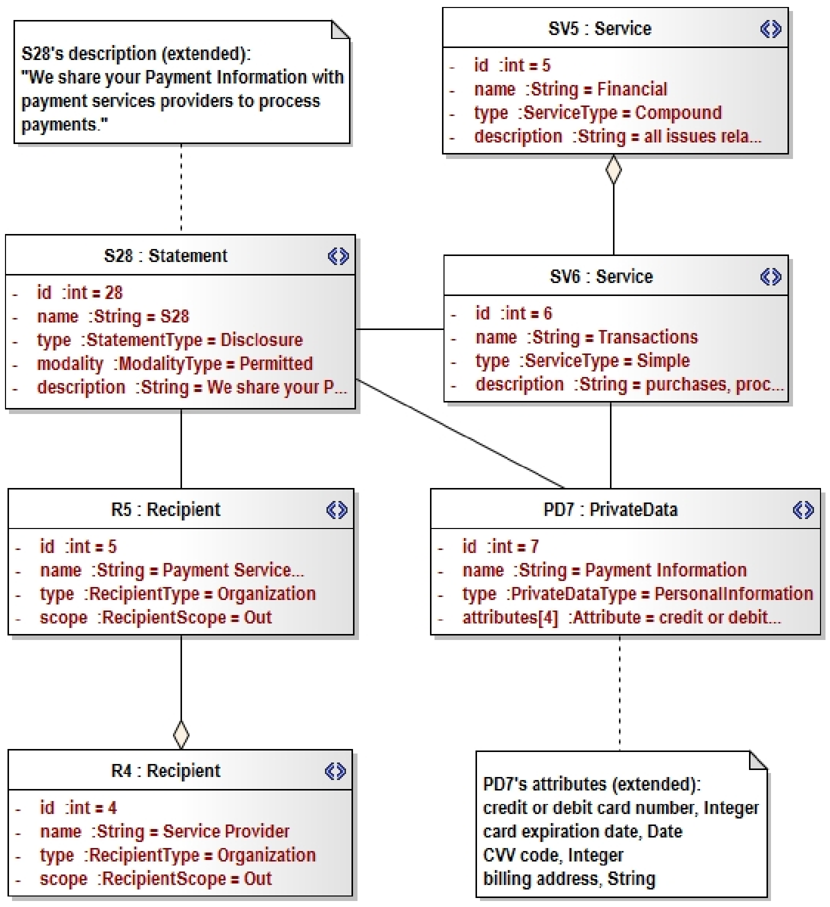

S28's description (extended):
"We share your Payment Information with payment services providers to process payments."
SV5 : Service
- id :int = 5
- name :String = Financial
- type :ServiceType = Compound
- description :String = all issues rela...
S28 : Statement
- id :int = 28
- name :String = S28
- type :StatementType = Disclosure
- modality :ModalityType = Permitted
- description :String = We share your P...
SV6 : Service
- id :int = 6
- name :String = Transactions
- type :ServiceType = Simple
- description :String = purchases, proc...
R5 : Recipient
- id :int = 5
- name :String = Payment Service...
- type :RecipientType = Organization
- scope :RecipientScope = Out
PD7 : PrivateData
- id :int = 7
- name :String = Payment Information
- type :PrivateDataType = PersonalInformation
- attributes[4] :Attribute = credit or debit...
R4 : Recipient
- id :int = 4
- name :String = Service Provider
- type :RecipientType = Organization
- scope :RecipientScope = Out
PD7's attributes (extended):
credit or debit card number, Integer
card expiration date, Date
CVV code, Integer
billing address, String

Fig 5. Object diagram for Twitter's statement S28 (partial view)

```
statement S28 : Disclosure {
   name “S28”
   modality Forbidden
   description “We share your Payment Information with payment services providers to process payments.”
   recipient R5,
   privateData PD7,
   service SV6
};

recipient R5 : Organization {
   name “Payment Service Provider”
   scope Out
   partOf R4
};

recipient R4 : Organization {
   name “Service Provider”
   scope Out
};

privateData P7 : PersonalInformation {
   name “Payment Information”
   attribute A1 : Integer {
      name “credit or debit card number”
   };
   attribute A2 : Date {
      name “card expiration date”
   };
   attribute A3 : Integer {
      name “CVV code”
   };
   attribute A4 : String {
      name “billing address”
   };
};

service SV6 : Simple {
   name “Transactions”
   partOf SV5
   description “purchases, process payments, acceptance of credit or debit cards […]”
   refersToPrivateData PD1-PD26
};

service SV5 : Compound {
   name “Financial”
   description “all issues related to financial reasons”
   refersToPrivateData PD1-PD26
};
```

Fig 6. Textual representation for Twitter’s statement S28 (partial view)

6. RELATED WORK

The analysis of privacy requirements and privacy policies has been surveyed by several researchers, as reported and discussed in [32-34]. To provide a common understanding of the concepts that comprise such policies some authors have suggested that the definition of privacy-aware languages, with respective tool support, can help stakeholders to better define, implement or understand these privacy-related concerns [3, 5, 13-21, 27]. For example, policy authors or requirements engineers have the ability to specify and author policies using (controlled) natural language approaches; developers can interpret these rigorous specifications as they are similar to other programming languages' syntaxes that their master, and so they can avoid misinterpretation and misalignment with the system implementation; and end-users can view and understand these policies in an easy and systematic way, eventually supported by multiple representations (e.g., textual, graphic, tabular) and multiple views (e.g., by crossing the various practices and actions by private data).

These approaches emphasize the importance of having a well-defined language that may provide the necessary concepts for a convenient privacy policy specification. We analyze and classify these proposals into two perspectives, namely: access control based policy languages; and semantic web based policy languages. We summarize in Tables VII and VIII the RSLingo4Privacy features and contrast them with the related approaches, mainly taking into account the analysis of their specific languages (Table VII) and the tool support they offer (Table VIII).

6.1 Access Control Policy Languages

Privacy approaches from the perspective of access control have focused on the definition and enforcement of authorization mechanisms with respect to disclosed user information. Popular approaches for defining website's data privacy based on such policies are the ones involving the standards P3P [15], XACML [16] and EPAL [17]. These languages appeared as an attempt to automate the data management practices of a website, due to their XML-based syntax that could be more easily processed by computers. However, their application domain tends to be broader than the privacy policies, which is the focus in our work. For instance, P3P/APPEL includes in addition the users' privacy preferences definition, while XACML and EPAL are more focused on access control aspects. Even they address slightly different aspects, when compared with the RSL-IL4Privacy, they deserve to be mention because they can be used complementary.

W3C's P3P (Platform for Privacy Preferences Project) [15] and ***APPEL (A P3P Preference Exchange Language)*** [19] are XML-based languages used for privacy definition and negotiation between a web site and its users, including the collection of user's preferences. By using P3P web sites can express their privacy policies that can be retrieved automatically and interpreted easily by user agents [18]. Web sites may declare in P3P why they need users' personal data, what data they collect, how long they will retain them, and who will use these data, etc. These elements are standardized in a policy, which is applied to a specific set of data resources. Then the web users can be informed of web sites' data-collection policies, which are encoded in the machine-readable XML format, with the help of a user agent such as P3P-enabled web browsers. In this way the user can rely on his or her agent to read and evaluate web sites' policies, and to further opt-in or opt-out data privacy related decisions. The user can use APPEL to express his preference rules, and then his user agent can make automated or semi-automated decisions on accepting or not accepting web sites' machine readable privacy policies from P3P-enabled web sites, by comparing web sites' declared practices with web user's preference rules. These decisions could be given out by simply informing the user, or prompting him for a decision.

eXtensible Access Control Markup Language (XACML) [16] is a declarative, XML-based policy language, mainly for access control management in distributed systems. XACML defines a declarative fine-grained, attribute-based access control policy language, but also includes the definition of an architecture and a processing model describing how to evaluate access requests according to the rules defined in its policies. XACML promotes common terminology and interoperability between access control implementations by different vendors. XACML is primarily an attribute-based access control (ABAC) system, where attributes associated with a user or action or resource are inputs into the decision of whether a given user may access a given resource in a particular way. The popular and simple role-based access control (RBAC) can also be implemented in XACML as a specialization of ABAC.

Enterprise Privacy Authorization Language (EPAL) [17] is another XML-based language proposed by IBM (its latest version EPAL1.2 was submitted to W3C in 2003 for consideration as a W3C standard, but has not been approved since then). EPAL is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems according to fine-grained authorization rights. EPAL concentrates on the core privacy authorization while abstracting data models and user-authentication from all deployment details such as data model or user authentication. An EPAL policy defines lists of hierarchies of concepts such as data-categories, user-categories, purposes, actions, obligations, and conditions. User-categories are the entities (users/groups) that use collected data. Data-categories define different categories of collected data that are handled differently from a privacy perspective. Purposes model the intended service for which data is used. Actions model how the data is used (e.g., disclose, read, share). Obligations define actions that must be taken by the environment of EPAL (e.g., delete after 30 days). Conditions are Boolean expressions that evaluate the context (e.g., "the user-category must be an adult"). These elements are then used to formulate privacy authorization rules that allow or deny actions on data-categories by user-categories, for certain purposes under certain conditions while mandating certain obligations, i.e. following a general rule like "ALLOW [Data User] TO PERFORM [Action] ON [Data Type] FOR [Purpose] IF [Condition] AND CARRY OUT [Obligation]". Backes et al. [35] reported some tools to implement and enforce EPAL, and some degree of policy refinement and conflict resolution was also considered.

SPARCLE [27, 36] is a privacy policy workbench to support privacy policy authoring, implementation and compliance monitoring. SPARCLE allows users to specify a privacy policy using a controlled natural language (NL) or using a structured format (tabular and form-based) based on a set of predefined rules. These rules are combination of the following concepts similar to EPAL language: user categories, actions (e.g., read, use or modify), data categories, purposes, conditions and obligations, such as "[User Category] CAN [Action] ON [Data Category] TO [Purpose] IF THE [Condition or Obligation]". SPARCLE tool supports the transformation of privacy policies written in this controlled NL into its structured rule-based representation (and vice-versa), as well as the transformation to a machine-readable format like XACML for use by an enforcement engine.

Several tools support the creation of policies written in P3P, XACML and EPAL, but some of them were discontinued [37]. IBM P3P Policy Editor is a proprietary tool that permits creating from a template or editing a website's privacy policy using a drag-and-drop graphical user interface (GUI) [18]. JRC Policy Workbench is an open-source tool that provides a GUI for creating, managing and testing P3P policies through a form-based policy editor where the user configures and fills some input fields by following wizards. This editor provides other useful features like viewing the corresponding XML structure in a tree view, a human-readable summary of the policy and the possibility to carry out tests with the default APPEL configuration. The JRC Policy Workbench provides an extendable API for building editing and testing environments for other types of XML-based privacy and access control policies like XACML or EPAL. P3PEdit website allows the generation of a P3P policy by following a web-based wizard. All these tools abstract the XML syntax, but users still need to properly understand these languages' concepts and how these apply to their website. Due to the insufficient support and adoption from Browser implementers, P3P/APPEL did not succeed. However, in spite of this situation, P3P keeps being the basis of a number of research initiatives in the area of privacy worldwide [15].

6.2 Semantic Web Policy Languages

Other approaches [5, 20, 21, 24] have focused on the definition of privacy policies as ontologies, based on semantic web techniques, and using formal knowledge representation languages such as OWL[7] or its predecessors such as DAML[8]. These approaches provide semantic reasoners that use deontic logic [26] and that, consequently, can determine if a policy is consistent and can check if there are conflicts among their rules. In spite of their interest due to the possibility to automatically detect such quality issues, the privacy policies represented by these formal languages tend to be compact and hard to be readable by humans.

As described in Section 4, ***Eddy*** [5] is a formal language defined for specifying privacy and related data flow requirements, which allows to trace data flows and to check for potential requirements conflicts. The specification of privacy requirements takes into account three main concepts that can be structured hierarchically: datum, actor and purpose. The Eddy approach consists of mapping requirements (that include those concepts) written in natural language text to actions (e.g., collection of data) and roles (i.e., relationships between datum, actor and purpose) in OWL-DL (Ontology Web Language – Description Logic) [10]. Then, the use of OWL-DL makes it possible to detect deontic conflicts. Eddy's website[9] provides examples that using its editor any user can specify a privacy policy in a free text area. Then, it runs the Eddy engine to analyze the policy for detecting any possible conflicts or for tracing the flow and showing it as a network chart. There is also an option to export the equivalent OWL file resulting from the analysis. The Eddy engine code and some examples of its invocation using Java are publicly available on Eddy's GitHub repository.

KAoS policy, as reported by Uszok et al. [20], is a language for policy specification originally represented in DAML but then in OWL. A KAoS policy definition is based on general concepts such as actor/agent, modality, action and context, and it addresses security concerns such as authorization, encryption, resource and access control. The modality corresponds to two main types: authorization and obligation rules. The set of permitted actions is determined by authorization rules that specify which actions an actor is allowed or not allowed to perform in a given context. On the other hand, obligation rules specify actions that an actor is required to perform (positive obligations) or for which such a requirement is waived (negative obligations). Other kinds of policies (e.g., delegation, teamwork coordination) can be defined from these two primitive types, combined with other aspects of KAoS policy semantics. Uszok et al. also reported their preliminary experiences with the implementation of KPAT (KAoS Policy Administration Tool), a graphical tool that allows users to specify, analyze, modify and test KAOS policies. KPAT also detects policy conflicts and allows managing sets of ontologies. KPAT offers a set of views of KAoS concepts such as domains, actors/agents, policies. Since the policies are specified using that tool, the corresponding DAML/OWL representations can be generated automatically using a generic template, avoiding its users to master DAML/OWL. Other language-specific templates or domain-specific templates for common classes of policies can be defined. KPAT also offers a wizard to guide the user throughout the policy creation process.

[7] https://www.w3.org/OWL

[8] http://www.daml.org/

[9] https://gaius.isri.cmu.edu:8080/eddy

Rei policy [21] is another logic-based language based on concepts like data resources, services, actors, actions, provisions/obligations, and constraints/conditions; and it also relies on the deontic concepts of rights, prohibitions, obligations and dispensations. Beside the general purpose text editors, there are some tools that support the specification of Rei policies. The first tool is a plugin for the Protégé-2000 ontology editor that provides features for creating policies, rules, meta-policies and queries through a custom tab and dialog boxes. The second is a text-based editor for specifying policies for Rei using the Notation3 (N3) language [22], in order to make the policies easier to read. This editor provides content assistance and context information while a user is typing a Rei policy. Finally, RIDE (Rei Integrated Development Environment) [23] is an Eclipse plug-in that uses a wizard-based approach. The creation wizard guides the creation of a policy and in the end automatically generates the corresponding policy file in OWL, based on the user input and selections. Once the policy file is created, the user can launch the test wizard that provides an interface for testing the policy and querying the Rei engine.

STS-ml [24] is a graphical language defined as a Tropos extension for modeling and reasoning about security concerns in socio-technical systems. STS-ml language supports a social and information view, including flows of information between agents and processes. The user can describe intentional relationships, including that an actor wants to achieve a goal that they possess data or read, modify and produce data, and that goals can be decomposed into sub-goals. In addition, the user can specify a number of properties, such as authorizations, delegation, non-repudiation, separation of duties, among others, and then automatically check whether the specifications respect those properties. It is supported by the STS-Tool[10] that allows STS-ml modelling through different views (social, information, and authorisation), and provides features such as inter-view consistency, requirements document generation, and automated reasoning for conflict detection. This tool encodes STS-ml models into disjunctive Datalog programs to support the automated reasoning [25].

6.3 Other Related Researches

Other researches are important to be mentioned because they introduced additional challenges [28-31] or because they provided a complementary understanding of our analysis by having surveyed in extensive way the area of policy languages [32] and privacy requirements engineering [33, 34].

Wishart et al. [28] proposed a privacy-aware social networking service and a collaborative approach to authoring privacy policies. That approach takes into account the needs of all parties affected by the disclosure of information and digital content. They developed the PRiMMA-Viewer tool that supports the collaborative specification of privacy policies for shared content on Facebook. PRiMMA-Viewer uses an architecture aligned with the Policy Core Information Model (PCIM) [29].

Nadas et al. [30] presented another model-based policy authoring framework called PATRN applied to the health information systems domain. PATRN uses a combination of a graphical language, for policy authoring, and the FORMULA specification language (based on logic programming) for policy analysis.

Winkler and Zeadally [31] analyzed privacy policies from four of the most popular web platforms (i.e., Google, LinkedIn, Twitter, and Facebook), compared and contrasted such policies to identify the platform that gathers the most information and the platform that offers the most information privacy for the user, analyzed the privacy policies to determine what the average user can comprehend, and they also recommended good practices for users to increase their privacy in spite of these policies.

Han and Lei [32] analyzed the researches and development of policy languages in the context of policy-driven management. In particular, they discussed these languages from two perspectives:

[10] https://www.sts-tool.eu

network management policy languages and security management policy languages. The former focused on how to drive the network devices and resources to meet system requirements, e.g. service level agreement assignments, whereas the latter focused on the protection of system resources, including users' privacy, as is the focus of our work.

Gharib et al. [33] proposed an extensive ontology for privacy requirements derived from their systematic literature analysis. This ontology includes many concepts found in the domain of security requirements engineering such as actor, threat, or vulnerability; and it also includes other privacy-specific concepts, like personal information, privacy goal or privacy requirement. However, this ontology does not directly include key concepts specific to privacy policies (e.g., it does not provide privacy actions or statements such as collection, disclosure or share) and, in addition, it is too much extensive and complex to be used as a starting language to define privacy policies as proposed with the RSL-IL4Privacy language.

Anthonysamy et al. [34] analyzed recently the area of privacy requirements engineering from four perspectives: compliance, usability, access control, and verification. According to the *perspective of compliance* privacy approaches have mainly consisted of deriving privacy requirements from data protection legislation. The focus of these approaches has been on eliciting and analyzing requirements that are necessary to make systems data protection legislation compliant [38-40]. These methods have used the theoretical frameworks [41, 42] and security/privacy standards frameworks to elicit the privacy requirements [39, 43] and model privacy expectations and practices [44]. Other approaches have focused on defining traceability relationships between various artifacts like legal documents, requirements and source code [45] and identifying inconsistencies in natural language software requirements for a successful software system development [46]. On the other hand, privacy approaches from the *perspective of usability* have focused on the evaluation and understanding of user behaviours, needs, and motivations through analysis of usability problems of existing privacy solutions. This perspective has involved a wide spectrum which includes user studies on privacy perceptions [47, 48], privacy breaches in social media [49], and improvement of user awareness [50, 51]. In addition, their two other categories – privacy approaches from the *perspective of access control* and from the *perspective of verification* – were further discussed in the subsections above because they are the most related with the scope of our paper.

6.4 Comparison

Tables VII and VIII compare the RSLingo4Privacy with the related approaches, mainly taking into account the analysis of their specific languages (Table VII) and the respective tool support they offer (Table VIII). Table VII compares the privacy-aware specification languages, in particular on the following aspects: application domain, key concepts, abstract and concrete syntaxes support, and semantics support. On the other hand, Table VIII compares the respective tool support regarding the following features: visualization and authoring, text to model (T2M) transformations, model to model (M2M) transformations, model to text (M2T) transformations, publishing, and authorization enforcement features. There are some aspects of this analysis that should be highlighted.

First, in spite of the differences and subtleties in their adopted terminology, in general all these languages share a common set of key concepts, namely: Data, Actor, Action and Service.

Second, some languages (e.g., P3P, XACML, EPAL) are mostly targeted to the specification of access control and privacy of web applications (P3P) or IT services and applications in general (XACML or EPAL); but the specifications produced with these languages follow a low-level XML format and so, they are not appropriated to be used directly by humans. This same happens with the other group of languages (e.g., Eddy, KAoS or Rei), mostly concerned with formal and compact representation of such policies.

Third, the closest approaches to RSLingo4Privacy, in what concerns the qualities of its policies being simultaneously used by humans and machines, are the SPARCLE and STS. However, to our knowledge, none of them was applied and validated with popular web sites' policies (such as Facebook, LinkedIn, Twitter, Dropbox and IMDb) as we did in our research.

Fourth, as it is suggested in Table VIII, each language is supported by software tools that provides at least authoring and visualization features. Some tools (e.g., XACML, EPAL and KAoS) include authoring enforcement engines and others (e.g., SPARCLE, STS, and RSLingo4Privacy) support publishing the policies in an easy-to-read NL format. In addition, RSLingo4Privacy is the tool that most supports multiple model transformations (e.g., T2M, M2M, and M2T), as illustrated in Figure 3, which is a consequence of its inherent "*interlingua*" (i.e., "intermediate language") nature.

Table VII. Comparison of privacy-aware specification languages

Language	**Domain**	**Key Concepts**	**Abstract Syntax,** defined as a…	**Concrete Syntax,** represented by…	**Semantics**
P3P/APPEL	Web Privacy	Data Collection, Statement, Practice, Service, User/Recipient, User Preference	XML Schema	Textual	Declarative
XACML	Access Control	Resource, Attribute, Rule, Action, Subject, Target	XML Schema	Textual	Declarative
EPAL	Access Control	Data Category, User, Action, Purpose, Condition, Obligation	XML Schema	Textual	Declarative
SPARCLE	Privacy	Data category, User category, Action, Purpose, Condition, Obligation	Grammar	Textual	Declarative
Eddy	Privacy	Datum, Actor, Action, Purpose	Grammar	Textual	OWL
KAoS	Generic	Actor/Agent, Modality, Action, Context	XML Schema	Textual	OWL
Rei	Generic	Data and Service, Actor, Action, Obligation, Condition	Prolog* constructs	Textual	OWL
STS-ml	Security	Data, Actor, Goal, Security property	UML Profile (Tropos extension)	Graphic	Datalog
RSL-IL4Privacy	Privacy	Private Data, Recipient, Statement, Service, Enforcement	UML profile + Grammar	Graphic + Textual	Declarative

Table VIII. Comparison of tools that support privacy-aware languages

Approach	**Languages**	**Tool Support**					
		Visualization & Authoring	**Transformations**			**Publishing**	**Authorization Enforcement**
			T2M	**M2M**	**M2T**		
P3P/APPEL	P3P/APPEL	IBM P3P, JRC Policy Workbench, P3PEdit	No	No	To HTML	No	No
XACML	XACML	UMU XACML Editor	No	No	To HTML	No	Yes
EPAL	EPAL	Privacy Authoring Editor, EPAL Editor	No	No	To HTML	No	Yes
SPARCLE	SPARCLE format	SPARCLE Policy Workbench	From CNL	XACML	To CNL	Yes	Yes (via XACML)
Eddy	Eddy	General-purpose text editors	No	Yes	No	No	No
KAoS	KAoS	KPAT	No	No	No	No	Yes
Rei	Rei (OWL-based)	Protégé-Plugin, N3 text editor, RIDE	No	No	No	No	No
STS	STS-ml	STS-Tool	No	No	Yes	Yes	Yes
RSLingo4 Privacy	RSL-IL4Privacy	RSLingo4Privacy Studio (Eclipse Xtext-based Plugin)	From Ad-hoc NL	Excel, JSON and Eddy	To Word and CNL	Yes	No

7. DISCUSSION

RSL-IL4Privacy is a language designed to provide a clear specification of privacy policies and that acts as an intermediate language between natural language and other controlled or more formal languages (as suggested in Figure 7). Its goal is to allow the specification of privacy-aware requirements bearing in mind the importance of natural language. The base of this approach is to map the original policy (written in natural language) into a renewed privacy requirements specification where each requirement represents a statement in the original policy, i.e., each privacy requirement holds the content of the one or more sentences that can be found on the original policy.

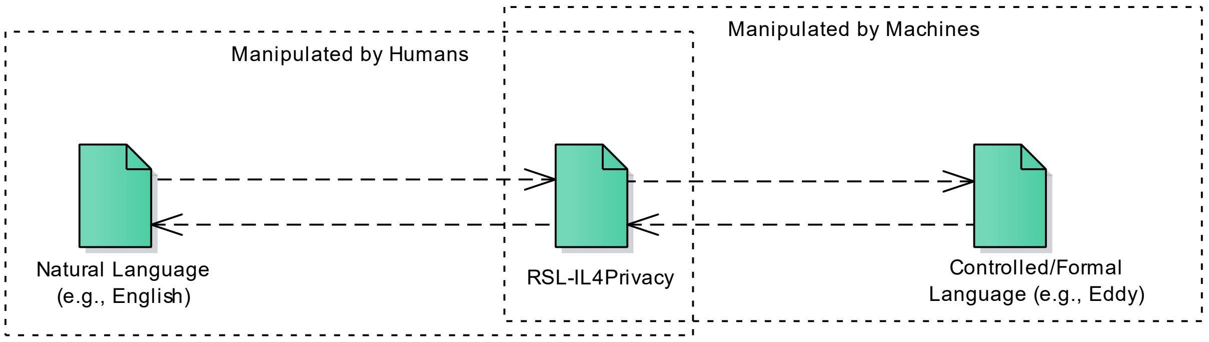

Manipulated by Humans
Manipulated by Machines
Natural Language (e.g., English)
RSL-IL4Privacy
Controlled/Formal Language (e.g., Eddy)

Fig 7. The Interlingua nature of RSL-IL4Privacy

7.1 Improving Privacy Specifications with RSL-IL4Privacy

In RSL-IL4Privacy each privacy requirement also maintains associations with other elements defined under the scope of the language. This is an extra-layer of information gathered from the statement at hand. The information about the remaining elements (e.g., private data, services, recipients) can be used directly to quickly infer new knowledge about a specific privacy-related aspect inside a policy, and it plays an important role in performing a deeper analysis of a policy as it may reduce the ambiguity that characterizes natural language.

Due to the objective of having a reasoner for the detection of conflicts in policies, we decided to provide interoperability features between Eddy and RSL-IL4Privacy. Even though that RSL-IL4Privacy language aims to preserve natural language as much as possible, it has the necessary elements that can be used in the formal analysis of privacy policies using reasoning from formal languages, such as Eddy. With regard to the syntax of both languages, RSL-IL4Privacy provides a more comprehensible and familiar syntax, despite the difference in size for the specification of a single statement. For example, Figure 8 shows the representation of the same Collection statement (S2) from Twitter's privacy policy both in RSL-IL4Privacy and in Eddy. Despite representing equivalent information, having the original statement included in the requirement specification is one significant difference between both languages. This aspect is particularly valuable in the way that it provides developers – who need to be aware of the different data practices and flows that occur in the systems that integrate – with the opportunity of checking the conflicting statements and taking measures to correct and improve these statements. On the other hand, this also gives end-users the possibility of having a better understanding of the statements that comprise a given privacy policy, and easily identify which personal information is being managed by the service provider.

Moreover, generic statements (classified as "informative") may still contain useful information in a slightly different context (e.g., explicit business rules, tools or other mechanisms). Using RSL-IL4Privacy to specify a policy previously written in natural language ensures that such information is still kept and even becomes documented in a better organized way. On the other hand, due to a narrower scope, privacy policies specified in Eddy or other formal languages, such as those described in Section 6, tend to restrict the original policies by discarding fragments that do not directly relate to data privacy concerns. In the end, this may cause the loss of valuable information as it ends up not being included in such regular policy specifications.

```
// RSL-IL4Privacy
statement S2 : Collection {
   name "S2"
   modality Permitted
   description "You may provide us with profile information to make public on the Twitter
Services, such as a short biography, your location, your website, or a picture."

   privateData PD2,
   service SV3
};


// Eddy
P COLLECT Profile-Information FOR Anything
```

Fig 8. Textual representation for Twitter's statement S2 using RSL-IL4Privacy and Eddy (partial view)

7.2 Threats to Validity

Even though RSLingo4Privacy approach relies on RSL-IL4Privacy as an intermediate language for specifying privacy policies, the services (and sub-services) need to be manually inferred because of the crucial role they play in conflict detection. Due to the ambiguity and broad vocabulary that characterizes natural language, we cannot rely on the policy phrases that correspond to a *Service* element (i.e., the "purpose"), alone, to find real conflicts. For example, a purpose "marketing" and "displaying ads" are semantically, but not lexically, similar. At the same time, having vague services, such as "for improving services," would trigger numerous false positives. Defining the services as clear and balanced as possible is still a very challenging task.

On the other hand, the interoperability between RSL-IL4Privacy and Eddy for the purpose of conflict detection is not complete due to the need for manual work. After generating an equivalent Eddy specification and submitting that specification to the reasoning engine, the conflicting statements are reported back with their original attribute "id" identified. At this moment, after identifying the conflicts in the equivalent Eddy specification, one needs to go back to the original RSL-IL4Privacy specification and manually modify the statements to remove conflicts. Future work will focus on generating a conflicting RSL-IL4Privacy specification with suggestions for removing conflicts. This would considerably help the manual process of revising policies to be conflict-free.

Lastly, and since our focus at the moment is to prove how viable and usable RSL-IL4Privacy can be in terms of detecting conflicts in a single specification, we decided not to have an equivalent Eddy "FROM" keyword, and instead we use the service provider as the default recipient. By doing this, we deliberately miss the possibility of finding conflicts between multiple specifications.

8. CONCLUSION

This paper proposes and discusses the RSL-IL4Privacy language for the specification and documentation of privacy policies. This work complements the current state-of-the-art by providing a clear and plain language for the specification of privacy requirements with multiple representations while taking into account the importance of having requirements documented in a format as close to natural language as possible. This language is supported by the RSLingo4Privacy-Studio built on top of the Eclipse IDE, and particularly leveraging Xtext and Eclipse Modeling Framework (EMF) technologies.

RSL-IL4Privacy provides the fundamental constructs to rigorously specify privacy policies. This revised set of constructs allow a better documentation of privacy-aware requirements, since they deal with the information that appears in a slightly different context and that may be used to enforce these policies. Additionally, a simple and well-formed syntax encourages its adoption and use by the interesting parties of this process: developers will recognize its concrete syntax as it is similar to other programming languages' syntax; policy authors will have the ability for specifying and authoring new or existing policies using natural language all the same; and lastly, end-users will get a privacy policy that clearly shows the various data practices and actions with all its related elements clearly defined.

The Twitter case study shows the potential of RSL-IL4Privacy as a rigorous language for expressing privacy requirements. This paper also details an interoperability mechanism between RSL-IL4Privacy and Eddy as a way to provide formal conflict detection between privacy policy statements. In addition, results based on five privacy policies (Dropbox, IMDb, Facebook, LinkedIn and Twitter), achieved using the RSL-IL4Privacy approach, were described and thoroughly discussed, as well as the positive impact of having such information defined in RSL-IL4Privacy, when compared to other formal, but less readable, languages. The concrete artifacts of the RSL-IL4Privacy representations from Dropbox, IMDb, Facebook, LinkedIn and Twitter privacy policies, as well as the analysis of other case studies under the scope of RSLingo4Privacy are available online[11].

Future work will focus on developing the information extraction tasks that comprise the initial step of RSLingo4Privacy approach [4]. This includes automatically classifying policy statements and extracting textual fragments (e.g., the recipients) from privacy policies written in natural language text. Such work would extend the approach proposed by Bhatia et al. to extract privacy goals from policies [53]. We envision that these fragments could then be used to automatically classify privacy policies (against a set of privacy-aware patterns), which would allow us to analyze the privacy policies from popular websites and assign them a qualitative value (e.g., a grade ranging from 1 to 5) in what privacy is concerned; these scores are sometimes called privacy grades[12]. We will also work on a possible solution to define the services automatically and check how RSL-IL4Privacy and the interoperability mechanism perform when dealing with multiple specifications. Finally, we will proceed with the development and enhancement of RSLingo4Privacy-Studio. We also plan on evaluating RSLingo4Privacy-Studio by carrying out usability tests through laboratory-controlled sessions with end-users (e.g. software developers, policy authors and requirement engineers) so that we are also able to receive feedback about the tool.

ACKNOWLEDGEMENTS

This work was partially supported by national funds under FCT projects UID/CEC/50021/2013, EXCL/EEI-ESS/0257/2012, CMUP-EPB/TIC/0053/ 2013 and the project TT-MDD-Mindbury/2014.

[11] https://github.com/RSLingo/RSLingo4Privacy
[12] http://www.privacygrade.org/

REFERENCES

[1] K. Pohl, Requirements Engineering: Fundamentals, Principles and Techniques, Springer 2010.

[2] B. Kovitz, Practical Software Requirements: Manual of Content and Style, Manning 1998.

[3] J. Caramujo and A. R. Silva, "Analyzing privacy policies based on a privacy-aware profile: the Facebook and LinkedIn case studies", IEEE 17th Conference on Business Informatics (CBI), July 2015.

[4] A. R. Silva et al., "Improving the Specification and Analysis of Privacy Policies: The RSLingo4Privacy Approach", 18th International Conference on Enterprise Information Systems (ICEIS), SCITEPRESS, April 2016.

[5] T. D. Breaux, H. Hibshi and A. Rao, "Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements", Requirements Engineering, 19 (3): 281—307, September 2014.

[6] A. Van Deursen, P. Klint and J. Visser, "Domain-specific languages: an annotated bibliography", ACM SIGPLAN Notices, 35(6): 26–36, March 2000.

[7] A.R. da Silva, "Model-Driven Engineering: A Survey Supported by a Unified Conceptual Model", Computer Languages, Systems & Structures, 43, 2015.

[8] L. Bettini, Implementing Domain-Specific Languages with Xtext and Xtend, Packt Publishing Ltd, 2013.

[9] D. Savic et al., "Preliminary experience using JetBrains MPS to implement a requirements specification language", 9th International Conference on the Quality of Information and Communications Technology (QUATIC), September 2014.

[10] F. Baader, The Description Logic Handbook: Theory, Implementantion and Applications, Cambridge University Press 2003.

[11] A. Antón, D. Bolchini and Q. He, "The use of goals to extract privacy and security requirements from policy statements", 26th IEEE International Conference on Software Engineering, 2003.

[12] C. Kalloniatis, E. Kavakli and S. Gritzalis, "Addressing privacy requirements in system design: the PriS method", Requirements Engineering, vol. 13 (3), pp. 241—255, September 2008.

[13] G. Kapitsaki and I. Venieris, "PCP: privacy-aware context profile towards context-aware application development", 10th International Conference on Information Integration and Web-based Applications & Services, pp. 104—110, November 2008.

[14] P. Kumari, "Requirements analysis for privacy in social networks", 8th International Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods, 2010.

[15] W3C, The Platform for Privacy Preferences (P3P) Project: http://www.w3.org/P3P/.

[16] eXtensible Access Control Markup Language (XACML) Version 3.0. 22 January 2013. OASIS Standard.

[17] Enterprise Policy Authorization Language 1.2 (EPAL) Specification, W3C. https://www.w3.org/Submission/2003/SUBM-EPAL-20031110

[18] Cranor, L.F. 2003. "P3P: Making privacy policies more useful", IEEE Security & Privacy. 6, 50-55.

[19] P3P Preference Exchange Language 1.0 (APPEL) Specification, W3C, http://www.w3.org/TR/P3P-preferences

[20] A. Uszok et al., "KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement", 4th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 93—96, June 2003.

[21] L. Kagal, T. Finin and A. Joshi, "A policy language for a pervasive computing environment", 4th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 63—74, June 2003.

[22] W3C, Notation3 (N3): A readable RDF syntax, 2011. https://www.w3.org/TeamSubmission/n3/.

[23] Shah, A.B. 2005. An integrated development environment for policies. Master Thesis. University of Baltimore.

[24] E. Paja, F. Dalpiaz, P. Giorgini, "Modeling and reasoning about security requirements in socio-technical systems", Data & Knowledge Engineering, 98:123-143, 2015.

[25] F. Dalpiaz, E. Paja, P. Giorgini, "Security Requirements Engineering: Designing Secure Socio-Technical Systems". MIT Press, 2016.

[26] J.-J. Meyer, "Deontic logic: A concise overview", Deontic Logic in Computer Science: Normative System Specification. John Wiley & Sons, 1993.

[27] Karat, J. et al., "Designing natural language and structured entry methods for privacy policy authoring", Human-Computer Interaction - INTERACT. Springer, 671-684, 2005.

[28] Wishart, R. et al., "Collaborative privacy policy authoring in a social networking context", In Proc. of the POLICY symposium. IEEE, 1-8, 2010.

[29] Moore, B. et al., 2001. Policy Core Information 1.0 Specification, RFC 3060, http://www.ietf.org/rfc/rfc3060

[30] Nadas, A. et al., "A model-integrated authoring environment for privacy policies", Science of Computer Programming. 89, Part B, 105-125, 2014.

[31] Winkler, S., and Zeadally, S. "Privacy Policy Analysis of Popular Web Platforms", IEEETechnology and Society Magazine 35.2, 75-85, 2016.

[32] Han, W., Lei, C., "A survey on policy languages in network and security management". Computer Networks. 56, 1, 477-489, 2012.

[33] M. Gharib, P. Giorgini, J. Mylopoulos, "Towards an Ontology for Privacy Requirements via a Systematic Literature Review", International Conference on Conceptual Modeling. Springer, 2017.

[34] P. Anthonysamy, A. Rashid, R. Chitchyan, "Privacy requirements: present & future", Proceedings of the 39th International Conference on Software Engineering, IEEE Press, 2017.

[35] Backes, Michael, Birgit Pfitzmann, and Matthias Schunter. "A toolkit for managing enterprise privacy policies", European Symposium on Research in Computer Security. Springer, 2003.

[36] C. A. Brodie, C.-M. Karat, and J. Karat, "An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench", *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006.

[37] W3C, P3P 1.0 Implementations. <http://www.w3.org/P3P/implementations>

[38] T. Breaux and A. Anton, “Analyzing regulatory rules for privacy and security requirements”, IEEE Trans. Softw. Eng., January 2008.

[39] A. Massey, P. Otto, L. Hayward, and A. Anton, “Evaluating existing security and privacy requirements for legal compliance”, Proceedings of the RE, 2010.

[40] J. Young, “Commitment analysis to operationalize software requirements from privacy policies”, Requirements Engineering, 2011.

[41] H. Nissenbaum, “Privacy as contextual integrity”, Wash. L. Rev., 2004.

[42] D. J. Solove, “A Taxonomy of Privacy”, University of Pennsylvania Law Review, January 2006.

[43] A. I. Anton, E. Bertino, N. Li, and T. Yu, “A roadmap for comprehensive online privacy policy management”, Commun. ACM, 2007.

[44] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum, “Privacy and contextual integrity: Framework and applications”, in Proc. 2006 IEEE Symposium on Security & Privacy, 2006.

[45] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, “A machine learning approach for tracing regulatory codes to product specific requirements,” in ICSE, 2010.

[46] V. Gervasi and D. Zowghi, “Reasoning about inconsistencies in natural language requirements,” ACM Trans. Softw. Eng. Methodol., 2005.

[47] A. Guha, M. Fredrikson, B. Livshits, and N. Swamy, “Verified security for browser extensions”, in 2011 IEEE Symposium on Security and Privacy, May 2011.

[48] M. L. Johnson, S. Egelman, and S. M. Bellovin, “Facebook and privacy: it’s complicated”, in SOUPS, 2012.

[49] S. Gurses, R. Rizk, and O. Gunther, “Privacy design in online social networks: Learning from privacy breaches and community feedback”, in ICIS 2008 Proceedings. ACM, 2008.

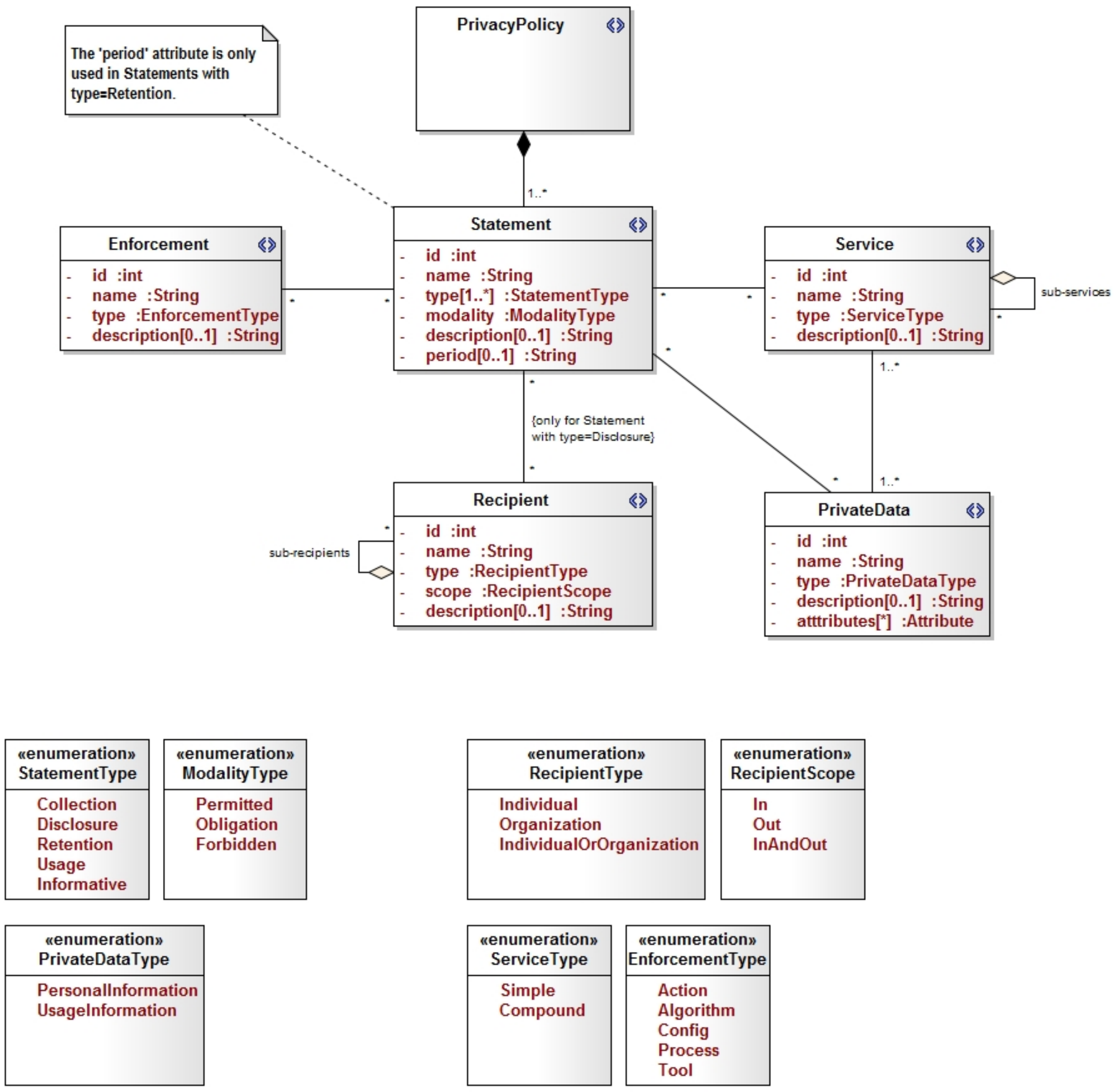
[50] J. Bonneau and S. Preibusch, “The privacy jungle: On the market for data protection in social networks”, in Economics of Information Security and Privacy. Springer, 2010.

[51] A. Acquisti and R. Gross, “Imagined communities: Awareness, information sharing, and privacy on the facebook”, in Privacy Enhancing Technologies. Springer, 2006.

[52] Horkoff, J., et al., “Goal-oriented requirements engineering: an extended systematic mapping study”, Requirements Engineering, 1-28, Springer, 2017.

[53] J. Bhatia, T. D. Breaux, F. Schaub. “Privacy Goal Mining through Hybridized Task Re-composition”, ACM Trans. on Soft. Engr. Method., 2016.

APPENDIX A: COMPLETE RSL-IL4PRIVACY METAMODEL


The 'period' attribute is only used in Statements with type=Retention.
PrivacyPolicy
1..*
Statement
- id :int
- name :String
- type[1..*] :StatementType
- modality :ModalityType
- description[0..1] :String
- period[0..1] :String
Enforcement
- id :int
- name :String
- type :EnforcementType
- description[0..1] :String
Service
- id :int
- name :String
- type :ServiceType
- description[0..1] :String
sub-services
{only for Statement with type=Disclosure}
1..*
1..*
Recipient
- id :int
- name :String
- type :RecipientType
- scope :RecipientScope
- description[0..1] :String
sub-recipients
PrivateData
- id :int
- name :String
- type :PrivateDataType
- description[0..1] :String
- atttributes[*] :Attribute
«enumeration» StatementType
Collection
Disclosure
Retention
Usage
Informative
«enumeration» ModalityType
Permitted
Obligation
Forbidden
«enumeration» RecipientType
Individual
Organization
IndividualOrOrganization
«enumeration» RecipientScope
In
Out
InAndOut
«enumeration» PrivateDataType
PersonalInformation
UsageInformation
«enumeration» ServiceType
Simple
Compound
«enumeration» EnforcementType
Action
Algorithm
Config
Process
Tool