Rigorous Specification of Software Usability Requirements

Carolina Lisboa Sequeira

Universidade Aberta\ Instituto Superior Técnico, Universidade de Lisboa,

Lisbon, Portugal

carolinasequeira@tecnico.ulisboa.pt

***Abstract*— [Context&Motivation] In recent years we have been witnessing the globalization of access to technology. Therefore, the interaction with software systems has ceased to be a privilege of some professionals and has become a reality for any-one in general. This phenomenon has created more demanding software users, not only in the functional features presented in a system, but also, in their end-users' satisfaction because they tend to opt for software labeled as "user friend-ly". Usability has therefore become a commodity, an essential quality of a prod-uct, if not an important differential in the business competition itself.**

[Question/Problem] The major challenge addressed in this research is to pro-pose a better approach to understand and define usability requirements. This is a difficult task because system requirements specifications tend to be poorly struc-tured and with little or no formal evidence regarding usability requirements. An-other common problem is the introduction of usability requirements late in the development process which results major rework and costs in projects.

[Principal ideas/results] This research identifies cross-cutting concerns in a general way to support the specification of usability requirements in a rigorous and structed way, and so, reducing their incoherence and ambiguity. To achieve this, we use the RSL Language (and respective tools) that allows improve the consistency of the specification process. In addition, a library of reusable usabil-ity requirements is proposed considering the concerns identified in the literature. This research has been conducted in a software house operating in the healthcare domain and has been applied and evaluated in its family of software products.

[Contribution] The most important contribution of this research is the proposal of a library of reusable usability requirements, rigorously specified in the RSL Language, that is both simple and understandable by humans as well as pro-cessable computer-based tools. Part of these usability requirements are defined at two complementary levels of abstraction, namely as system goals and as quality requirements (in measurable terms) that help to better specify a system which meets users' satisfaction in terms of usage.

***Index Terms* — Usability Requirements, Reusability, RSLingo RSL**

I. INTRODUCTION

Usability is a quality attribute that assesses how user interfaces are easy to use. The word usability also refers to methods for improving ease-of-use during the design process [1]. In the context of digital transformation scenarios, people are more demanding of the products they use, the experiences they have and how fast and simple they get information. Everything is getting "online" and organizations, including government, want to make better decisions, speed core business processes and get closer to their customers or citizens [2]. Digital transformation scenarios demand that the respective digital systems shall be simple, easy and accessible; end-users want to access the information they need and to get their work easily done. End-users like mobile applications that they can access on their phones and tablets to ensure that work continues even when they're on the go. That means that software systems *must be designed for people, and so that usability has become a commodity, an essential characteristic of a product.*

However, this context influences the software product's development process, *requiring that usability concerns shall be considered from the early stage of the development cycle*. Usability requirements must be identified during the specification and design's stage to prevent that later on (mainly during the product implementation) usability problems force software changes that can imply high rework and product costs [3]. This principle raises major challenges to the software engineering discipline. For example, during this research it was possible to identify that *usability is an interdisciplinary field* which contributes to the quality of the software; that usability is manifested in interaction design and even in software architecture [4], driving to functional and non-functional requirements and that sometimes *usability requirements meaning is ambiguous and hard to assign.*

After understanding that usability must be considered together with the analysis of other system concerns since the beginning of the development cycle, it is also important to stress that usability requirements are an important concern as well. They can be of different nature for example *Responsive Web Design or Reusing Information* [5]. It is essential that the *structures of the requirements specification bring accuracy to the process to reduce ambiguous meaning in the requirements*.

However, not only must the specification be more accurate but is also shall to be specified in a *natural language so that stakeholders can easily understand it*. RSLingo [6] is a possible solution in this scope because it uses natural language processing techniques to translate informal requirements into a formal representation defined in a rigorous language: the RSL language.

This research clarifies the usability attributes' comprehensiveness, through the proposal of a usability classification scheme and a set of reusable usability requirements rigorously specified in this RSL Language. This proposal was tested and validate in a real-world environment, namely in the context of a software house operating in the healthcare domain.

This paper is structured in 6 sections. Section 2 introduces the research background on non-functional requirements (classification schemes and representation structures) and usability attributes. Section 3 explores some usability concerns such as its definition, usability taxonomies and usability patterns. Section 4 presents the proposed library of reusable usability requirements, rigorously specified in RSL as System Goals and Quality Requirements. Section 5 discusses a hands-on requirements specification session performed in the context of a software house. Finally, section 6 presents the main conclusion of this research and suggested future.

II. BACKGROUND

Usability is commonly defined as a non-functional requirement (NFR), so it is important to reflect about the NFR's main classification schemes and representation structures. To understand how usability can be understandable as an important product's quality attribute [7], a usability taxonomy review is also conducted.

A. Non-functional requirements classification schemas

For this research, it is set up a brief review of ways of expressing NFR, that tend to define global attributes of a system, such as *integrity, safety, usability, reliability or performance*. For example, Chung and Leite, describe the state of the art about the treatment of NFR in software engineering, considering different definitions and representation schemes. In this paper, the emphasis is placed on functional requirements (FR), neglecting the fact that the functionality is not useful or usable without the characteristics commonly considered in NFRs. They consider that FRs are focused on "what makes software" while NFRs on "how well" software does something [8].

ISO/IEC 25010 (replaces ISO/IEC 9126) [7] is the ISO standard for software product quality that defines the quality characteristics that all software systems shall have. This standard distinguishes two classes of qualities: *Quality in use*, that relates to the outcome of the interaction when a product is used and *Product quality* that relates to software attributes and dynamic properties of the computer system. This model is applicable to both computer systems and software products. ISO/IEC 25010 is detailed with greater focus in Section 3.

There are process-oriented approaches that describe NFR like *goals to be achieved* [8] (functional hardgoals, nonfunctional hardgoals, functional softgoals and nonfunctional softgoals), or approaches oriented to the decision making [9], *requirements design decision types* (REDT), about different views like task, domain, interaction and system. REDT approach is interesting, because although in other similar approaches the process is initiated by the objectives, the support by tasks becomes more important, the system will only be accepted if the users can realize their tasks, making it easier to identify and validate tasks than objectives. Other works provide attributes, or categories to classified NFR, Roman [10] define requirements taxonomy, introducing NFR concepts like *Interface, Performance, Operating, Lifecycle, Economic or Political requirements*, in Glinz' s proposal [11] the four most important NFR classes for software architects, are *Performance, Usability, Security and Availability* (in this order). In some way, FURPS + [8] define a classification model of software quality attributes developed at Hewlett-Packard. In FURPS +, the following attributes are considered: *Functionality, Usability, Reliability, Performance and Compatibility*.

It can be considered that exists an explicit inconsistency between classification schemes that can be justified by the complexity of understanding, identifying and knowing the scope of an RNF.

B. Requirements representation structures

The analysis of several NFRs specification schemas raises some issues like the role they play in the full system development life cycle and the diversity of forms they can assume. The review shows that is possible to have structured schemas to repre-sent NFRs, such as SADT or SIG. SADT (Structured analysis and design technique) is a structured analysis modeling language that uses diagrammatic notation to describe systems and to represent its entities and activities relations. SIG (softgoal interde-pendency graph), used by NFR Framework, is a graphical representation to model NFRs, specifying the interdependencies among them, and determining how to opera-tionalize them [8].

A common way of representing NFRs is to list them separately from FR in different sections. This point of view is defended by IEEE 830, a good practice standard for specifying software requirement. In this standard NRFs and FRs shall be specified apart. NFR Assistant, a tool for the treatment of quality requirements during the development process of a system, also defended the separation of treatment between NFRs and FRs [11]. However, this view differed from other approaches that defend a holistic view of software, not separating FRs from NFRs, and that an entire modeling would only bring benefits. The separation of treatment between NFRs and FRs can compromise the system's overall view and the perception of dependencies among FRs and NFRs.

An important fact detected in background review is that all these approaches defended quality requirements elicitation since the beginning of software life cycle. In some way, NFRs are decisions or constrains on FRs, and in the end, this is all about goals to achieved, tasks to be done, and decision to be taken by users, in a context, using a software system [8] [12].

C. RSLingo

RSLingo is an approach for the formal specification of software requirements that uses natural language processing techniques to translate informal requirements into a formal representation through a specific language of requirements engineering [6] [13].

This approach concerns requirements specification in a natural language, which allows it to be more easily understood by the various actors in the process, stakeholders, developers, functional analysts or designers, but which is then mapped to a formal language. This process's phase allows computer-processed specifications to automatically check for certain requirements quality criteria, such as consistency, and integrity [6].

The RSLingo RSL language is expressed by a bi-dimensional and multi-view architecture, based in two levels of abstraction (business and system) and several requirements engineering concerns (context, active structure, behavior, passive structure and requirements), as shown in the Table 1 [14]. RSL includes constructs logically organized into views according to the specific requirements engineering they address. These constructs are defined as linguistic standards (represented as mandatory or facultative) and are represented textually by multiple linguistic styles [14].

Table 1 Classification of RSL Viewpoints: Abstraction Levels vs RE specific Concerns

Concerns / Levels	Package	Context	Active Structure (Subjects)	Behavior (Verbs, Actions)	Passive Structure (Objects)	Requirements
Business	package-business	Business System(s) relations	Stakeholder	BusinessProcess (BusinessEvent, BusinessFlows)	GlossaryTerm	BusinessGoal
System	package-system	System	Actor	StateMachine (State, Transition, Action)	DataEntity DataEntityView	SystemGoal QR Constraint FR UseCase UserStory

The possibility of defining and representing requirements (and co-related information) at different abstraction levels allows a richer and flexible approach. On one hand, the business level covers business context, key stakeholders, business glossaries, business processes, events and flows. On the other hand, the system level allows describing the requirements from the system perspective and based on multiple available styles, such as, system goals, quality requirements, functional requirements, constraints, use cases and user stories. In addition, RSL allows to establish mappings between all these type of requirements, for example, allows to establish relationships between business stakeholders and system actors, or between business goals and system requirements.

D. Usability attributes

NFRs are referred by different authors with different terminology for example as: goals [15] quality criterion (ISO\IEC 25010) [7] or quality attributes [8] [12]. Usability relates to the ability of an application to be understood, learned, used and attractive to the user, under specific conditions of use.

The article "Usability Meanings and Interpretations in ISO Standards" states that the definition and usability properties are not absolute in terms of interpretation, varying according to the target audience of the software product: managers, developers or end-users may have different interpretations of what a product' usability means [16]. 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. Finally, for an end-user the usability might means how quick and efficient she can do her 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 construction of a definition of usability and respective attributes, this research was deepened using a comparative analysis between several authors and published standards, as show in Table 2.

Table 2 Comparative table of usability attributes

	ISO/IEC 25010:2011	ISSO-9241-11	Nielsen	Schackel	CISU-R
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	
Attitude					
Accessibility	X				

Due to space constraint, we only summarize the main conclusions of this analysis.

Conclusions can be made that, 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 the definitions are ambiguous or show different perceptions, or different ways of combining attributes. Some of the attributes, such as "learnability" or "memorability" can be considered the same attribute, but with different names. The definition of "error" by ISO / IEC 25010: 2011 [7], as part of efficiency, is distinct from Nielsen's "error" design [17].

Considering the importance of usability as a requirement of a system, it becomes important to define techniques to evaluate its success. This issue was extensively conducted by Nielsen that proposes a set of ten usability heuristics to make it easier to perceive and evaluate usability common problems [17].

III. USABILITY CONCERNS

The ability to refine the concept of usability and usability attributes into more measurable and concrete concepts is essential to be able to include in the software product properties that make the software more "user friendly".

A number of usability concerns have been selected from literature that embody heuristics [17], patterns collections [18] [19] [4] [20], usability scenarios [5] and design principles [21] that researchers in this usability field consider to have a direct positive influence on usability.

A. A Taxonomy for Usability Requirements

ISO/IEC 25010:201 defines usability as "degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [7]. ISO/IEC 25010:201 was chosen in our proposal as a classification schema of usability requirements because it defines usability in a way that can be measured as software product property, not as a quality of software in use (user perceived quality). This standard defines a quality in use model that relate to the outcome of interaction when a product is used in a particular context of use, and a product quality model that relate to static properties of software and dynamic properties of the computer system.

Table 3 Usability's six sub-characteristics, ISO/IEC 25010:2011

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 has attributes that make it easy to operate and control.
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.

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, usability is break-down into six sub-characteristics (see Table 3) that help to further detail and classify each requirement. In some way, these usability sub-characteristics help to define high-level usability attributes.

B. Usability Patterns

To identify which usability concerns, must be taken into account in the requirements elicitation step one must first focused in usability patterns approaches that investigate the relationship between usability and software architecture.

In Guidelines for Eliciting Usability Functionalities [19] the various usability attributes are broken down into what the authors called "functional usability features," that is, software product features that introduce relevant usability benefits. The authors explain these features by "usability patterns" like wizards, shortcuts (key and tasks), context-sensitive help, history logging or action for multiple objects [22].

Some usability patterns defined by Folmer and Bosch [23] [24] were also consider in our research. Data validation, workflow model and users' modes are patterns namely by the authors.

Taking the users' perspective, Welie and Trætteberg describe interaction patterns in user interfaces [20]. These patterns are focused on solutions to common problems end-users have when interacting with systems. For instance, Grid Layout, Shield, Contextual Menu, Unambiguous Format, navigating between spaces, Continuous Filter and Command Area, are patterns that were also considered in our research.

Bass, John and Kates also 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 set of stimulus and response. For every scenario, they provided an architecture pattern that implements its aspect of usability.

From research, some important usability concerns were considered and included in our research, namely the following patterns: Maintaining Device Independence, Recovering 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 [5].

Furthermore, usability and software design are truly linked: good screen design requires understanding many things and develop usable software includes the user's entire experience. In this way, it shall be considered design guidelines, including aspects, like graphical issues (colors, menus, windows characteristics), that follows user interface design guides [21] and usability heuristics [1].

IV. A LIBRARY OF REUSABLE USABILITY REQUIREMENTS

The initial effort in determining usability concerns (as introduced in Section 3) allows to continue these research mapping all this concepts to a rigorous requirements specification. As referred in section 2, RSL language includes constructs logically organized into views according to the

specific concerns they address [14]. All the 25 usability concerns previously identified (in section 3) are specified in RSL namely as a set of reusable System Goals (SG) and Quality Requirements (QRs).

Figure 1 overviews the RSL constructs used to define the proposed library of usability requirements. System requirements can be defined in different types, according the nature of the requirement itself. System Goal is the result toward which effort is directed, an objective that should be achieved, expresses what is desire and the context in which the motivations and rationales behind it can be understood. Quality Requirements express "quality" of the system and define specific metrics that the system must meet to be accepted. In order to ensure consistent requirements rules is important to define relations among them. Finally, user stories are short descriptions of functionality told from the user's perspective. The focus is on why and how the user interacts with the software. This perspective is important to check SG or QRs applicability, in a use context, but usually shall not be defined as a reusable requirement.

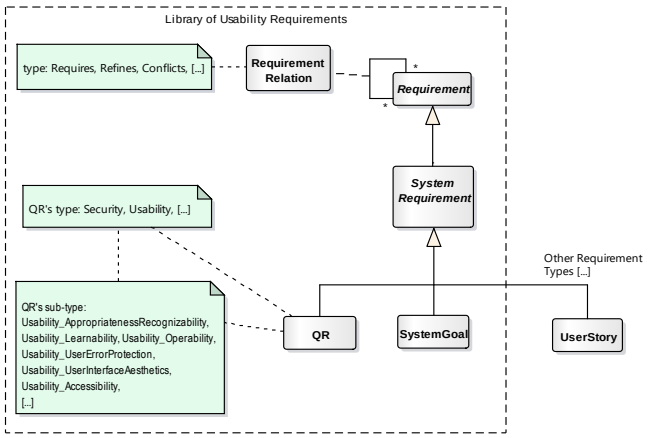


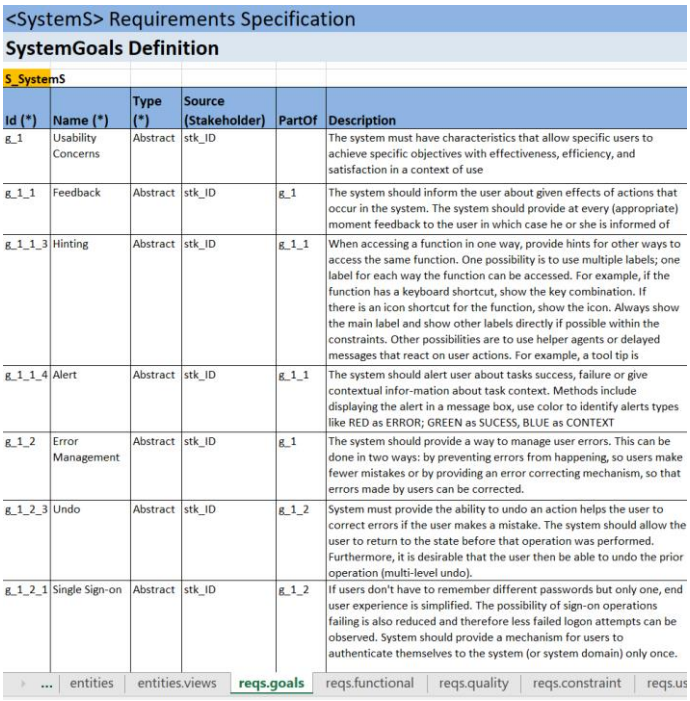
Figure 1 Overview of the Library of Usability Requirements

RSL is supported by some tools namely by: RSL-Excel-Template, that allows requirements specification based on an Excel template; RSLingo-Studio that supports RSL automatic validation as well as model-based transformations; and ITBox that is a collaborative web platform for document management of requirements specification and other technical documentation.

A. *Usability System Goals*

Figure 2 shows some of these usability concerns as system goals. This view shows the system goals to be achieved in scope of usability, it can also be aggregated in goals and sub-goals in a way that is part of relations can be expressed between them.

During requirements writing process RSL Excel-Template presents guidelines and grammar structure to check for certain requirements quality criteria, such as clarity, consistency, and integrity. All requirements can be prioritized to be gradually assign in development process and can even be related with a source, a stakeholder that make a valid requirement in a context of use.



<SystemS> Requirements Specification

SystemGoals Definition

S_SystemS

Id (*)	Name (*)	Type (*)	Source (Stakeholder)	PartOf	Description
g_1	Usability Concerns	Abstract	stk_ID		The system must have characteristics that allow specific users to achieve specific objectives with effectiveness, efficiency, and satisfaction in a context of use
g_1_1	Feedback	Abstract	stk_ID	g_1	The system should inform the user about given effects of actions that occur in the system. The system should provide at every (appropriate) moment feedback to the user in which case he or she is informed of
g_1_1_3	Hinting	Abstract	stk_ID	g_1_1	When accessing a function in one way, provide hints for other ways to access the same function. One possibility is to use multiple labels; one label for each way the function can be accessed. For example, if the function has a keyboard shortcut, show the key combination. If there is an icon shortcut for the function, show the icon. Always show the main label and show other labels directly if possible within the constraints. Other possibilities are to use helper agents or delayed messages that react on user actions. For example, a tool tip is
g_1_1_4	Alert	Abstract	stk_ID	g_1_1	The system should alert user about tasks success, failure or give contextual infor-mation about task context. Methods include displaying the alert in a message box, use color to identify alerts types like RED as ERROR; GREEN as SUCESS, BLUE as CONTEXT
g_1_2	Error Management	Abstract	stk_ID	g_1	The system should provide a way to manage user errors. This can be done in two ways: by preventing errors from happening, so users make fewer mistakes or by providing an error correcting mechanism, so that errors made by users can be corrected.
g_1_2_3	Undo	Abstract	stk_ID	g_1_2	System must provide the ability to undo an action helps the user to correct errors if the user makes a mistake. The system should allow the user to return to the state before that operation was performed. Furthermore, it is desirable that the user then be able to undo the prior operation (multi-level undo).
g_1_2_1	Single Sign-on	Abstract	stk_ID	g_1_2	If users don't have to remember different passwords but only one, end user experience is simplified. The possibility of sign-on operations failing is also reduced and therefore less failed logon attempts can be observed. System should provide a mechanism for users to authenticate themselves to the system (or system domain) only once.

... | entities | entities.views | reqs.goals | reqs.functional | reqs.quality | reqs.constraint | reqs.us

Figure 2 Requirements Goals in RSL-Excel-Template

B. *Usability Quality Requirements*

Usability may depend on several variables, like goals of use, context of use, tasks or users. Like any other goal, usability can be detailed in quantitative and measurable ways. Clear and concrete goals also provide objectives for usability testing and ensure that a faulty or unsatisfactory software will not be released. To this purpose, RSL quality requirements (QRs) shall be used. These usability QRs can be still classified as a subtype, as defined by ISO/IEC 25010:201's usability sub-characteristics.

Figure 3 shows the quality requirements' classification schema, and Figure 4 presents a concrete set of quality requirements.

All this usability requirements were defined with no particular system as context, defining common notations to introduce usability concerns in a software system. So, any software system specification can adopt this usability requirements as common base to define its own user-friendly system.

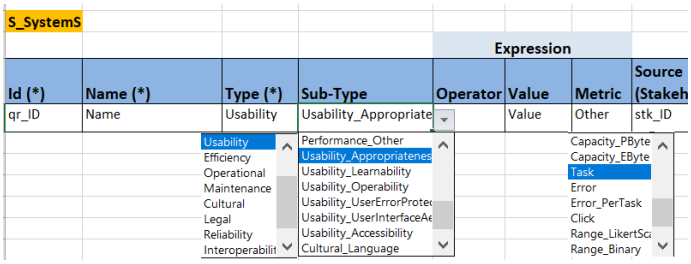


Figure 2 Quality Requirements in RSL-Excel-Template

<SystemS> Requirements Specification

Quality Requirements Definition

S_SystemS

Id (*)	Name (*)	Type (*)	Sub-Type	Metric	Operator	Value	Source (Stakeholder)	Part Of	Description
QR_1	Error prevention	Usability	Usability_UserErrorProtection				stk_3		Error messages and or success messages are clear and perceptible, but even better than good error messages is a careful design which prevents a problem from occurring in the first place.
QR_1_1	Warning	Usability	Usability_UserErrorProtection	Time_Sec	=	30	stk_3	QR_1	Warn the user before continuing the task and give the user the chance to abort the tasks. Error warning should be visible to user during 30 seconds.
QR_1_3	Feedback - success operation	Usability	Usability_UserErrorProtection	Time_Sec	=	10	stk_3	QR_1	User must be warned of the successful operation end, without one more click intervention do confirm.
QR_2	Navigation: Multi-tasking	Usability	Usability_Operability	Click	=	3	stk_3		User should not have to give more than 3 clicks to navigate to menu for a new task.
QR_3	Users perceptions	Usability	Usability_UserInterfaceAesthetics	Range_LikertScale	>=	3	stk_6		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 between 3 and 5 in the Likert Scale.

Figure 4 QR Implementation details in RSL-Excel-Template

V. RESEARCH EVALUATION

The results of this research have been applied and assessed in the scope of a software house that operates in the healthcare domain. This software house makes part of a multinational technology and consulting company, with € 66.1 million turnover in 2016 and it has approximately 900 employees, operating from 10 offices in 6 countries: Portugal, Spain, 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 it was used a software product (the patient administration system), as considered in requirement specification pilot project using RSL. Figure 5 shows a list of requirements specification based on user-stories (that is because the involved software house has adopted the SRUM process).

A user story represents a task to a user, and has relation with usability system goals, that can be evaluated using a usability QR. All these relations can be explored in RSL-Excel-Template, as shown in Figure 6.

PAS Requirements Specification

User Stories Definition

s_PAS

Id (*)	Name (*)	Type (*)	as a Actor (*)	I want Goal (*)	so that Reason	Part Of	Description	Priority
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	Must
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.	Must

Figure 5 User story in 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	ct_1	Responsiveness	Requires	Softweare system supports multi-device access, so system should use responsive

Figure 6 Requirements relations in RSL-Excel-Template

It is important to notice that some usability concerns have impact in other requirements, in a way that they can be identical, supports, conflicted or obstruct. For instance, Single Sign-on have security concerns, and feedback can have performance conflicts with usability, with RSL all these issues can be expressed and validated.

The action research methodology was followed through different cycles that allowed us to continuously improve the requirements specification.

A multidisciplinary team (developers, QA specialists and product managers) with a median of 12-year-old work experience took part in a focus group section, of 12 participants, to analyze and evaluate not only the library of reusable usability requirements, but also the experience of learning and the using RSLingo.

The focus group toke place in software house's environment, concepts and RSL-Excel-Template experience was new for all team. The focus groups have a 1 hour duration. In first 30 minutes was presented RSL approach, and the library of reusable usability requirements was discussed. In other 30 minutes, a hands-on training was conducted. The team follow a script describing a s set of pre-defined requirements like: "Opening of list of values, 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 section, team made suggestions about how to improve RSL-Excel-

template usability. In the end, team were asked to rate RSL-Excel-template and the library of reusable usability requirements. The questionnaire focused in two groups:

Group 1 - RSL-Excel-Template: the relevance and how appropriated are concerns typologies described in RSL-Excel-Template (RSLingo approach concepts) and its usability and learnability.

Group 2: The library of reusable requirements and usability types appropriateness evaluation.

The answers were classified in a scale of: 0 (N/A – Do not know), 1 (Very Low), 2 (Low), 3 (Medium), 4 (High) and 5 (Very High).

The possible answers for the questions regarding to group 1 are presented below.

1. Evaluate the relevance of the requirements definition at the business level presented in the RSL-excel template?
2. Classify the appropriateness of the concerns' typology in Business Level (Business System Relations, Stakeholder Business Processes, Glossary Term, Business Goal).
3. Evaluate the relevance of the requirements definition at the System Level presented in the RSL-excel template.
4. Classify the appropriateness of the concerns' typology in System Level (System, Actor, State Machine, System Goal, QR, Constraint, FR, Use Case, User Story).
5. Classify the learnability of the requirements specification using RSL-Excel-Template.
6. Classify the usability of the RSL- Excel template for requirements specification.

Table 4 Survey average score (in a scale of 0-5) by Group 1: RSL-Excel-Template

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

All the questions had a very positive score, however it is noted that the overall RSL-Excel template usability highlights some difficulties during hands one training. For some team elements, RSL should evolve to a web platform with a user interface that make possible to novice stakeholders work more easily.

The possible answers for the questions regarding to group 2 are presented below:

1. Classify the appropriateness of the usability subtypes typified as QR in the RSL grammar (Appropriateness recognizability, Learnability, Operability, User interface Aesthetics Accessibility).
2. Evaluate the relevance of the usability requirements library, so that a software system becomes more user friendly.
3. Evaluate the impact that the usability requirements presented in the RSL-Excel-template have in product development regarding system functionality and architecture.

Table 5 Survey average score (in a scale of 0-5) by Group 2: Usability Concerns

Question	Q1	Q2	Q3
Average	3.7	4.0	4.2

An import review about usability concerns, is that the library of reusable usability requirements has important concepts and usable definitions to support elicitation, analysis, and operationalization of these requirements for software systems.

To evaluate the overall initiative were made two questions were possible answers were: (1) Yes and (2) No.

1. Would you use this RSL-Excel-Template to requirements specifications?
2. Do you consider that the list of usability requirements (identified in the RSL-Excel-Template) is representative and comprehensive so that it can be applied to a whole software system family?

All team considered that would use RSL-Excel-Template on their engineering projects, and considered that the library of usability requirements is representative and comprehensive so that they can be applied to a complete family of software products.

VI. CONCLUSION

This research proposed usability requirements specification approach based on the RSL language. This approach was enhanced by considering important usability concerns available in the literature that were rigorously defined in RSL system goals and quality requirements (these in measurable terms). As a result, a library of usability requirements was specified; this library helps requirements engineers to be more productive and effective in their tasks, and developers to better design their systems which meets users' satisfaction in terms of usage.

There are satisfactory solutions by introducing usability requirements in early stage of development process, because in the relations created during specification processes it was possible to identify that usability requirements affected several software domains like architectural and interface design.

Usability requirements affect other requirements like security or performance which affects software quality modifications lately in the process and it can minimize engineering costs and provide customer's satisfaction. In the requirements analyses and specification process, a system and business goals overview is advised.

Requirements specification process can be improved using natural language because it is easier to write and understand. However, it is important to introduce validation methods to validate the coherence and rigor of requirements specification.

Contributions over usability concerns are collected as this research is still in progress. Understanding the usability

comprehensiveness is a hard work and must be developed in continuous learning and experimentation. An important work is mapping interface design decisions with functional and architectural software issues.

On the other hand, in the scope of RSLingo approach, we suggest that the future research should focus on improve RSL Excel Template usability (e.g., support some contextual help for the template filling, or even include some data validation rules). In addition, extending the RSL language with some constructs for testing specification (with a tight relation with requirements) shall also an important contribute for future work.

References

[1] Nielsen, J., Mack, R.L.: Usability Ispection Methods, John Wiley & Sons, 1994.

[2] Aleixo, C., Nunes, M., Isaias, P.: "Usability and Digital Inclusion: Standards and Guidelines," *International Journal of Public Administration*, vol. 35, no. 3, pp. 221-239, 2012.

[3] Ferré, X., Juristo, N., Windl, H., Constantine, L.: "Usability basics for software developers," *IEEE Software*, pp. 22-29, Jan/Feb 2001.

[4] Folmer, E., Gurp, J. V., Bosch, J.: "Software Architecture Analysis of Usability," in *Engineering Human Computer Interaction and Interactive Systems*, Berlin, Heidelberg, Springer, 2004, pp. 38-58.

[5] Bass, L., John, B.E., Kates, J.: "Achieving Usability Through Software Architecture," Carnegie Mellon Software Engineering Institute, Pittsburgh, 2001.

[6] Ferreira, D.A., Silva, A.R.: "RSLingo: An Information Extraction Approach," in *Model-Driven Requirements Engineering Workshop (MoDRE)*, Chicago, Illinois, USA, 2012.

[7] "International Organization for Standardization," International Organization for Standardization, 2017. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en. [Accessed 28 9 2017].

[8] Chung, L., Leite, J.C.S.P.: "On Non-Functional Requirements in Software Engineering," in *Conceptual Modeling: Foundations and Applications*, Springer Berlin Heidelberg, 2009, pp. pp 363-379.

[9] Paech, B., Kohler, K.: "Usability Engineering integrated with Requirements Engineering," in *ICSE Workshop on SE-HCI*, 2003.

[10] Roman, "A taxonomy of current issues in requirements engineering," *IEEE Computer*, vol. 18, no. 4, pp. 14-23, April 1985.

[11] Glinz, M.: "Rethinking the notion of non-functional requirements," in *Third World Congress for Software Quality, Munich, Germany, September 2005*, 2005.

[12] Eckhardt, J., Vogelsang, A., Fernández, D.M.: "Are "Non-functional" Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice," in *38th International Conference on Software Engineering, Austin, Texas*, 2016.

[13] Ferreira, D.A., Silva, A.R.: "RSL-IL: An interlingua for formally documenting requirements," in *2013 3rd International Workshop on Model-Driven Requirements Engineering (MoDRE)*, Rio de Janeiro.

[14] Silva, A. R. d.: "Linguistic Patterns and Styles for Requirements Specification: The RSL/Business-Level Language," in *EuroPLOP'2017*, Baviera, 2017.

[15] Chung, L., Nixon, B. A., Yu, E.: " Using Quality Requirements To Systematically Develop Quality Software," in *Fourth International Conference on Software Quality McLean, VA, U.S.A. October 3–5*, 1994.

[16] Abran, A., Khelfi, A., Suryn, W.: "Usability Meanings and Interpretations in ISO Standards," *Software Quality Journal*, p. 325–338, 2003.

[17] Nielsen, J.: "Enhancing the Explanatory Power of Usability Heuristic," in *CHI '94 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Boston, Massachusetts, USA , 1994.

[18] Ferre, X., Juristo, N., Moreno, A. M.: "Framework for Integrating Usability Practices into the," in *Product Focused Software Process Improvement, 6th International Conference, PROFES 2005*, Oulu, Finland, Springer, 2005.

[19] Juristo, N., Moreno, A.M., Sanchez-Segura, M.: "Guidelines for Eliciting Usability Functionalities," in *IEEE Transactions on Software Engineering* , 2007.

[20] Welie, M.V., Trætteberg, H.: "Interaction Patterns In User Interfaces," in *Proc. Seventh Pattern Languages of Programs Conference: PLoP 2000*, Monticello, Illinois, USA, 2000.

[21] Galitz, W.O.: The Essential Guide to User Interface Design, New York: Robert Ipsen, 2002.

[22] Juristo, N., Lopez, M., Moreno, A.M., Sanchez-Segura, M.: "Improving software usability through architectural pattern," in *ICSE Workshop on SE-HCI*, 2003.

[23] Folmer, E., Bosch, J.: "A Pattern Framework for Software Quality Assessment and Tradeoff Analysis," *International Journal of Software Engineering and Knowledge Engineering* , pp. 515-538, 2007.

[24] Folmer, E., Bosch, J.: "Architecting for usability: a survey," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 61-78, 2004.