Towards a Library of Usability Requirements

Alberto Rodrigues da Silva
INESC-ID, IST, Universidade de Lisboa
Portugal
alberto.silva@tecnico.ulisboa.pt

Carolina Lisboa Sequeira
IST, Universidade de Lisboa & Universidade Aberta
Portugal
carolinaslsequeira@gmail.com

ABSTRACT

[1]Usability is an essential quality of software products if not an important concern in any business competition. This research discusses an approach to better define and specify reusable usability requirements. This is difficult because requirements specifications tend to be poorly structured, and tend to be introduced late in the development process, which results in major cost and rework. This research starts by identifying concerns related the classification and specification of usability requirements commonly found in literature. Then, it extends the ITLingo RSL language, which is a controlled natural language that allows to specify requirements and tests in a rigorous and structured way, and uses the respective tools to define a library of reusable usability requirements. This research was conducted in a software house operating in the healthcare domain and was applied and evaluated in its family of software products. The most important contribution of this research is the initial proposal of a library of reusable usability requirements, rigorously specified in a language like RSL, which may promote both quality and productivity.

KEYWORDS

Usability; Requirements Specification; Requirements Reuse; RSL.

ACM Reference format:

G. Gubbiotti, P. Malagò, S. Fin, S. Tacchi, L. Giovannini, D. Bisero, M. Madami, and G. Carlotti. 2020. In *Proceedings of ACM SAC Conference, Brno, Czech Republic, March 30 - April 3, 2020*, 8 pages. DOI: https://doi.org/10.1145/3341105.3373939

1 INTRODUCTION

Usability is a quality attribute that assesses how easy is to use some product. Usability also refers to methods for improving ease-of-use during the design process [1]. In the context of digital transformation scenarios, people are demanding products that they use, the experiences they have and how fast and simple they get their information. Organizations need to make better decisions, speed core business processes and get closer to their customers and users [2]. Digital transformation scenarios demand that digital systems would be simple, accessible and easy to use. That means that software products must be designed for people, and that usability is becoming a commodity and an essential property.

However, this context influences the software product's development process, requiring that usability concerns would be considered since early stages. Usability requirements shall be identified during the analysis and design's stages to prevent that later on usability problems force software changes that can imply high rework and costs [3]. This fact raises major challenges to the software engineering discipline. For example, during this research it was verified the following aspects: that usability is indeed an interdisciplinary field which contributes to the quality of the software; that usability is manifested in interaction design and even has impact in software architecture [4], driving to functional and non-functional requirements; and that usability meaning tends to be ambiguous and hard to validate.

After understanding that usability must be considered together with the analysis of other system concerns, since the beginning of the development process, it is important to stress that usability requirements are an important concern as well [5]. It is essential that the structures of the requirements specification bring accuracy to the process to reduce common ambiguousness and inconsistency. On the other hand, these requirements shall be specified in a controlled natural language so that stakeholders could easily understand them.

ITLingo [35] is a research initiative that has proposed a set of languages for specify technical documentation considering different IT aspects. One of its languages is the RSL (Requirements and Tests Specification Language) [14-16]. RSL is used in the context of our proposal because it allows us to rigorously specify requirements and, in particular, to define a library of reusable usability requirements. Currently this library includes around 60 requirements divided into ~40 goals and ~20 quality requirements, which are publicly available [35]. Furthermore, this work is in line with others like the proposals of Gonçalves et al. on security requirements [27-28], Caramujo et al. on privacy policies [39], or Fernandes et al. on legal requirements [29].

This proposal was validated in a real-world environment, namely in the context of a software house operating in the healthcare domain, as discussed in Section 5.

SAC'2020, March 30 –April 3, 2020, Brno, Czech Republic
 978-1-4503-6866-7/20/03. . . $15.00
DOI: https://doi.org/10.1145/3341105.3373939

This paper is structured in 6 sections. Section 2 overviews the research background and related work on non-functional requirements and usability attributes, reuse techniques and in particular usability patterns. Section 3 briefly introduces the RSL language and discusses the reasons that justify its adoption. Section 4 discusses the proposed library of reusable usability requirements. Section 5 presents a hands-on session performed in the context of a software house that allowed us to preliminary assessed this proposal. Finally, section 6 presents the conclusion and identifies open issues for future work.

2 RELATED WORK

Usability is commonly defined as a quality requirement (QR) but also as a non-functional requirement (NFR), so it is important to reflect about QR/NFR's classification schemes and representation structures. To understand how usability can be properly defined as an important product's quality attribute, a usability taxonomy review is discussed. Finally, it is introduced an analysis of requirements reuse approaches, namely at different levels like document templates, sentences, parameterized sentences, linguistics patterns.

2.1 Non-Functional Requirements

Non-functional requirements (NFRs) tend to define global properties of a system such as integrity, safety, usability, reliability or performance. Chung and Leite discuss the state of the art of NFRs in software engineering, considering different definitions and representation schemes [9]; they consider that functional requirements (FRs) are focused on "what makes software" while NFRs on "how well" software does something. However, NFRs are referred by different authors with different terminology, for example as goals [34] quality criterion (ISO/IEC 25010) [7] or quality attributes [9] [12]. Even some authors, like Pohl [25], consider that NFRs are a superset of QRs and *under-specified requirements* (i.e. poorly defined requirements), and suggest the use of QR instead of the broader term NFR.

ISO/IEC 25010 (that replaced ISO/IEC 9126) [7] is a standard for software product quality that defines the quality characteristics that software systems shall have. This standard distinguishes two classes of qualities: (i) *Quality in use*, which relates to the outcome of the interaction when a product is used, and (ii) *Product quality*, that relates to software attributes and dynamic properties of computer systems.

Other works propose different categories to classified NFRs, for example: Roman [10] proposes a NFR taxonomy based on categories like Interface, Performance, Operating, Lifecycle, Economic or Political requirements; Glinz' s proposal [11] considers that the four most important NFR categories for software architects are: Performance, Usability, Security and Availability (in this order); or FURPS + [9] defines a classification of software quality attributes based on the following categories: Functionality, Usability, Reliability, Performance and Compatibility.

2.2 Usability

Usability relates to the ability of an application to be understood, learned, used and attractive to the user, under specific conditions of use. However, as stated by Abran et al. [37], definitions of usability properties are not absolute in terms of interpretation, varying according to the stakeholders' interest in the software product: managers, developers and end-users have different interpretations of what a product's usability mean. For instance, for a manager, usability might be a distinguishing feature in the selection of a software product in that it directly influences the learning ability and productivity of its team. In turn, for a developer, usability might represent a set of internal attributes that should be introduced in the development process, assessing problems with design quality or maintenance of documentation issues. On the other hand, for end-users, the usability might mean how quick and efficient they can do their tasks. This fact shows that usability cannot be defined as absolute sense, but rather as a set of references and contexts. From them it is possible to find the concept of usability with greater formalism, associating its attributes with a context, user, and concrete goals to achieve.

For the definition of usability and respective attributes, this research was deepened using a comparative analysis of several proposals, as showed in Table 1. Due to space constraints, we just summarize the key conclusions of that analysis: that is, although there is a consensus on the term usability, there are distinct approaches to its attributes; many of these differences are a result of the authors' analysis of how to measure usability. Notwithstanding, some of these definitions are ambiguous or show different ways of combining attributes. For example, attributes like "learnability" or "memorability" can be considered as the same concern; but the definition of "error" by ISO/IEC 25010 [7], as part of efficiency, is distinct from Nielsen's "error" design [17].

In addition, it is also important to identify techniques to evaluate the usability success level. This issue is extensively discussed by Nielsen that proposes a set of 10 heuristics to make it easier to perceive and evaluate usability common problems [17].

Table 1. Comparative table of usability attributes

Usability Attributes	ISO/IEC 25010:2011 [7]	ISO 9241-11:2018 [8]	Nielsen [17]	Schackel [32]	CISU-R [33]
Learnability	X		X	X	
Effectivness		X		X	X
Memorability			X		
Errors	X		X		
Efficiency		X	X		X
Operability	X				
Understandability	X				
Flexibility				X	
Satisfaction	X	X	X		X
Attractiveness	X			X	
Accessibility	X				

2.3 Taxonomy for Usability Requirements

The ability to refine the concept of usability and usability attributes into more measurable and concrete concepts is essential to include in the software product properties. A number of usability aspects have been selected from literature that embody heuristics [17], patterns collections [4,18-20], usability scenarios [5] and design principles [21].

ISO/IEC 25010 defines usability as the "degree to which a product or system can be used by specific users to achieve specific goals with effectiveness, efficiency and satisfaction in a specific context of use" [7]. ISO/IEC 25010 was chosen in our proposal as the adopted classification schema of usability requirements because it defines usability in a way that can be measured as software product property, not just as a quality of software in use (user perceived quality). This standard defines a quality model that relates to the outcome of interaction when a product is used in a particular context.

As a product characteristic, usability is defined in the product quality model with other seven characteristics: functional suitability, reliability, performance efficiency, security, compatibility, maintainability and portability. Each of these characteristics are then break-down into sub-characteristics. In particular in this paper, usability is break-down into six sub-characteristics that help to further classify each QR, namely and as summarized in Table 2: Appropriateness recognizability, Learnability, Operability, User error protection, User interface aesthetics, and Accessibility.

Table 2. Usability's 6 sub-characteristics, ISO/IEC 25010

Usability Sub-characteristics	Description
Appropriateness recognizability	Degree to which users can recognize whether a product or system is appropriate for their needs. The information provided by the product or system can include demonstrations, tutorials, documentation or, for a web site, the information on the home page.
Learnability	Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
Operability	Degree to which a product or system has attributes that make it easy to operate and control.
User error protection	Degree to which a product or system can protect its users against user errors.
User interface aesthetics	Degree to which a user interface enables pleasing and satisfying interaction for the user.
Accessibility	Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

2.4 Reuse and Usability Patterns

To identify common usability concerns that can be taken into account during the elicitation, specification and documentation of requirements we shall identify reuse practices. Several authors have already discussed this issue from different perspectives, namely as requirements specification document templates, requirements templates or requirements patterns. For example, Palomares et al. [31] analyze this issue by identifying and classifying 294 papers on the subject, considering different dimensions of analysis: artifacts to be reused, size of the artifacts, process involved, repository, scope, abstraction and purpose. In addition, from their exploratory survey, they report that although they found that a high majority of respondents declared some level of reuse in their projects (in particular, non-functional requirements were identified as the most similar and recurrent among projects), only a minority of them declared such reuse as a regular practice.

Toval et al. propose the SIREN method [30] that includes a spiral process model, requirements documents templates, a reusable requirements repository which is organized by catalogs and a supporting tool called SirenTool. SIREN is a practical approach for selecting and specifying the requirements of a software system based on requirements reuse. The experience gained through the application of this method, has led them to discuss 8 key issues for any reuse-based requirements method to succeed, namely [36]: Organization of the reusable requirements; Search engine for reusable requirements; Requirements selection and reuse with different granularity levels; Requirement attributes reuse; Traceability relationships reuse; Parameterized requirements management; Repository improvement; and Tool support to reuse.

Juristo et al. [19] proposes to break down usability-specific attributes into "functional usability features", i.e., software product features that introduce relevant usability benefits. They describe these features as "usability patterns" like [22]: Wizards, Shortcuts (key and tasks), Context-sensitive Help, History Logging or Action for Multiple Objects. Folmer and Bosch [23] [24] also propose relevant patterns like: Data Validation, Workflow Model, and User Models. Taking the user perspective, Welie and Trætteberg [20] discuss interaction patterns in the scope of user interfaces such as: Grid Layout, Shield, Contextual Menu, Unambiguous Format, Navigating between Spaces, Continuous Filter, Command Area.

Bass et al. [5] present an approach to improve the usability of software systems by means of software architectural decisions; they formulated each aspect of usability as a scenario based on a set of stimulus and response. For each scenario, they discuss architectural patterns that implement a usability aspect, such as the following patterns [5]: Maintaining Device Independence, Re-Covering from Failure, Retrieving Forgotten Passwords, Reusing Information, Supporting International Use, Predicting Task Duration, Supporting Comprehensive Searching, Reduces the Impact of System Errors, Aggregating Commands, and Supporting Undo.

Usability and software design are truly embraced: this means that UI design requires understanding technical aspects, and that develop usable software requires understanding and designing the complete user experience, and also including aesthetics and graphical issues, UI design guides [21], or usability heuristics [1].

3 RSL LANGUAGE

ITLingo adopts a linguistic approach to improve the rigor of technical documentation and, as a consequence, to promote productivity through re-usability and model transformations, as well as to promote quality through semi-automatic validation. In particular, ITLingo/RSLingo is the approach for the specification of software requirements that uses natural language processing techniques to translate informal requirements into rigorous representations [6,13].

This approach defends that requirements shall be represented in a natural language, which allows to be more easily writable and readable by various stakeholders (e.g., requirement engineers, business analysts, developers or designers), but which may then be mapped to a constrained language, called "ITLingo RSL" (or just "RSL" for brevity), based on the design of former languages like RSL-IL [14], XIS* [38], RSL-IL4Privacy [39] and TSL [16]. RSL is a controlled natural language designed to help the production of requirements and tests specifications in a systematic, rigorous and consistent way [14-15]. RSL includes a rich set of constructs (e.g. stakeholders, actors, data entities, requirements and tests) logically arranged into views according to RE-specific concerns that may exist at different abstraction levels. These constructs are defined as linguistic patterns and represented textually by mandatory and optional fragments [14].

Figure 1 illustrates a partial view of the RSL metamodel that shows a hierarchy of requirement types, such as: Quality Requirement (QR), Goal and User Story (visible), but also Constraint, Functional Requirement (FR), and Use Case (not visible). A requirement can aggregate other requirements through the "isPart" relationship and may establish different types of relationships with other requirements, through the "Requirement Relation" relationship; these relationships can be further classified as Requires, Supports, Obstructs, Conflicts or Identical.

The key reasons to adopt the RSL language in our research were the following: First, RSL goals and quality requirements' classification schema in RSL also adopt the ISO/IEC 25010 schema as discussed in Section 2.3. Second, RSL has been implemented with the Xtext framework (https://www.eclipse.org/Xtext/) in the scope of an Eclipse-based tool called ITLingo-Sudio [35]. Consequently, RSL specifications are rigorous and can be automatically validated and transformed into other representations and formats. Third, and key for our research, it is also provided an RSL Excel template, publicly available at ResearchGate [35] and github (https://github.com/ITLingo). This Excel template is composed by several sheets, each one corresponding to a given RSL construct (e.g., "stakeholders", "reqs.goals", "reqs.userstories", "reqs.qrs"). This research was mainly developed on top of this template and so, the examples showed in the following sections use it. Fourth, Gonçalves and Silva [27-28] showed recently that RSL could be adopted as a suitable language to define reusable and rigorous security requirements, and they preliminary discussed the proposal of a catalogue of reusable requirements for such profile. Also, Fernandes et al. [29] showed how to capture and properly define GDPR-specific requirements with the RSL language.

4 LIBRARY OF USABILITY REQUIREMENTS

As referred above, RSL includes constructs logically organized into views according to the specific concerns [14-15]. Some usability patterns briefly identified in Sub-section 2.4 were specified in RSL namely as a set of reusable Goals and QRs, as suggested in Figure 1. A *Goal* is an objective that should be achieved, it expresses what is desired by some stakeholder and the context in which the motivations and rationales behind it can be understood. On the other hand, a *QR* expresses a "quality" of the system and may express specific metrics that the system must meet to be accepted.

In addition, we also used the RSL UserStory construct that allow describing some functionality from the user perspective, namely according the popular linguistic template "As a <User>, I want <This-Feature> so that <Some-Reason>"; this perspective of user stories is relevant to validate the Goals or QRs applicability, in a use context, but usually are defined as concrete (i.e., not reusable) requirements.

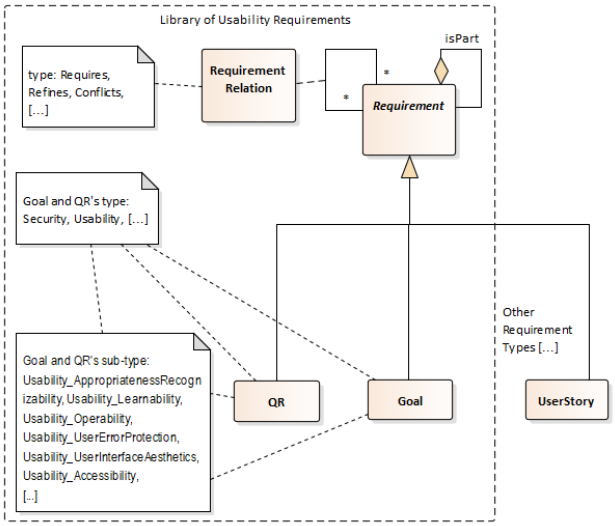

Fig. 1. Overview of the Library of Usability Requirements based on the RSL language.

4.1 Usability Goals

Figure 2 shows general usability concerns as defined with the RSL Goal construct. This view shows a set of usability goals to be achieved in general by any software system. These requirements are aggregated in goals and sub-goals in a way that a more abstract goal can be decomposed by other sub-goals.

For example, the general goal "Feedback (g_1_1): The system should keep the user informed about what is happening" is further detailed into the following sub-goals: (i) Progress bar (g_1_1_1): The system must present information about the current action's status, action´s total duration and how much longer the user will need to wait […]; (ii) Alerts (g_1_1_2): The user may cause or be confronted with an error or problematic situation […]; and (iii) Resource Verification (g_1_1_3): The system must verify that the necessary resources exist before starting […].

Library of Reusable Usability Requirements

SystemGoals Definition

Id (*)	Name (*)	PartOf	Description
g_1	Usability Concerns		The system must have characteristics that allow specific users to achieve specific objectives with effectiveness, efficiency, and satisfaction in a specific context of use
g_1_1	Feedback	g_1	The system should keep the user informed about what is happening.
g_1_1_1	Progress bar	g_1_1	The system must present information about the curent action's status, action´s total duration and how much longer the user will need to wait. For long tasks, the user must be able to take informed decisions to decide whether to continue the task while the system completes an operation or if it is going to work on another task while it waits.
g_1_1_2	Alerts	g_1_1	The user may cause or be confronted with an error or problematic situation that needs to be resolved. The system should assist the user by giving them important information on the error presentation. Example, The condition that triggered the error, detailed description of the situation to help the user make the appropriate decision. Choices should always be declarative, including a verb that refers to the desired action and not just YES or NO. The choices presented should be the answer to the question triggered by the system.
g_1_1_3	Resource Verification	g_1_1	The system must verify that the necessary resources exist before starting an operation so that there are no unexpected errors and failures during execution. The user must be informed of the conditions necessary to perform the task and the necessary resources, if the conditions are not met, the user must be informed about current conditions and the necessary conditions
g_1_2	Error management	g_1	The system must allow the user to control their actions by canceling or changing in progress operations or operations already performed.
g_1_2_1	Undo	g_1_2	The system must allow the user to return to the previous state before completing the current task or to undo a previous completed operation - unfold the undo in several levels, as many as the levels of accomplishment of the task. It allows the user to have interaction' s control and to
g_1_2_2	Cancel	g_1_2	The system should allow the user to cancel a command issued and not yet completed. The system must be prepared to process actions that are interrupted and may even have to be

... b.processes b.flows b.relations b.systems.relations s.systems s.actors

Fig. 2. Usability Goals defined in RSL-Excel-Template

Library of Reusable Usability Requirements

Quality Requirements Definition

				Expression				
d (*)	Name (*)	Type (*)	Sub-Type	Metric	Ope-rator	Value	PartOf	Description
qr_1	Message quality	Usability	Usability_UserErrorProtection					Error messages and success messages shall be clear and perceptible.
qr_1_1	Error message comprehension time	Usability	Usability_UserErrorProtection	Time_Sec	=	<value1>	qr_1	Error warnings shall be active on the screen for <value1> seconds.
qr_1_2	Functionality used without error	Us	Usability_AppropriatenessRec Usability_Learnability Usability_Operability Usability_UserErrorProtection Usability_UserInterfaceAesthet Usability_Accessibility Cultural_Language Cultural_Currency	or_Per sk	=	<value2>	qr_1	When filling a Form, an user shall not err in filling the form due to bad fields (e.g., error in entering data type or required field to fill) more than <value2> times.
qr_2	Users perceptions	Us		nge_Li rtScale	>=	<value3>		In the product evaluation questionnaire, in the evaluation of the topic "the product has a user friendly interface" the product must have 80% of the answers upper or equal <value3>, in the Likert Scale.

Fig. 3. QRs in RSL-Excel-Template (Usability sub-types)

```
Goal g_1_1_Feedback "Feedback": Usability: Usability_Operability
[isAbstract  partOf g_1_Usability
description "The system should keep the user informed about what is
happening."]
Goal g_1_1_1_Progress_Bar "Progress bar": Usability:
Usability_Operability [partOf g_1_1_Feedback
description "The system must present information about the curent
action's status, action´s total duration and how much longer the user
will need to wait. For long tasks, the user must be able to take
informed decisions to decide whether to continue the task while the
system completes an operation or if it is going to work on another
task while it waits."]
Goal g_1_1_2_Alerts"Alerts": Usability: Usability_Operability [
part of g_1_1_Feedback
description "The user may cause or be confronted with an error or
problematic situation that needs to be resolved. The system should
assist the user by giving them important information on the error
presentation. Example, The condition that triggered the error,
detailed description of the situation to help the user make the
appropriate decision. Choices should always be declarative, including
a verb that refers to the desired action and not just YES or NO. The
choices presented should be the answer to the question triggered by
the system."]
Goal g_1_1_3_Resource_Verification "Resource Verification": Usability:
Usability_Operability [partOf g_1_1_Feedback
description "The system must verify that the necessary resources exist
before starting an operation so that there are no unexpected errors
and failures during execution. The user must be informed of the
conditions necessary to perform the task and the necessary resources,
if the conditions are not met, the user must be informed about current
conditions and the necessary conditions."]
```

Spec. 1. Partial RSL specification of reusable Goals.

Spec. 1 also illustrates these reusable goals as directly specified with the RSL concrete syntax.

Currently this usability library includes 41 reusable goals, corresponding to some usability patterns identified in Section 2.4, with the following top level goals: Feedback, Error management, Support international use, Responsive Web design, Help, Wizard, Workflow, Action on multiple objects, User profile, Make information accessible, Logging history, Reuse Information, Scripting, Supporting Comprehensive Searching, Single Sign-on.

4.2 Usability Quality Requirements

Usability may depend on several aspects like goals of use, context of use, tasks or even users. However, usability can be further detailed in quantitative and measurable ways. General or concrete goals also provide objectives for usability testing and ensure that faulty or unsatisfactory software will not be released. For this purpose, RSL QR was applied: as suggested in Figure 2, usability QRs may be further classified by a subtype as defined by ISO/IEC 25010's usability sub-characteristics.

```
QR qr_1_Message_Quality "Message quality": Usability:
Usability_UserErrorProtection [
description "Error messages and success messages shall be clear and
perceptible."]
QR qr_1_1_Error_Message_Comprehension_Time "Error message
comprehension time":Usability: Usability_UserErrorProtection
[expression [<= <value> Time_Sec] partOf qr_1_Message_Quality
description "Error warnings shall be active on the screen for <value>
seconds."]
QR qr_1_2_Functionality_Used_Without_Errors "Functionality used
without errors": Usability: Usability_UserErrorProtection [
expression [<= <value> Error_PerTask] partOf qr_1_Message_Quality
description "When filling a Form, an user shall not err in ..."]
```

Spec. 2. Partial RSL specification of reusable QRs.

Figure 2 shows the QR classification schema with a concrete set of reusable QRs. In a consistent way as seen above for Goals, QRs can be break-down into other (more specific) QRs, through the PartOf relationship. The current version of this library includes several QRs structured around the following top-level QRs: Message quality (qr_1), Users perceptions (qr_2), Interface complexity (qr_3), Learning facility (qr_4), and Results of a list of values (qr_5).

These top-level QRs are decomposed in more specific QRs. For example (see Figure 3 and Spec. 2) the general usability QR "Message quality Usability (qr_1)" is decomposed in the following specific QRs: (i) Error message comprehension time (qr_1_1): sub-type Usability_UserErrorProtection: Error warnings shall be active on the screen for <value> seconds; and (ii) Functionality used without errors (qr_1_2): sub-type Usability_UserErrorProtection: When filling a Form, an user shall not err in filling the form due to bad fields (e.g., error in entering data type or required field to fill) more than <value> times.

In addition, as also illustrated in Figure 3, we can assign to each QR a logical expression, based on a logical operator, one or more values, and the respective metric of these values (e.g., seconds, days, number of tasks, errors per task). Finally, because we are defining a library of reusable requirements, these QRs shall be defined in such reusable way, in particular, some values shall be specified in a generic way with "template variables" (aka. macros or parameterized elements) expressed in the examples with text delimited by the characters "<" and ">" (e.g., "<value1>").

These QRs and Goals were defined with no particular system's needs into consideration: they just define common concerns found in enterprise information systems. Consequently, any requirements specification can adopt these requirements as a starting base for reusing its own requirements.

5 EVALUATION

The results of this research have been applied and assessed in the scope of a software house that operates in the healthcare domain. It makes part of a multinational IT consulting company, with € 66.1 million turnover in last year and it has approximately 900 employees, operating from 10 offices in 6 countries: Spain, Portugal, Angola, Brazil, United Kingdom and Ireland. Its core business is healthcare market (hospitals and pharmacies), however, it has representative businesses in Financial Services, Telecommunications and Public Administration.

For a thorough evaluation of this research it was analyzed the PAS (Patient Administration System) software product, as considered for the requirement specification pilot project using the RSL language.

Figure 4 shows that the PAS's requirements specification was mainly defined by using RSL User Story construct. This happen because the software house adopts the Scrum process, so using user stories better fits with these company procedures. A user story describes a user task, and may have relations with usability goals and usability QRs. All these relations can be defined in RSL-Excel-Template, as shown in Figure 5. It is important to notice that usability requirements impact with other concerns with different relation types like requires, conflicts or obstructs. For instance, the Single Sign-on usability goal (g_1_18) may require security concerns; or the Feedback usability goal (g_1_1) may conflict with performance-specific requirements. With RSL these issues can be properly expressed and validated.

PAS Requirements Specification

User Stories Definition

s_PAS

			as a	I want	so that		
Id (*)	Name (*)	Type (*)	Actor (*)	Goal (*)	Reason	Part Of	Description
us_30	Required fields	UserStory	a_1	I want to be informed of the mandatory fields that I must fill in the form of the patient's file	In advance of the process of recording the form I have all information about minimum fields for patient registration	us_7	The mandatory fields that the user must complete in the form to patient dentification must be checked: fields shaded in red and with * red, remains user after submit patient information
us_31	Required fields validation	UserStory	a_1	When recording the form, I must be advised of the fields that I did not fill in and that are required to complete the process	When recording the form, I must be advised of the fields that I did not fill in and that are required to complete the process		In the submission of the form must be validated incomplete fields and system presents error message: it was not possible to write the form without all required information complete.

Fig. 4. User stories of the PAS system (with the RSL-Excel-Template).

PAS Requirements Specification

Requirement Relations Definition

s_PAS

Goals, Functional Reqs, Quality Reqs, Constraint, UseCases, UserStories				DependsOn Type (*)	Description
Source (*)		Target (*)			
Id	Name	Id	Name		
QR_1_1	Warning	us_31	Required fields validation	Requires	us_31 should be evaluated from the metric defined in QR_1_1, warning
g_1_1	Feedback	us_31	Required fields validation	Supports	In system development process the goal g_1_1 feedback suports us 31, as functional description about user warning.
g_1_4	Responsive Web Design	ct_1	Responsiveness	Requires	Softweare system supports multi-device access, so system should use responsive Web technology to allow g_1_4.

Fig. 5. Requirements relations of the PAS system.

This evaluation took part as a focus group session with a multi-disciplinary group (i.e., with developers, QA specialists and product managers). This group involved 12 subjects with a median of 12-years of work experience. This group of subjects analyzed and evaluated not only the library of reusable usability requirements, but also they learned and they used the RSL language with the respective Excel template (they did not have any previous experience with this tool). The focus group took place in the company environment. The focus group session spans for 1-hour with two parts: The first part (the first 30 minutes) included a brief introduction and discussion to the ITLingo approach, the RSL language, and the library of reusable usability requirements. The second part (the remaining 30 minutes) included a hands-on training session, in which the subjects followed a script describing a predefined set of requirements to be defined in the RSL-Excel-template (e.g., "Open a list of entities, in any form, should not take more than 5 seconds to present results, for lists with more than 50 items the first results may be partial. (QR: Performance)". During this session the subjects made some suggestions, namely how to improve RSL-Excel-template usability itself.

Table 3. Theme-1: Assessment of RSL and RSL-Excel-Template (average score in a 0-5 scale)

Question	Q1	Q2	Q3	Q4	Q5	Q6
Average	3.5	4.4	4.1	4.2	4.4	3.4

Table 4. Theme-2: Assessment of the library of reusable usability requirements (average score in a 0-5 scale)

Question	Q7	Q8	Q9
Average	3.7	4.0	4.2

In the end of the session the subjects were asked to rate the evaluation, namely based on a questionnaire focused in two themes: (Theme-1) assessment of the RSL and the RSL-Excel-Template; and (Theme-2) assessment of the library of reusable requirements. The answers were classified in a Likert scale of: 0 (Not relevant or Do not know), 1 (Very Low), 2 (Low), 3 (Medium), 4 (High) and 5 (Very High).

The questions regarding Theme-1 were the following:

1. How do you assess the relevance of the requirements definition at the business level supported by the RSL-excel template?
2. How do you assess the appropriateness of the RSL constructs at Business Level (e.g., System Relations, Stakeholder, Processes, Glossary Term, Business Goal)?
3. How do you assess the relevance of the requirements definition at the system level supported by the RSL-excel template?
4. How do you assess the appropriateness of the RSL constructs at System Level (System, Actor, State Machine, Goal, QR, Constraint, FR, Use Case, User Story)?
5. How do you assess the learnability of requirements specification using RSL-Excel-Template?
6. How do you assess the usability of the RSL-Excel-Template for requirements specification?

On the other hand, the questions regarding the Theme-2 were the following:

7. How do you assess the appropriateness of the usability subtypes defined for the QR in the RSL grammar (i.e. the subtypes Appropriateness recognizability, Learnability, Operability, User error protection, User interface aesthetics, Accessibility)?
8. How do you assess the relevance of the usability requirements library, so that a software system may become more user-friendly?
9. How do you assess the impact that the usability requirements presented in the RSL-Excel-Template have in your product development regarding system functionality and architecture?

As summarized in Tables 3 and 4, the majority of the questions received a very positive score. However, it is relevant to mention that the RSL-Excel-Template usability highlights some difficulties as reported by the subjects of this evaluation. Some subjects also mentioned that RSL-Excel-Template could evolve to a web application that would be possible for novice users to work more easily.

A key conclusion from this evaluation is that the library of reusable usability requirements contributes positively to the elicitation, analysis and specification of software requirements.

Finally, to assess the overall approach two additional (yes/no) questions were put: (i) Would you consider to use the RSL-Excel-Template to improve your requirements specification activities? and (ii) Do you consider that the set of usability requirements (defined in the RSL-Excel-Template) is representative and comprehensive enough so that it can be applied to your own projects? To these two questions, all subjects considered that they would consider to use RSL on their own projects, and they considered that the library of usability requirements, in spite of not being exhaustive, is representative and comprehensive enough so that they could apply it to their own projects.

The number of participants involved in this assessment is sufficient to take preliminary but meaningful conclusions, namely considering that usability experts have noted that a group of 5 testers is enough to uncover over 80% of the usability problems [26]. Also, since this evaluation focuses on the usability of the language, tool support and approach, 12 participants is a reasonable number for an exploratory assessment, at least to identify challenges on the usability of these aspects.

6 CONCLUSION

Requirements specification is better done by using natural languages so it becomes easier to write and understand. However, this requires the need for validation methods to keep the coherence and rigor of such specifications. This research discusses the creation of a library of reusable usability requirements defined with the RSL language. This approach is enhanced by considering important usability concerns that are rigorously defined with RSL's Goal and QR constructs. The main objective of this library was to promote reuse and to help requirements engineers to be more productive and effective in their tasks, and developers to better design their systems with usability concerns.

It is popularly recommended to define usability requirements since the early stages of development process because such they may have a strong impact in subsequent tasks like software architecture and UI design. They also impact on other requirements like security or performance, which may affect software quality modification lately. However, if properly managed they can minimize rework and increase general customer satisfaction.

The proposed library includes ~40 goals and ~20 QRs. However, understanding and properly capture usability concerns in a generic way is a hard process that shall be developed in a continuous learning and research task in real world settings. As future work in this respect we mention the following areas. First, these requirements shall be further defined in order to explore both reusability and customizability. This means that requirements shall be further specified to express meaningful "parameterized elements" (or "template variables"), that would allow to be instantiated by concrete values when they were selected and instantiated to a concrete system. Second, we shall conduct more and extensive evaluation sessions to better evaluate and improve the library but also the supported tools. Third, map the dependencies among UI design decisions with functional and architectural software issues as well as to research and discover dependencies among different types of requirements.

ACKNOWLEDGEMENTS

Work supported by funds under FCT UID/CEC/50021/2019 and 02/SAICT/2017/29360.

REFERENCES

1. J. Nielsen, R. L. Mack. 1994. *Usability Inspection Methods*. John Wiley & Sons.
2. C. Aleixo, M. Nunes, P. Isaias. 2012. Usability and Digital Inclusion: Standards and Guidelines. *International Journal of Public Administration*, 35(3), 221-239.
3. X. Ferré, N. Juristo, H. Windl, L. Constantine. 2001. Usability basics for software developers. *IEEE Software*, 22-29.
4. Folmer, E., Gurp, J. V., Bosch, J. 2004. Software Architecture Analysis of Usability. In *Engineering Human Computer Interaction and Interactive Systems*, Springer, pp. 38-58.
5. Bass, L., John, B.E., Kates, J. 2001. *Achieving Usability through Software Architecture*. Carnegie Mellon Software Engineering Institute.
6. D. Ferreira, A.R. Silva. 2012. RSLingo: An Information Extraction Approach. In *Proc. of Model-Driven Requirements Engineering Workshop (MoDRE)*, IEEE.
7. ISO. 2011. *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. [Available online from https://www.iso.org/standard/35733.html, accessed on September 2019]
8. ISO. 2018. *ISO 9241-11:2018, Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. [Available online from https://www.iso.org/standard/63500.html]
9. L. Chung, J. Leite. 2009. On Non-Functional Requirements in Software Engineering. In *Conceptual Modeling: Foundations and Applications*. Springer.
10. G. Roman. 1985. A taxonomy of current issues in requirements engineering. IEEE Computer, 18(4), 14-23.
11. M. Glinz. 2005. Rethinking the notion of non-functional requirements. In *Proc. Third World Congress for Software Quality*.
12. J. Eckhardt, A. Vogelsang, D. M. Fernández. 2016. Are Non-functional Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice. In *Proc. IEEE/ACM 38th International Conference on Software Engineering*, IEEE.
13. D. Ferreira, A.R. Silva. 2013. RSL-IL: An Interlingua for formally documenting requirements. In *Proc. Model-Driven Requirements Engineering Workshop (MoDRE)*, IEEE.
14. A.R. Silva. 2017. Linguistic Patterns and Linguistic Styles for Requirements Specification (I): An Application Case with the Rigorous RSL/Business-Level Language. In *Proc. EuroPLOP*, ACM.
15. A. R. Silva. 2019. Rigorous Specification of Use Cases with the RSL Language. In *Proc. of International Conference on Information Systems Development*, AIS.
16. A. R. Silva, A. C. R. Paiva, V. R. Silva. 2018. Towards a Test Specification Language for Information Systems: Focus on Data Entity and State Machine Tests. In *Proc. of MODELSWARD'2018*, SCITEPRESS.
17. J. Nielsen. 1994. Enhancing the Explanatory Power of Usability Heuristic. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, ACM.
18. X. Ferre, N. Juristo, A. M. Moreno. 2005. Framework for integrating usability practices into the software process. In *Proc. Product Focused Software Process Improvement, 6th International Conference (PROFES)*, Springer.
19. N. Juristo, A. M. Moreno, M. Sanchez-Segura. 2007. Guidelines for Eliciting Usability Functionalities. In *IEEE Transactions on Software Engineering*, 33(11), 744-758.
20. M.V. Welie, H. Trætteberg. 2000. Interaction Patterns In User Interfaces. In *Proc. Seventh Pattern Languages of Programs Conference (PLoP)*.
21. W.O.Galitz. 2002. *The Essential Guide to User Interface Design*. Wiley & Sons.
22. N. Juristo, M. Lopez, A. M. Moreno, M. Sanchez-Segura. 2003. Improving software usability through architectural pattern. In *Proc. Workshop on SE-HCI*.
23. E. Folmer, J. Bosch. 2007. A Pattern Framework for Software Quality Assessment and Tradeoff Analysis. *International Journal of Software Engineering and Knowledge Engineering*, 17(04), 515-538.
24. E. Folmer, J. Bosch. 2004. Architecting for usability: a survey. *Journal of Systems and Software*, 70(1-2), 61-78.
25. K. Pohl. 2010. Requirements Engineering: Fundamentals, Principles, and Techniques. Springer.
26. J. Nielsen, T. K. Landauer. 1993. A Mathematical Model of the Finding of Usability Problems. In *Proc. INTERCHI'93*.
27. L. P. Gonçalves, A. R. Silva. 2018. A Catalogue of Reusable Security Concerns: Focus on Privacy Threats. In *Proc. of IEEE Conference on Business Informatics (CBI)*, IEEE.
28. L. P. Gonçalves, A. R. Silva. 2018. Towards a Catalogue of Reusable Security Requirements, Vulnerabilities and Threats. In *Proc. of International Conference on Information Systems Development (ISD)*, AIS.
29. M. Fernandes, A. R. Silva, A. Gonçalves. 2018. Specification of Personal Data Protection Requirements Interpretation of Legal Requirements from the GDPR. In Proc. of ICEIS'2018 Conference, SCITEPRESS.
30. A. Toval, J. Nicolás, et al. 2002. Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach. *Requirements Engineering Journal*, 6(4), 205-219, Springer.
31. C. Palomares, C. Quer, X. Franch. 2017. Requirements reuse and requirement patterns: a state of the practice survey. *Empirical Software Engineering*, 22(6), 2719-2762, Springer.
32. B. Schackel. 2009. Usability – Context, framework, definition, design and evaluation. *Interacting with Computers*, 21(5-6): 339-346.
33. M. Theofanos. 2007. Common Industry Specification for Usability - Requirements (CISU-R), NIST Pubs.
34. L. Chung, B. A. Nixon, E. Yu. 1994. Using Quality Requirements to Systematically Develop Quality Software. In *Proc. 4th International Conference on Software Quality*.
35. ITLingo - Specification Languages for the IT domain. 2018. [Available online from https://www.researchgate.net/project/ITLingo-Specification-Languages-for-the-IT-domain, accessed on September 2019].
36. A. Toval, et al. 2008. Eight key issues for an effective reuse-based requirements process. *Computer Systems Science and Engineering*, 23(6), 373.
37. A. Abran, A. Khelfi, W. Suryn. 2003. Usability Meanings and Interpretations in ISO Standards. *Software Quality Journal*, 11(4), 325-338.
38. A. Ribeiro, A. R. Silva. 2014. XIS-Mobile A DSL for Mobile Applications. In *Proc. ACM SAC'2014 Conference*, ACM.
39. J. Caramujo, et al. 2019. RSL-IL4Privacy A Domain-Specific Language for the Specification of Privacy-Aware Requirements. *Requirements Engineering*, 24(1) Springer.