ABSTRACT
Requirements engineering is an extremely important area on the development process of software systems that requires a good communication between stakeholders to achieve a shared vision of the problems to be addressed. Therefore, an efficient collaborative tool to manage requirements specification documents is crucial to guarantee an alignment of expectations between all involved parties. Collaborative RE tools currently available already offer ways to manage those documents, but are more focused on the organization and traceability of requirements, rather than the quality and rigorousness of their representation. Furthermore, there are still RE concerns which have not yet been properly addressed, such as the lack of proposals to efficiently model variability aspects of software systems at the requirements level. Therefore, a platform that could offer good baseline usability features, a formal structure for modeling RE concerns and a way to efficiently manage requirements variability would not only provide a familiar but powerful editor, as well as a set of strong reusability features. In addition, it offers an innovative approach for managing requirements variability, based on the recent OMG proposal for a domain-independent variability modeling standard: the Common Variability Language (CVL).

Keywords
Requirements Engineering, Collaborative Platforms, Reusability, Variability Modeling

1. INTRODUCTION
Requirements Engineering defines a set of tasks to build a Requirements Specification Document, most of which demand a high level of collaboration between business and technical stakeholders in order to achieve a mutual understanding of the software system to be developed. The intrinsically abstract nature that characterizes the software conceptualization phase can often lead to a misalignment of the views amongst the various stakeholders, which ultimately leads to a system that doesn’t match the initial vision [12]. Correcting the consequences of those early misunderstandings is often a costly endeavor [5], requiring a new elicitation of the requirements and a rework of the project to accommodate the changes.

A good collaborative environment to manage these requirements specifications is, therefore, crucial to guarantee an alignment of expectations and an overall better communication of the problem amongst stakeholders. Several software tools already offer ways to organize and share those documents, ranging from simple text and spreadsheet editors to more advanced commercial platforms specifically designed for the purpose. Furthermore, recent academic works have proposed wiki-based collaborative systems with integrated semantic assistants for analyzing requirements supported by RE-specific ontologies.

However, trade-offs must be made when choosing between these tools. Widely used text and spreadsheet editors can be free and easy to learn, but lack many of the advanced capabilities present in RE management commercial systems. Likewise, these systems focus on strong traceability features but are often little more than databases, lacking any meaningful ways of analyzing the requirements in regards to their correctness, consistency, completeness and ambiguity. Semantic Wikis like the ones described before do address this lack of analysis by providing mechanisms to extract relevant information contained within the requirements and verifying it against pre-defined RE ontologies in an effort to detect inconsistencies. However, and given the difficult task of analyzing contextless, ad-hoc natural language texts, most of these wikis have not ultimately provided practical results.

Moreover, there are still concerns in the field of RE for whom none of these tools provide clear solutions, such as the lack of efficient proposals to model variability in requirements specifications. This is a relevant issue because during the development of many software projects, it is common that some of the original features will need to be adapted in order to meet specific customers’ needs. To manage this variability, companies often begin developing customer-specific versions of the original product, changing and adding features for each new version. As the number of versions grow, however, so does the effort needed to model all the variation points and respective variants (alternatives for each point).
In some cases, the sheer number of these variation points increases the complexity of the project to the point of making it almost impossible to manage.

Naturally, this variability will need be reflected at the requirements specification level, but software companies prefer to express product variability in terms of features because it is closer to how designers understand and represent it [3].

This happens mainly because since its first introduction, feature modeling [11] has been the most popular technique to model commonality and variability (C&V) of products of a product line. Commonalities and variabilities are modeled from the perspective of product features, “stakeholder visible characteristics of products” in a product line that are of stakeholdersâ€™ concern [4]. With the advent of software product lines for building multiple and related products in a given domain, variability models have equally increased in popularity.

Common Variability Language (CVL) [10] comes as the OMG’s latest proposal for a Variability Modeling standard, defining itself as a domain-independent language for specifying and resolving variability. In practice, this allows the representation the system’s features variability to be independent from the DSL used to model those features. This domain-independence makes CVL an ideal language to develop a framework for modeling variability at the requirements level, incentivizing experimentation and the proposal of new approaches in the field.

This research proposes REBox, a collaborative platform for managing requirements specifications with a focus on reusability and variability modeling.

REBox is a web-based platform that offers a management system for requirements specifications. Its reusability features include a Document Template System for uploading/managing SRS templates and a Requirements Specifications Library (RSLib) for creating and reusing modular sets of reusable requirements. It also offers an innovative variability modeling framework that leverages the concepts of the CVL language to model variability aspects in the context of Requirements Engineering. The framework defines a process for applying the language’s central variability notions - the Variation Points - to the views of a structured software requirements specification document, modeled using a rigorous requirements specification language: RSL-IL [7].

The structure of this paper is as follows. Chapter 2 provides an analysis of proposals and technologies that constitute this paper’s background, namely the RSLingo proposal and the RSL-IL language. Chapter 3 introduces the REBox platform main goals and chapter 4 details its CVL-based variability approach to requirements modeling, further illustrating the concepts introduced though the use of a concrete example. Chapter 5 discusses the strengths and limitations of this approach, based on a pilot-user test session and the comparison with the related work on the area of requirements variability modeling. Finally, Chapter 6 concludes the paper and lays down some ideas for extending the current work.

2. BACKGROUND

REBox is a web-based collaborative platform for managing requirements specification documents that’s integrated in the RSLingo initiative for Requirements Engineering. This initiative is based on the RSLingo [6] proposal, which states that natural language, although being the most common and preferred form of representation used within requirements documents, is prone to produce ambiguous and inconsistent specification documents [9].

The proposal presents an approach to use simplified Natural Language Processing (NLP) techniques for capturing relevant information from these Ad-Hoc natural language requirements specifications and then applying lightweight parsing techniques to extract domain knowledge encoded within them. This is achieved through the use of two Requirements Specific Languages (RSL): the RSL-PL(Pattern Language), designed for encoding RE-specific linguistic patterns and RSL-IL(Intermediate Language), a Domain Specific Language (DSL) designed to address RE concerns and better support RE-related tasks. 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 formal structure, reducing its original ambiguity and creating a new and more refined software requirements specification document.

This approach foresees the whole process of knowledge extraction and conversion to a more rigorous representation as a way to help business stakeholders without knowledge in the area of Requirements Engineering in having a deeper insight on the problem at hand, and thus better understanding the implications of the set of stated natural language statements that represent their real requirements [6].

However, to a technical user already familiar with the concepts of requirements engineering, the natural language process phase can be omitted, as there is already a much better understanding of the RE concerns and a deeper knowledge of its documentation process. As such, the REBox system doesn’t focus on the automatic extraction part of the process, but instead on the formal representation of RE-specific concerns provided by the RSL-IL language [7]. The language defines several constructs that are logically arranged into viewpoints according to the specific RE concerns they address. For example, the people and organizations that can influence or will be affected by the system are represented in the Stakeholders viewpoint. Likewise, the objectives of business stakeholders regarding the value that the system will bring are expressed in the Goals viewpoint. This multi-view architecture defined in the RSL-IL language has since been adapted to an Excel format: the “RSL-IL Excel Template” [16]. This Software Requirements Specification (SRS) template encodes the different viewpoints within their own particular sheet, defining a set of properties for each one.

3. THE REBOX PLATFORM

With the SRS template established, the goal was to create a collaborative environment for its management, given the requirements engineering process intrinsic need for a straight cooperation between the all the involved stakeholders in an effort to achieve a shared understanding of the system.

With that in mind, the REBox system’s main goals can be summarized as follows:

Create a collaborative environment for managing requirements specification documents: The platform...
major purpose is to provide a collaborative editor for software requirements documents. Spreadsheet editors, although lacking many of the fundamental features of the commercial dedicated RE tools, have always been a popular choice to manage requirements specifications due to their widespread availability and simple interface, making them a good baseline tool for nontechnical stakeholders. Until recently, one of the main problems with this desktop software was to maintain synchronized and updated versions of the documents in decentralized projects requiring multiple concurrent changes. Recent cloud storage services, however, such as Google Drive or OneDrive, have helped on that regard by offering collaborative mediums to manage these type of documents with automatic synchronization and lightweight versioning tools. As such, the REBox platform makes use of the Google Drive and Google Sheets APIs to provide both the editor and a cooperative environment. The first is used for uploading the local SRS documents, sharing them and granting access permission to its users, while the second offers all the data manipulating functions for extracting, and editing the information contained within the platform documents;

Integrate with RSLingo concepts and technologies: As stated before, this includes in particular the RSL-IL language and its current Excel representation. Although the platform allows for the usage of any format of SRS document, all the features that involve data manipulation expect the current format of the RSL-IL template. Besides the RSL-IL language, REBox also aims to integrate its features with other tools currently being developed in the scope of the RSLingo field of investigation;

Provide a set of Reusability and Variability mechanisms: To complement the document editor, the system makes use of the data manipulating capabilities provided by the integration with the Google APIs to offer a set of features to increase the efficiency of its processes. The main feature is the variability modeling approach based on CVL that will be described in depth in the next chapter. The other two main features focus on Reusability and consist of a Templating System for requirements specification documents and a Requirements Library system for requirements reuse. The former allows for any previously developed template in a spreadsheet format to be uploaded to the platform and later used to create new projects based on its structure, while the latter allows the creation of libraries of coarse-grained and potentially generic requirements that can afterwards be edited if necessary and added to any new project;

The collaborative tools described in the related work section can be broken into three types: Spreadsheet Editors, Dedicated RE platforms and RE Wikis. REBox, through Google Sheets, wholly incorporates the first type - the spreadsheet editor - and all its benefits, namely its familiarity to most users and its focus on usability. Regarding the second type - RE Dedicated Tools - their main strength is often related to strong management and productivity features. Although not as extensively, REBox also strives to tackle this, through its Templating System, Requirement Library and even though the Google Sheets, who besides its editing capabilities, also provides lightweight versioning and rollback mechanisms. Finally, regarding the third type - RE Wikis - they propose innovative ideas to analyze and validate the quality of requirements based their syntax and semantics. Again, REBox is often not as ambitious as some proposals, but guarantees textual validation though its integration with the RSLingo Studio system.

Figure 1 provides an overview of the REBox platform, its internal features and the way it integrates with the Google Drive Service and the RSLingo Studio tool. The next sections describe each of those components in detail.

3.1 Template Management System

The template management system, like its name implies, offers the possibility to create and manage reusable requirements templates that can be later utilized to bootstrap new projects. Like the RSDoc editor, this system also makes full use of the APIs provided by Google. The process of creating a template in the platform encompasses two steps. The first one consists on the manual upload of a spreadsheet file containing the template into the platform. To note that this document doesn’t need to hold the finalized template (it can even be empty), as REBox allows it to be further edited once its uploaded. Once the file is uploaded, it is saved into the local filesystem and the second step begins. This time, the file is automatically re-uploaded, this time to the Drive cloud storage server and then shared with the users of the platform. From this point on, it can be accessed and edited concurrently like any other document, and it can be used as the base structure for new project created in REBox.

3.2 RS Library

REBox’s Requirements Specifications Library offers the possibility to create modules of reusable requirements that can then be added to any RSProject. Requirements reusability across multiple projects is already one of the main ideas being explored by some of the platforms presented in the Related Work in regards to increasing the efficiency of RE activities. To that end, REBox RS Library serves as an open “database” of modular, agnostic and coarse-grained requirements, that can be grouped by areas of interest, such as Security or Multi-Language, etc.. These groups of requirements can be created manually, extracted from an existing project or uploaded directly into the platform. Furthermore, after created, the user can still add and remove requirements to the module, as well as edit the existing ones, allowing for an iterative process of refinement. The process of adding a Requirements Module to a project is very simple, since the new requirements only need to be “appended” to the existing

---

2https://developers.google.com/drive/
3https://developers.google.com/sheets/
3.3 RSLingo Studio Integration

As described in chapter 3.3, the RSLingo Studio tool is another system currently being developed in the scope of the RSLingo initiative. Like REBox, it provides an editor for creating and editing RSL-IL requirement specification documents. However, while REBox focuses on a more user-friendly spreadsheet representation, Studio focuses on the textual representation of the language. By defining a DSL containing the language’s grammar, it used the xText framework to generate a parser and typechecker for RSL-IL. This tools, allied to features like syntax highlighting, error checking, auto-completion and source-code navigation, provide very powerful means for automatically analyzing and validating the correctness of a RSL-IL document. Therefore, the integration with the RSLingo Studio tool would provide a very big improvement in terms of requirements quality validation. From the start of the project that the integrating with other RSLingo technologies was defined as one of the main focuses, and indeed, the choice to use the Google Drive technology allows the possibility not only to upload, but also to download the spreadsheet files. Those files can then be manually uploaded to the RSLingo Studio tool and converted into the textual RSL-IL representation through its M2M transformations. The user can then receive visual feedback about the correctness of the document.

4. THE REBOX VARIABILITY MODELING APPROACH

The REBox process of variability modeling closely follows the proposed CVL approach, as it can be seen in Figure 2. A detailed description of the process is presented below.

4.1 Variation Points Configuration

At the start of the variability modeling process, the user selects the SRS document to be used as the Base Model. In REBox, the Base model is an instance of a requirements specification document created using the RSL-IL language. This document can either be part of a project already present in the platform, or can be manually uploaded into the system from an outside source, provided it conforms to the RSL-IL language. The system then automatically parses the document and extracts the requirements from the RSL-IL supported views, namely the Goals, Functional Requirements and Quality Requirements. It is then, regarding these extracted requirements, that the user will define the variation points. As defined by CVL, variation points are specifications of concrete variability in the base model, defining specific modifications to be applied to it during the materialization phase. Variation points can take various types, which will define the nature of the dependency between a VSpec and the corresponding Variability Point in the materialization phase.

In REBox, variation points can be defined on two levels: (1) the requirements themselves and (2) their properties. Variation points defined over a requirement are of the Existence type, meaning that in the materialization phase, they will be bound to a Choice VSpec. On the other hand, variation points defined over a requirement property are of the Value Assignment type, and will be bound to a Variable VSpec.

4.2 VSpecs Binding

After the user defines all the variation points over the base model, the REBox system automatically generates the first part of the variability model: the var.specs view. In CVL each variation point references a single VSpec, in a pro-
Figure 2: The REBox Variability Modeling process.

Table 1: Properties of the var.specs View

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Unique Identifier of the VSpec</td>
<td>String</td>
</tr>
<tr>
<td>Name</td>
<td>Descriptive name of the VSpec</td>
<td>String</td>
</tr>
<tr>
<td>Type</td>
<td>The associated Element Type</td>
<td>VSReqGoal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VSReqFunctional;</td>
</tr>
<tr>
<td>ElementId</td>
<td>The associated Element Id</td>
<td>&lt;Elements Ids&gt;</td>
</tr>
<tr>
<td>ElementName</td>
<td>The associated Element Name</td>
<td>&lt;Elements Names&gt;</td>
</tr>
<tr>
<td>VarTokens</td>
<td>List of the Element's</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>variable properties</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Description of the VSpec</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 2: Properties of the var.specs.constraints View

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Unique Identifier of the Constraint</td>
<td>String</td>
</tr>
<tr>
<td>Name</td>
<td>Descriptive name of the Constraint</td>
<td>String</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the constraint</td>
<td>Simple; LogicExpression</td>
</tr>
<tr>
<td>VSpecsIds</td>
<td>The list of VSpecs that are</td>
<td>&lt;VSpecs Ids&gt;</td>
</tr>
<tr>
<td>LogicalExpression</td>
<td>The propositional logic statement</td>
<td>String</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the Constraint</td>
<td>String</td>
</tr>
</tbody>
</table>

cess defined as Binding, which provides the linkage between variability realization and variability abstraction. Figure 3 details the variability binding model used by the REBox approach. The var.specs view contains all the information about the newly generated VSpecs and its full list of attributes are presented in Table 1. To note that this view introduces a new concept associated to VSpecs: the VarTokens. VarTokens are nothing more than "children" VSpecs that are associated with a particular attribute of the requirement that serves as its variation point. A VSpec can have any number of VarTokens, depending on the number of attributes that were considered as variation points in the last step. Regarding the type, "parent" VSpecs that are associated with requirements are defined as Choices, meaning that in the materialization phase they will require a binary yes/no decision to define if that particular requirement is going to exist in the resolved model. Contrarily, the VarTokens are of type Variable, determining that in the materialization phase these VSpecs will require providing a resolution value of their specified type.

Furthermore, the user is then able to define constraints over the VSpecs that were created. In CVL, constraints are used to express intricate relationships among VSpecs that cannot be directly captured by hierarchical relations. To achieve this, it relies on a simplified version of the OCL[14] language. The user can utilize both the parent VSpecs and their children VarTokens when building the logical expressions that serve to express these constraints. A more in-depth example of this process is shown in chapter 4.4.

After all the constraints are created, the system then generates the var.specs.constraints view, shown in Table 2. With both views created, the Variability Model is finalized and saved.

4.3 The Materialization Process

The final step in the variability modeling approach is the Materialization process. As stated in the CVL proposal, this process consists of transforming a base model into a new document by applying variation points. Materialization is driven by a resolution model which provides resolutions for the VSpecs defined against the base model. In REBox, after the Variability Model is created in the previous steps, its information is stored in the system’s database. From that moment on, whenever a new requirements specification document creation process starts, the user is given the option
to, instead of creating a new empty document, generate a new one by defining a set of resolution values for an existing variability model. When this happens, the platform crosses the information contained in the Variability Model with the resolution values provided by the user and checks every constraint for possible inconsistencies. If nothing is found, a new Resolved Model is generated.

4.4 Example - Reliability System Requirements

Figure 4 provides a concrete example of Variability Model generation process described above. This example describes a fictional system’s set of reliability-focused quality requirements, which specify its expected uptime over an year, required maintenance procedures and fault logging details.

The first table of the image shows an exact representation of a RSL-IL Excel template Quality Requirements view. The view defines a set of attributes to characterize the requirement, such as an unique Id, a name, a type and sub-type, a quantitative metric and its value (if applicable), the stakeholder who proposed the requirement, its parent requirement(s), a description and finally its priority in the context of the current project.

The second table presents an example of the “Variation Points Config” step defined in Figure 4. As described before, this step starts with the automatic extraction of the Goals, Functional Requirements and Quality Requirements from the base model defined in the RSL-IL language and consists in the manual configuration of the model’s Variation Points. To define a requirement as a variation point, the user just needs to check the “VarPoint?” field on that requirement’s row. This action will mark that requirement as an Existence variation point, meaning that it be linked to a Choice VSpec in the Binding phase. Furthermore, field “VarTokens” lets the user select which of that requirement’s attributes will also be modeled as children variation points of the type Value Assignment. In the example, from the five initial requirements, two were selected as variation points: QR_R_1 and QR_R_3. The first one has two attributes modeled as children variation points: “Description” and “Priority”. However, not all of the content of the “Description” field is to be considered for variability purposes, but only the substring “99%”, denoted by the suffix .value#99%.

The third table shows a simplified version of the .var.specs view (for the sake of space, some attributes were omitted; the full list is displayed in Table 1). This view is automatically generated by the REBox system after the user selects the variation points in the previous step. This process represents the Binding phase of CVL, where each of those points are linked to their abstract representation in the variability model. To the right is an illustrative example of the corresponding VSpecs tree representation, which shows the six VSpecs created based on the input provided in the second table. The parent VSpec corresponds to the system itself. On the second row, the QR_R_1 and QR_R_3 requirements are represented as Choice VSpecs, meaning that in the Materialization phase, both resolutions will require a binary yes/no decision. These elements are linked to their parent by a dashed edge, representing a False “IsImpliedByParent” value, making their materialization value independent of the one their parent takes. Finally, on the last row are shown thee children VSpecs of type Variable, generated by the VarTokens defined in the second table and their corresponding types.

The fourth table presents an equally simplified version of
5. EVALUATION

The evaluation of this work consisted in two approaches. The first one focused on the assessment of the platform by test users that were not previously involved in the research work. This assessment was done by conducting a session where the participants had to make full use of the platform’s functionalities. In the end of the session, the participants filled in a questionnaire focused on three aspects: (1) the Platform, (2) the Variability Modeling Framework (3) the General Approach. The results obtained from the questionnaires were generally encouraging with positive scores in all three analyzed aspects. The least positive aspects were regarding the variability modeling process and its concepts, which were not considered very intuitive and need to be better explained to the users.

The second approach consisted in evaluating the proposed variability framework against related proposals and is presented below.

Extensive research has already been conducted regarding requirements variability in the context of SPLs, leading to proposal of models[13] and languages[1] to address its challenges. Furthermore, with the advent of CVL, proposals have been made to leverage its concepts to software product lines. However, and given that CVL is a relatively recent language, not much research has yet been conducted to define clear ways of applying its concepts and models to the specific constructs of the RE field.

Regarding requirements variability in the context of SPLs, the work presented in [13] proposes to integrate that variability directly into its requirements metamodel, REMM (Requirements Engineering MetaModel). Like RSL-IL, REMM encompasses a comprehensive set of RE concepts and relationships between them. Furthermore, REMM allows requirements engineers to define optional requirements by means of constraints on VSpecs, which are defined in a constraint language and are used to express relationships between VSpecs.
of OR parent-child traces and supports the definition of parameterized requirements, both being variability mechanisms that are often modeled using separate feature models. With this approach, they are provided by a unique metamodel (REMM), making it possible to keep all the information about the requirements and their variability points connected, and to make explicit the relationships between the requirements parameters (variability points) by means of the explicit relationships existing between requirements.

Another proposal suggests the use of a compositional strategy. VML4RE (Variability Modeling Language for Requirements)[1] presents a multi-view composition language for SPL requirements with two main contributions. First, the VML4RE language - a Domain Specific Language (DSL) capable of composing requirements with feature models; and second, a requirements process that uses: 1) a feature model to perform the variability identification; 2) use cases and activity diagrams to describe the SPL domain requirements; and 3) a VML4RE model to relate the feature model with the requirement models.

Additionally, projects to leverage CVL concepts for variability management in Software Process Lines have also been proposed. The work presented in [15] uses CVL to define an SPL and automatically derive a process according to that projectâ€™s requirements. It defines its base model as a base process model and then build a variability abstraction model(VAM) that specifies the projects requirements variability. The main benefits of this approach is the ability to capture the variability requirement in the VAM and its materialization in the base model using the VRM, which defines a binding between the projects requirements variability and the processes variability.

A more recent proposal introduces a multimodel approach for specifying requirements variability on SPLs[2]. This proposal is based on the Feature-Driven Requirements Engineering approach(FeDRE)[8], which provides support to the requirements specification of SPLs. It proposes an extension of FeDRE to enable the specification of requirements variability in multiple models at the domain engineering stage by applying model-driven engineering principles. Furthermore, the proposal defines an integration with CVL in order to obtain the product requirements from the domain requirements by applying model transformations at the application engineering stage.

To conclude, there are already proposals for modeling requirements variability in the context of product lines, with different kinds of approaches and addressed concerns. Yet, whether using the newly proposed CVL language or not, most of them still focus greatly on the mapping between requirements and the feature modules of the product. The REBox variability approach main focus, however, is to correctly model the variability of the RE concepts, regardless of the software features they define. It achieves this by using the RSL-IL rigorous language to encode its requirements specifications, which encompasses a comprehensive set of RE concepts and relationships between them. Furthermore, it makes full use of REBox web platform capabilities to greatly automate this process and provide distributed access to its data.

6. CONCLUSIONS

This research has proposed REBox, a collaborative web-based platform for managing requirements specification documents with an innovative approach for modeling variability aspects in RE using the CVL language. To promote the efficiency of its processes, it offers a set of reusability features, including a Document Template System for uploading/managing SRS templates and a Requirements Specifications Library (RSLib) for creating and reusing modular sets of reusable requirements. Its variability modeling framework defines a process for applying the CVL language’s central variability notions, the Variation Points, to the views of a structured software requirements specification document. The language used to model that rigorous requirements specification in REBox is RSL-IL.

The web platform has been built using the DotNetNuke .NET web CMS and employs the Google Cloud Drive storage service. This technology provides both the Requirements Specification Document (RSDoc) editor though Google Sheets, as well as the collaborative environment through its synchronization system and distributed architecture. In addition, it uses a series of Google provided APIs for automating processes and managing data. Thanks to this, the REBox variability modeling approach can automatically extract the information present in a specification document, modify it and generate new documents and views as needed. This allowed to define an automatized process for applying CVL concepts in the context of requirements engineering.

We believe that the REBox platform provides a well-rounded collaborative RE platform which combines some of the strong points of other tools available, while offering an innovative way to model variability aspects in the field of requirements engineering using the CVL language. Furthermore, it is based on an extensible technology, the Google Drive and its APIs, making it very easy to develop new features regarding the extraction and analysis of data from RE documents.

Future work will focus on expanding the current RSL-IL language supported views, allowing to progress from simply modeling variability at the requirements level (Goals, Functional and Quality requirements), to all the other viewpoints, like the Stakeholders, Entities, Use Cases etc., enabling a much wider scope of variability points within the spectrum of requirements engineering concerns. Furthermore, the long-term objective is to fully integrate this process within the RSLingo natural language information extraction approach, expanding the entry point of the variability modeling process from formal RE documents to more unstructured Ad-hoc natural language requirements specifications.

7. REFERENCES


