Variability Specification and Resolution of Textual Requirements

Alberto Rodrigues da Silva and João Costa Fernandes

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

Keywords: Requirements Specification, Variability Modelling, Reusability, Software Product Line.

Abstract: Since software product lines emerged, various techniques have provided for commonality and variability modelling of functionally similar products within a given domain. However, so far the emphasis of variability modelling proposals has mostly been on the solution rather than the requirements level, which is mainly due to stakeholders often associating variability with the software implementation instead of the problem analysis. Taken into consideration the positive impact that a high-quality system requirements specification plays within a software project, this paper proposes and evaluates an innovative approach for the modelling and management of variability at the requirements level, based on the Common Variability Language (CVL), the OMG proposal for a domain-independent variability modelling standard. This approach has been implemented as a core feature of the ITBox system, a Web-based collaborative platform for the management of technical documentation.

1 INTRODUCTION

In order to meet specific customer needs, companies often begin with developing customer-specific versions of their original products by changing and/or adding features. On one hand, defining a product by a hierarchized structure of functional capabilities, such as a feature model, does not include neither cross-cutting capabilities, nor multiple abstraction levels and constructs or views (for instance, a traditional billing system can be break-down into features such as customers, suppliers, invoices, payments, products and services, whereas a requirements specification for the same product would involve not only those functional capabilities, but also cross-cutting capabilities such as non-functional requirements like security, availability, usability, etc.) On the other hand, a high number of customer-specific versions demands for a high effort in variability and even project management e.g. the specification of variation points and respective variants. OMG's CVL (Common Variability Language) is the proposal of a domain-independent language for specifying and resolving variability (OMG, 2012).

Diverse authors argue that the activity of variability modelling shall be initiated as early as during the RE (Requirements Engineering) stage (Coplien et al., 1998; Verelst et al., 2013; Silva et al., 2014; Blanes et al., 2014). Given the vital role

that a well-defined SRS (System Requirements Specification) document plays in the success of a project (Davis, 2005), rigorous means of expressing variability concerns in those documents are of extreme importance. Despite this, little research has investigated ways of applying the concepts of CVL to the RE spectrum, which does not stand for a recognition of OMG's effort to establish a standard targeted at the representation of variability. The approach proposed in this paper, however, is targeted at specifying the C&V (Commonalities and Variabilities) of RE concepts.

Fernandes (2016) and Silva, Fernandes, and Azevedo (2017) proposed an approach for leveraging the concepts of CVL and its domain-independence to the specification and resolution of variability in the context of RE, specifically in structured SRS documents defined with a rigorous RSL (Requirements Specification Language). This paper discusses how this approach was translated into a core feature of the ITBox system, a Web-based collaborative platform for the management of SRSs. That feature is focused on the reusability of requirements starting from C&V modelling.

The remainder of this paper is structured as follows. Section 2 refers to the ITLingo initiative, the RSL and the OMG's CVL. Section 3 introduces the ITBox collaborative platform. Section 4 details the CVL-based variability approach to requirements modelling supported by the ITBox. Section 5

provides for a demonstration case of that approach. Section 6 presents and discusses the evaluation of the proposed approach. Section 7 analyses some related work. Finally, section 8 presents some concluding remarks and future work.

2 BACKGROUND

2.1 RSLingo Initiative

RSLingo is a long-term research initiative in the Requirements Engineering area (Ferreira and Silva, 2012; Silva, 2017). It is a linguistic approach to improve the quality of requirements specifications. Although being the most common and preferred form of representing requirements, natural language is prone to producing ambiguous and inconsistent documents, hard to automatically validate or transform into other kinds of artefacts.

Originally, RSLingo proposed two languages: the RSL-PL (Pattern Language) (Ferreira and Silva, 2013a), designed to support the encoding of RE linguistic patterns, and RSL-IL (Intermediate Language) (Ferreira and Silva, 2013), a domain-specific language designed to primarily support the elaboration of SRSs. Together, these two languages allow domain knowledge written in natural language to be extracted, parsed and converted into a more structure format, reducing its original ambiguity, and creating a new and more rigorous SRS document (Silva, 2015a). This process of extracting knowledge and converting it into a more rigorous representation has in itself a way of providing business stakeholders with a better understanding of natural language statements that represent requirements.

2.2 RSL

Recently, broader language, called RSL, has been designed (Silva, 2017; Silva, 2018). It is a comprehensive domain-specific language (Silva, 2015) designed to address general-purpose RE activities such as the rigorous specification, automatic validation, persistence and management of software requirements. It is based on other languages such as Pohl's (Pohl, 2010), XIS* (Silva et al. 2007; Ribeiro and Silva, 2014; Ribeiro and Silva, 2014a) and SilabReq (Savić et al., 2015).

RSL is, in fact, a controlled natural language to help with the production of SRSs in a more systematic, rigorous and consistent way. It is a language that includes a rich set of constructs logically arranged into views according to multiple

RE-specific constructs situated either at the business or at the system abstraction levels. These constructs are defined as linguistic patterns and textually represented by mandatory or optional fragments (text snippets). For example, people and organizations that can influence or can be affected by the system are represented by the construct *Stakeholder*. Likewise, the goals of business stakeholders towards the system and the value it represents to them are represented by the construct *BusinessGoal*.

RSL is a process- and tool-independent language that can be used and adapted by multiple users or organizations with different processes, as well as supported by multiple types of software tools. However, in practical terms, RSL has been implemented with the Xtext (http://www.eclipse.org/Xtext/) framework (Bettini, 2016), which means that RSL specifications are rigorous, and can be automatically validated and transformed into other representations and formats. A lightweight tool support is provided with the ITLingo RSL Excel Template (https://github.com/RSLingo/RSL-Excel-Template) publicly available at GitHub. This Excel template encloses different viewpoints (shortly views) organized into sheets and described by a set of properties for each one of them. As an example, the key properties of the *Goals* view are shown in Table 1.

Table 1: Properties of the *Goals* view from RSL.

Name	Description	Type/Values
Id	Unique identifier of the goal	string
Name	Descriptive name of the goal	string
Type	Type of goal	Concrete; Abstract
Source	Reference to the stakeholder who has or defined the goal	<stakeholder id>
PartOf	Reference from the current goal to its parent goal	<parent goal id>
Description	Description of the goal	string
Priority	Level of priority for the goal	Must; Should; Could; Won't
Progress	Current status of the development process	Plan; Design; Develop; Test; Deploy; Concluded

2.3 CVL

CVL is the OMG's proposal of a domain-independent language for specifying and resolving variability over models of any MOF-based language (OMG, 2012). Figure 1 shows the CVL execution (or materialization) process, which involves the following models: the *Base Model*, the *Variability Model*, *Resolution Models* and *Resolved Models*. Although involved in the CVL execution process, the *Base Model* is not part of the CVL. It consists of the definition of the product line in any MOF-compliant language. This compliance makes of CVL a domain-independent language. The *Variability Model* is a collection of variation points, VSpecs (further explained in section 3) and constraints used in the specification of variability over the Base Model. These concepts and the integration between them constitute the core of the CVL. The *Resolution Model* consists of a collection of VSpec resolutions that resolve the VSpecs of a Variability Model. Finally, the *Resolved Model* is a model produced by the materialization of the Base Model according to a Resolution Model.
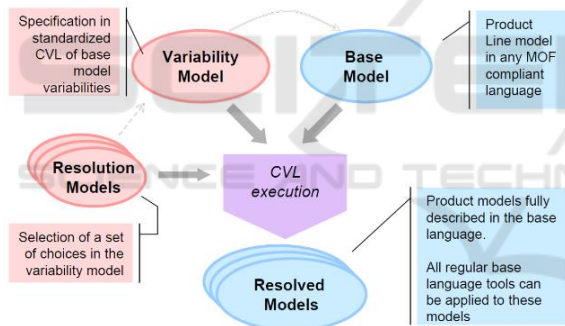

Figure 1: Overview of the models involved in the CVL execution process (extracted from (OMG, 2012)).

3 THE ITBox PLATFORM

ITBox is a Web-based collaborative platform for the management of SRSs and other technical documents. Although ITBox can support multiple types of documents, this paper focuses only on SRSs. Figure 2 shows a screenshot of ITBox. The users of this platform can author, review and validate requirements based on the constructs of the RSL. Given that RE processes involve an intense cooperation between many stakeholders (e.g. customers, domain experts, requirements engineers and software developers), the key design goals of ITBox were the following.
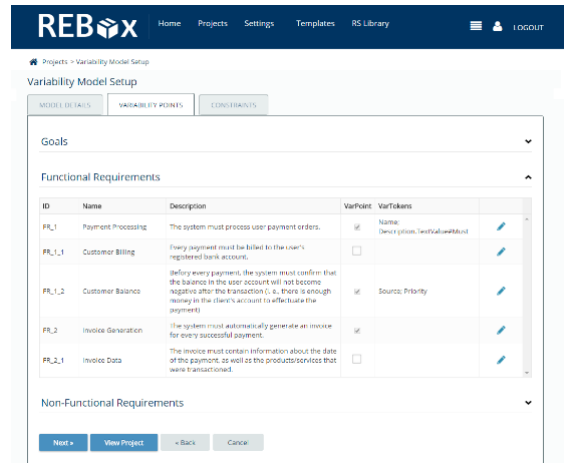

Figure 2: The ITBox (variability point's selection screen).

Provide for a collaborative environment to support the management of SRSs with an easy to learn and easy to use interface

The major purpose of ITBox is to provide for a collaborative editor of SRS documents. Spreadsheet editors, although lacking many of the fundamental features of the commercial RE-dedicated tools, have always been a popular choice to manage SRSs 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 documents in decentralized projects that require multiple concurrent changes. However, recent cloud storage services, such as Google Drive or OneDrive, offer collaborative means to manage this type of documents like automatic synchronization and lightweight versioning. In that sense, ITBox uses Google Drive and Google Sheets APIs to provide for authoring in a cooperative environment. The first is used for uploading the local copies of SRS documents, sharing them and granting access permissions to its users, whereas the second offers all the data manipulation functions for extracting and editing the information contained within the documents.

Integrate with ITLingo concepts and technologies

This key design goal is related to the adoption of the constructs defined in the RSL. Although ITBox allows for the usage of any format of SRS document, all the features that involve data manipulation expect the current version of the RSL Excel template.
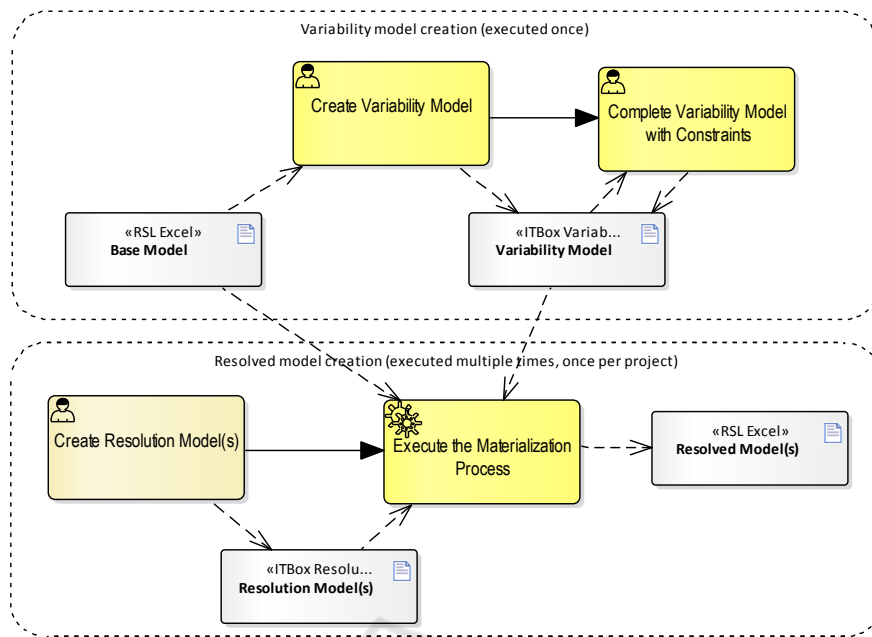
Figure 3: The ITBox Variability Modelling approach (represented as a BPMN diagram).

Include reusability and variability capabilities

The variability features have to do with the variability modelling approach based on the CVL that this paper reports, which is described in more depth in the next section. It is of utter importance to note that a model in not necessarily a graphical, rather a representation of requirements and their variability, in the case of this paper. The reusability features provided by the ITBox contemplate the management of *SRS templates* and of *SRS libraries*. The former (management of SRS templates) allows any previously developed template (defined in the format of a spreadsheet) to be uploaded into the platform and later used to create new SRSs based on the structure of that template. The later (management of SRS libraries) allows the creation of libraries of coarse grained and potentially generic requirements that can afterwards be added (and edited if necessary) to any new project. The ITBox variability modelling process is intimately related to the management of SRS libraries.

4 THE ITBox VARIABILITY MODELLING APPROACH

The ITBox variability modelling process closely follows the approach proposed by CVL, as Figure 3 depicts. Complementarily, Figure 4 illustrates the main concepts of the approach and the relationships between them. A detailed description of the process is presented below.

4.1 Create Variability Model

The variability modelling process begins with the user selecting the SRS document (or SRS library) to be used as Base Model. In ITBox the Base Model is a SRS document based on the RSL Excel Template. That document can either be part of a project already in the platform or manually uploaded into the platform (it has, though, to conform to the RSL). ITBox, then, automatically parses the document and extracts the requirements for each of the views supported by RSL (e.g. *Goals*, *Functional Requirements* and *Quality Requirements*). Afterwards, the user can define variation points over the extracted requirements. As defined by CVL, *variation points* are specifications of "concrete" variability in the Base Model and are part of Variability Models. They indicate modifications that the Base Model suffers during materialization. Variation points are bound to VSpecs, which means that the application of a variation point to a Base Model, during materialization, implies the resolution of a VSpec.

According to CVL, a *VSpec* is a specification of "abstract" variability and is also part of a Variability Model. In order to materialize a Base Model, a Variability Model is applied over it and VSpecs are resolved. The nature of the dependency is specific to
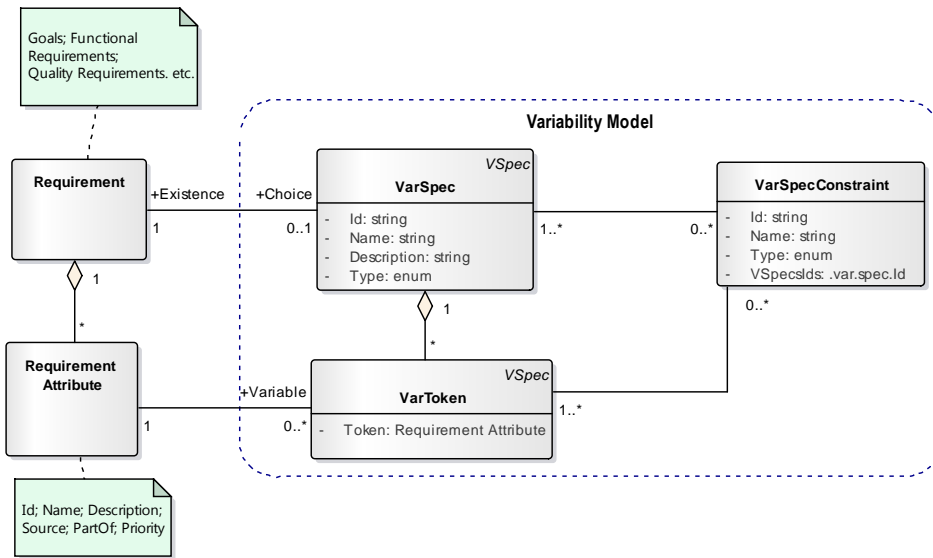
Figure 4: The ITBox Variability Model (represented as a UML class diagram).

the type of variation point. In ITBox, variation points can be defined over (1) requirements themselves and (2) their properties. *Variation points defined over a requirement* are of the `Existence` type, meaning that, during materialization, they will be bound to a `Choice` VSpec. *Variation points defined over a requirement property* are of `Value Assignment` type, meaning that, during materialization, they will be bound to a `Variable` VSpec. Once the user has defined all the variation points over the Base Model, ITBox automatically generates the first part of the Variability Model: the VarSpecs view. In CVL, each variation point references a single VSpec. Figure 4 details the variability binding model used by ITBox. *VarSpec* is the name given by the ITBox approach to the concept of VSpec in CVL. Table 2 presents the key properties of the *VarSpecs* view. This view introduces a new concept associated with VarSpecs (which refers also to the concept of VSpec in CVL): the VarToken. *VarTokens* are children VarSpecs associated with a particular property of the requirement in the Base Model. Due to space limitations, we are not presenting a table with the key properties of the *VarTokens* view. A VarSpec can have any number of VarTokens, depending on the number of properties previously considered as variation points. VarSpecs, which associated with requirements, are defined as of type `Choice`, which means that, during materialization, its resolution requires a binary (yes/no, true/false) decision to define if that particular requirement is going to exist in the Resolved Model. VarTokens are of type

`Variable`, which means that, during materialization, its resolution requires providing a resolution value of their specified type.

Table 2: Properties of the *VarSpecs* view from RSL.

Name	Description	Type/Values
Id	Unique identifier	String
Name	Descriptive name	String
Type	VarSpec type	VSReqGoal; VSReqFunctional; VSReqQuality
ElementId	The associated Requirement id	<Requirement Id>
ElementName	The associated Requirement Name	<Requirement Name>
VarTokens	List of VarTokens	List of <VarTokens>
Description	Description of the VarSpec	String

4.2 Complete Variability Model with Constraints

After the first part of the Variability Model is generated, the user is able to define constraints over the VarSpecs that were created. In CVL, constraints are used to restrict the allowed resolutions of VSpecs and can be defined either globally or in the context of a particular VSpec. To achieve this, CVL relies on a restricted subset of the OCL (Object Constraint Language). Both parent VarSpecs, as

RSL-IL Quality Requirements View

Id (*)	Name (*)	Type (*)	Sub-Type	Metric	Value	Source (Stakeholder)	PartOf	Description	Priority (*)
QR_R_1	System Uptime	Reliability	Other	Hours	~88	stk_enterprise		The system should assure an annual Uptime of 99%.	Should
QR_R_2	System Maintenance	Reliability	Other	Hours	1	stk_enterprise		The system should have a weekly maintenance period of 1 hour.	Should
QR_R_2_1	Maintenance Actions	Reliability	Other	...		stk_tech_admin	QR_R_2	The maintenance should be performed by a technical administrator of the system and should consist of a verification of the system fault log and any other reported problems that occurred since the last maintenance.	Should
QR_R_3	Fault Logging	Reliability	Other	...		stk_enterprise		The system must register all the faults occurred at runtime in a specific log file.	Must
QR_R_3_1	Fault Log Information	Reliability	Other	...		stk_enterprise	QR_R_3	Each logged fault must contain the file and line where the fault originated, the date and hour of the ocurrence and a stack dump from the moment when it occurred.	Should

REBox Variation Points Selection

ReqId	Name	VarPoint?	VarTokens
QR_R_1	System Uptime	x	Description.Value#99%; Priority
QR_R_2	System Maintenance	-	-
QR_R_2_1	Maintenance Actions	-	-
QR_R_3	Fault Logging	x	Description
QR_R_3_1	Fault Log Information	-	-

.var.specs view

VSpecID	ReqID	VarTokens
VS_1	QR_R_1	Description.Value#99%; Priority
VS_2	QR_R_3	Description

.var.specs.constraints view

ConstraintId	VSpecs	VarTokens
VC_1	VS_1	VS_1:Description.Value#99% ≥ 99.9% → VS_1:Priority = Must

VSpecs Tree

System
- QR_R_1
  - Description.Value#99%: int
  - Priority: string
- QR_R_3
  - Description: string

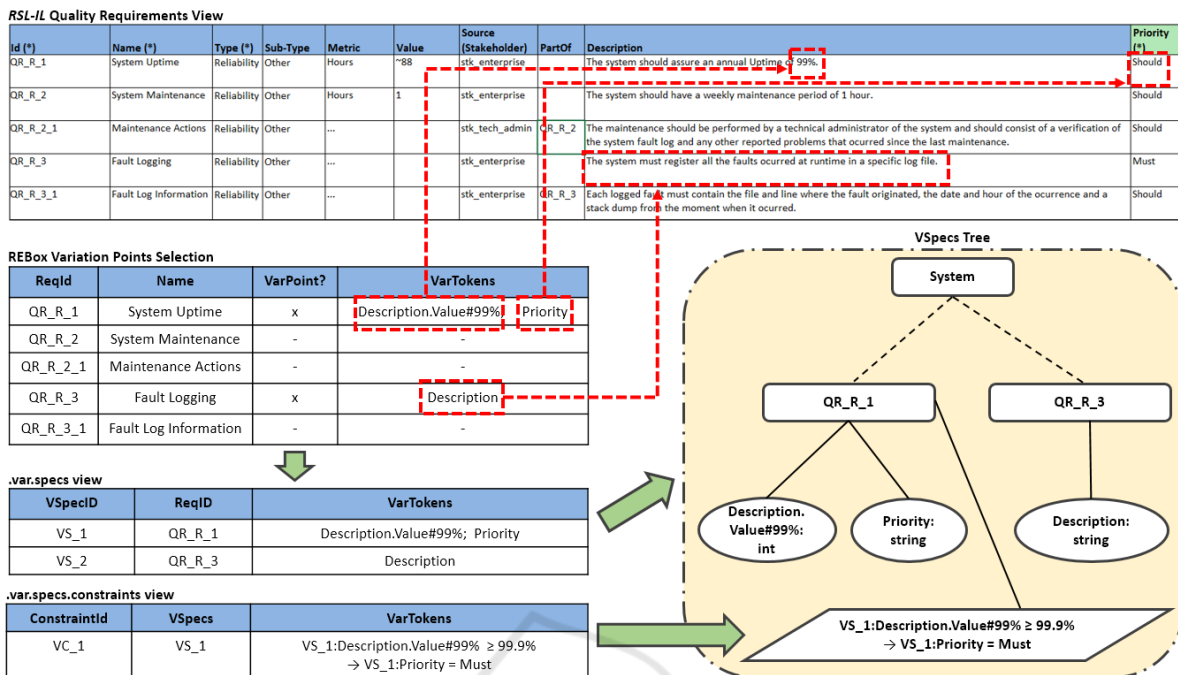VS_1:Description.Value#99% ≥ 99.9% → VS_1:Priority = Must

Figure 5: An example case of the definition of variation points in ITBox, and the associated VSpecs and constraints, together with a representation of the corresponding VSpecs tree.

well as their children VarTokens can be used when defining the logical formulas or expressions that express those constraints. An in-depth example of this process is shown in section 5, as well as the *VarSpecsConstraints* view from RSL.

4.3 Create Resolution Model and Execute Materialization Process

The CVL-based variability modelling approach of ITBox ends with the *materialization process*. As stated in CVL, this process consists of transforming a Base Model into a new document by resolving the variation points in that model. Materialization is driven by a Resolution Model which provides for resolutions of VSpecs to which variation points are bound. In ITBox, the information in the Variability Model is stored in the database of the system. From that moment on, whenever a new SRS document creation process starts, the user is given the option to, instead of creating a new empty document, generate a new document by defining a set of resolution values (i.e. *creating the Resolution Model*) for an existing Variability Model. When this happens, the platform crosses the information contained in the Variability Model with the Resolution Model provided by the user and checks every constraint for possible inconsistencies. If no inconsistency is found, a new Resolved Model is produced.

5 RUNNING EXAMPLE

Figure 5 depicts a simple, yet effective example of application of the proposed CVL-based variability modelling approach of ITBox to a set of reliability-specific quality requirements, which defines the expected uptime of a system throughout a 1-year period, required maintenance procedures and fault logging details.

5.1 Create a Variability Model

5.1.1 Base Model

The first table in Figure 5 presents the *Quality Requirements* view with the quality requirements extracted from the SRS document of the example case. This view uses a set of properties to characterize each quality requirement: unique identifier, name, type, subtype, metric and corresponding value (if applicable), stakeholder who owns the requirement, parent requirement (if applicable), description and, finally, priority.

5.1.2 Variability Model

The second table in Figure 5 shows an example of the manual configuration of variation points. To define a requirement as a variation point, the user checks the `VarPoint?` field in the row corresponding to that requirement. This action will mark that requirement as an `Existence` variation point, meaning that it will be bound to a `Choice` VSpec. Furthermore, the field `VarTokens` is intended for the user to select which of the properties of that requirement will be modelled 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 properties modelled as children variation points: `Description` and `Priority`. However, not all of the content from the `Description` field is to be considered for variability purposes, only the substring `99%`, denoted by the suffix `.value#99%`.

The third table of Figure 5 depicts a simplified version of the *VarSpecs* view (for the sake of space, some of the properties mentioned in Table 2 were omitted). This view is automatically generated by the ITBox from the variation points previously selected by the user. In the example, the user selected the requirements `QR_R_1` and `QR_R_3`, which were transformed by ITBox into selected variation points `VS_1` and `VS_2`, respectively. This process corresponds to the binding process of CVL, in which each of the variation points is linked to its abstract representation in the Variability Model (or VSpec).

To the right of the third table in Figure 5 is the tree representation of the VarSpecs, VarTokens and VarSpecsConstraints in the example case. The parent element corresponds to the system itself. `QR_R_1` and `QR_R_3` requirements are represented as `Choice` VSpecs. These elements are linked with their parent by a dashed association, representing a false `IsImpliedByParent` value, which means that their materialization value is independent from the one their parent takes. In the third level of the tree three children VSpecs of type `Variable` reside. They were generated by the VarTokens defined in the third table and their corresponding types.

The fourth table of Figure 5 illustrates a simplified version of the *VarSpecsConstraints* view. The constraints are neither generated by the ITBox system, nor could they be since they are user-defined by definition. The user defined constraints based on

the VSpecs defined in the VarSpecs table and on the context of the system under specification. In Figure 5, a propositional constraint determines whether (during materialization) the annual uptime of the example case is greater than or equal to 99.9%. If so, the value defined for the `Priority` VSpec has to be `Must`. Note that both VSpecs involved in the constraint are children of the same VSpec (`QR_R_1`, with id `VS_1`). Thus, the constraint was defined in the context of the parent node. It is possible, however, to combine multiple parent VSpecs and their children (e.g. `VS_1->VS_2:Description = "something"`, which means that if `QR_R_1` (`VS_1`) is resolved as true, then the `Description` of `QR_R_3` (`VS_2`) must be resolved to the string "something").

5.2 Create Resolved Models

As depicted in Figure 3, various Resolved Models can semiautomatically be produced from a pair of Base and Variability Models, and various resolution models, with an appropriate tool support. For instance, considering the example case, two Resolved Models could be produced for two distinct systems (system A and system B) as suggested in the Tables 3 and 4, where the yellow background (or grey in grayscale printings) of some text snippets corresponds to the materialization process.

Table 3: Resolved Model for system A.

Id (*)	Name (*)	Description
QR_R_1	System Uptime	The system should assure an annual Uptime of 66%.
QR_R_3	Fault Logging	The system shall register all the faults occurred at runtime in the log file of system A.
…	…	…

Table 4: Resolved Model for system B.

Id (*)	Name (*)	Description
QR_R_1	System Uptime	The system should assure an annual Uptime of 90%.
…	…	…

6 ITBox EVALUATION

To evaluate the approach to the modelling and management of variability at the SRS level supported by the ITBox platform, a pilot user test session was conducted. The assessment focused on three aspects of ITBox: (1) the overall usability and

the quality of its features; (2) its capabilities in what requirements variability is concerned; and (3) the general approach enclosed in the platform.

The session involved a group of 7 participants with ages ranging from 22 to 28 years-old and with at least a Bachelor of Science degree. All participants had previous knowledge and academic experience in the field of RE and half of them had professional experience in the same field. The session was conducted under the following conditions: (1) it took place in a laboratory, therefore a controlled environment; (2) the assigned tasks were performed without previous use or learning of the platform; (3) all participants had a computer with a Web browser and Internet access; (4) while participants performed the assigned tasks, their behaviour and performance was directly observed; and (5) participants were free to think out loud and share ideas. All participants received a 20-minute presentation of the ITBox fundamentals (concepts and features), particularly its variability modelling capabilities. Afterwards, they were given a script describing a simple case study (the Billing System example) and its corresponding SRS document in the RSL-IL Excel template. The task consisted in uploading the template into ITBox and used it to test all the features of the platform within 40 minutes. In the end, participants were asked to fill in a questionnaire to rate the ITBox platform, its variability capabilities and the overall RE approach. The answers were expressed in the following scale: 0 (Not Applicable or Do Not Know), 1 (Very Low), 2 (Low), 3 (Medium), 4 (High) and 5 (Very High). The evaluation of the overall usability of ITBox and the quality of its features included 4 questions:

QP.1. How do you rate the overall usability of the Web platform?
QP.2. How do you rate the usefulness of the main RE productivity features (RSDoc Spreadsheet Editor, Template Manager and RSLibrary)?
QP.3. How easy to learn (or how familiar) was the main document editing tool (Google Sheets)?
QP.4. How suitable is the platform for a collaborative management of Requirements Specifications?

Table 5 summarizes the average score of the answers to the questions regarding the overall usability of ITBox and the quality of its features. In general, scores were very positive, which implies that the platform was successful at accomplishing its general goals. The lowest score was given to question QP.1, which demonstrates that the usability of ITBox can still be improved, however the answers to the other three questions were very positive, which indicates that the participants considered the

features of ITBox to be useful and easy to learn. Furthermore, the platform itself was considered very suitable for the collaborative management of SRSs.

Table 5: Average score (in a scale of 0-5), by question, for the overall usability of ITBox and the quality of its features.

QP.1	QP.2	QP.3	QP.4
4	4.29	4.29	4.71

The evaluation of the capabilities of ITBox in what requirements variability is concerned included 5 questions:

QV.1. How easy to understand was the overall variability modelling process?
QV.2. How easy to understand were the concepts (VarSpecs, VSpecs, VarTokens, VarSpecsConstraints, etc.)?
QV.3. How do you rate the usefulness of the variability modelling approach?
QV.4. How do you rate the simplicity of the Variability Model creation phase?
QV.5. How do you rate the simplicity of the Resolved Model creation phase?

Table 6 summarizes the average score of the answers to the questions regarding the capabilities of ITBox in what requirements variability is concerned. Similarly to the previous questions package, in general, scores were very positive, which means that the platform was successful at accomplishing its requirements variability goals. However, the scores were slightly lower than the ones of the previous questions package, especially those of questions QV.1 and QV.2. This indicates that the participants considered both the variability modelling process and its concepts sometimes hard to understand, which is relatively understandable since the proposal is innovative and the participants were not familiarized with it. Despite that, ITBox can still incorporate informative tooltips and a detailed user guide to better convey the variability process and its terms.

Table 6: Average score (in a scale of 0-5), by question, for the capabilities of ITBox in what requirements variability is concerned.

QV.1	QV.2	QV.3	QV.4	QV.5
3.48	3.43	4.57	4	4.14

Finally, the evaluation of the general approach enclosed in ITBox included 2 questions:

QA.1. How do you rate the productivity of ITBox when compared to the traditional requirements specification process?

QA.2. How likely would you use this platform on your own Requirements Engineering projects?

Finally, Table 7 summarizes the average score of the answers regarding to the questions regarding the general approach enclosed in ITBox. The score obtained for both questions was highly positive, which demonstrates that the participants considered ITBox to be quite productive to specify requirements as well as useful.

Table 7: Average score (in a scale of 0-5), by question, for the general approach enclosed in ITBox.

QA.1	QA.2
4.43	4.43

As illustrated in Table 8, the results of the pilot user test session were generally encouraging, with positive scores in all of the 3 aspects of ITBox analysed. Nevertheless, the variability modelling process and its concepts can still be improved with regards to its simplicity towards the user. Regarding the relatively low number of participants, studies have noted that a group of 5 testers is enough to uncover over 80% of the usability problems in a proposed solution (Nielsen and Landauer, 1993), so the conclusion that the results extracted from the pilot user test session are representative of what could be expected from a bigger number of participants can be drawn. Also, we believe 7 participants is a reasonable number for an exploratory assessment to identify challenges associated with the overall usability and features of ITBox, its variability requirements approach and the general approach enclosed in the platform.

Table 8: Average score (in a scale of 0-5) for each of the aspects of ITBox analysed.

Platform	Variability	Approach
4.32	3.92	4.43

7 RELATED WORK

Coplien et al. (1998) argued that the analysis decisions on C&V shall be made during the requirements analysis stage, rather than during the implementation stage by professionals who are not so familiar with the implications and impact of such decisions. They referred that early decisions on C&V contribute to large-scale reuse and the automated generation of family members. In 2002, Bosch et al. also mentioned the need for describing

C&V within different modelling levels such as the requirements one (Bosch et al., 2002).

Table 9 synthetizes prior research contributions in what C&V at the analysis stage is concerned, namely at the level of requirements representation (e.g. with use case models). The most recent effort to establish a unified variability language is CVL. However, given that CVL is relatively new, not much research has yet been conducted to define clear ways of applying its concepts to the RE domain. Research contributions referred in Table 9 between #1 and #9 are prior to the CVL, therefore, they convey different approaches over variability representation, including at the requirements point of view, and they have contributed to the existence of CVL.

Particularly research contribution #10, like RSL, covers a comprehensive set of RE concepts and relationships between them, and avoids having to keep the consistency between requirements models and variability models, yet it is not compliant with CVL. The same goes for the research contribution #11 in what compliance with the CVL is concerned. So far, CVL has not been specifically applied to RE. The approach proposed in this paper, however, is targeted at specifying the C&V of RE concepts in compliance with CVL. It achieves this because RSL allows encoding rigorous SRSs, but also because it is possible to associate variability points in a non-intrusive way for many of the RSL constructs (e.g. goals, quality requirements, actors and use cases).

8 CONCLUSION

This paper discusses a CVL-based approach for modelling variability in requirements specification documents. The approach was integrated into the ITBox[1] platform and uses the SRS template based on the multiview architecture defined in the RSL as its Base Model. RSL is a language that includes a rich set of constructs logically arranged into views according to multiple RE-specific constructs situated either at the business or at the system abstraction levels. Those constructs are defined as linguistic patterns and textually represented by mandatory or optional fragments (Silva, 2017; Silva, 2018).

RSL is a process- and tool-independent language that can be used and adapted by multiple users or organizations with different processes, as well as supported by multiple types of software tools. However, in practical terms, RSL has been

---

[1] ITBox currently available at http://itbox.inesc-id.pt

Table 9: Research contributions concerned with the representation of requirements variability.

#	Author, year [reference]	Super-structure	Concepts related to C&V	Key Contribution
1	Muthig (2002)	UML: use cases	*«variant»* use cases	The *«variant»* stereotype, targeted at representing variability in use cases
2	Maßen and Lichter (2002)	UML: use cases	Option, alternative and optional alternative use cases	Extension to the UML metamodel to incorporate two new relationships to represent variability at the level of requirements modelling with use cases
3	Halmans and Pohl (2003)	UML: use cases	Mandatory and optional variation points	Extensions to use case diagrams to represent and communicate variability relevant to the customer (additional graphical elements proposed to explicitly represent variation points and variability cardinality in use case diagrams)
4	Gomaa (2004)	UML: use cases	*«kernel»*, *«optional»* and *«alternative»* use cases *«common feature»*, *«optional feature»* and *«alternative feature»* *«zero-or-one-of feature group»* and *«exactly-one-of feature group»*	PLUS (Product Line UML-based Software engineering), a model-driven approach for variability analysis, namely at the requirements (use cases) level
5	Gomaa and Shin (2004)	UML: use cases	Kernel use case Optional use case Alternative use case	Multiple-view variability metamodeling approach, using UML, namely in use case modelling
6	Webber and Gomaa (2004)	VPM	Four types of variation points: Parameterization Information hiding Inheritance Callback	VPM (Variation Point Model), a method that contemplates a modelling view to capture requirements together with variation points during the domain analysis phase
7	Bachmann et al. (2004)	*No specific name*	Variation point Variant Asset Rationale	Variation (meta)model for the representation of variability as a dedicated view connected to all the other views of a system, namely the requirements view (e.g. use cases)
8	Bühne et al. (2005)	*No specific name*	Mandatory/optional variation point Variant Requirements artefact	Metamodel representing the structure of variability information used for the documentation of requirements across a single SPL or a set of SPLs, based on the metamodel of Bachmann et al.
9	Bayer et al. (2006)	CVM	Model and variation elements Variability specification (variability constraint and transformer) Variation and resolution models (resolution elements: value resolution and type resolution)	CVM (Consolidated Variability Metamodel), which systematizes different kinds of variability recurrently present in SPL models and contemplates different approaches of variability capturing, namely UML and DSLs
10	Moros (2008)	REMM	Requirement Mandatory level type Reusable and product catalogues Reusable and product requirements Parameter and parameter instance …	REMM (Requirements Engineering MetaModel), which allows specifying catalogues of reusable requirements models, as well as defining specific product requirements, namely by reusing previously modelled requirements; furthermore, REMM allows requirements engineers to define, in the same model, both optional and parameterized requirements, which usually are represented in feature models on the side
11	Alférez et al. (2010)	VML4RE	Commonalities Variabilities (optional feature, variation point and variant)	VML4RE (Variability Modelling Language for Requirements), which provides for a SPL requirements modelling approach, comprised of variability identification at the feature modelling level, as well as of domain requirements description by means of use cases and activity diagrams, and ultimately a (meta)model to relate modelled features and requirements
12	Rouillé et al. (2012)	*No specific name*	*C&V concepts inherited from the CVL*	Model-driven approach for the automatic derivation of processes from software process lines using CVL to bind requirements variability to process variability
13	Oliveira et al. (2013)	FeDRE	Mandatory, optional and alternative (OR or XOR) features Requirement, requirements specification, use case…	FeDRE (Feature-Driven Requirements Engineering) approach, in which variability modeled through features is realized into functional requirements, as well as features themselves are further taken as input for use case specification

implemented with the Xtext framework, which means that RSL specifications are rigorous, and can be automatically validated and transformed into other representations and formats. A lightweight tool support is provided with the ITLingo RSL Excel template[2] publicly available at GitHub.

To provide for distributed access to its SRS documents, as well as data manipulation features, the ITBox Web-based platform extensively uses two Google Web APIs: Google Drive API and Google Sheets API. Thanks to this, the ITBox variability modelling approach can automatically extract the information in a SRS document, modify it, and generate new documents and views if necessary. This allowed automating the application of CVL concepts to the context of RE Future work will focus on expanding the views supported by RSL, allowing to progress from modelling variability at the level of *Goals*, *Functional Requirements* and *Quality Requirements* to modelling variability at the level of *Stakeholders*, *Entities*, *Use Cases*, etc., enabling a much wider scope of variability points within the spectrum of RE concerns. Furthermore, the long-term goal of this research is to fully integrate this variability modelling process within the ITLingo approach for domain knowledge extraction from natural language documents, expanding the source of the variability modelling process from semiformal SRS documents to more unstructured ad-hoc SRSs in natural language.

ACKNOWLEDGEMENTS

REFERENCES

Alferez, M., et al., 2010. Multi-view composition language for software product line requirements. LNCS, 5969:103–122.

Bachmann, F., et al., 2004. A Meta-model for Representing Variability in Product Family Development. 5th International Workshop on Product-Family Engineering (PFE-5) Springer-Verlag.

Bayer, J. et al., 2006. Consolidated Product Line Variability Modeling," in Software Product Lines - Research Issues in Engineering and Management, Springer-Verlag.

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

Blanes, D., González-Huerta, J., and Insfran, E., 2014. A multimodel approach for specifying the requirements variability on software product lines. 23rd International Conference on Information Systems Development, 329–336.

Bosch, J., et al., 2002, "Variability Issues in Software Product Lines," in 4th International Workshop on Product Family Engineering (PFE-4) Bilbao, Spain: Springer-Verlag.

Bühne, S., Lauenroth, Pohl, K., 2005. Modelling Requirements Variability across Product Lines," in 13th IEEE International Conference on Requirements Engineering, IEEE Computer Society.

Coplien, J., Hoffman, D., Weiss, D., 1998. Commonality and Variability in Software Engineering, IEEE Software, vol. 15, pp. 37-45.

Davis, A., 2005. Just enough requirements management: Where Software Development Meets Marketing.

Fernandes, J., 2016. REBox: Collaborative Environment for Requirements Engineering, MSc Thesis, IST, Universidade de Lisboa.

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

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.

Gomaa, H., 2004. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley.

Gomaa, H., Shin, M. E., 2004. Multiple-View Meta-modeling Approach for Variability Management in Software Product Lines," in 8th International Conference on Software Reuse (ICSR-8), Springer-Verlag.

Halmans, G. Pohl, K., 2003. Communicating the Variability of a Software-Product Family to Customers, Software and Systems Modeling, vol. 2, pp. 15-36.

Maßen, T. v. d., Lichter, H., 2002. Modeling Variability by UML Use Case Diagrams, in International Workshop on Requirements Engineering for Product Lines (REPL 2002).

Moros, B., Vicente-Chicote, C., Toval, A., 2008. Metamodeling variability to enable requirements reuse. CEUR Workshop Proceedings, 337:140–154.

Muthig, J. D., 2002. Product Line Modeling with Generic Use Cases, in Workshop on Techniques for Exploiting Commonality Through Variability Management, Springer-Verlag.

Nielsen, J., Landauer, T. K., 1993. A Mathematical Model of the Finding of Usability Problems. In Proceedings of the INTERACT '93 and CHI '93. ACM.

---

[2] https://github.com/RSLingo/RSL-Excel-Template

Oliveira, R. P., et al., 2013. A feature-driven requirements engineering approach for software Product Lines. Proceedings of 7th Brazilian Symposium on Software Components, Architectures and Reuse.

OMG, 2012. OMG Common Variability Language (CVL) OMG Revised Submission.

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

Ribeiro, A., Silva, A. R., 2014. XIS-Mobile: A DSL for Mobile Applications, Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC), ACM.

Ribeiro, A., Silva, A. R., 2014a. Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development. Journal of Software Engineering and Applications, 7(11), 906-919.

Rouille, E., et al., 2012. Leveraging CVL to manage variability in software process lines. Proceedings Asia-Pacific Software Engineering Conference, APSEC, 1:148–157, 2012.

Savić, D., et al., 2015. Use Case Specification Using the SILABREQ Domain Specific Language, in Computing and Informatics Journal, 34(4):877–910.

Silva, A. R., et al., 2014. Towards a System Requirements Specification Template that Minimizes Combinatorial Effects, Proceedings of QUATIC'2014 Conference, IEEE CS.

Silva, A. R., 2015. Model-driven engineering: A survey supported by the unified conceptual model. Computer Languages, Systems & Structures, 43(October 2015), 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., 2017. Linguistic Patterns and Linguistic Styles for Requirements Specification (I): An Application Case with the Rigorous RSL/Business-Level Language, Proceedings of EuroPLOP'2017, ACM, 2017.

Silva, A. R., 2018. Rigorous Requirements Specification: Specification of Use Cases with the RSLingo RSL Language. INESC-ID Technical Report.

Silva, A. R., et al., 2007. XIS – UML Profile for eXtreme Modeling Interactive Systems, in Proceedings of the MOMPES 2007, IEEE Computer Society.

Silva, A. R., Fernandes, J., Azevedo, S., 2017. Variability Aspects at a Textual Requirements Specification Level. IEEE 25th International Requirements Engineering Conference Workshops (REW), IEEE Computer Society.

Verelst, J., et al., 2013. Identifying Combinatorial Effects in Requirements Engineering, Proceedings of Third Enterprise Engineering Working Conference (EEWC 2013), Advances in Enterprise Engineering, LNBIP, Springer.

Webber, D. L., Gomaa, H., 2004. Modeling Variability in Software Product Lines with the Variation Point Model, Science of Computer Programming, vol. 53, pp. 305-331.