Towards a framework for improving requirement traceability

Hacia el desarrollo de un marco de trabajo para mejorar la trazabilidad de requisitos

Marco Antonio **Toranzo** Céspedes [1], Gilberto **Cysneiros** Filho[2], Yessica **Gómez**[3], Oscar **Rodríguez** Mendoza[4]

RESUMEN

Muchos trabajos de trazabilidad de requisitos están focalizados en aspectos de programación en vez de la identificación, análisis y modelamiento de todas las informaciones trazables de un proyecto de *software*. Este artículo trata del desarrollo de un marco de trabajo para mejorar la trazabilidad de requisitos de *software*. El marco de trabajo consiste en la clasificación de las informaciones trazables; la definición y uso de tipos de relaciones entre las informaciones trazables; un conjunto de directrices para elaborar un modelo de trazabilidad de requisitos en un proyecto de *software* y el desarrollo de la herramienta MyMT (My Management Tool) para apoyar el desarrollo de un modelo de trazabilidad de requisitos. Un sistema universitario de administración de biblioteca es empleado para ilustrar la aplicación del marco de trabajo.

Palabras clave: trazabilidad, calidad de *software*, modelo de referencia, marco de trabajo.

ABSTRACT

Work regarding requirement traceability focuses on programming aspects instead of identifying, analysing and modelling all traceable data in a software project. This paper describes the development of a framework for improving software requirement traceability. The framework consisted of classifying traced information, defining and using relationship types regarding traced information, a set of guidelines for developing a requirements traceability model for a software project and using my management tool (MyMT tool) to support developing a requirement traceability model. A university library management system was used to illustrate applying the framework.

Keywords: Traceability, software quality, reference model, framework.

Received: June 2nd 2011
Accepted: February 27th 2012

Introduction

Requirements traceability refers to, "the ability to describe and follow the life of a requirement, in both forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)" (Gotel, 1994). According to Cleland-Huang (Cleland-Huang, 2003), traceability involves several challenges, such as training and certification, support regarding the evolution of relationships between artefacts, the semantics of traceability relationships and traceability throughout an organisation. Moreover, Gotel (2008, 2009) has identified other traceability problems and challenges, such as changing the perception that traceability is a tedious and repetitive activity, distributing responsibility regarding traceability amongst several people, planning a strategy to ensure traceability, continuity and maintenance, determining traceability reliability levels, traceability by product and identifying traceability stakeholders and their respective needs. More detailed information on these and other traceability challenges can be found in Cysneiros (Cysneiros, 2011).

This work concerns reference models and meta-models, traceability relationship representation, visualising traceability relationships and requirement management. This paper describes automating traceability relationship types and applying a set of guidelines for developing a traceability model using MyMTtool, the successor of Labrador tool (Toranzo and Mello, 2002). A traceability model identifies all related artefacts in software development concerned with requirements. This study was based on experience of applying software traceability improvement matrix in a financial company (Villarroel, 2009), Toranzo's work (Toranzo, 2002) being adapted by Castro (Castro, 2003), Castor (Castor, 2004) and Pinto (Pinto, 2005; Pinto, 2007) for tracing agent-orientated systems and reviewing the literature concerned with requirement traceability (Cysneiros, 2011).

[1] Mathematics Pedagogue, Universidad Arturo Prat, Chile. MSc in Computer Science, Universidade Federal de Pernambuco, Brazil. PhD in Computer Science, Universidade Federal de Pernambuco, Brazil. Departamento de Computación e Informática, Universidad Católica del Maule, Chile. E-mail: mtoranzo@ucm.cl

[2] BSc in Computer Science, Universidade Federal Rural de Pernambuco, Brazil. MSc in Computer Science, Universidade Federal de Pernambuco, Brazil. PhD in Computing, University of City University, London. Statistics and Informatics Department, Universidade Federal Rural de Pernambuco, Brazil. E-mail: g.cysneiros@gmail.com

[3] BSc. in Computer Engineering, Universidad de Santiago, Chile. MSc in Information Technology, Universidad Técnica Federico Santa María. Departamento de Computación e Informática, Universidad Católica del Maule, Chile. E-mail: jgomez@ucm.cl

[4] BSc. in Civil Engineer Informatic, Universidad Católica del Maule, Chile. Computer Science Student, Universidad Católica del Maule, Chile. Facultad de Ingeniería, Universidad Autónoma de Chile. E-mail: orodriguezm@uautonoma.cl

The framework described in this paper consisted of:

1. Classifying information which may form part of a traceability model;
2. A set of relationship types regarding a traceability model (this paper proposes a set of predefined relationships for a traceability model);
3. A set of guidelines for developing a traceability model; and

Developing the MyMT tool to support creating a traceability model and requirement management.

Classifying information

Most research on traceability does not formulate systematic questions and processes for identifying artefacts, information and stakeholders (interested in traceability) forming part of a traceability model. This paper proposes an approach for addressing how to classify the traced information to help answer questions such as, Who are the stakeholders interested in traceability information? Which artefacts will be traced? How are these artefacts related to? An organisation having a tool to manage requirements does not mean that it can identify and manage them. Figure 1 shows four levels of classification: external, organisational, management and development. Information levels are not necessarily disjointed. Classification is based on which organisations are inserted in a changing political and economic environment which can affect their information systems.

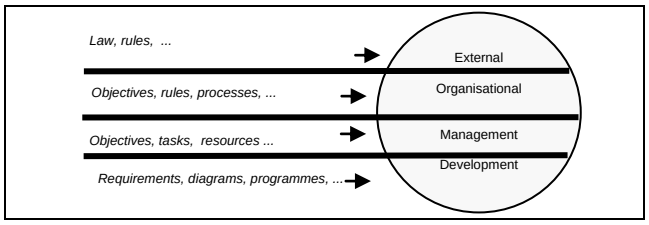

Figure 1. Classifying traceable information

The external level includes all information related to an organisation's external political and economic context which can affect an organisation's information systems (for example, tax laws affecting pay systems and forcing organisations to pay such tax).

The organisational level represents concepts, such as objectives, strategies and organisational goals, and organisational processes consisting of several activities conducted by different departments within an organisation. The combination of items at this level contributes towards software acquisition and/or development. It is important for system analysts to know where a process begins and ends, where an organisation's needs begin and end to develop software meeting an organisation's business needs, and not otherwise.

In terms of management, with the exception of Ramesh (Ramesh, 2001), most research has not addressed how traceability can be integrated with project management software. Poor project management is a major problem in software development (Humphrey, 2010). Software project managers should thus consider that software projects plans are based on satisfaction, changing requirements (Wiegers, 2003) and paying attention to requirement traceability being done properly because it forms part of the development and maintenance involved in producing a quality product. The current research was paper particularly interested in establishing the relationship between tasks, project management and project requirements to help managers improve control and monitoring requirements during different software development stages.

The level of development represents the artefacts produced by a development team using software development methodology. A strategy is needed for manual, automatic or semi-automatic traceability (i.e. How are requirements identified within a text? How are relationships between requirements identified? How are requirements related to artefacts?)

Some benefits of such four levels would be:

1. Presenting an alternative way to identify, organise and analyse the information which may form part of a traceability model;
2. Each level has stakeholders who might be interested in traceability to verify that their requirements and restrictions are satisfied by particular software; and
3. Understanding traceability in developing and maintaining software can contribute towards improving practice. This is a form of awareness that traceability involves a problem-solving team and not just a person.

Types of relationship

All relationships were organised into a proposed meta-model (Toranzo, 2002). All relationships are described below:

- Resource (<REC>) states that source class has physical dependence or information with instances of the target class;
- Satisfaction (<Sat>) specifies that source class instances depend on something's satisfaction (or compliance) regarding target class instances. The term is used to express satisfaction that something must be done to achieve something (for example, meeting organisational goals partly depends on requirements);
- Responsibility (<Resp>) specifies that source class instances depend on responsibility for target class instances. The term responsibility is used to express that a person is responsible for an artefact/element within a project;
- Representation (<Rep>) reflects that a requirement is expressed (transformed) into another notation regarding another device, such as a requirement (a text) being expressed in graphic notation (for example UML) in a diagram;
- Localization (<loc>) states that a requirement has been assigned to a subsystem; and
- Explanations for aggregation and generalisation relationships are analogous to the definitions in the literature.

This set of relationships was used to associate different information as being traceable within a particular software project.

Guidelines for developing a traceability model

This section describes and illustrates a set of guidelines which were used for developing a requirement traceability model using the MyMT tool. A library system for managing and integrating bibliographic data from a university's different campus was used to present the guidelines. Figure 2 shows the creation of the project for the library system and Figure 3 shows a screen for creating a specific task (i.e. analysis) in a project.

- **Guideline 1:** Identifying stakeholders interested in traceability to ascertain how a system satisfies organisational activities;
- **Guideline 2:** Identifying information outside an organisation which might affect the system. Some questions to be asked would be: Which requirements depend on external organisations? Which requirements depending on senior management can affect the system? Applying this guideline to a library system led to creating the *InformationFormat* class to reference information for formatting and cataloguing books, called machine-readable cataloguing and Anglo-American cataloguing rules;
- **Guideline 3:** Identifying the objectives, strategies and business rules to be traced. Some questions might be: Which organisational objectives must a system satisfy? Which business rules are implemented in a system? Where were the business rules documented / modelled? Which requirements are related to the business rules? Which business rules affect the requirements? After applying the guideline, the *OrganisationalObjective* class was created in the traceability model for documenting a system's organisational objectives;

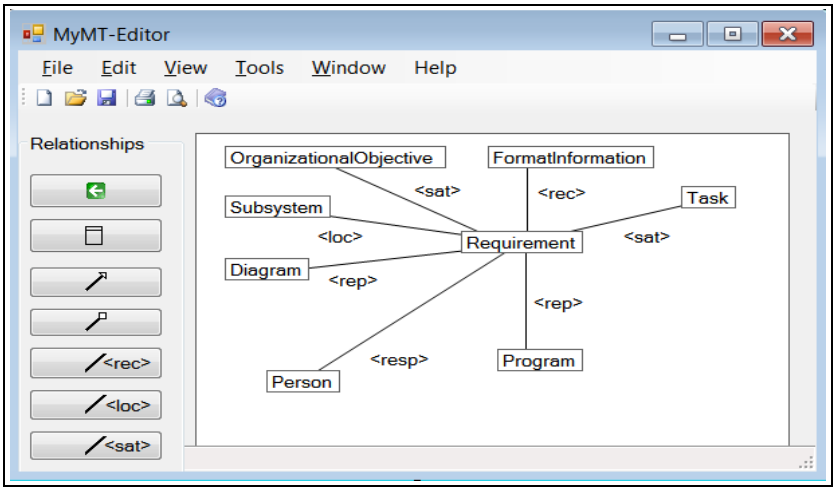

Figure 5: Traceability model

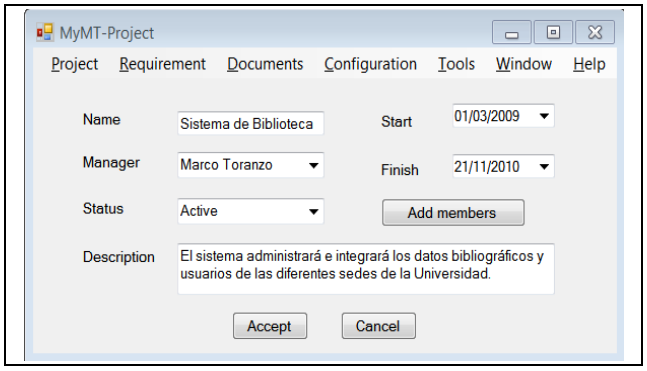

Figure 2: Creating a project

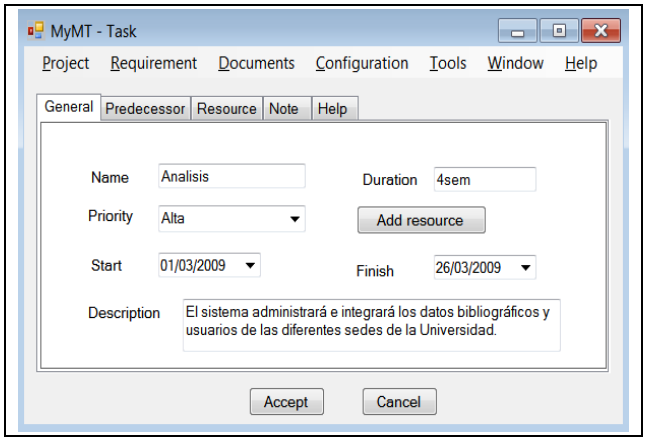

Figure 3: Creating a task

- **Guideline 4:** Including project management information in the traceability model. The guideline recommends including the *Task* class in the traceability model;
- **Guideline 5:** Identifying subsystems. Implementing the guideline created a *subsystem* class to represent, for example, user management and loan administration subsystems;
- **Guideline 6:** Including the *Requirement* class. Creating the *Requirement* class is mandatory and obvious;
- **Guideline 7:** Identifying diagrams used to model requirements. This guideline identified diagrams used to model requirements and define logical paths. A logical path is the sequence of folder names and file name (at the end) where a diagram is stored;
- **Guideline 8:** Identifying programmes. This guideline identified programmes implementing the requirements and logically defining how to recover them. The explanation of the logical path is similar to that indicated in Guideline 7. The *Programme* class was created In MyMT; it was related to the *Requirement* class;
- **Guideline 9:** Identifying documents. Project documents must be identified. Figure 4 shows the requirements specification template available in MyMT;

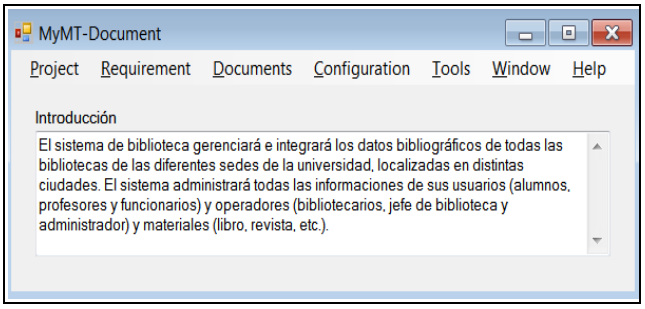

Figure 4: Requirement specification document

- **Guideline 10:** Excluding irrelevant classes;
- **Guideline 11:** Integrating classes having the same meaning; and
- **Guideline 12:** Integrating new classes.

Guidelines 1 to 12 were used in developing and identifying candidate classes in the traceability model.

- **Guideline 13:** Organising classes. The classes forming part of a traceability model must be organised and structured; and

- **Guideline 14:** Establishing relationships between classes. Candidate classes should be related. Several relationships were identified, for example *Requirement* into classes: Programme, Person and Subsystem. The notations for the types of relationships are shown on the left-hand side of Figure 5, while the work area shows the relationships between classes.

MyMT implements the types of relationships: aggregation, generalisation, application, location, satisfaction, performance and accountability. They were defined in section 3.

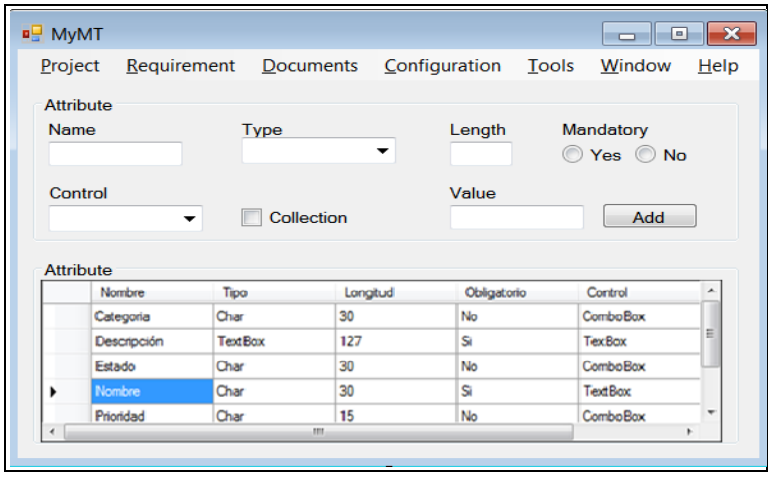

Figure 6: Defining requirement class attributes

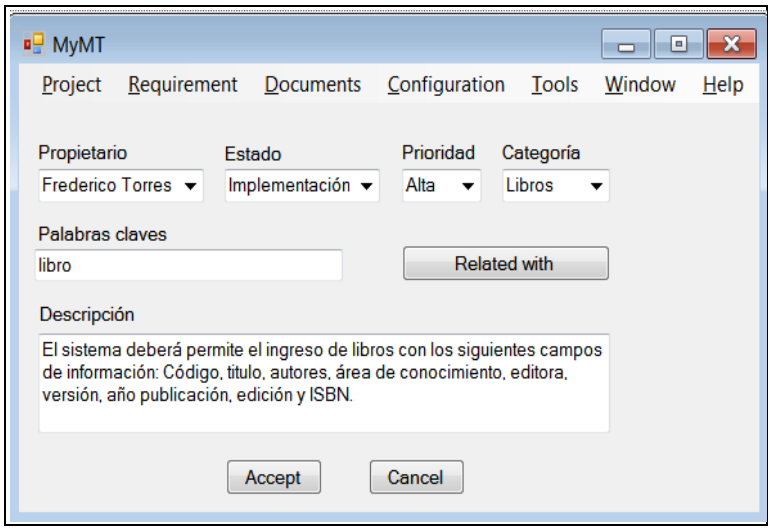

Figure 7: Creating a requirement

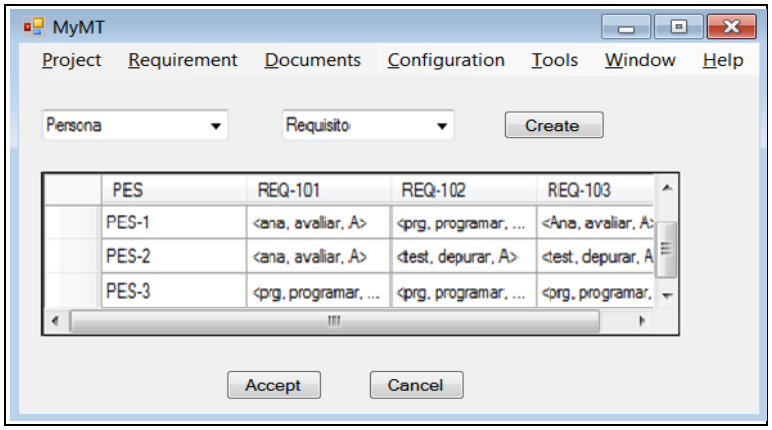

Figure 8: Representation

- **Guideline 15:** Defining class attributes. Traceability model classes' attributes must be defined. Figure 6 defines Requirement class attributes.

The definition should state the field name, data type, field length, whether it is mandatory or optional, the associated visual controller (textbox, combobox, for example) and whether it is a collection of other predefined elements. Figure 7 presents the data entry for requirements.

- **Guideline 16**: Defining a matrix for each of the model's relationships. The objective was to develop several matrices for each relationship in the traceability model. Figure 8 gives an example of representing the responsibility relationship (between *Person* and *Requirement* classes) in a matrix presenting three arguments: the person's role, the activity to be performed and the degree of responsibility for a particular activity.

A row's intersection with a column represented the relationship between instances of class origin and destination, for example Figure 8 indicates that the intersection between requirement "REQ-102" and Person "PES-2" was represented by " <test, refine, A> ", this meant that the person tested PES-2 (first component "test ") to refine (second component "refine") implementing requirement REQ-102, and was highly responsible (third component "A") in this task.

Conclusions

In Cysneiros' review (Cysneiros, 2011) the researchers assumed that traceable artifacts were identified and did not provide guidelines and activities for developing a traceability model. The work presented in this paper has been more focused on analysing a traceability methodology. Traceability cannot be applied without identifying the pertinent stakeholders and their respective needs. Some lessons learned from the study were:

- Levels of information led to identifying potential stakeholders involved in traceability;

- A traceability model should be agreed amongst team members and also represent an opportunity to distribute project tracking according to each member's role;

- Manual traceability is an arduous and time-consuming task; and

- The MyMT tool was used to support the case study library system, represent traceability relationships (Figures 5 and 7) and visualise traceability relationships (Figure 8).

Future work should include implementing automatic mechanisms such as text identifying requirements and establishing relationships between the requirements expressed in text. MyMT continuous development and

implementation in visual basic, NET and MySQL 5.051b. The SimplyRequirement system is currently being developed for managing requirements.

References

Castro, J., Pinto, R., Castor, A., J Mylopoulos., Requirements traceability in agent oriented software engineering, lecture notes in computer science, LNCS 2603, V. 2603, 2003, Springer-Verlag. pp. 57-72.

Castor, A., Rastreamento de requisitos no proceso de desenvolvimento de software orientado a agentes, MSc thesis, Universidade Federal de Pernambuco, Recife, 2004.

Cleland-Huang, J., Chang, C., Christensen, M., Event-based traceability for managing evolutionary change, IEEE Transactions on Software Engineering, 2003, N- 9 : Vol. 29, pp. 796-810.

Cysneiros, G., Software Traceability for multi-agent system implemented using BDI architecture, DPhil thesis, City University London, London, 2011.

Cysneiros, G., Zisman, A., Traceability and completeness checking for agent-oriented systems, 23rd Annual ACM Symposium on Applied Computing - Technical Track on Agent-Oriented Programming, Systems, Languages, and Applications, 2008.

Gotel, O., Finkelstein A., An analysis of the requirements traceability problem. International Conference on Requirements Engineering. - Colorado, USA, IEEE Computer Society, 1994, pp. 94-101.

Gotel, O., Traceability – Problems in a Word. The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society, RQ49, 2008.

Gotel, O., Traceability – Putting the 'y' First. Requirements Quarterly: The Newsletter of the Requirements Engineering, Specialist Group of the British Computer Society, RQ50, 2009.

Humphrey, W., Thomas, W., Reflections on management: how to. manage your software projects, your teams, yours boss, and yourself. Addison Wesley, 2010

Pinto, R., Castro, J., Toranzo, M., Requirements Traceability, in International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, 2002, Orlando. http://www.teccomm.les.inf.puc-rio.br/selmas2002/. Last consulted: 15/10/2010.

Pinto, R., Castro, J., Tedesco, P., Silva M., Alencar F., A Traceability Reference Model for Agent Oriented Development. Proceedings of the Third Workshop on Software Engineering for Agent-oriented Systems. João Pessoa. pp. 27-38. Paraíba/UFPB, 2007.

Pinto, R., Silva, C., Lima, T., Castro, J., Support for requirement traceability: The Tropos case, XIX Simpósio Brasileiro de Engenharia de Software - SBES'05, 2005, Uberlândia, Anais do XIX Simpósio Brasileiro de Engenharia de Software - SBES'05. 2005. pp. 40-55.

Ramesh, B., Jarke, M., Towards reference models for requirements traceability, IEEE Transactions on Software Engenieering, Vol. 27, Jan, 2001. pp. 58-93.

Toranzo, M., A Framework para melhorar o rastreamento de requisitos. PhD thesis, Universidade Federal de Pernambuco, Brasil, 2002.

Toranzo, M., Mellon, E., Uma proposta para melhorar o rastreamento de requisitos. Workshop de Engenharia de Requisitos.. Valencia, Spain. 2002. pp. 194 -209

Toranzo, M., Mejorando la trazabilidad de requisitos. 8th Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software 2005.. Valparaíso, Chile, 2005. pp. 28-42

Villarroel, R., Gómez, Y., Gajardo, R., Rodriguez, O. Implementation of an improvement cycle using the Competisoft methodological framework and the Tutelkan platform. In Proceedings of SCCC'2009. pp.97-104.

Wiegers, K., Software Requirements, Microsoft Press, 2nd edition, 2003