Cognitive Engineering meets Requirements Engineering

Bridging the Traceability Gap

Alexandra Mazak
Junior Research Studio Cognitive Engineering (CoE)
Research Studios Austria Forschungsgesellschaft
Vienna, Austria
alexandra.mazak@researchstudios.at

Horst Kargl
SparxSystems Software GmbH
Vienna, Austria
horst.kargl@sparxsystems.eu

***Abstract*—Support for various stakeholders (customer, project manager, system architect, requirements engineer) involved in the design and management of large software systems is needed since frequently, misinterpretations occur already when specifying customer requirements into system requirements. This problem is mainly caused by the various perspectives and intentions of the involved parties that may lead to diverging interpretations during said process. Therefore, the focus of our work in progress is on the requirements engineers when transforming customer requirements into system requirements. There is still a gap to trace design decisions especially at this early stage of the system development life cycle. We introduce a heuristic-based approach in order to make a contribution to bridge this gap. We propose to consider the requirements engineer's "cognitive perspective" on traceability links by a heuristic-based weighting procedure that can be performed during the design process. We enhance the established relationship or traceability matrix to make it possible for requirements engineers to annotate their informal knowledge to the linkage (i.e., visualized realizations) in that matrix.**

Keywords-Requirements management; requirements traceability; cognitive engineering; traceability matrix; design decisions.

I. INTRODUCTION

There is a variety of stakeholders involved in large software projects, each having a different set of goals and priorities [9]. Generally, requirements are prioritized by stakeholders (e.g., based on the software projects' purpose, certain functionalities which should be targeted by the system). Various methods and supporting tools exist for guiding this process of requirements' prioritization. What we are missing is another prioritization when linking customer requirements to system requirements and system requirements to design artifacts (i.e., model components) in the specification task. Misinterpretations at this stage of a software project are resulting from misconceived or misvalued linking. Validation continues to be a "big pain" due to the lack of trusted documentation of traceability drives validation teams to leave no doubt (aka double or triple effort). There is a multitude of tools (e.g., Rational DOORS [12]) by which requirement lifecycle management is supported. Often, tools produce non- or less compelling documentation [13]. However, experience has shown that a "distributed traceability of requirements" guided by external tools or methods, which means that it is not integrated in the system's architecture design process itself, is often error-prone [13].

We propose focusing on the requirements engineers' informal knowledge when design decisions are made during the specification task of customer requirements. According to the used modeling language a certain traceability link can be used (*realization* in UML [2], *satisfy* in SysML [3]). Generally, a trace chain is constructed through linking by which an impact analysis can be made. Mainly, such information is presented in form of a matrix called *relationship matrix*. In Fig. 1, which represents an excerpt of a relationship matrix from the tool Enterprise Architect (EA) [13], the linkage is visualized in form of arrows in the cells. This type of linking merely express for instance that *Receive Orders* is realized by two system artifacts *REQ019* and *REQ032* (e.g., Fig. 1). It cannot be derived to what extent the realization is proceeded, or the system requirements' level of utility to fulfill this customer requirement. Generally, a certain customer requirement is covered by more than a single system component and not each component has the same relevance to realize that requirement within the system's architecture; just as customer requirements do not have equal prioritizations. Thus, each system component has a different level of utility (relevance).

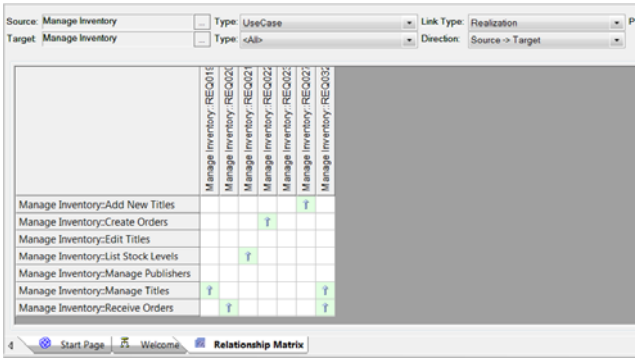

Figure 1. Relationship matrix in EA.

 ISBN: 978-1-61208-230-1

In the early past, methods have emerged by which informal knowledge of the original engineers can be annotated to the model itself, as for example presented in the field of *Ontology Alignment* in [7][8]. There the modelers' intention involved during the design process, that is *modeler's cognition* or *cognitive perspective*, is made visible to users.

Currently, there is no tool support to requirements engineers in order to aid them to make their informal knowledge visible. This open issue motivated us to work on an heuristic-based approach to bridge the traceability gap at the early stage of system design that is of commercial interest, too.

II. BACKGROUND

A. Requirements Traceability

Requirements traceability in all stages of the system development life cycle is an important field of requirements management. Gotel and Finkelstein [4] define requirements traceability as *"the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases"*. The authors differentiate between *pre-requirements specification (pre-RS) traceability* and *post-requirements (post-RS) specification traceability*. The first refers to those aspects of a requirement's life prior to inclusion in the specification task (e.g., when stakeholders prioritize requirements depending on their expectations they place on the system); whereas the latter refers to those aspects that result from inclusion in the requirement's specification [4].

In our work in progress, we focus on post-RS traceability by which system components and their relations to certain customer requirements are considered. Today, requirements traceability is a key factor in the project management of large-scale systems and it plays an important role in the quality control of software engineering processes [9]. It acts as an indicator to define the system's maturity of development [8]. The aim is to support reliable, up to date, and high-quality traceability—right from the start.

B. Requirements Traceability Matrix

Validation pain can be reduced when using the *traceability matrix* [6]. This is a kind of "completeness indicator" by which the relationships among requirements and artifacts can be traced. It is an aid to determine the complexity of dependencies. There are tools by which a graphical as well as a textual traceability is supported to engineers. For example, in the tool Enterprise Architect the relationship matrix is a spreadsheet display of relationships between different sets of model elements (e.g., Fig. 1). A source package and a target package, the relationship type, and the direction can be defined. All relationships among source and target elements can be identified by highlighting a grid square and displaying an arrow indicating the relationship's direction. The matrix is a convenient method of visualizing relationships quickly and definitively. It also enables users to create, modify and delete relationships between elements with a single mouse click - another quick way to create complex sets of element relationships with a minimum of effort.

Nevertheless, the literature-based research has shown that the traceability matrix is a useful template to trace if a certain requirement is covered by a single or more components. We argue that a simple representation of linkage alone is not enough. For instance, it cannot be derived from the matrix how important a system component is for implementation (i.e., when transforming system components to design artifacts, or when coding) an aspect which is also relevant in agile software development. For instance, in Scrum [5] in order to provide a reference value for the product owner when prioritizing stories for the product backlog.

III. THEORETICAL APPROACH

Requirements engineers decide about how to transform customer requirements into system requirements, i.e., they decide on the realization of customer requirements in the system architecture. Generally, this design step is visualized in the traceability matrix, where the realizations can be traced (e.g., Fig. 1, the customer requirements form the source and the system requirements or components the target nodes). The better this task is guided and monitored, the fewer problems (e.g., over-engineering) will occur during implementation which leads to time and cost savings.

Therefore, we propose to use the engineers' informal knowledge of system design specification by introducing a *cognitive heuristic* in form of a relevance weighting procedure. This procedure is based on an intuitive mental judgement made by the engineers during the design process. We adapt the approach from concepts, which we have introduced in [8]. In our current approach, the requirements engineer determines the importance of a linkage (source → target) by annotating the linkage with a weighting in the range of [1, 9]. The weighting is mainly based on his/her intuition (or cognitive perspective [8]) of the system component's relevance in order to cover a certain customer requirement.

Customer requirements		Prioritization (Stakeholder value)	Manage Inventory::REQ019 8%	Manage Inventory::REQ020 18%	Manage Inventory::REQ021 4%	Manage Inventory::REQ022 7%	Manage Inventory::REQ023 0%	Manage Inventory::REQ027 4%	Manage Inventory::REQ032 39%
Manage Inventory::	Add New Titles	6						2	
	Create Orders	3				7			
	Edit Titels	4							
	List Stock Levels	2			6				
	Manage Publishers	2							
	Manage Titles	6	4						8
	Receive Orders	8		7					9

Caption:	Weighting class	Relevance of linkage
	[7, 9]	high relevance
	[4, 6]	middle relevance
	[1, 3]	low relevance

Figure 2. Design Decision Traceability Grid (DDT-Grid).

 ISBN: 978-1-61208-230-1

In the tool-guided procedure (i.e., the engineer is prompted by the system), he/she affixes a relevance weighting on each realization in the matrix. This quantified mental judgement is made visible in an enhanced form of the traceability matrix, which we denote as *design decision traceability grid (DDT-grid)* (e.g., Fig. 2). We introduce this DDT-grid as a controlling instrument for engineers in the form of a weighted decision matrix, where the engineer can reflect on the made decisions and its alternatives at any time during the design process. The requirements engineer can distinguish among three weighting classes (i) high relevance in the range of [7, 9], (ii) middle relevance in the range of [4, 6], and (iii) low relevance in the range of [1, 3] (e.g., Fig. 2).

The cognitive heuristic can be classified as post-requirements specification traceability method. The system components level of utility or (functional) degree of fulfillment is automatically computed by an algorithm by which additionally the customer requirements' prioritizations, made by the stakeholders, are taken into account. The computation is based on (i) the number of covered customer requirements, (ii) their prioritization, and (iii) the relevance weightings of linkage. For each system component a relative utility value is expressed as a percentage, which constitutes the component's importance in the entire system architecture and thereby, a pairwise comparison between components is facilitated. On the one hand, also alternative modeling solutions of equal value can be identified, and on the other hand system components with a low level of importance can be easily detected in order to minimize a possible over-engineering risk even before the next step in the system's development process will be started (i.e. when transforming system requirements to design artifacts, or when coding).

The introduced cognitive-based approach to formalize the requirements engineers' informal knowledge by an importance weighting metric and turning this knowledge into operating figures provides a new dimension of requirements traceability controlling at the early stage of requirements management.

IV. EXPECTED RESULTS

Making informal design knowledge explicit (i.e., visible to users by the DDT-grid) would favorably impact the interpretation of the system's architecture. This means that the DDT-grid forms an innovative communication base for system architects and customers. For example, by identifying what are the critical, highly relevant system components in the model in order to assess the developers' efforts when implementing them. Additionally, stakeholders can track, at any time, how the scoring of system artifacts is affected when customers vary the prioritization, delete, or add requirements. Thus, the cognitive-based approach makes a contribution to the fulfillment of an efficient *requirements negotiation* as proposed in [1]. The DDT-grid and the calculated metrics provide a kind of cognitive map for engineers to rethink about their design decisions (e.g., to check if they are on the "right way"). Each map represents a personal view of prioritized solutions of the system. By comparing these maps a hint to a different system understanding can be provided. Additionally, a quick overview can be given through the highlighted visualization in the DDT-grid by which different weightings of realizations are made visible. By describing and discussing its own view, a common understanding between stakeholders can be established.

ACKNOWLEDGMENT

The introduced approach will be realized in the course of the *TraCo* project. TraCo (Traceability Controlling) is supported and promoted by the Austrian Research Promotion Agency (FFG) by the funding program BRIDGE. Launch of the 18-month project is October 01 2012. The cooperation partner in this FFG - project is SparxSystems Software GmbH based in Vienna. SparxSystems specializes in high-performance and scalable visual modeling tools for planning, design and construction of software intensive systems. TraCo is to be conceptualized as plug-in for the core product of the company, the Enterprise Architect, and it is to be implemented as a component of the business logic of EnArWEB (Enterprise Architect in the WEB), where the Enterprise Architect is used as repository.

REFERENCES

[1] L. Cao and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study," IEEE Software, vol. 25, IEEE Computer Society, January/February 2008, pp. 60–67, doi: 10.1109/MS.2008.1.

[2] M. Flower, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3th edition, Addison-Wesley Object Technology, 2003.

[3] S. Friedenthal, A. C. Moore, and R. Steiner, A Practical Guide to SysML: The Systems Modeling Language, The Morgan Kaufmann Omp Press, Elsevier Inc., 2012.

[4] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," In Proceedings of the 1th International Conference on Requirements Engineering, Colorado Springs (CO US), April 1994, pp. 94–101, doi: 10.1.1.137.5052.

[5] H. Kniberg, Scrum and XP from the Trenches, How we do Scrum, Enterprise Software Development Series, C4Media Inc, 2007.

[6] J. Macmillan and J. R. Vosburgh, "Software Quality Indicators," Final Report, Scientific Systems Inc Cambridge (MA US), September 1986, Accession Number: ADA181505.

[7] A. Mazak, M. Lanzenberger, and B. Schandl, "iweightings: Enhancing Structure-based Ontology Alignment by Enriching Models with Importance Weightings," In Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 10), Krakow (PL), February 2010, IEEE Press, pp. 992–997, doi=10.1109/CISIS.2010.164.

[8] A. Mazak, "CoMetO: A Cognitive Design Methodology for Enhancing the Alignment Potential of Ontologies," doctoral thesis, Information & Software Engineering Group (ifs), Department of Software Technology and Interactive Systems, Vienna University of Technologie, Vienna, April 2012.

[9] K. Pohl, Requirements Engineering: Grundlagen, Prinzipien, Techniken, 2. Auflage, dpunkt.verlag, 2008.

[10] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards, "Requirements traceability: Theory and practice," in Journal

 ISBN: 978-1-61208-230-1

Annals of Software Engineering, vol. 3, J. C. Baltzer AG, Science Publishers, January 1997, pp. 397–415.

[11] B. Ramesh and M. Jarke, “Toward Reference Models for Requirements Traceability,” in IEEE Transactions of Software Engineering, vol. 27, January 2001, pp. 58–93.

[12] IBM, Rational DOORS version 9.2, available at http://publib.boulder.ibm.com/infocenter/rsdp/vlr0m0/index.jsp?topic=/com.ibm.help.download.doors.doc/topics/doors_version9_2.html.

[13] D. Steinpichler and H. Kargl, “Enterprise Architect, project management with UML and EA,” Manual revised edition for Version 9.3, SparxSystems Software GmbH, Vienna, January 2011, available at http://www.sparxsystems.de/?gclid=CNap8rfwr7MCFYq7zAodxR4AXg.

 ISBN: 978-1-61208-230-1