Journal of Software Engineering and Applications, 2014, 7, 592-601
Published Online June 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.77055

Computer Aided Call for Tenders: A Tool for Software Bidding

Jorge Hochstetter, Carlos Cares

Systems Engineering Department, University of La Frontera, Temuco, Chile
Email: jorge.hochstetter@ceisufro.cl, carlos.cares@ceisufro.cl

Received 20 February 2014; revised 20 March 2014; accepted 27 March 2014

Abstract

Under a broad comprehension of Software Engineering, it is preferred the term software life cycle instead of just software production. The reason is that cycle starts at software conception and stops when the software is relegated. Given contemporary companies' market strategies of focusing on their competitive advantages, most of them have externalized their software production to outsourced services. Therefore, the call for software tenders has become a common step in the software life cycle. In this paper, we present a tool which aids non-experts to specify call for software tenders. The motivation was Chile situation of most of rural and semi-urban city halls which do not have engineering teams to build call for software tenders. We describe the problem in terms of lack of competitiveness and empty biddings generated by low quality calls for tenders. As a second step, we show the technical considerations to develop the proposed tool and its features. We include an initial tool perception from a first group of users.

Keywords

Call for Tenders, Software Bidding, Requirements Specifications, Software Outsourcing

1. Introduction

From three decades ago, outsourcing has been a clear industry mainstream [1] [2] which also has covered IT services and particularly software development [3]. Nowadays, both private and public organizations call for software tenders (CSTs) when they have business process automation necessities [4].

Due to transparency issues, most of the countries publish their demands of external services on public web sites, e.g. Brasil on www.comrprasnet.gov.br, Mexico on compranet.gob.mx, Ecuador on compras publicas.gob.ec.

Therefore, different enterprises study these elicitation documents and make the decision of applying based on their contents. Acquiring products and services using this way normally means getting specialized consulting and ensured results.

In particular, since October 2004, in Chile came into force a law about public acquisitions which regulates government's demands and makes mandatory to publish elicitation documents on its purchases' web site. Although the law is not explicit, it is understood that it includes the purchase of software products and services as developing or maintaining software systems [5].

In spite of requirements, engineering approaches assume and recommend building a call for tenders after requirements elicitation [6]-[8]. We have found that, at least in Chile, most of the public calls for software tenders (CSTs) do not have enough information to sustain that they have included a requirements elicitation stage; moreover, several of them explicitly require a software requirements stage as part of the solution.

In addition, we have also detected a big proportion of low quality CST documents. Although we could speculate and theorize about reasons, the fact is that these documents do not include a minimal set of wanted functions, they do not describe organizational actors, and most of the time they include neither non-functional requirements, nor basic project stages, nor their corresponding deliverables.

We have previously sustained that software-bidding process requires new supporting mechanisms and specific procedures for both, tenders and demanders [6] [8]. In particular, these kinds of procedures and tools seem more urgent in rural-municipality governments of Chile because they have few professionals who play too different roles. Thus, most of the time, there is the role of purchases responsible which has in charge of buying a broad diversity of materials and services. The education and training of these professionals does not allow considering them as software engineers.

In this paper, we present a web-based software for supporting the creation of CSTs. We have joined main management topics such as budget, project stages and deliverables to the known structure of IEEE830 software requirements standard [9] in order to generate the basic structure of a CST. Besides, we have selected a set of basic contents and examples which can be added edited to generate the main outcome of the tool: a CST document. We have named this tool T2R2-Software. We also present some initial perception of a group of users from purchasing units of three different rural and semi-urban Chilean municipalities.

This paper has been sorted under the following structure: In Section 2, we present the problem in terms of quantitative and qualitative descriptions: quantitative descriptions correspond to metrics based on content analyses whilst qualitative descriptions are based on descriptions which were gathered in a focus group session of experienced software tenders and by our own contact with these governmental units. In Section 3, we present the assumptions which have guided us to the current design of the human-computer interaction (HCI) and functionality. Here, we briefly present the first experience and perception of a small set of initial users. In Section 4, we show the basics of software architecture which allow us easily modifying sections and examples of CSTs. Finally, in Section 5, we summarize the conclusions and present the planned future work, mainly related to validation of the tool and its impact on local governments. Besides, we add ideas about how to improve the collective construction of CSTs in new versions of T2R2-Software.

2. Problem Description

The main motivation for developing the tool for software bidding were our findings related to the gap between requirements engineering recommendations (e.g. see [2] [10]-[12]) and the current behavior of software purchasers from the Chilean state. On one side we have the assumptions and recommendations sustaining that a call for tender is a stage, which runs after requirements elicitation. On the other side we have a wide set of real CST documents, on which is not possible to sustain that they contain a requirements specification.

Moreover, we have previously affirmed that CSTs are different from software requirements specifications (SRSs) [7]. CSTs normally add methodology constraints, stages related to project's deliverables and their associated pays, among other contents. However, we have also detected high dispersion on long, technical specifications, business goal specifications, even for similar software products. In order to measure this lack, we have created a metric based on the similarity between the actual content of a CST and the expected content of a SRS based on IEEE830. We analyzed 477 CSTs corresponding to Chilean governmental purchasing initiatives. As a result, we get that 80% of them (383) do not reach the 5% of similarity (even the wide consideration of using multiple synonymous to find some similarity). Even the resting 20% (94) do not go beyond 37% of an expected SRS. In this study we let out measures related to project constraints and business constraints. Therefore, at least theoretically, analyzed CSTs could have a lower quality than the reported by the applied metric.

In order to complete the scenario, other studies support the idea that CSTs only contain a first approach to a requirements elicitation stage [13]-[15]. However, at the same time, a proper expectation is that provided information allows making a feasible technical proposal having regard to the contract sum.

Our first qualitative approach to the problem was by reviewing the set of best proposals. These calls correspond to municipalities of main cities and ministries. As a constant, all worst proposals (from the point of view of expected content) were from rural municipalities. This approach also included gathering some initial information of these offices. Calling to the corresponding people in charge we normally found technical people from management (normally supporting purchasers in general) and few times technical computing professionals. We called to more than 30 rural municipalities from the south zone of Chile; none of them have a software engineer for carrying out the software procurement process. Therefore, a relevant finding here was that a critical stage of a software life cycle is being developed by people without training on software project matters.

We are absolutely convinced that studying this phenomena worth the effort. But, at the same time, we do not want to wait to have a complete understanding of this phenomenon to try to improve it. We would like to contribute now to some initial solution by providing a recommender system which aids to existing people in charge to build better CSTs. We believe that studying some eventual adoption and their results could give us a better understanding than studying the phenomenon just as external observers.

Therefore, we have not proposed a general-term solution to the problem of building a proper CST document, but, we are trying to provide a tailored tool for the specific situation of software public procurements from rural and semi-urban municipalities of Chile.

3. Applying HCI Principles

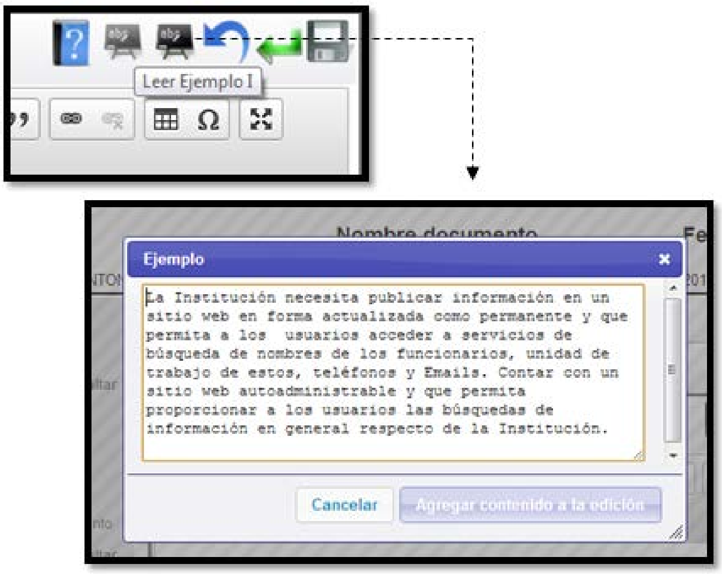
In this section, we review traditional concepts of HCI design and how they have been applied to the design of our T2R2-Software tool.

We start applying the concept of affordance. In [16] four types of affordance are recognized. The first of them regards to cognitive affordance, considered *a design feature that helps*, *aids*, *supports*, *facilitates*, *or enables thinking and*/*or knowing about something*. In the case of our tool the background of technical managers, we considered that known computing environments are word editors. Therefore the main interaction space looks like a word editor where the CST can be specified. The second type is physical affordance; it helps to relate visible objects to functions. Thus we use traditional icons from word processors (and many other applications) such as old storing device for saving, a question symbol for helping, among others. The third type of affordance is functional affordance referred to the feature that *helps or aids to the user to do* something. Thus, we provide two buttons which activate the visualization of examples of the needed sections of a CST. These examples can be—by one click—added to the main text and then modified. Thus we implement the principle that the user clicks to accomplishing a goal and the purpose achieved is to complete a specific section of the CST (**Figure 1**).

The fourth affordance type is sensory affordance, defined as the feature that *helps*, *aids*, *supports*, *facilitates*, *or enables the user in sensing*. At this respect the main added feature is the seeing the visual summary of completed, empty and editing document sections. **Figure 2** illustrates the implementation of this feature.

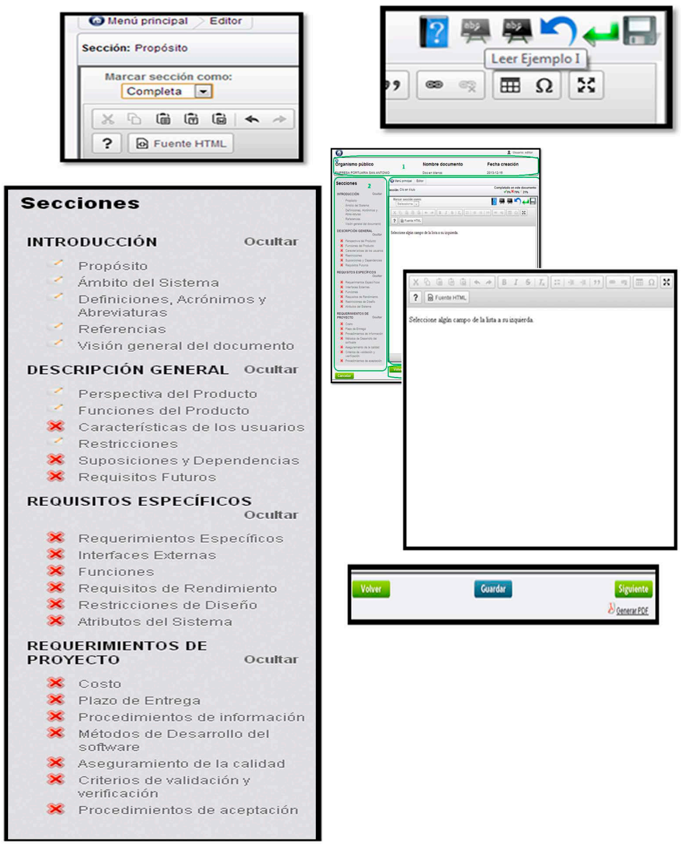
A second HCI design principle is reducing cognitive overhead. It has simply defined as the challenge of simultaneously maintaining several tasks [17]. Actually, in rural municipalities is very scarce to find two CSTs in the same period, therefore it does not seem necessary to go on a theoretical problem of summarily showing the state of several CSTs at the same time. What is necessary is to offer some help for knowing the individual CST's advancing state. Thus the implemented feature of sections' states (**Figure 2**) helps for reducing cognitive overhead. Moreover, helping feature, which describes the expected content of each section, helps to users to remember a desirable CST structure.

Learnability has been another point where we have focused our design. Given that the concept of learnability has several acceptations, we have used some ideas from the framework presented in [18]. The first main idea is that learnability depends on users' experience. Therefore, we have already considered that users are management people who know word processors, but we also considered the expansion of internet browsers. Therefore, one of our first decisions was implementing a web based application. It also facilitates the process of installing and configuring the software. Concerning Internet penetration, Chile has good numbers respect to its region [19] [20] thus it was not considered a problem.



La Institución necesita publicar información en un sitio web en forma actualizada como permanente y que permita a los usuarios acceder a servicios de búsqueda de nombres de los funcionarios, unidad de trabajo de estos, teléfonos y Emails. Contar con un sitio web autoadministrable y que permita proporcionar a los usuarios las búsquedas de información en general respecto de la Institución.

Figure 1. Accessing examples for implementing functional affordance.



Secciones

INTRODUCCIÓN Ocultar
- ✓ Propósito
- ✓ Ámbito del Sistema
- ✓ Definiciones, Acrónimos y Abreviaturas
- ✓ Referencias
- ✓ Visión general del documento

DESCRIPCIÓN GENERAL Ocultar
- ✓ Perspectiva del Producto
- ✓ Funciones del Producto
- ✗ Características de los usuarios
- ✓ Restricciones
- ✗ Suposiciones y Dependencias
- ✗ Requisitos Futuros

REQUISITOS ESPECÍFICOS Ocultar
- ✗ Requerimientos Específicos
- ✗ Interfaces Externas
- ✗ Funciones
- ✗ Requisitos de Rendimiento
- ✗ Restricciones de Diseño
- ✗ Atributos del Sistema

REQUERIMIENTOS DE PROYECTO Ocultar
- ✗ Costo
- ✗ Plazo de Entrega
- ✗ Procedimientos de información
- ✗ Métodos de Desarrollo del software
- ✗ Aseguramiento de la calidad
- ✗ Criterios de validación y verificación
- ✗ Procedimientos de aceptación

Figure 2. It shows section states for implementing sensory affordance.

The second idea of this learnability framework includes the concept of extended learnability, which means to consider how efficiency could be improved over time. To this point we consider that existing CST can be used as initial patterns of new ones. Therefore, experience is not just a matter of recognizing buttons and functionality, but also of having a growing set of documents which can be continually improved and reused.

Finally, also a classical HCI design principle is user satisfaction which is considered an elusive concept having different approaches to measure it [21]. A contemporary approach review main variables influencing user satisfaction [22]. Given its general terms we have followed its refinement in the factors: 1) system effectiveness, 2) user effectiveness, 3) user effort and 4) user characteristics.

We have considered each of these items in the following adapted manners: 1) system effectiveness measure how well the proposed tool accomplish its goal, *i.e.* the construction of a CST document. In the case of T2R2, we have not just provided an adequate and known user interface but also we have provided an additional function for producing the final document in PDF format, due to the Chilean public platform accepts documents in this format. We have also been award of 2) user effectiveness by providing explanations and examples of the needed sections of the expected document. In general, this factor is closely related to the concept of affordance. 3) Under the same principle we have designed the tool to facilitate the current manual task of specifying a CST. Therefore, the user total effort (given their features) is effectively reduced. Finally, in all cases we have considered 4) user characteristics. In fact, from the beginning we have characterized the expected user in his/her business context due to he/she presents particular education and role in their institutions.

Finally, and in spite of the different approaches to measure user satisfaction, there is a common way which includes applying some instrument in order to know their perception. In the case of T2R2-Software, we have presented our resulting tool to a group of 6 users that works on rural municipalities and their job include specifying CSTs. The related profile of participants is presented in **Table 1**. Here we consider years of experience as purchasers (second column), the percentage of journey spend on making CSTs (third column) and their perception about spent time on making a simple (fourth column) and a complex (fifth column) CST.

To these participants we explained the tool on two sessions of one and half hour. After that, we applied a usability questionnaire. The results are showed in **Table 2**. The participants correspond to the same id from **Table 1**. We have summarized here three questions qualified between 1 (lowest level) to 10 (highest level). The first question regards the improvement of the final CST's quality (second column), *i.e.* efficacy. The second question regards the time for producing the CST (third column), *i.e.* efficiency. Finally, the third question regards the quality of HCI in order to get previous goals (fourth column), *i.e.* usability. As it can be seen, their initial perception has been very good. However, we have reasonable doubts about the feasibility of reaching a professional quality of CST, mainly because neither of them thinks that specifying a complex software product may spend more than two weeks.

Table 1. Profile of participants.

Participant	Exp (Years)	Dedication to CSTs (%)	Hours to specify a simple CST	Hours to specify a complex CST
1	5 to 6	0% - 15%	0 - 4	21 - 50
2	Less than 1	0% - 15%	-	-
3	Less than 1	0% - 15%	0 - 4	0 - 4
4	More than 6	31% - 50%	5 - 20	51 - 100
5	1 to 2	0% - 15%	5 - 20	5 - 20
6	Less than 1	0% - 15%	0 - 4	0 - 4

Table 2. Answers from participants.

Participant	T2R2 helps to get a high quality CST	T2R2 helps to save time	T2R2 is simple of using
1	9	8	9
2	10	5	6
3	10	10	5
4	10	9	8
5	10	10	7
6	10	10	10

4. T2R2's Architecture

In order to show the architecture we describe three main components: 1) technological platform, 2) conceptual architecture; 3) functional architecture. Finally given secondary research goals, *i.e.* to understand the process of generating call for tenders, we have also reported the basis of T2R2-Software's log system.

4.1. Technological Platform

From the technical point of view the application is a web-based software product using a MySQL database. It has been programmed in Java Server Pages (JSP) using a package of Java classes corresponding to the business layer. Therefore, T2R2 only requires the HTML virtual machines incorporated into Internet browsers and an Internet point for accessing the web site.

4.2. Conceptual Architecture

We have used UML class diagrams to show the main modeled concepts and the corresponding object collections which constitute the conceptual model (business layer) of our application. In **Figure 3**, the conceptual model is illustrated.

The *Governmental Office* class represents the collection of all purchasing organizations from state. The representation corresponds to a simple model of state management offices or units. The dependencies (hierarchy) among them have not been modeled. This class has been associated to *Software Specifier*, which represents the collection of persons who have the task of purchasing some software product; thus, they have to make a CST. The multiplicity of the association between *Software Specifier* and *Governmental Office* is many-to-many because we have considered that a person can change his/her job from one state unit to another or even it is possible that a person works part-time for more than one state unit. Moreover, *Software Specifier* has the stereotype <<actor>> because objects from this class also represent system actors (from use cases view).

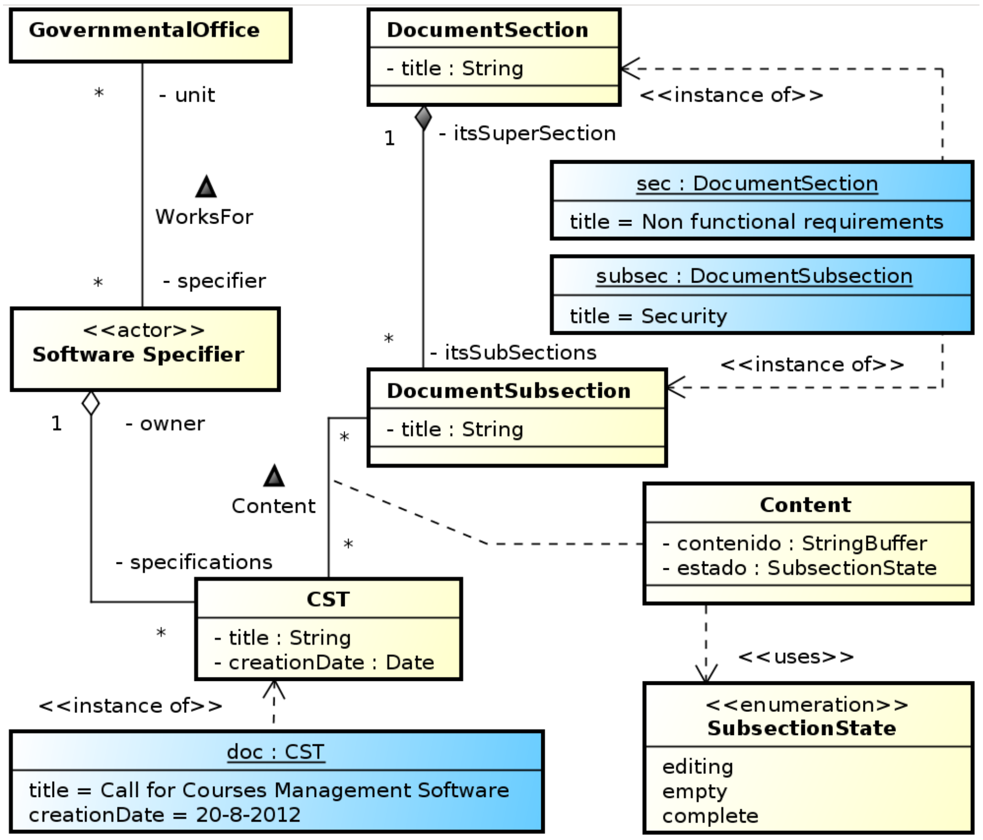


Figure 3. Conceptual model for T2R2.

We also have modeled that an object from *Software Specifier* could have many specifications or CSTs. Therefore we have an association from *Software Specifier* to CST. However, an object from CST must have just only one creator or owner. At once, an object from CST has a set of sections and subsections. In our model, it is not necessary to specify some aggregation between CST and *DocumentSection* because it is assumed that each CST has all active document sections which conforms the collection represented by the class *DocumentSection*. However, the association to *DocumentSubsection* is justified because we need to know the specific text content and its corresponding current state (represented by the enumeration *SubsectionState*).

In order to make a clear model we have added some objects from more relevant classes (blue boxes). Thus a specific CST may be for acquiring a Courses Management Software—in Chile, municipalities control most of public schools. This document has a section namely "non-functional requirements" (*sec* object) and into this, there is a subsection dedicated to security (*subsec* object). For simplicity of both, implementation and document structure, we have made the decision of keeping these classes as two different classes and not as the same class having some recursive association.

4.3. Functional Architecture

We have used UML use cases for specifying the functionality components. Due to the common misuse of specifying use cases as actors' actions or intentions, it is worth to clear that according to UML [23] use cases should represent system behaviors and neither actors' tasks, goals nor intentions. In **Figure 4**, we illustrate the main functions of T2R2.

Firstly the main system' actor is the *Software Specifier*, but not the only one, we also have a separated diagram for system administrator who should be in charge of users management. Due to the system was conceived as part of purchasing state platform then the *Software Specifier* is someone that has access to publish calls for purchasing. We have called him/her state purchaser.

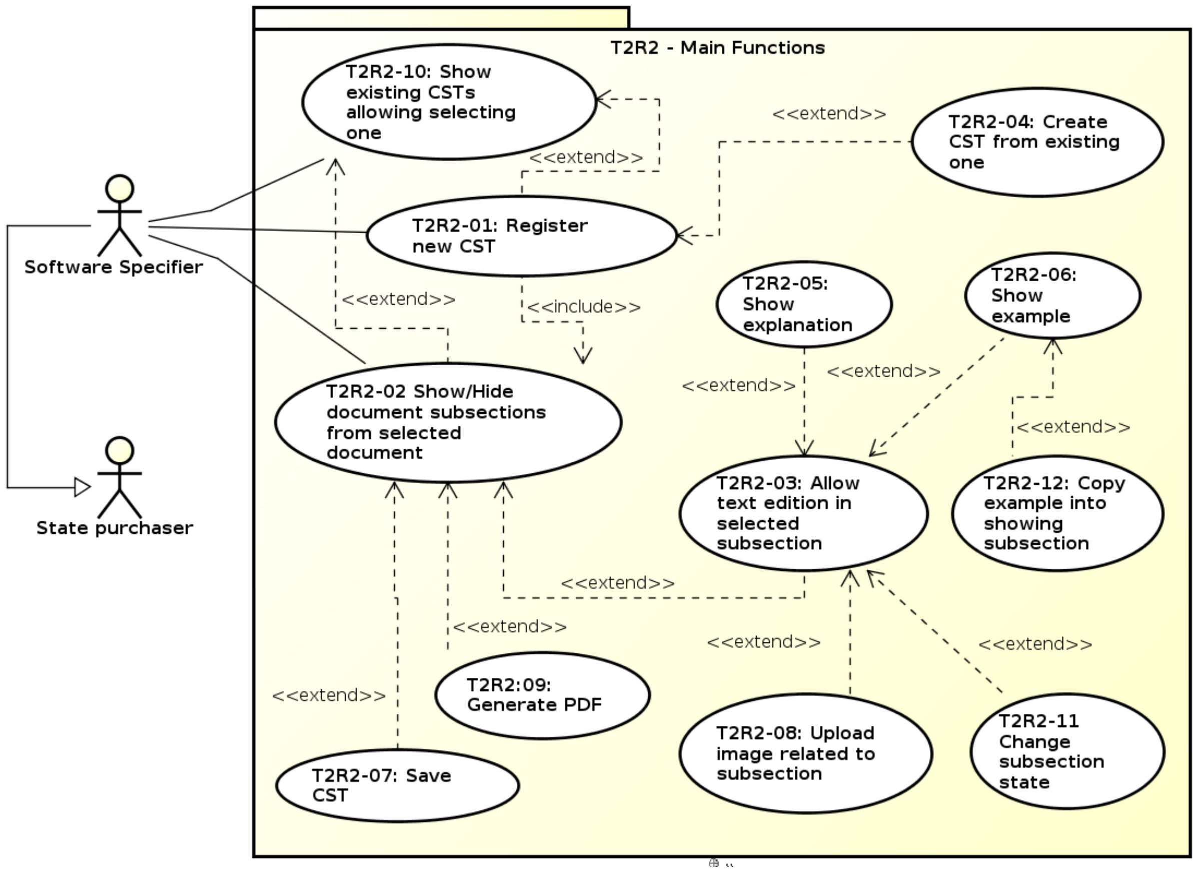

T2R2 - Main Functions
T2R2-10: Show existing CSTs allowing selecting one
<<extend>>
T2R2-04: Create CST from existing one
<<extend>>
Software Specifier
T2R2-01: Register new CST
T2R2-05: Show explanation
T2R2-06: Show example
<<extend>>
<<include>>
<<extend>>
<<extend>>
T2R2-02 Show/Hide document subsections from selected document
<<extend>>
T2R2-03: Allow text edition in selected subsection
T2R2-12: Copy example into showing subsection
State purchaser
<<extend>>
<<extend>>
<<extend>>
<<extend>>
<<extend>>
T2R2:09: Generate PDF
T2R2-08: Upload image related to subsection
T2R2-11 Change subsection state
T2R2-07: Save CST

Figure 4. Functional components of T2R2-Software.

In order to track the project we have enumerated the use cases. The main interactions are offered for viewing existing CSTs (T2R2-10), or creating a new one (T2R2-01). However, we have added a third main function which deals with offering to work on the last document under edition (T2R2-02). These three main use cases have an association to the *Software Specifier*. We have also included the choice of creating a new CST starting from an existing one (T2R2-04). In order to get some help we provide two functionalities, an abstract explanation of a specific subsection (T2R2-05) and two examples of right paragraphs related to specific domains (T2R2-06). For these examples a relevant and useful function is putting the example as part of the editing document (T2R2-12). As part of the documentation we designed the choice of uploading images (T2R2-08), however, it is a feature that we have let down in order to avoid the common practice of uploading the entire call for tender as an image or set of images—due to documents are first printed and then scanned. Finally we have also represented two simple but fundamental explicit actions such as saving the CST (T2R2-07) and producing the PDF (T2R2-09) which should be later upload into the state purchasing platform.

4.4. Log Model

As it is a problem over which we are still under study, a secondary goal is to gather additional information about the phenomena of formulating a CST. We are very interested on study de behavior of software specifiers. Therefore, a log system seems crucial. Hence we have designed a product that store dates, times and changes on documents. In **Figure 5**, we show the corresponding data model. Here we have an entity for storing each http sessions (LOG_Session). In order to store the changes on CSTs we store individual changes (LOG_CSTChange) and, in a separate table, the text changes if any (LOG_TXTChange). This last split has been planned for accelerating search analysis over both, by one side dates, times, delays (CSTChange) and on the other side content changes (LOG_TXTChange).

Finally, in regard to testing effort, we have made a white box exhaustive testing, which most of the time implied refactoring tasks in order to keep separate user interface and business concerns. Unit testing did not really mean a problem. However, at the level of integrated testing, we needed to add some additional functionality in order to allow that two or more users were working using the same user account (which was detected as a common practice in governmental offices). Stress testing has not been considered, under a normal year it is possible to reach 400 CSTs, if we consider three months of low activity (September by independence holidays, December by Christmas and February by summer vacations) we have a maximum of 45 CSTs by month, which do not really stress a contemporary web-based system.

5. Conclusions and Further Work

In this paper, we have presented T2R2-Software, a tool for aiding software purchasers in the task of generating a call for software tenders. We have presented the problem as a software engineering problem because software life cycle includes from initial conception to software relegation. Therefore, making a call for tender is part of the cycle. We have also argued that there are differences between theoretical and practical issues about producing calls for software tenders. We have considered the quality of state calls for software tenders in Chile. According to this, we have provided a specific solution for rural and semi-urban municipalities because their calls comparatively present lower quality.

As part of the tool presentation, we have showed main HCI and architectural considerations which have included an initial users' perception of the tool and the design of the log system to gather additional information for trying to get an improved understanding about the variables behind the creation of calls for software tenders.

As future work, we hope incorporating features such as: 1) ability to import/export technical documents from XML format, 2) ability to share technical documents in XML format between different users, and 3) ability to easily search complete documents or existing paragraphs. The features 2) and 3) mean to have a collective impulse in order to share not only phrases but also, what they have implied in the particular software projects which can be imported by the editing call.

In the scientific part, we want to empirically validate T2R2-Software as a useful tool. It means to approach a good measure of quality of the published calls. Moreover, we are interested in the results referred to the software quality of improved calls. We have already progressed on this quality model and also on other tool which allows getting simple metrics from CST documents.

From the social point of view, we expect benefits for state units from rural and semi-urban municipalities

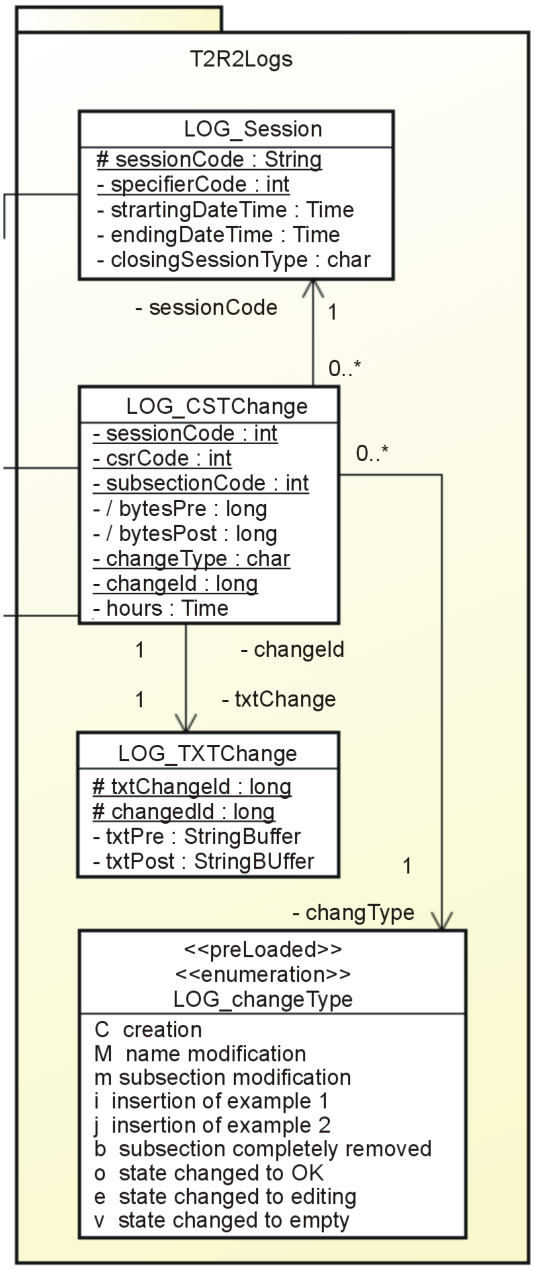


Figure 5. Log model of T2R2.

which may imply that they have better software products to really manage their resources and accomplish their social goals.

Acknowledgements

This work has been sponsored by the Investigation Direction, University of La Frontera, Temuco, Chile, DIUFRO DI13-0068 Project.

References

[1] McIvor, R. (2010) Global Services Outsourcing. Cambrigde University Press, Cambrigde.

[2] Anda, B.C.D., Sjoberg, D.I.K. and Mockus, A. (2009) Variability and Reproducibility in Software Engineering: A

Study of Four Companies that Developed the Same System. *IEEE Transactions on Software Engineering*, **35**, 407-429. http://dx.doi.org/10.1109/TSE.2008.89

[3] Reyes, M., Llopis, J. and Garcó, J. (2006) El offshore outsourcing de sistemas de información. *Universia business review-actualidad económica*, **38**, 80-91.

[4] Assar, S. and Boughzala, I. (2008) Empirical Evaluation of Public E-Procurement Platforms in France. *International Journal of Value Chain Management*, **2**, 90-108. http://dx.doi.org/10.1504/IJVCM.2008.016120

[5] Hussmann, K.A. (2004) La Promesa de las compras públicas Electrónicas: El caso de Chilecompra (2000-2003). Unpublished Masters, Universidad de Chile, Santiago.

[6] Hochstetter, J., Cachero, C., Cares, C. and Sepúlveda, S. (2012) Call for Tenders Challenges in Practice: A Field Study. *Proceedings of the XV Congreso Iberoamericano en "Software Engineering"* (*CIbSE*'12), Buenos Aires, 24-27 April 2012.

[7] Hochstetter, J., Díaz, C. and Cares, C. (2012) Software Call for Tenders: Metrics Based on Speech Acts. *Proceedings of the Conferencia Ibèrica de sistemas y Tecnologías de Infomación* (*CISTI*'12), Madrid, 20 al 23 de Junio de 2012, 451-456.

[8] Hochstetter, J. and Cares, C. (2012) Call for Software Tenders: Features and Research Problems. *Proceedings of the* 7*th International Conference on Software Engineering Advances* (*ICSEA*'12), Lisboa, 18-23 November 2012.

[9] IEEE (1998) IEEE Std 830-1998 IEEE Standard for Software Requirements Specification. IEEE Computer Society, Piscataway.

[10] Carvallo, X. and Franch, J.P. (2009) On the Use of Requirements for Driving Call-for-Tender Processes for Procuring Coarse-Grained OTS Components. 17*th IEEE International*, *Proceedings of the Requirements Engineering Conference*, Atlanta, 31 August 2009, 287-292.

[11] Biffl, S., Winkler, D., Höhn, R. and Wetzel, H. (2006) Software Process Improvement in Europe: Potential of the New V-Modell XT and Research Issues. *Software Process Improvement Practice*, **11**, 229-238. http://dx.doi.org/10.1002/spip.266

[12] Team, C. (2010) Cmmi for Acquisition, Version 1.3. http://repository.cmu.edu/sei/288/

[13] Brender, J. and McNair, P. (2001) User Requirements Specifications: A Hierarchical Structure Covering Strategical, Tactical and Operational Requirements. *International Journal of Medical Informatics*, **64**, 83-98. http://dx.doi.org/10.1016/S1386-5056(01)00190-3

[14] Costa, C.B.E., Corrêa, E., Corted, J.-M.D. and Vansnickd, J.-C. (2002) Facilitating Bid Evaluation in Public Call for Tenders: A Socio-Technical Approach. *Omega*, **30**, 227-242. http://dx.doi.org/10.1016/S0305-0483(02)00029-4

[15] Renault, S., Ménez-Bonilla, Ó., Franch, X. and Quer, C. (2009) A Pattern-Based Method for Building Requirements Documents in Call-for-Tender Processes. *International Journal of Computer Science and Applications*, **6**, 175-202.

[16] Hartson, R. (2003) Cognitive, Physical, Sensory, and Functional Affordances in Interaction Design. *Interaction Design*, **22**, 315-338.

[17] Nah, F.F.-H. and Davis, S. (2002) HCI Research Issues in E-Commerce. *Forrester Research*, **3**, 98-113.

[18] Grossman, T., Fitzmaurice, G. and Attar, R. (2009) A Survey of Software Learnability: Metrics, Methodologies and Guide-Lines. *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*: *ACM*, Boston, 4-9 April 2009, 649-658.

[19] (2014) Tendenciasdigitales. Conociendo el mercado de internet latinoamericano: Chile, Venezuela y Uruguay se ubican en el top 3 del Índice Web 2.0. http://tendenciasdigitales.com/1330/chile-venezuela-y-uruguay-se-ubican-en-el-top-3-del-indice-web-2-0/

[20] (2014) Tendenciasdigitales. Conociendo el mercado de internet latinoamericano: Tendencias digitales. Penetración de Internet en Latinoamérica se ubica en 37%. http://tendenciasdigitales.com/1386/penetracion-de-internet-en-latinoamerica-se-ubica-en-37

[21] Lindgaard, G. and Dudek, C. (2003) What Is This Evasive Beast We Call User Satisfaction? Oxford University Press, Oxford, 429-452.

[22] Al-Maskari, A. and Sanderson, M. (2010) A Review of Factors Influencing User Satisfaction in Information Retrieval. *Wiley Online Library*, **61**, 859-868.

[23] OMG (2014) OMG Unified Modeling Language TM (OMG UML), Superstructure. http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/