Context-sensitive Traceability Controlling

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

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

***Abstract*—In the course of a research project funded by the Austrian Research Promotion Agency (FFG), we integrate cognitive engineering in the field of requirements management. In doing so, we go in the question of how utility and quality of design components could be operationalized in the context of modeling at the design phase in a software development project. We named this project *Traceability Controlling (TraCo)*. Our focus in TraCo is on the quality assurance of the modelers' design decisions made at the phase of problem-solving and solution specification. For the purpose of implementing TraCo we choose an interdisciplinary approach. We present an appropriate method to ensure the level of quality of the created design solution already during its creation. The TraCo-method can be used for continuously monitoring and controlling content quality issues like adequacy, appropriateness, and impact. By using the introduced method, requirements-engineers (e.g. system-architects, modelers) are able to validate whether the design model or fine-grained the model's design components exist in sufficient quality and whether these components meet the predetermined requirements' prioritization (i.e. their stakeholder values).**

Keywords—requirements management; traceability, cognitive engineering; quality control; decision theory.

I. INTRODUCTION

The domain of interest is the field of *requirements traceability*. 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 [13]. Firstly introduced in [6], 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 [6]. In our approach, we focus on post-RS traceability by which system or design components and their relations to certain customer requirements are considered. Our objective is to support reliable, up to date and high-quality traceability right from the start.

In the course of the opening session IKT 2012, organized by the FFG [17], the key note speaker Manfred Broy propagated that the emphasis should be placed on an early starting quality control in requirements engineering, since software-intensive systems become increasingly complex [18]. Also, other research colleagues are demanding an requirements' quality control from the very beginning [12][2][7][16]. They state that an early detection of design errors helps to prevent "developer gold plating" also known as "over-engineering" [12]. These design pitfalls lead to a more complex architecture/design model that is more prone to error, which negatively affect time and costs of development [16]. Customer requirements be "over anxious" and this causes additional costs. Often, customers are not willing to pay extra costs since the extra effort was not required.

Often design errors, made in the early stages of system design, are detected posteriori in subsequent project phases or in the first place in the operation of the system [12]. In these subsequent project phases, requirements which have been implemented incorrectly or incompletely are interpreted by developers as subjective coherent, since they rely on the requirement engineers' work. In the final stage, the system created doesn't meet the stakeholders' expectations. Frequently, misinterpretations among system architects/modelers and stakeholders already occur when transforming customer requirements into system requirements or design components resulting from misconceived or misvalued linking. This problem is mainly caused by various perspectives, priorities and intentions of the parties involved which may lead to diverging interpretations during said process [13]. Unfortunately, extensive quality analysis only found little acceptance in the face of high time and cost pressure just at the beginning of a project. Particularly in large software projects, it is very difficult to control software quality in addition to time and budget, since continuous and comprehensive feedback is often missing on the current status of the project [16].

In recent years, agile approaches (e.g. Scrum, Extreme Programming [8]) have been introduced to avoid expensive trouble shooting in the final phase of a project. Agile approaches are aiming for an early customer feedback, often based on early prototyping. Based on the feedback, requirements may be changed. However, working with agile processes does not mean developing without a plan.

To addressing these aforementioned problems, we introduce a heuristic-based approach for visualizing the engineers' design decisions in relation to certain (quality-based) contexts in order to make them transparent and traceable for stakeholders (e.g. project manager and customer). We named this approach *Traceability Controlling (TraCo)* and

introduce it at the operational level where system architects or modelers realize customer requirements in the design model. The TraCo-method can be used by modelers for continuously reviewing content quality issues such as *adequacy*, *appropriateness*, and *impact*. In TraCo, the end-user's perspective is foresighted involved at various points in the design process. For this purpose, we implement (i) a procedure adapted from the concept of *Cognitive Walkthrough*, an inspection technique of usability testing [23], (ii) the concept of the *Reflective Practitioner* introduced by Schön [20] and (iii) the *utility analysis*, a compositional method of the field of decision theory.

The core idea is to formalize *tacit (non-formal) knowledge* (i.e. the engineers' intuition when modeling) by explicitly pour this situational context-related knowledge into numbers and to store it as meta-information in the model. That meta-information facilitates the comparability among engineers and their individual views when modeling. In TraCo, design decisions related to certain contexts are analyzed based on human perception. The property measured in the particular context of perception is the *relevance* of a realization (linking) between a customer requirement and design components. This means that the modeler quantifies the quality-related *contextual effect* (e.g. functional suitability, performance efficiency, usability) a linking has in his expert opinion.

In this sense, TraCo fits in agile approaches by using prioritized customer requirements (e.g. provided by Scrum [8]) as input for the extended linking among requirements and design artifacts. The calculated ratios (e.g. Fig. 3, Fig. 4) help stakeholders (i.e. all people directly and indirectly involved in the project) to get a better understanding of the requirements and their implementation. This will help to provide a common understanding of the project's objectives and it leads to a ranking of the relevance of design components (modules) to be developed. The stakeholders' feedback of the early prototype is to be taken into account in the computations. That enables modelers to continuously monitor and control individual design components and gives stakeholders and customers a quick overview of the relevant parts of the architecture model which in turn leads to highest customer satisfaction.

This paper is organized as follows: the domain of interest, a short introduction to the main problem and a brief insight in our approach are discussed in Section 1. Section 2 presents background knowledge and the theoretical basis of our contribution. Section 3 describes the theoretical approach and the method realized in the TraCo-project. Finally, we present our conclusion in Section 4.

II. BACKGROUND

A. Cognitive Perspective in Modeling

In [1] the authors refer to the existence of a *cognitive perspective* in the field of knowledge base (i.e. ontology) engineering. They state that this perspective *"is very important in the analysis of what is generally called an intentional context"* [1]. In previous works in the field of ontology engineering and alignment [9][10], we distinguish between formal design decisions and non-formal ones. We relate formal

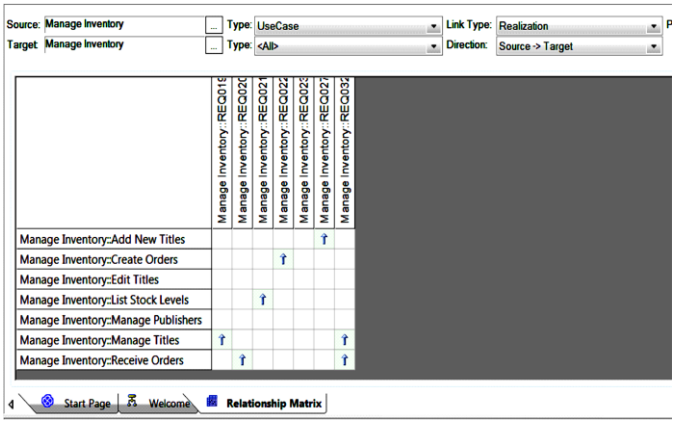


Fig. 1. Relationship matrix in the tool Enterprise Architect (EA)

decisions to the modelers' logical perspective and non-formal decisions to their cognitive perspective. The logical view posits the kind of knowledge modelers have when describing the concepts of a domain well-formed by using the syntax and semantics of a language (e.g. OWL DL's necessary and sufficient conditions [24]); whereas the usage of the language in certain contexts, which the *use-conditional* meaning of semantics is meant, is based on the modelers' cognitive perspective [11]. Thereby, the focus is on the importance of a decision that cannot be detected by truth condition.

Also surveys have shown that design decisions are only to a small extent logical and deductive driven when describing a domain of interest [3][14][7]. Based on these surveys and previous work in [11], the author presents a *Cognitive Design Methodology (CoMetO)* in the field of Ontology Alignment. The objective in CoMetO is to provide users–in combination with model-based semantics–a "complete package" for meaning interpretation as input in the alignment process. The author introduces an alignment support that already starts when developing ontologies (i.e. in ontology engineering). In her approach, the modelers' cognitive perspective on the concepts of a domain to be described and certain contexts are taken into account. In CoMetO, ontology engineers determine the contextual effects of logical statements in the domain description. For this purpose, she adapts the relevance-based inferential model of verbal communication [22] for supplementing the ontology's rational structure with context-based (cognitive) semantics. She implements a method by which the importance of ontology entities can be evaluated based on their usage in certain (domain- and modeling-related) contexts. The methodology introduced in CoMetO makes it feasible to visualize heterogeneities among entities of two ontologies to users prior to starting an alignment process.

B. Requirements Traceability Matrix

Generally, validation pain can be reduced when using a *relationship matrix* by which a traceability view is facilitated during design [15]. The relationship matrix is a kind of "completeness indicator" by which the relationships among requirements and design components can be traced. According to the used modeling language a certain traceability link can be used (*realization* in UML [4], *satisfy* in SysML [5]). There is a multitude of tools (e.g. Rational DOORS [19]) by which

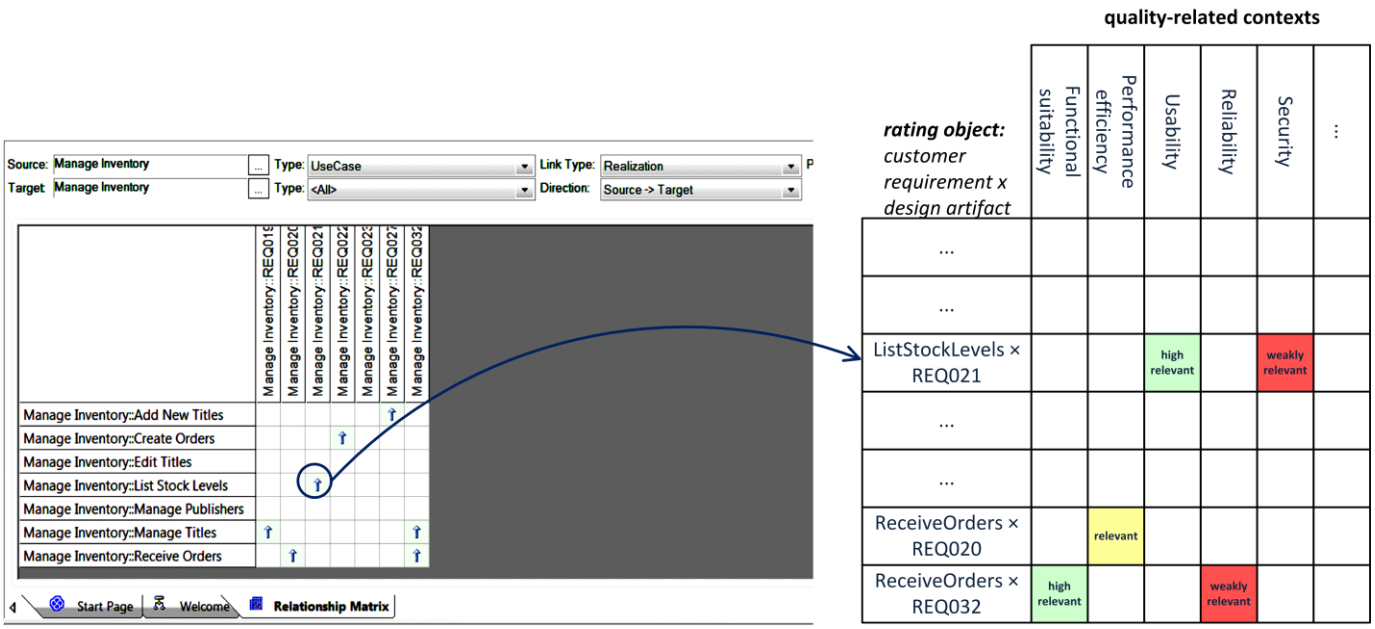


Fig. 2. EA-Relationship matrix and Weighted Design Decision Matrix

requirement lifecycle management is supported. There are tools available by which graphical as well as textual traceability are supported. Often, tools produce non- or less compelling documentation [15].

Fig. 1, shows an example of an excerpt of a relationship matrix from the tool *Enterprise Architect (EA)* [15]. In the EA-tool, the template of the relationship matrix has the form of a spreadsheet where the relationships between different sets of model elements can be visualized. Users can define the direction for displaying the source and target packages and the relationship type. All linkings among source and target elements can be identified by highlighting a grid square and displaying an arrow indicating the direction of the relationship.

The matrix is a convenient method of visualizing relationships quickly and definitely. It guides users to create, modify and delete relationships between elements with a single mouse click which is another quick way to create complex sets of element relationships with a minimum of effort. The matrix is an aid in order to trace which customer requirements are to be implemented by which design components. Based on this, the modeler can make sure that all customer requirements are taken into account in the solution *(fulfillment of completeness)* and he can check whether the requirements are jointly satisfiable *(fulfillment of consistency)*. The matrix helps to keep the amount of requirements manageable and to track linkings comprehensible.

We argue that a simple representation of linkage alone is not enough for a sufficient quality control at this stage of system design. For instance, there is a lack to verify to what extent the realization has proceeded, or to specify the design components' level of utility to fulfill customer requirements by additionally taking into account the stakeholder values (i.e. requirements' prioritization). In other words, the relevance of the linking from an "integration perspective"–with regard to the substantive aspects of quality such as impact, adequacy, and appropriateness–is ignored. This means that the modeler's tacit and context-based knowledge, which he has in mind when modeling, cannot be recorded and therefore is lost. Currently, this knowledge is stored implicitly in the design model's structure and therefore not transparent to other stakeholders–indirectly as well as directly–involved in the design process (e.g. project manager, customer, developer).

III. Theoretical Approach

Our aim is to implement an "intentional model" in the tool Enterprise Architect. The effect size of this model is *cognitive processes*. In our approach, the modeler continuously goes in reflection with the model's design components and the "utility" of their usage. Thereby, he evaluates each linking among customer requirements and design components referring to its effect in certain quality-based contexts regarding the actual use of the system. That supports modelers to the effect that they have in mind the fulfillment of quality-properties of the system (e.g. security, usability, maintainability) already during design.

We present a technique which facilitates the systematization and (non-monetary) quantification of thought processes during modeling in order to make the modelers' intention (their cognitive perspective) explicit to stakeholders. We provide engineers an aid so that they are able to express their internal mental state at this phase of development. For this purpose, we implement a formalism based on the concepts of *Cognitive Function Analysis* introduced by Boy [21]. Cognitive Functions attempt to characterize the activity of a human involved in the execution of task. They are defined by attributes related to the transformation of a task into an activity [21]. In his work he presents the *AUTO-pyramid* where an artifact, a user-profile, a task and an organizational environment are interlinked and so the processes that take place between them. In TraCo, the modelers' situational (tacit)

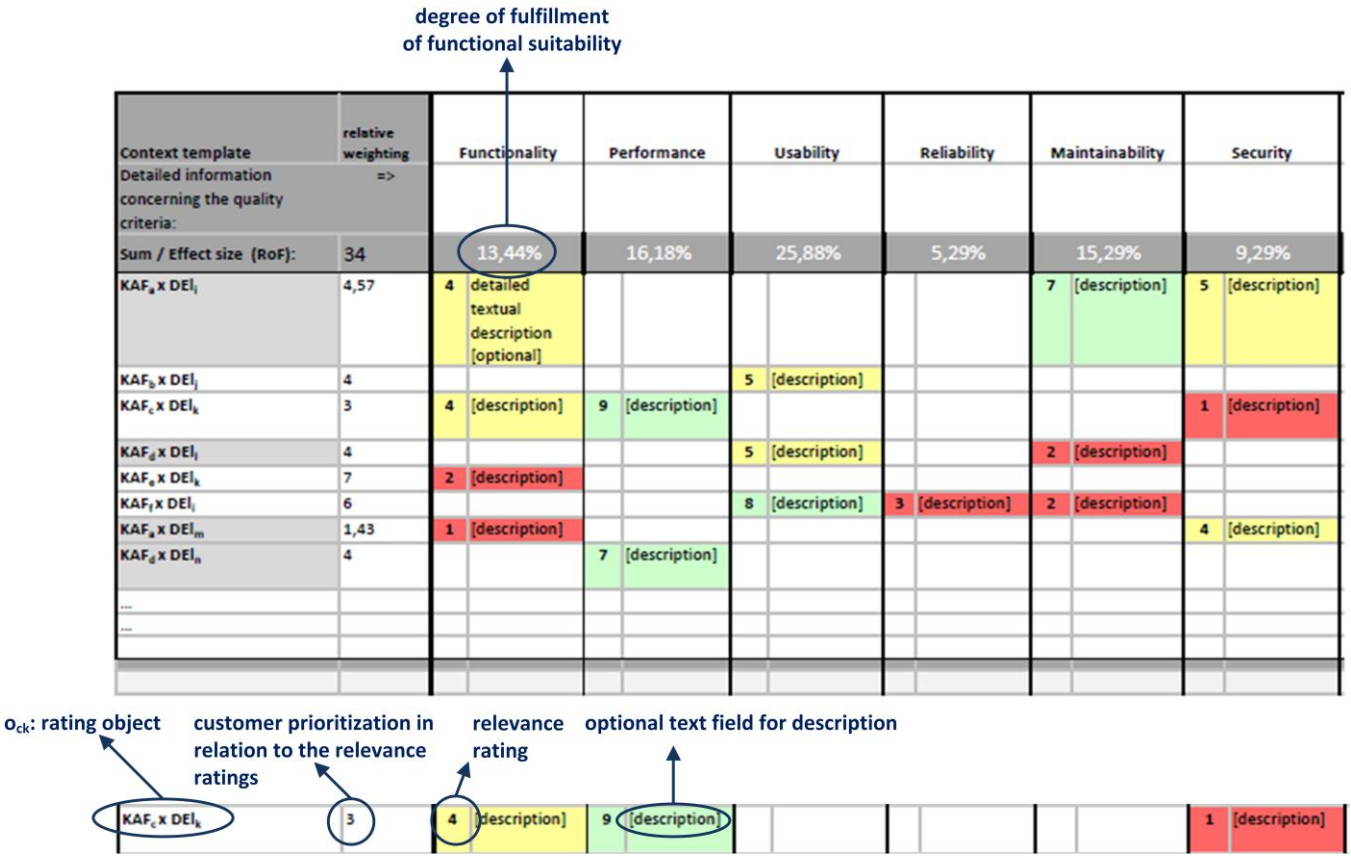


Fig. 3. Weighted Design Decision Matrix (WDDM)

knowledge–their *modeling focus*–is formalized and explicitly poured into numbers by *contextual constraints* or *cognitive markers*.

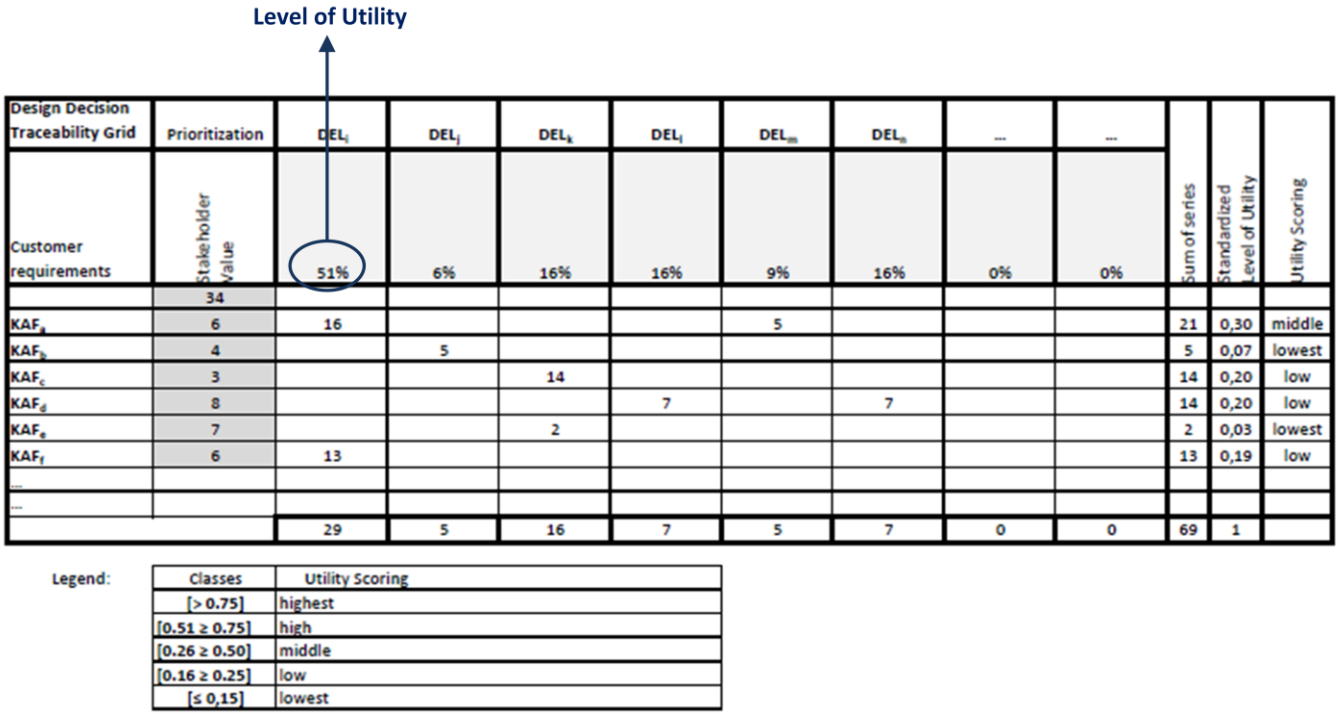
Software architects decide about how to transform customer requirements into design components. Generally, this design step is visualized in the traceability matrix in which the realizations can be traced (e.g. Fig. 1). The better this task is monitored and controlled, the fewer problems will occur during implementation which leads to time and cost savings. Several solutions are based on a design decision. This means that there are several variations to model a customer requirement. Thus, not all design components are equally important in the model equally as customer requirements have different stakeholder values. We take this circumstance into account by the aforementioned cognitive constraints, which are manually assigned in form of *relevance ratings* on each linking. The ratings are based on an intuitive mental judgement made by the modelers. The weighting assessment is directly carried out when transforming customer requirements into design components. The procedure is automatically initiated by the system when a realization is generated. The modeler is prompted by the system to select a predefined context or to create a new one and to assign a *relevance rating* in form of points. The modeler can distinguish between three classes (i) *"high relevant"* in the range of [7, 9], (ii) *"relevant"* in the range of [4, 6], and (iii) *"weakly relevant"* in the range of [1, 3] (e.g. Fig. 2, on the right side and Fig. 3).

In the EA-modeling tool the relationship of which design components realize which customer requirements is represented using the UML *realization link* (e.g. Fig. 2, on the left side visualized by the arrow icon). Each decision situation when linking customer requirements with design components forms the starting point for our cognitive walkthrough. This evaluation procedure, similar to the reflective practitioner [20], forces modelers to an intensive analysis with the quality aspects of the model along the design process. They can continuously monitor the made decisions and its alternatives at any time. By doing so, the consequences and implications of design decisions can be detected early and modified as needed. Moreover, they can track how the scoring of design artifacts is affected when stakeholders vary their prioritization, or delete, or add new requirements (*fulfillment of impact*).

A. Method

The modelers' contextualized design knowledge is annotated by contextual constraints (cognitive markers) in form of relevance ratings and implemented as *tagged values*. This context-based knowledge itself is a data structure storing contextual constraints consisting of *ID, source, context facet, context criterion, rating object, weighting and description* directly in the EA-repository.

Fig. 3, shows the *rating object* (o_{ck}) which is a key value pair consisting of the unique IDs of a customer requirement (KAF_c, foreign key of the requirement c) and a design component (DEl_k, foreign key of the artifact k). This means,



Design Decision Traceability Grid	Prioritization	DEL_i	DEL_j	DEL_k	DEL_l	DEL_m	DEL_n	...	...	Sum of series	Standardized Level of Utility	Utility Scoring
Customer requirements	Stakeholder Value	51%	6%	16%	16%	9%	16%	0%	0%			
	34											
KAF_a	6	16				5				21	0,30	middle
KAF_b	4		5							5	0,07	lowest
KAF_c	3			14						14	0,20	low
KAF_d	8				7		7			14	0,20	low
KAF_e	7			2						2	0,03	lowest
KAF_f	6	13								13	0,19	low
...												
...												
		29	5	16	7	5	7	0	0	69	1	

Legend:

Classes	Utility Scoring
[> 0.75]	highest
[0.51 ≥ 0.75]	high
[0.26 ≥ 0.50]	middle
[0.16 ≥ 0.25]	low
[≤ 0,15]	lowest

Fig. 4. Design Decision Traceability Grid (DDTG)

we couple the modeler's perspective with the customer's perspective also taking into account the stakeholder values, using a relative weighting. The *relative weighting* is a ratio value that is calculated from the predetermined customer prioritization (this information for each KAF is received from the EA-repository) and the previously conducted relevance ratings (in respect of each row of the matrix in which the current KAF is a part).

The weighting assessment is carried out directly by the modeler. He is prompted by the system to select a predefined context (based on the standard ISO/IEC 25010 [25]) or to create a new one and to affix a value from a 9-point rating scale. The relevance of semantics underlying the cognitive walkthrough-based evaluation means: the more relevant a design solution is with respect to a given context, the greater is its contextual or quality assurance effect in the model in terms of the final product and its real use.

For instance: [1, 9] is a scale of relevance (9 = highest relevant, 1 = weakly relevant), then annotating the rating object o_{ij} with 8 and o_{ik} with 3 implies that o_{ij} has more contextual effect than o_{ik}.

As a result of the rating procedure, the engineer receives a matrix (e.g. Fig. 3) reflecting the design choices expanded by a "relevance view". We implement this matrix–the *Weighted Design Decision Matrix (WDDM)*–in the EA-tool to foster a better communication among (directly and indirectly involved) stakeholders during the design phase. If the matrix is filled by multiple modelers a map of different cognitive perspectives in different contexts is obtained to them, so that they can validate a common understanding of different modeling views. The *degree of fulfillment* is an indicator of the extent (in percent) to which a certain quality-related context (e.g. functionality, usability) was included in the modeling. Additionally, stakeholders receive valuable metrics (e.g. Fig. 4) by which they are able to estimate the *impact* on the design model and its components when changings of customer requirements occur (e.g. delete, add requirements or change its weights).

Fig. 4, presents the *Design Decision Traceability Grid (DDTG)* where the design components' *level of utility (LoU)*–an importance indicator–is automatically computed and continuously updated with the WDDM by an algorithm based on (i) the number of modeled customer requirements (KAFs), (ii) their initial prioritization made by stakeholders, and (iii) the sum of relevance ratings (absolute frequency) with respect to each covered KAF, derived from the WDDM (e.g. Fig. 3). The LoU for each component is calculated as a percentage of the sum product. Besides a *utility scoring* is computed in order to check for the engineer if he is on the "right way". For instance, he can continuously monitor and control whether low prioritized requirements were not modeled too detailed in the design model (*fulfillment of adequacy*).

IV. Conclusion

In the presented approach, we use the EA-relationship matrix for identifying the realizations among customer requirements and design components as well as their predetermined stakeholder values (e.g. Fig. 1). We extend this matrix to a Weighted Design Decision Matrix (e.g. Fig. 2) for making it feasible for modelers to quantify the realizations' relevance in certain contexts, based on their cognitive perspective when modeling (e.g. Fig. 3). In a next step, we introduce the Design Decision Traceability Grid (e.g. Fig. 4). The values of this grid are automatically determined as a function of the relevance ratings of the WDDM. The relevance ratings as well as the stakeholder values are persisted in the EA-repository and the WDDM and DDTG are continuously

updated. There is a "reciprocal relationship" between these two controlling instruments.

The two matrizes and the calculated metrics provide a kind of *cognitive map* for engineers to rethink about their design decisions during the process. Each map represents a personal view on parts of the design model's solution. By comparing these maps a hint to a different system understanding can be prepared for the stakeholders involved. Additionally, they are aids for stakeholders to monitor and control the internal software quality by means of indicators (degree of fulfillment and level of utility). Both instruments are designed to guide modelers by identifying which requirements were not, or not enough or even transferred too detailed in the model (*fulfillment of adequacy*). Thereby, realizations can be identified that cause low benefit (*fulfillment of appropriateness*). As a result, a possible over-engineering risk can be minimized even before the implementation task starts. Moreover, engineers can identify significant design components to give them priority, e.g., as a starting point for the product owner in Scrum [8].

ACKNOWLEDGMENT

The introduced approach is realized in the course of the TraCo project. TraCo is supported and promoted by the Austrian Research Promotion Agency (FFG) by the funding program BRIDGE. The cooperation partner in this 18-month project (launched October 01 2012) 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 (EA), and it is to be implemented as a component of the business logic of EnArWeb (a web-based interface to EA repositories).

REFERENCES

[1] M. Benerecetti, P. Bouquet, and C. Ghidini, "On the dimension of context dependency," LNCS, vol. 2116, pp.59-72, Springer, Heidelberg (2001) [Digests 3th International and Interdisciplinary Conference, Dundee (UK)].

[2] 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.

[3] S.M. Falconer, N.F. Noy, and M.-A. Storey, "A cognitive support framework for ontology mapping," LNCS, vol. 4825, pp. 114-127, Springer, Heidelberg (2007) [Digests 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, Busan].

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

[5] 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.

[6] 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.

[7] Ch. Janiesch, "Situation vs. Context: Consideration on the Level of Detail in Modelling Method Adaption," IEEE Computer Society, pp. 1-10, 2010 [Digests 43rd International Conference on System Science, Hawaii].

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

[9] A. Mazak, L. Lanzenberger, and B. Schandl, "iweightings: Enhancing Structure-based Ontology Alignment by Enriching Models with Importance Weightings," IEEE press, pp. 992-997, doi: 10.1109/CISIS.2010.164, February 2010 [Digests 3th International Workshop on Ontology Alignment and Visualization (OnAV), Krakow (PL)].

[10] A. Mazak, B. Schandl, and M. Lanzenberger, "align++: A Heuristic-based Method for Approximating Mismatch-at-Risk in Schema-based Ontology Alignment," In Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD), pp. 17-26, October 2010, Valencia (ES).

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

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

[13] K. Pohl and Ch. Rupp, Basiswissen Requirements Engineering, dpunkt.verlag, Heidelberg 2011.

[14] P.R. Smart and P.C. Engelbrecht, "An Analysis of the Origin of Ontology Mismatches on the Semantic Web," LNCS, vol. 5268, Eds. A. Gangemi, J. Euzenat, pp. 120-135, Springer, Heidelberg 2008.

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

[16] A. Schatten, S. Biffl, M. Demolsky, E. Gostischa-Franta, T. Östereicher, and D. Winkler, Best Practice Software-Engineering, Spektrum Akademischer Verlag, Heidelberg 2010.

[17] Austrian Research Promotion Agency (FFG), http://www.ffg.at/auftaktveranstaltung-ikt-2012.

[18] M. Broy, TU München, http://www4.in.tum.de/broy.

[19] 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.

[20] D.A. Schön, The Reflective Practitioner: How Professionals Think in Action, Basic Books Inc. 1983, Ashgate Publishing Limited, 2011.

[21] G.A. Boy, Cognitive Function Analysis, Ablex Publishing Corporation, London 1998.

[22] D. Wilson and D. Sperber, Meaning and Relevance, Cambridge University Press, Cambridge (UK) 2012.

[23] T. Tullis and B. Albert, Measuring The User Experience, Collecting, Analyzing, and Presenting Usability Metrics, ELSEVIER, 2008.

[24] W3C, World Wide Web Consortium, OWL 2 Web Ontology Language, October 2009, available at http://www.w3.org/TR/owl2-guide/.

[25] ISO/IEC 25010: 2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, available at http:// http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733.