RSLingo4Privacy Studio
A Tool to Improve the Specification and Analysis of Privacy Policies

André Ribeiro and Alberto Rodrigues da Silva
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal

Keywords: Privacy Policy, Requirements Specification, Domain Specific Language, Software Tool.

Abstract: Popular software applications collect and retain a lot of users' information, part of which is personal and sensitive. To assure that only the desired information is made public, these applications have to define and publish privacy policies that describe how they manage and disclose this information. Problems arise when privacy policies are misinterpreted, for instance because they contain ambiguous and inconsistent statements, what results in a defective application of the policy enforcement mechanisms. The RSLingo4Privacy approach aims to improve the specification and analysis of such policies. This paper presents and discusses its companion tool, the RSLingo4Privacy Studio, which materializes this approach by providing the technological support for users being able to specify, analyze and publish policies based on the RSL-IL4Privacy domain specific language. We validated its feasibility using popular websites policies such as Dropbox, Facebook, IMDB, LinkedIn, Twitter and Zynga. We conclude this paper with a discussion of the related work, namely a comparative analysis of pros and cons of RSLingo4Privacy Studio with other previous proposals.

1 INTRODUCTION

Popular web and mobile applications attract and support a huge number of users. They collect data from these users without ensuring traceability between privacy policies and application design decisions. A particular challenge for policy authors and application developers is the need to use a common language and companion tools that supports translating important privacy policy statements into actionable requirements. For example, European Union and United States employ privacy policies as "notices" to end users and, in the U.S., these policies are often the sole means to enforce accountability. Given the pressure to post privacy policies and the pressure to keep policies honest, companies must do more to align their policies and practices. More should be accomplished by enabling 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.

A privacy policy (or just "policy" for the sake of brevity) is a technical document that states multiple privacy-related requirements that a system should satisfy. These requirements are usually defined as ad-hoc natural language (NL) statements. NL is an ideal medium to express these 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 (Ferreira and Silva, 2012) (Ferreira and Silva, 2013). However, NL has intrinsic characteristics that become the root cause of quality problems, such as incorrectness, inconsistency and incompleteness (Pohl, 2010) (Silva, 2015a).

Caramujo and Silva proposed the definition of a domain-specific language for the specification of privacy-aware requirements, called ***RSL-IL4Privacy language*** (Caramujo and Silva, 2015). Recently this language evolved for a more updated and consistent version available as a technical report (Caramujo et al, 2017). This language provides several constructs such as statements, private data, recipients and enforcement mechanisms, which are necessary to specify and document privacy-related requirements.

The goal of the proposed approach is to use this language as the necessary mechanism for the specification of policies while providing features for better analyzing and validating the corresponding policies.

Ribeiro, A. and Silva, A.
RSLingo4Privacy Studio - A Tool to Improve the Specification and Analysis of Privacy Policies.
DOI: 10.5220/0006310400520063
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 52-63
ISBN: 978-989-758-248-6

The adoption of this language was defined in the ***RSLingo4Privacy approach*** by the integration of the following key processes (Silva et al., 2016): (i) automatic classification, extraction and translation of statements from policies written in NL into RSL-IL4Privacy specifications; (ii) visualization and authoring; (iii) analysis and validation; and (iv) (re)publishing in a structured and both human and machine-readable formats.

This paper extends that prior work with the following novel contributions: description of a tool (RSLingo4Privacy Studio) that materializes the RSLingo4Privacy approach and uses RSL-IL4Privacy as an intermediate language for the specification of privacy policies, and an extensive discussion on how this tool supports several transformations to support the multiple policies representations.

RSLingo4Privacy Studio is mainly targeted for requirement engineers, policy authors and software developers so they can edit, analyze and (re)publish privacy policies in different formats using a single tool. The major merit of this tool is that it allows both technical and non-technical users to easily author and analyze policies using a language close to NL, but that is simultaneously readable and executable by machines and so providing automatic validation.

This work was validated using six privacy policies extracted from popular web sites and social networks (Dropbox, Facebook, IMDB, LinkedIn, Twitter and Zynga). For the sake of brevity we only consider here Dropbox's policy to support the discussion and exemplify the usage of RSLingo4Privacy Studio.

The paper is organized as follows: Section 2 introduces the background, namely providing an overview of the RSL-IL4Privacy language and explains the scope of this research. Section 3 introduces RSLingo4Privacy Studio, highlighting its principal features and technological aspects; and in particular the transformations supported by this tool. Section 4 refers and discusses the related work. Finally, Section 5 presents the conclusion and future work.

2 BACKGROUND

RSLingo is a long-term research initiative in the RE area that recognizes that natural language, although being the most common and preferred form of representation used within requirements documents, is prone to produce such ambiguous and inconsistent documents that are hard to automatically validate or transform. Originally RSLingo proposed an approach to use simplified Natural Language Processing (NLP) techniques as well as human-driven techniques for capturing relevant information from ad-hoc natural language requirements specifications and then applying lightweight parsing techniques to extract domain knowledge encoded within them (Ferreira and Silva, 2012). This was achieved through the use of two original languages: the RSL-PL (Pattern Language) (Ferreira and Silva, 2013a), designed for encoding RE-specific linguistic patterns, and RSL-IL (Intermediate Language), a domain specific language designed to address RE concerns (Ferreira and Silva, 2013). Through the use of these two languages and the mapping between them, the initial knowledge written in natural language can be extracted, parsed and converted to a more structure format, reducing its original ambiguity and creating a more rigorous SRS document (Silva, 2015a).

In the scope of the RSlingo initiative we have developed a focused analysis of privacy policies to discover common linguistics patterns found throughout these policies. As a consequence, we defined most of these patterns as a privacy-aware profile based on software language technologies (Mernik et al., 2005; Voelter et al., 2013; Silva, 2015; Ribeiro et al., 2016): the RSL-IL4Privacy language.

RSL-IL4Privacy enables a more rigorous specification of privacy requirements than writing in just NL. The adoption of a language such as RSL-IL4Privacy allows that its specifications become simpler to read and understand which facilitates the communication between the involved stakeholders (Silva et al., 2016). Figure 1 depicts a partial view of the RSL-IL4Privacy metamodel including its core elements: Statement, Recipient, PrivateData, Service and Enforcement.

Statement describes what rules or actions are specified in a policy, thus it can be seen as a privacy requirement. A policy comprises a set of statements.

Each statement can be classified into five different categories, according to its features or qualities: *Collection* defines which data is collected; *Disclosure* defines which data is disclosed and to what entities; *Retention* defines for how long data will be stored; *Usage* defines what is the purpose of having the data; and *Informative* is a statement with just generic information. It is also noteworthy that one statement may refer to multiple services and act on different private data.

PrivateData represents the users' data that is

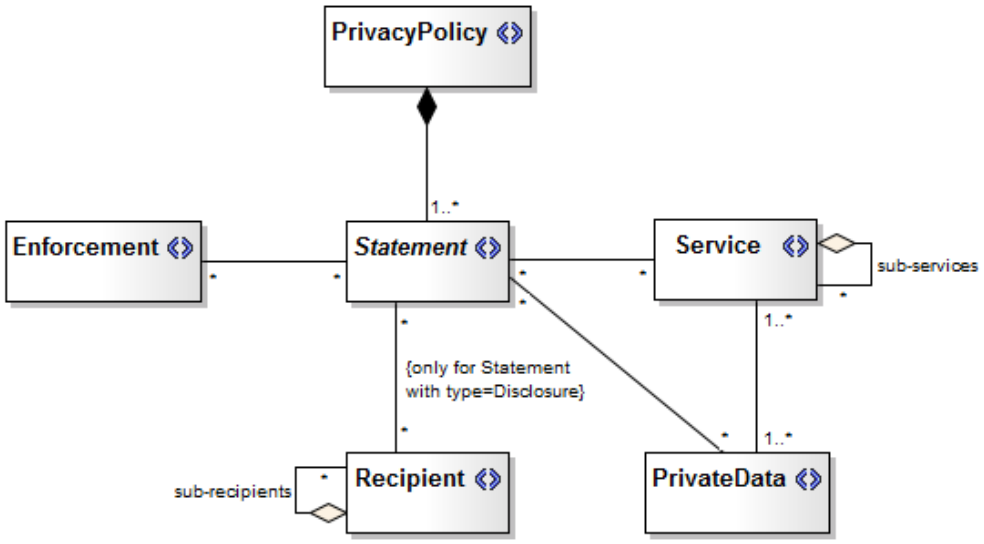

Figure 1: RSL-IL4Privacy metamodel (partial view).

collected and managed by the software application (or service provider). A PrivateData 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); or *usage*, if the data is gathered based on the user's activity on the system (e.g., device specifications).

Service describes the multiple high-level services that are provided through the users' point of view. It is important to point out the association between Service and PrivateData because it makes possible to track what personal information is being employed in each service.

Finally, *Enforcement* element is particularly useful because through the description of the mechanisms and tools that are documented in a policy, one can have some insight about how it can be possible to enforce privacy requirements of such privacy policies. On the other hand, it also encompasses rules and specific actions with regard to the use of the system that are important for the enforcement of a given privacy policy.

The RSL-IL4Privacy language is implemented with different technologies to provide multiple representations depending on the respective formality, namely: tabular, graphical, and textual representation.

The *tabular representation* is supported by a MS-Excel template that includes some predefined sheets. For example, it includes a specific sheet with the statements, other sheet with the private data, etc. It might also include some analysis reports and graphics on top of that source information.

The *graphical representation* is based on UML tools, such as Sparx Systems Enterprise Architect that has been used. In this case, RSL-IL4Privacy is implemented as a UML profile and consequently we can represent any policy as a UML package with its associated 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 case specifications) nor do they easily support automatic validation of such requirements. Therefore, the RSL-IL4Privacy language is also formalized and defined with a *rigorous* and *textual representation* using the Xtext framework (Bettini, 2016). Figure 2 shows an example of RSL-IL4Privacy statements specified according to this representation. On the other hand, Figure 3 shows partially the Xtext grammar of RSL-IL4Privacy. (From now on, whenever we refer "RSL-IL4Privacy", we mean this textual representation.)

3 RSL-IL4PRIVACY STUDIO

RSLingo4Privacy Studio (or simply "Studio" for the sake of brevity) is a software tool that supports and materializes the RSLingo4Privacy approach providing the technological support for a user being able to perform the processes proposed. Studio allows a user to specify, analyze and publish privacy policies into multiple formats. Studio is built on top of the Eclipse IDE, more specifically leveraging the Eclipse Modeling Framework (EMF). It relies on a

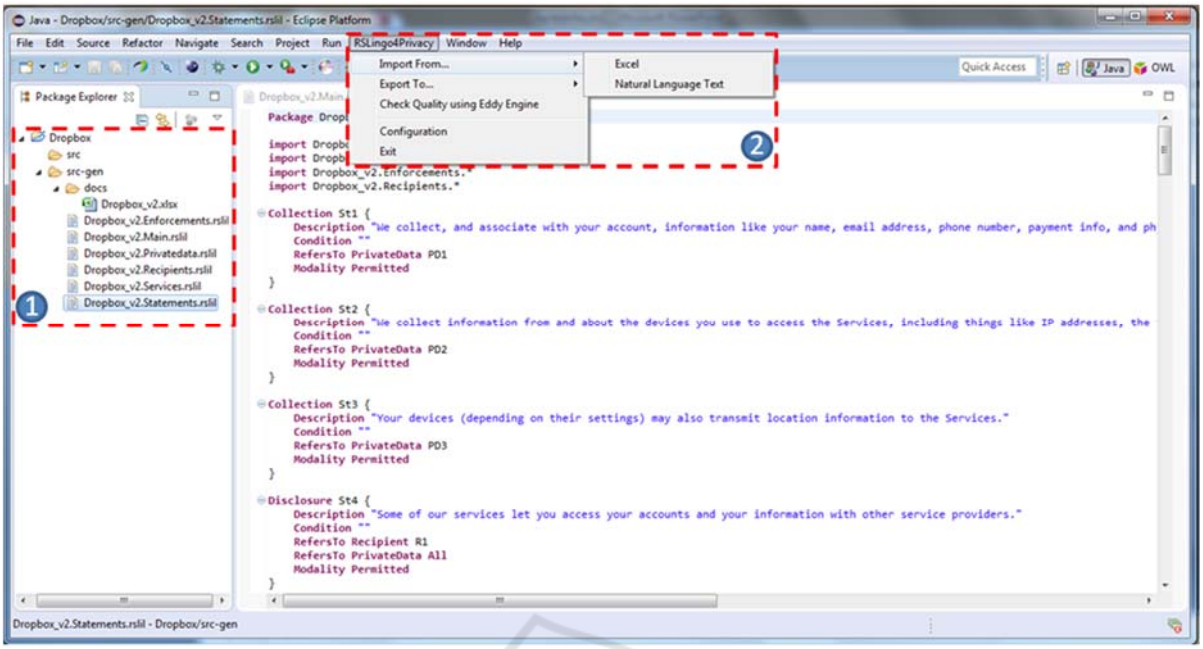

Figure 2: Structure of Dropbox RSLingo project (1) and RSLingo4Privacy Main Menu bar (2).

```
    Enforcement:
  'Enforcement' name=ID '{'
  'Name' enforcementName=STRING
  'Description' description=STRING
  'Type'
type=('Action'|'Algorithm'|'Config'|'Process'|'Tool')
'}';

    Service:
  'Service' name=ID '{'
  'Name' serviceName=STRING
  ('Description' description=STRING)?
  ('RefersTo                          PrivateData'
(refPrivateData+=RefPrivateData* | refPDAll='All'))?
  ('Service_Part' servicePart+=ServicePart*)? '}';

    Recipient:
  'Recipient' name=ID '{'
  'Name' recipientName=STRING
  'Description' description=STRING
  ('Recipient_Part' recipientPart+=RecipientPart*)?
  'Scope'
scope=('Internal'|'External'|'Internal/External')
  'Type'
type=('Individual'|'Organization'|'Individual/Organizati
on') '}';
```

Figure 3: Part of the Xtext grammar of RSL-IL4Privacy.

multi-transformation approach where the RSL-IL4Privacy language acts as an intermediate language used to represent the policies and then, from that specification, being able to generate more readable representations of these policies (e.g. Excel or Word files) or to check their quality using the Eddy engine.

Eddy is a formal language based on Description Logics (DL) (Baader et al., 2003) that allows specifying privacy requirements, actors, data, and data-use purpose hierarchies based on the DL subsumption. It also allows specifying the deontic modality (i.e., permission and prohibition) of such data purposes and then automatically detects conflicts between what it is permitted and what it is prohibited, and traces permissible, required and prohibited data flows within the specification. Eddy language is supported by the Eddy engine, which is implemented using two OWL reasoners. Eddy's source code is available on GitHub[1].

Subsection 3.2 provides a detailed description of the model transformations involved, namely based on three main features: Import, Export and Check Quality. The **Import** feature allows a user to import an Excel file or an ad-hoc natural language text file containing the policy and produce its corresponding RSL-IL4Privacy files. The **Export** feature allows transforming a RSL-IL4Privacy file to other file formats, namely Word, Excel, JSON, Eddy and Text (controlled NL). 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 policy specified as an Eddy file. The output of this process is a log file, containing the possible conflicts, an OWL (Web Ontology Language) (Bechhofer et al., 2004) file with the representation of the rules used, and an image with the graphical hierarchical representation of the equivalent ontologies (produced using the OwlViz plugin[2]).

As an Eclipse-based tool, Studio takes advantage

[1] https://github.com/cmu-relab/eddy
[2] https://github.com/protegeproject/owlviz

of Eclipse's extension points[3] in order to offer custom menus, commands and project wizards. It provides a menu in the Main Menu bar, Context Menus for RSL-IL4Privacy and Eddy files, and wizards to ease the creation of RSL-IL4Privacy files or RSLingo4Privacy projects. The Main Menu bar option provides features such as (see Figure 2): import of Excel files and generation of the corresponding RSL-IL4Privacy files; transformation of RSL-IL4Privacy files to all the supported formats; and launch of the Eddy engine for the existing equivalent Eddy files.

In addition, a Context Menu is shown by right-clicking a RSL-IL4Privacy file or an Eddy file in the Package Explorer View. The Context Menu of a RSL-IL4Privacy file provides the options to export that file to all the supported formats. This menu also shows up when the user right-clicks inside the editor of an open RSL-IL4Privacy file. The Context Menu of an Eddy file allows to validate this file by launching the Eddy engine.

The "New RSL-IL4Privacy file" wizard creates a new file containing an example structure of a file complying with the RSL-IL4Privacy grammar defined in Xtext. The user can then freely edit this file with all the capabilities offered by the RSL-IL4Privacy editor. The "New RSLingo4Privacy project" wizard allows the creation of a new project with the Xtext nature with a source folder containing one or more RSL-IL4Privacy files.

3.1 RSL-IL4Privacy Editor

Studio's textual editor allows creating and editing RSL-IL4Privacy files. This editor was developed using the Xtext framework[4]. From a grammar definition it is possible to automatically generate the language infrastructure (e.g. parser and typechecker) and a fully customizable Eclipse plugin containing the DSL editor with helpful features like syntax highlighting, error checking, auto-completion or source-code navigation (Bettini, 2016). 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 entities and their relations. Figure 3 shows a fragment of the RSL-IL4Privacy grammar definition for the Enforcement, Service and Recipient elements.

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 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 a unique file, but instead defined in separated and different files with different purposes.

3.2 Transformations

This section must be in one column. Studio relies on several transformations to support multiple representations of a policy based on the common and intermediate RSL-IL4Privacy language. Figure 4 summarizes this multi-transformation approach, which involves T2M, M2M and M2T transformations.

According to the proposed approach, Studio deals with policies represented in multiple representations, namely: ad-hoc and controlled NL text, Excel, Word, JSON and Eddy. We considered an Excel file as model, since it is a tabular and highly structured representation. In contrast, we considered 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 that 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. For instance, JSON, Eddy and Text are generated using Xtend[5], while Word and Excel are generated using the Apache POI library[6].

T2M Transformations. 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

[3] http://goo.gl/jBRVuM
[4] https://eclipse.org/Xtext
[5] http://www.eclipse.org/xtend
[6] https://poi.apache.org

and classifies them into a set of five distinct categories (Collection, Disclosure, Retention, Usage and Informative). The second process extracts the relevant elements from the original statements into their equivalent representation in RSL-IL4Privacy. These processes are implemented using RapidMiner[7], which is a popular open source platform for predictive analytics and data mining (Kotu and Deshpande, 2014). The implementation of this transformation is a complex task that involves the integration and tuning of feature models and tools, and it is still a working in progress task.

M2M Transformations. M2M transformations are used both 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).

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).

3.2.1 JSON and Text

The transformations from RSL-IL4Privacy into JSON (M2M-3) and Text (M2T-2) are performed using Xtend. Xtend is a general purpose high-level programming language derived from Java that is commonly used with Xtext to develop code generators. More specifically, 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, because it allows the definition of code templates. Code templates are portions of code that contain dynamic parts that change according to the Xtext-based model given as input.

[7] http://rapidminer.com

3.2.2 Word and Excel

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 with Java, because Apache POI 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 Private Data) 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.

3.2.3 Eddy (and OWL)

The transformation from RSL-IL4Privacy into Eddy (M2M-4) is performed using Xtend, similarly to what is done for transformations M2M-3 and M2T-2. Silva et al. provide a detailed description on how RSL-IL4Privacy concepts are mapped into equivalent Eddy concepts in what concerns the transformation M2M-4 (Silva et al., 2016). In addition, the transformation from Eddy into OWL (M2M-5) is performed internally and at the

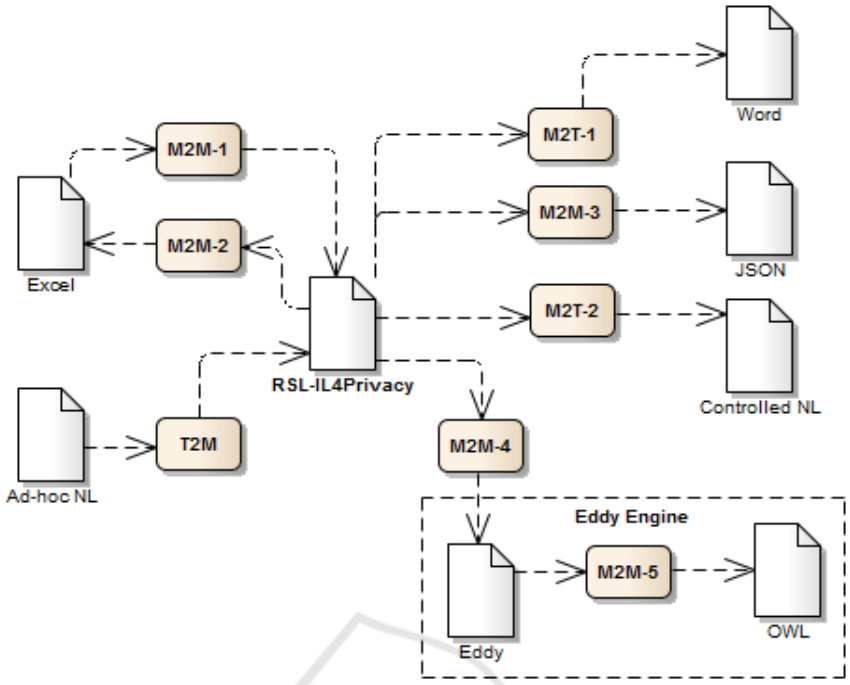

Figure 4: RSLingo4Privacy Studio model transformations.

execution-time by the Eddy engine that we integrate as a third-party tool. These Eddy statements are mapped to action and roles in DL based on OWL reasoners (Breaux et al., 2014).

In what respect the Dropbox case study, we generated the corresponding Eddy file and then ran it using the Eddy engine. The result of that execution did not detect any conflicts; that means that there are not any contradictory statements in what concerns the deontic modality analysis. This can be explained by the fact that Dropbox's policy is relatively small and simple (for instance, when compared to Facebook's policy) and so it is easier to maintain manually by a person and to detect any contradiction between each statement.

4 RELATED WORK

As far as we know from the literature analysis, there are not any tool like RSL-IL4Privacy Studio that addresses simultaneously the same goals and provides the proposed features. Despite that, there are some initiatives focused on security or privacy policy specification and analysis. From our analysis we identify three types of initiatives. First, initiatives concerned with web and authorization privacy policies. Second, semantic web approaches that mainly allow specifying and validating privacy policies based on ontologies and modal logic. Finally, we still mention other relevant work.

Table 1 summarizes the RSLingo4Privacy features and contrasted it with other approaches taking into account their base languages and the tool support they offer. With the exception of RSLingo4Privacy, none of these approaches provide a tool that simultaneously supports visualization and authoring, multiple types of model transformations (T2M, M2M and M2T) and publishing of policies. Despite that, RSLingo4Privacy does not support authorization enforcement. Adding the support to this feature constitutes a possible future research direction, namely by supporting the transformation into languages used for that purpose or even extending RSL-IL4Privacy with concepts commonly used for policy authorization enforcement.

4.1 Web and Authorization Approaches

The commonly mentioned approaches for defining website's data privacy management and/or authorization/access control policies are the ones involving the standards P3P (Cranor et al., 2006), XACML (OASIS, 2013) and EPAL (Ashley et al., 2003). 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 is broader than the data privacy domain, which is the focus in our work. For instance, P3P is focused on the users' privacy preferences, while XACML and EPAL are focused on access control aspects, i.e., authorization. Therefore, they address slightly different concerns comparing with RSL-IL4Privacy.

Several tools support the creation of policies written in P3P, XACML and EPAL, but the majority of them were discontinued. IBM P3P Policy Editor[8] 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) (Cranor, 2003). 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 (Cranor et al., 2002) 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[9] 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. For this reason these approaches are often considered too complex and difficult to adopt in practice. Also the fact that nor websites neither browsers (only Internet Explorer used) are obliged to use P3P, has contributed to its decreasing use.

4.2 Semantic Web Approaches

The commonly mentioned approaches for defining website's data privacy management Semantic web approaches are commonly used to both specify and analyze privacy policies. The use of ontologies, represented using formal knowledge representation languages (e.g. OWL, DAML[10], RDF[11]), allows the use of reasoners that can determine if the privacy policy is consistent and check if there are any conflicts between its rules that typically adopt deontic modal logic.

Eddy's website[12] provides three examples using an editor where the user can specify a privacy policy in a free text area. Then, it is possible to run the Eddy engine to analyze the policy for detecting any possible conflicts or for tracing the flow and showing it in 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 (Uszok et al., 2003) is goal-oriented software requirement language that can also be used to express high-level goals of a privacy policy. KAoS supports a formal specification based on DAML. KPAT (KAoS Policy Administration Tool) is a graphical tool that allows users to specify, analyze, modify and test policies using KAoS. KPAT also detects policy conflicts and allows managing sets of ontologies. KPAT offers a set of views of KAoS (e.g. Domains, Actor Classes, Policies, Policy Templates). Since the policies are specified using the GUI, the corresponding DAML representations are generated automatically using a generic template, avoiding the user to master DAML. 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.

The Rei policy language (Kagal et al., 2003) is another logic-based language that relies on the deontic concepts of rights, prohibitions, obligations and dispensations. Beside the general purpose text editors, there are three 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 (Berners-Lee, 2005), 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) (Shah, 2005) 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

[8] https://www.w3.org/P3P/imp/IBM
[9] https://www.p3pedit.com
[10] http://www.daml.org
[11] https://www.w3.org/RDF
[12] https://gaius.isri.cmu.edu:8080/eddy

Table 1: Comparison of privacy-aware specification approaches.

	Approach	Languages	Tool Support					
			Visualization & Authoring	Transformations			Publishing	Authorization Enforcement
				T2M	M2M	M2T		
Web and Authorization	P3P/APPEL	P3P/APPEL	Yes (IBM P3P, JRC Policy Workbench, P3PEdit)	No	No	To HTML	No	No
	XACML	XACML	Yes (UMU XACML Editor)	No	No	To HTML	No	Yes
	EPAL	EPAL	Yes (Privacy Authoring Editor, EPAL Editor)	No	No	To HTML	No	Yes
Semantic Web	Eddy	Eddy	Yes (General-purpose text editor)	No	Yes	No	No	No
	KAoS	DAML	Yes (KPAT)	No	No	No	No	Yes
	Rei	Rei (OWL-based)	Yes (Protégé-Plugin, N3 text editor, RIDE)	No	No	No	No	No
Other	Ponder	Ponder	Yes (Ponder Policy Editor)	No	To XML	To Java	No	Yes
	PATRN	PATRN & FORMULA	Yes (GME-based editor)	No	No	To FORMULA	No	No
	SPARCLE	NL or structured form	Yes (Policy Editor)	No	To EPAL	No	No	No
	PRiMMA-Viewer	Datalog	Yes (Datalog editor)	No	No	No	No	Yes
	RSLingo4 Privacy	RSL-IL4Privacy	Yes (Eclipse Xtext-based Plugin)	Ad-hoc NL to RSL-IL4Privacy	Excel, JSON and Eddy	Word and Controlled NL	Yes	No

interface for testing the policy and querying the Rei engine.

4.3 Other Approaches

Damianou described a set of tools for Ponder, another goal-oriented language that can be applied in the privacy policy domain (Damianou et al., 2002).

Han and Lei analyzed and compared eleven policy languages grouped in two sets: network and security management (Han and Lei, 2012).

Nadas presented a model-based policy authoring framework called PATRN applied to the health information systems domain. PATRN uses a graphical DSL, defined using GME, for policy authoring, and the FORMULA specification language, based on logic programming, for policy analysis (Nadas et al., 2014).

Karat proposed a privacy policy workbench named SPARCLE to support privacy policy authoring, implementation and compliance monitoring. SPARCLE allows users to specify a privacy policy using natural language or using a structured format (tabular and form-based). It supports the transformation of privacy policies written in natural language into the structured form (and vice-versa), as well as the transformation to a machine-readable format like EPAL (Karat et al., 2005).

Wishart developed a tool called PRiMMA-Viewer to support the collaborative specification of privacy policies, specified in Datalog, for shared content on Facebook (Wishart et al., 2010). PRiMMA-Viewer uses an architecture aligned with the Policy Core Information Model (PCIM) (Moore et al., 2001).

5 CONCLUSIONS

This paper proposes RSLingo4Privacy Studio, a software tool for better supporting the specification, analysis and documentation of privacy-aware requirements in the scope of privacy policies. This work complements the current state-of-the-art by providing a versatile tool designed around the RSL-IL4Privacy language, with multiple representations while taking into account the importance of having requirements documented in a format as close to natural language as possible. Studio is built on top of

the Eclipse IDE, and particularly leveraging and integrating technologies such as: Xtext, Xtend, Eclipse Modeling Framework (EMF), RapidMiner, Eddy engine and Apache POI library.

The validation with six real-world policies (Dropbox, Facebook, IMDB, LinkedIn, Twitter and Zynga, available at the GitHub repository) shows the potential of RSL-IL4Privacy as a rigorous language for expressing privacy requirements and, in addition, shows the relevance of the provided interoperability features. Figure 5 shows the model transformations that are performed in each process of the RSLingo4Privacy approach.

First, T2M transformations intend to automatically classify NL statements and extract from them text snippets using text mining and text extraction algorithms. The implementation of such transformations is a complex task that involves the integration and tuning of tools like RapidMiner, and is still a working in progress research.

Second, M2T transformations produce a consistent and easy-to-read version of a privacy policy. These versions can be produced in multiple formats, such as structured NL in Word, plain text or even HTML.

Third, M2M transformations may include two variants: M2M transformations that support multiple representations of the RSL-IL4Privacy; for example, from plain text format (defined with Xtext) into tabular format in Excel, and vice-versa; and finally, M2M transformations between RSL-IL4Privacy with other languages and formats, such as JSON or Eddy.

The major merit of Studio is that it allows both technical and non-technical users to easily author and analyze policies using a language close to NL, but that is simultaneously readable by machines and so providing automatic validation at both syntactic and semantic levels. This fact permits RSL-IL4Privacy to act as an intermediate language that when supported by an environment that integrates multiple representations of a privacy policy addressing concerns of multiple stakeholders.

As future work we plan to conduct other case studies and laboratory-controlled sessions with end-users (e.g. software developers, policy authors and requirement engineers).

In addition, we intend to apply and integrate these privacy-related concerns with other concerns, namely those defined in the original RSL-IL language (Ferreira and Silva, 2013), or more recently with RSLingo's RSL (Silva, 2017), which are related to requirements engineering in a broader perspective. Consequently, these privacy policies and respective requirements should be combined with other security concerns such as authorization, confidentiality, integrity, authenticity, and accountability requirements (Haley et al, 2008).

Other interesting research directions are dealing with modularity and variability aspects (e.g. using OMG's CVL (Haugen et al., 2012)) and with other privacy-related aspects, like: domain-specific privacy definitions for multiple domains (e.g. healthcare, finance, government), accountability mechanisms for detecting privacy violations, mechanisms for tracking data flow and privacy and authorization enforcement at application level (Landwehr, 2016).

6 CALL FOR ACTION!

Download the ready-to-use Eclipse IDE version of

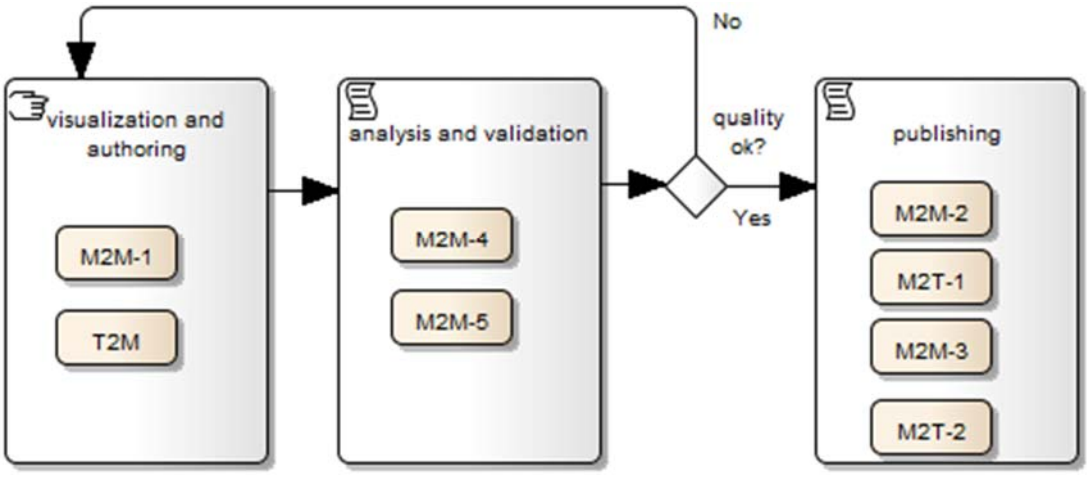

Figure 5: RSLingo4Privacy Approach versus the supported model transformations.

RSLingo4Privacy Studio, that it is available at its GitHub repository[13].

Furthermore, the concrete artifacts of the RSL-IL4Privacy representations for Dropbox, Facebook, IMDB, LinkedIn, Twitter and Zynga privacy policies, as well as the analysis of other case studies under the scope of RSLingo4Privacy are available and can be found on its GitHub repository[14].

ACKNOWLEDGEMENTS

This work was partially supported by national funds under FCT projects UID/CEC/50021/2013, and CMUP-EPB/TIC/0053/2013.

REFERENCES

Ashley, P. et al., 2003. Enterprise Policy Authorization Language 1.2 (EPAL) Specification, W3C. https://www.w3.org/Submission/2003/SUBM-EPAL-20031110.

Baader, F. et al., 2003. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press.

Bechhofer, S. et al., 2004. OWL: Web Ontology Language Reference. *W3C Recommendation*.

Berners-Lee, T., 2005. An RDF language for the Semantic Web. https://www.w3.org/DesignIssues/Notation3.

Bettini, L., 2016. Implementing Domain-Specific Languages with Xtext and Xtend. Packt Publishing Ltd.

Breaux, T. D., Hibshi, H. and Rao, A. 2014. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*. 19, 3, 281-307.

Caramujo, J., Silva, A.R., 2015. Analyzing Privacy Policies based on a Privacy-Aware Profile: the Facebook and LinkedIn case studies. In *Proc. of the 17th CBI conference*. IEEE, 1, 77-84.

Caramujo, J., et al., 2017. A Domain-Specific Language for the Specification of Privacy-Aware Requirements. INESC-ID Technical Report.

Cranor, L., 2002. P3P Preference Exchange Language 1.0 (APPEL) Specification, W3C, https://www.w3.org/TR/P3P-preferences.

Cranor, L., 2003. P3P: Making privacy policies more useful. *IEEE Security & Privacy*. 1, 6, 50-55.

Cranor, L. et al., 2006. Platform for Privacy Preferences 1.1 (P3P) Specification, W3C, https://www.w3.org/TR/P3P11.

[13] https://github.com/RSLingo/RSLingo4Privacy-Studio

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

Damianou, N. et al., 2002. Tools for domain-based policy management of distributed systems. *Network Operations and Management Symposium*. IEEE, 203-217.

Ferreira, D., Silva, A. R, 2012. RSLingo: An Information Extraction Approach toward Formal Requirements Specifications. In *Proc. of the 2nd MoDRE workshop*. IEEE, 39-48.

Ferreira, D., Silva, A. R., 2013. RSL-IL: An Interlingua for Formally Documenting Requirements. In *Proc. of the 3rd MoDRE workshop*. IEEE CS.

Ferreira, D., Silva, A. R. 2013a. RSL-PL: A Linguistic Pattern Language for Documenting Software Requirements, in Proceedings of RePa'13, IEEE CS.

Haley, C., Laney, R., Moffett, J., Nuseibeh, B., 2008. Security requirements engineering: A framework for representation and analysis. IEEE Transactions on Software Engineering, 34(1), 133-153.

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

Haugen, Ø., Wasowski, A., Czarnecki, K., 2012. CVL: Common Variability Language. In *SPLC*. ACM, 2, 266-267.

Kagal, L., Finin, T., Joshi, A., 2003. A policy language for a pervasive computing environment. In *Proc. of the 4th POLICY workshop*. IEEE, 63-74.

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

Kotu, V. and Deshpande, B., 2014. Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner. Morgan Kaufmann.

Landwehr, C., 2016. Privacy research directions. *Communications*. ACM, 59, 2, 29-31.

Mernik, M., Heering, J., Sloane, A. 2005. When and how to develop domain-specific languages. ACM Computing Surveys, 37(4):316-344.

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

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

OASIS, 2013. eXtensible Access Control Markup Language 3.0 (XACML) Specification. .http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

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

Ribeiro, A., Sousa, L., Silva, A. R., 2016. Comparative Analysis of Workbenches to Support DSMLs: Discussion with Non-Trivial Model-Driven Development Needs, in Proceedings of MODELSWARD'2016, SCITEPRESS.

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

Silva, A. R., 2015. Model-Driven Engineering: A Survey Supported by a Unified Conceptual Model, Computer

Languages, Systems & Structures 43 (C), 139–155.
Silva, A. R., 2015a. SpecQua: Towards a Framework for Requirements Specifications with Increased Quality, in Lecture Notes in Business Information Processing (LNBIP), LNBIP 227, Springer.
Silva, A. R., et al., 2016. Improving the Specification and Analysis of Privacy Policies: The RSLingo4Privacy Approach. In *Proc. of the 8th ICEIS conference*. SCITEPRESS, 336-347.
Silva, A. R., 2017. RSLingo's RSL: Requirements Specification Language Based on Linguistic Patterns. INESC-ID Technical Report.
Uszok, A. et al., 2003. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proc. of the 4th POLICY workshop*. IEEE, 93-96.
Voelter, M., Benz, S., Dietrich, C., Engelmann, B., Helander, M., Kats, L. C., Visser, E., Wachsmuth, G. 2013. DSL engineering: Designing, implementing and using domain-specific languages, dslbook.org.
Wishart, R. et al., 2010. Collaborative privacy policy authoring in a social networking context. In *Proc. of the POLICY symposium*. IEEE, 1-8.