Problem-Aware Traceability in Goal-Oriented Requirements Engineering

Grace Park, Lawrence Chung
University of Texas at Dallas
{exp130530, chung}@utdallas.edu

Jang-Eui Hong
Chungbuk National University
jehong@chungbuk.ac.kr

José Luis Garrido, Manuel Noguera
University of Granada
{jgarrido, mnoguera}@ugr.es

***Abstract*— Requirements traceability helps to ensure that a requirements specification is aligned with the intended stakeholders' needs. Such alignment should involve the consideration of why such needs arise, in terms of what problems the stakeholders are faced with, and what kinds of software system may help alleviate or eliminate the problems. However, little work can be found on requirements traceability that explicitly considers the problems. In this paper, we propose an problem-aware framework for establishing requirements traceability, in the context of goal-oriented requirements engineering, which explicitly models problems and their root causes, together with other important ontological concepts, stakeholders' goals, and both functional and non-functional requirements as a solution, and issues. In this framework, ontological concepts are partitioned into layers, reflecting which traceability links are classified into intra- and inter-traceabilities leading to several kinds of links. Additionally, undesirable consequences of inappropriate traceabilities are also categorized. A case study shows some key benefits of the framework.**

***Index Terms*—Requirements Traceability, Ontology, Problem-Aware, Goal-oriented Requirements Engineering**

I. Introduction

Requirements traceability [1] refers to the ability to relate various concepts about requirements, such as problems, goals and requirements, to each other, and helps ensure that the requirements specification is aligned with the intended stakeholders' needs. It aids impact analysis, process visibility, verification and validation, and change evaluation, hence being a critical success factor not only for an initial software development phase but also for its subsequent maintenance phase [2].

Alignment between a requirements specification and the intended stakeholders' needs should be considered in terms of why such needs arise – this in turn in terms of what *problems* the stakeholders are faced with, and what kinds of software system may help alleviate or eliminate the problems.

However, little work can be found on requirements traceability that explicitly considers the problems, even though the key role of a software system is supposedly to help alleviate or eliminate the problems that the stakeholders are suffering from.

In this paper, we propose a problem-aware framework for establishing requirements traceability, in the context of goal-oriented requirements engineering (*GORE*). In this framework, problems and their root causes are explicitly modeled as key ontological concepts (or vocabulary), together with other important ontological concepts, i.e, stakeholders' goals, and both functional requirements (*FRs*) and non-functional requirements (*NFRs*) as a solution to the problems and at the same time a means for the goals, and issues (e.g., ambiguity and inconsistency) with any such individual concept or any relationship between two individual concepts. In this framework, ontological concepts are partitioned into five layers – problem, goal, requirements, prototype and issue layers. Traceability links then are classified into intra- and inter-traceabilities (layer link). An intra-traceability refers to a relationship between concepts in the same layer, while an inter-traceability refers to a relationship between concepts in two different layers. The layer link can be combined with refinement link (satisficing and decomposition-links), which results in requirements product links for requirements themselves. Additionally, issue link can combine together with refinement link, which leads to requirements process links for rationale of the requirements process. This framework also identifies several classes of undesirable consequences of improper traceability links.

We have carried out a case study, involving about 60 teamwork-oriented course projects in several senior- and graduate-level requirements engineering (*RE*) classes with about 500 students in total. Through this case study, we have observed how the presence of proper traceabilities, or lack thereof, affects the quality of the resulting requirements and prototype. More specifically, our observation shows how frequently omissions or commissions of traceability links are made by students, and of what kinds – leading to incomplete or incorrect requirements. To provide a sense of how this case study has been carried out, a comparison of two groups' work will be shown. A statistical analysis result shows that our framework can indeed help improve the quality of the requirements and corresponding prototypes, by making them more complete, correct and clear, while capturing important rationales.

There has been some work on requirements traceability (e.g., see [1]). Our work is similar to [3], which provides a rich set of issues and ontological concepts such as stakeholder, object, and source as a reference model for requirements traceability, and to [4], which takes a goal-oriented approach to requirements traceability; in our framework, problems and their root causes are treated as a central and fine-grained ontological concept, along with goals, layers and different types of traceability links. The notions of goals and problems are also mentioned in [5], but without ontological layers, layer links (inter-/intra- links), refinement links (satisficing-/decomposition-links), root causes and issues. Model-driven traceability has

DOI reference number: 10.18293/SEKE2016-210

been addressed in general [6] and more specifically for requirements in the context of MDA [7], but with different ontological concepts and traceabilities for the life after requirements established; this framework more focus on the life before them.

Section 2 introduces goal-oriented models that have been adopted for the ontology of the framework. Section 3 presents our problem-aware framework for goal-oriented requirements traceability. Sections 4 and 5 respectively describe a case study and a discussion. At the end, a summary of the paper is described, along with some future work.

II. Adopted Goal-oriented Models

For our problem-aware traceability framework in the context of goal-oriented requirements engineering (GORE), we adopt work, on the one hand, on representing functional and non-functional requirements, and on representing problems, on the other.

A. Representing Functional and Non-Functional Requirements

Concerning the representation for both FRs and NFRs, there are several goal-oriented frameworks, including KAOS [8], i* [9], and the NFR Framework [10], each with its own emphasis and characteristics. We adopt the NFR Framework, since it is also common to many other goal-oriented models.

An NFR, such as accuracy, security and performance, typically has no clear-cut definition or criteria whether it is absolutely satisfied or not. Accordingly, in the NFR Framework, an NFR is treated as a softgoal, and the notion of "satisficing" is used. Each softgoal has an associated NFR type and one or more topics (See Fig. 4 for examples).

There are three types of softgoals (See Fig. 1): NFR, Operationalizing and Claim softgoals. An NFR softgoal is an NFR to be satisficed; an operationalizing softgoal is a concrete means (e.g., an operation to be carried out by people – expectation, or a FR to be implemented in the projected software system), which can achieve an NFR softgoal; and a claim softgoal is an argument/justification. Each softgoal can be either AND or OR decomposed into sub-softgoals or make a contribution towards satisficing another softgoal, fully or partially positively (MAKE or HELP), or fully or partially negatively (BREAK or HURT). A bottom-up label propagation mechanism evaluates the effect of a decision on upper softgoals, with a label - Satisficed, Denied, Conflict, or Undetermined. Softgoals and relationships between softgoals are represented in a softgoal interdependency graph (SIG).

B. Representing Problems

There are several different kinds of models for problem representation and root cause analysis, including notably FTA (Fault Tree Analysis) [11], Fish Bone Diagram [12], and PIG (Problem Interdependency Graph) [13]. While FTA is suitable when information is available about AND/OR logical relationships among root causes, Fish Bone Diagram is adequate when uncertainties exist about relationships among root causes. We adopt PIG, since it accommodates both conventions, and additionally it closely resembles SIG, offering NFR softproblem and Operationalizing softproblem, together with the same kinds of satisficing relationships.

III. A Problem-aware Traceability Framework in Goal-Oriented Requirements Engineering

Our problem-aware traceability framework in GORE offers an ontology, with problems as among the essential concepts or vocabulary, by extending the two adopted ontological concepts of goals and problems. The framework then introduces five layers of ontological concepts, which in turn lead to several kinds of traceability links. Additionally, the framework offers several different categories of undesirable consequences of omissions or commissions of traceability links.

A. Overall Ontology

Fig.1 shows a somewhat simplified ontology on our adopted goal-oriented models described in the previous section, with detailed refinements of some concepts omitted. It shows key concepts such as problems, goals (including softgoal), requirements, prototypes for requirements visibility and issues for requirements process. As stakeholder is the owner of problems or goals, it representst whose problems and goals. As for trace-

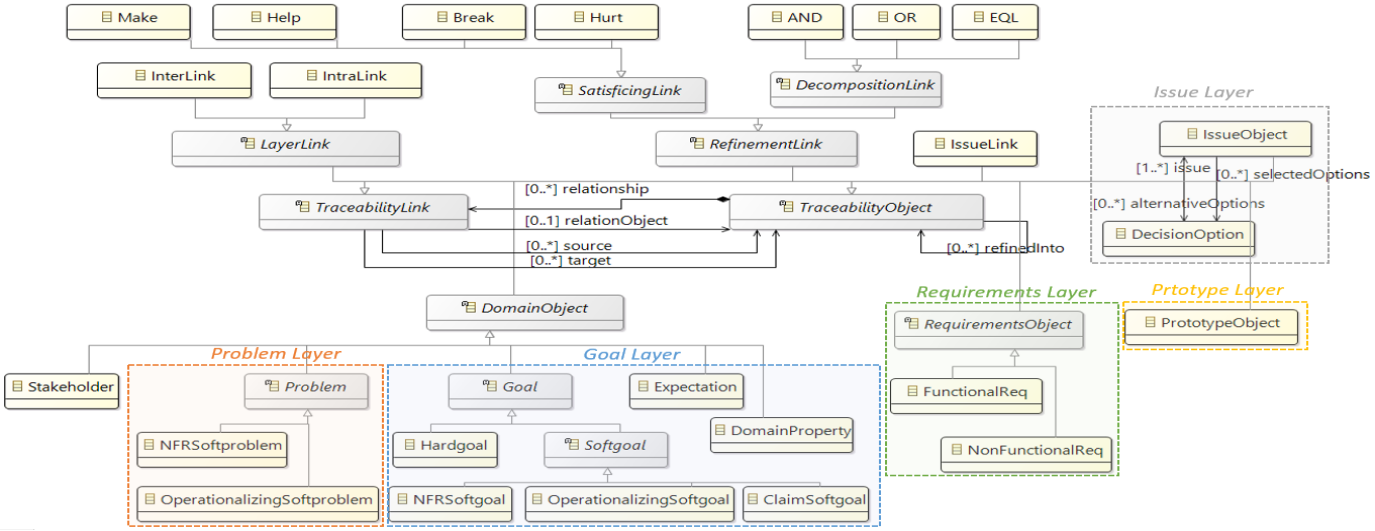

Fig. 1. Ontology for Problem-Aware Traceability in Goal-Oriented Requirements Engineering (GORE)

bility links, there are refinement links such as satisficing- and decomposition- link, and layer link such as inter- and intra-links. Essentially, a traceability link can exist wherever there can be a relationship between the source and target traceability objects. Traceability links are mostly bi-directional, i.e., forward and backward.

B. Layers of Traceability Objects

The ontological concepts are partitioned into five layers, as shown in Fig.1 and Fig. 2, and the example is in Fig. 3.

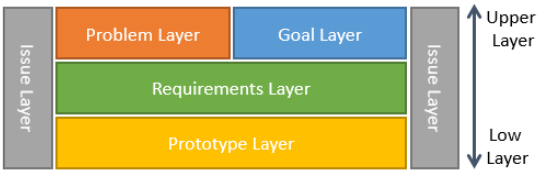

Fig. 2. Layers of Traceability Objects

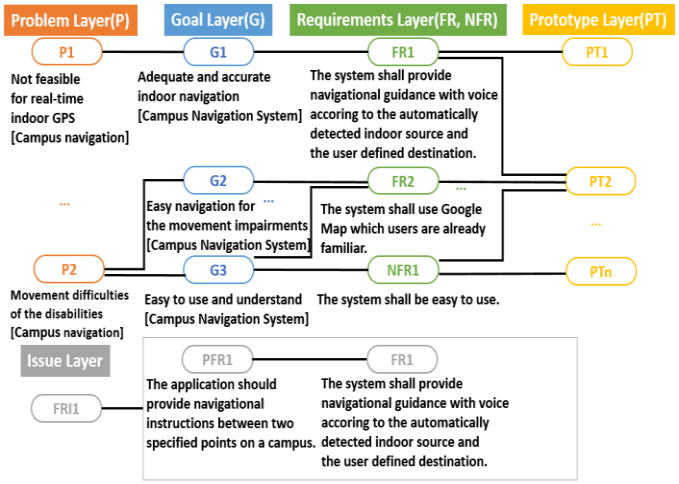

Fig. 3. Examples of Inter Links of Campus Navigation Application for the disabled

- Problem Layer: to represent phenomena that are against stakeholders' goals. This layer is for representing problems and their root causes, which can also help discover critical goals. For example, in Fig. 3, *P1.Not feasible for real-time indoor GPS [Campus Navigation]* is a root cause which handicapped people have in campus navigation. The procedure to extract the root causes is shown in the upper portion of Fig. 4.
- Goal Layer: to represent stakeholders' intentions, which helps to explore alternatives in requirements and select among them. For example, in Fig. 3, *G1.Adequate and Accurate indoor navigation [Campus Navigation System]* is a refined goal, which is against *P1.*
- Requirements Layer: to represent functional & non-functional requirements. For example, *the system shall provide navigational guidance with voice according to the automatically detected indoor source and the user defined destination (FR1)* is a functional requirement, for achieving the goal, *G1*.
- Prototype Layer: to represent a user interface as a prototype which implements FRs and NFRs.
- Issue Layer: to represent issues that happen with any statement such as ambiguous, inconsistent or conflicting statements, options to resolve them, and trade-off analyses with rationales. More details are discussed in Section C.3.

Relationships between the problem layer and goal layer are generally negative satisficing (e.g., BREAK or HURT). Additionally the requirements layer has positive satisficing relationships (e.g., MAKE or HELP) for the goal layer, as a means, and negative satisficing relationships for the problem layer, as a solution. That's why the problem layer and the goal layer are located in the same layer and requirements layer is under the layers.

C. Classification of Traceability Links

In our framework, there are two kinds of traceability links: Requirements Product- and Requirements Process- Traceability links. While Requirements Product Traceability are links between problems and requirements themselves, Requirements Process Traceability are links for rationales on how the requirements were created.

1) Requirements Product Traceability can be defined as a cross product between Layer Link and Refinement Link:

Requirements Product Traceability Link =
Layer Link X Refinement Link

1.1) Layer Link: This concerns the boundaries of traceability links, and there are two kinds:

a) Intra Link: refers to any relationship among traceability objects within the same layer. For example, links among NFR, Operationalizing, and claim softgoals are intra links because they belong to the same Goal Layer.

b) Inter Link: any link among traceability objects across different layers (except for the issue layer), including:

- Problem-Goal (*PG*) Link: helps ensure all the problems are linked to goals as their context.
- Goal–Requirements (*GR*) Link: helps ensure goals are at least partly achieved (satisficed) by requirements.
- Operationalizing goal-Requirements (*OR*) Link: helps ensure some operationalizing (soft)goals are linked to functional requirements. All goals should be refined into sub-goals small enough an agent can be assigned. If the leaf goal is assigned to a software system, that will be a requirements. This link shows the relationships between the assigned leaf goal and its requirements.
- Requirements-Prototype (RP) Link: helps validate the correctness of the requirements, using a prototype.

Fig. 4 shows some intra and inter traceability links within, as well as across, PIG (the upper portion) and SIG (lower portion). Using PIG, abstract problems are further refined, and they can help to find refined critical goals (i.e., leaf goals) by breaking or hurting the leaf problems.

1.2) Refinement Link: This concerns refinement using parent and child relationships:

a) Satisficing Link: refers to any link showing how a traceability object contributes to its parents, through Make,

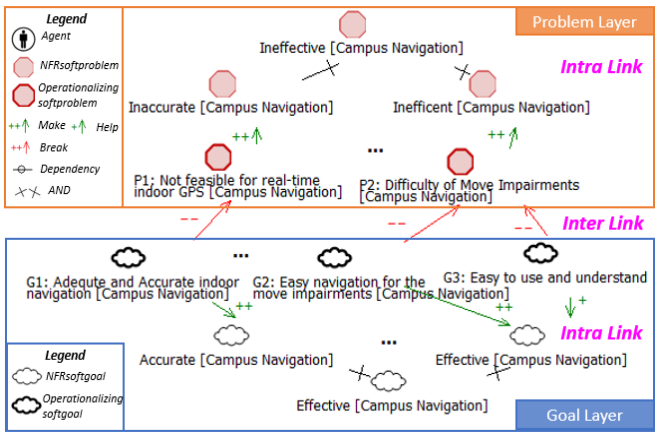

Fig. 4. Examples of Intra and Inter Links

Help, Hurt or Break.

b) Decomposition Link: any link showing how a traceability object is composed through AND or OR relationships.

2) Requirements Process Traceability can be defined into a cross product between Issue Link and Refinement Link:

Requirements Process Traceability Link =
Issue Link X Refinement Link

- *Issue Link*

This concerns any link, involving issues (ambiguity, inconsistency or conflict) on anything in any layer, i.e, any problem, goal, FRs/NFRs or prototype features. An issue is associated with options for resolving it, together with rationale for choosing one or more of the options, through tradeoff analysis. Table 1 is an example of issue traceability for a preliminary functional requirement (PFR1), concerning the scope of a campus, which is ambiguous. To alleviate the ambiguity, there are two options and option 2 is selected with a rational, concerning the project time and resource constraint.

TABLE 1. An Example of Issue Traceability

ID	Description	
FRI 1	PFR1	The application shall provide navigational instructions between two specified points on the ABC campus.
	Ambiguous. Does the campus include outdoor parking lots?	
	Option 1	Define the campus as indoor such as classroom and service buildings.
	Option 2	Include all the campus, from classroom and service buildings to parking lots.
	Choice	Option 1
	Rationale	Given the time and resource constraint of the project, Option 1 is better.

D. Undesirable Consequences of Improper Traceability Links

If traceability links are not properly established or omitted, the following undesirable consequences (i.e. defects) can arise.

- Incompleteness: all leaf elements in an upper/same layer are not properly dealt with in a lower/same layer due to the inappropriate forward inter traceability. For example, given a goal "offer a voice job search capability for those with disabilities" in goal (upper) layer, if there are no corresponding requirements to deal with the goal in requirements (lower) layer, then it will cause incompleteness defects.
- Incorrectness: an element in a lower/same layer is not correctly described according to a corresponding element in its upper/same layer due to both improper backward inter and intra traceability links. For example, let us suppose that for the above goal, "the system shall provide a function to learn job skills" is a requirement. The requirement is incorrect because the requirement does not adequately achieve the goal, since learning job skills is not much related to job search.
- Risk of Gold Plating: the addition of expensive and unnecessary features to a system due to lack of backward inter and intra traceability.
- Ambiguity: there are more than one interpretations and no rationale for a decision because of improper issue links.

IV. Case Study

We have conducted experiments to validate our problem-aware traceability framework, through the group projects of several undergraduate-, graduate-level and industry- requirements engineering courses, which one of the co-authors has been teaching for more than 10 years. Most of undergraduate student were consist of senior student whose majors were computer science, but there were some students who had been working more 10 years in Information Technology (IT) industry. This were similar to the graduate level course. The students of the industry course had IT industry experience more than 3years. We selected projects which were conducted in those courses from year 2011 to year 2015. Most projects were similar domain about building smartphone apps for people with mental or physical difficulties, but not the same apps. There have been about 60 projects which the average number of members were about 8 (12 from senior-level courses, 38 from graduate-level courses, 10 from executive courses for industry people), with about 500 students in total. Students learned several kinds of goal-oriented requirements models, such as PIG or SIG, and applied their knowledge to their projects, using the provided our guidelines and a traceability template which is available [14]. Each project selected the combination of traceability link types on its own decision.

This study has shown the presence of traceability links, or lack thereof, affects the quality of requirements and prototypes. To show the correlation, Teaching Assistant (TA) who is one of coauthors have carried out an overall defect analysis, and analyzed the kinds of defects which students frequently made. To provide a sense of the analysis, we show a comparison between an exemplary project which applied most of the proposed traceability links and another one which did not, which resulted in the functionality differences of their prototypes. Various student documents, which have been used for this case study, are publicly available (including [14]).

A. Overall Defects Analysis

Fig. 5 shows how different kinds of inappropriate traceability links affect the different kinds of defects in terms of the num-

ber of defect projects. The most common kind of defects was requirements incompleteness (as in [15]), and our observation shows that this is caused by omission of traceability links from an upper layer to a lower layer (i.e., forward traceability links). The next frequently-occurring is incorrectness, due to the lack of semantic correspondence between concepts in a lower layer and those in an upper layer that are backward traceable from them.

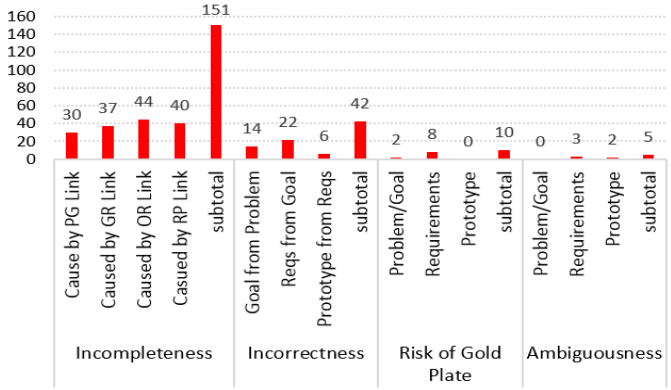

Fig. 5. Overall Defects Analysis

B. Common Defects & Effectiveness of Our Proposal to Reduce Defects

1) Incompleteness

Incompleteness defect, as described in the previous section, arises when all leaf elements of an upper layer are not properly dealt with in a lower layer due to the inappropriate forward inter traceability. This kind of defects mostly result from different inter satisficing links such as Problem-Goal (PG), Goal-Requirements (GR), Operationalizing goal-Requirements (OR) or Requirement-Prototype (RP) and the defects can lead to incompleteness of requirements. To measure the incompleteness defect rate, we divided the target projects into two groups: the projects which applied our suggested link vs. those of not applied each link. Then, for each group, we used the following formula.

$$IncomDR = \frac{IncomDO}{TOT}$$

, where IncomDR is Incompleteness Defect Rate, IncomDO is the number of projects which Incompleteness Defect Occurred and TOT is the TOTal project number, in a specific group.

As Fig. 6 shows, applying inter satisficing links positively affects to reduce the incompleteness defects in all inter link types.

2) Incompleteness

Incompleteness defect arise when elements in a low layer does not semantically correspond to elements in an upper layer due to both improper backward inter and intra traceability links. Similar to the Incompleteness defect measurement, per group, we calculate incorrectness defect rates using the following formula.

$$IncorDR = \frac{IncorDO}{TOT}$$

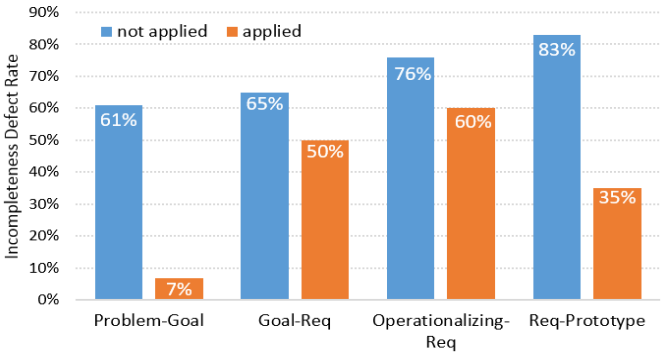

Fig. 6. Incompleteness Rates: When links present vs. absent.

, where IncorDR is Incorrectness Defect Rate, IncorDO is the number of projects which Incorrectness Defect Occurred and TOT is the TOTal projects number, in a specific group.

Fig.7 shows that the comparison of incorrectness defects between when the intra and inter links are applied and when they are not applied. This also showed positive effects on reducing incorrectness defects.

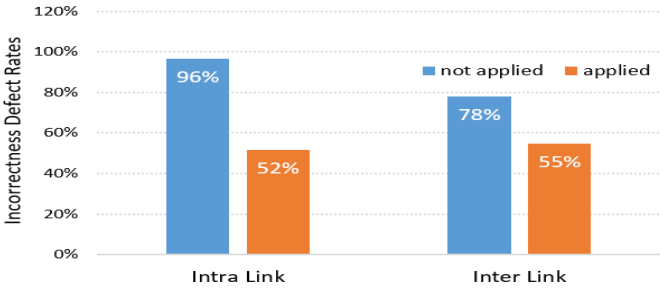

Fig. 7. Incorrectness Rates: When links present vs. absent.

C. Projects Comparison

In order to provide a sense of our case study, a comparison is given of two projects on the same subject - here, medication alert to help remind those elder people who often forget to take their medications. As Table 2 shows, while Team A applied most of the proposed links, including inter, intra and issue, Team B applied only Issue Link. Consequently, as Fig. 8 shows, while team A has more detailed information, such as Caregiver Notification and Refill Reminder which can help elder people in diverse ways, team B only has time setting.

TABLE 2. Traceability Links Applied by Two Teams

Link Type		Team A	Team B
Inter	Problem-Goal	O	X
	Goal-Requirements(Req)	O	X
	Operationalizing-Req	O	X
	Req-Prototype	O	X
Intra	Problem	O	X
	Goal	O	X
Issue Link		O	O

V. Discussion

A. Overall Observation: Although many students knew each concept, such as problem, goal, requirements and prototype, they did not well recognize relationships between problems and goals, goals and requirements, or operationalizing softgoals and

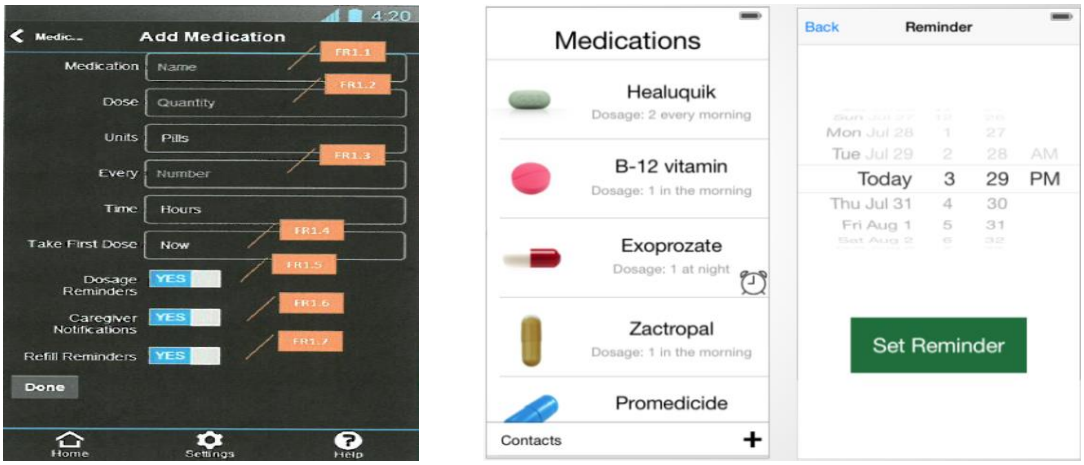

Fig. 8. Prototypes of Team A (left black background) vs. Team B (right white background)

functional requirements. Overall, the various kinds of traceability links we provided helped the students (better) understand such relationships. Through our case study, we also have observed that such links help reduce requirements defects, hence improving the quality of the requirements.

B. Threats to Validity: The quality of requirements depends on the manpower and capabilities – both individual and teamwork skills - of project team members. In our experiment, each team had different members and also the team size varied from one team to another. Moreover, the project scopes were diverse, although all of them had to do with a smartphone app for people with mental or physical disabilities, hence our experiment was not entirely homogeneous. Also, the sample size for this case study was small for a significant statistical analysis, concerning scalability, although the total number of students who participated was about 500. Additionally, the analyzer may not fully understand the each project.

VI. Conclusion

In this paper, we have proposed a problem-aware approach to requirements traceability in goal-oriented requirements engineering, in order to help ensure that the requirements specification and its corresponding prototype are well aligned with their intended stakeholders' needs. More specifically, this paper has presented 1) an ontology, which incorporates such key concepts as problems, goals, requirements, prototypes and issues; 2) five layers, which partition the ontological concepts, according to their levels of abstractions; 3) several classes of traceability links, which group relationships, according to the semantic closeness of the ontological concepts; and 4) several classes of undesirable consequences of improper traceability links. A case study, we feel, shows that our approach helps develop better requirements specifications and prototypes, and importantly in a traceable manner. This study, we feel, also has shown that our approach facilitates the detection of several different kinds of undesirable consequences of inappropriate traceabilities, such as incompleteness, inconsistencies and ambiguities in the requirements specifications and the prototypes – these defects will likely result in an implemented software system with the same kinds of defects.

There are several lines of future work. One is regarding (semi-)automatic mapping between layers and detection of omission or commission of traceabilities by using MDA (Model Driven Architecture). Another line of future research concerns provision of better templates for the various kinds of traceabilities. More case studies are also needed in various types of application domains.

References

[1] O. C. Z. Gotel and A. C. W. Finkelstein, "An Analysis of the Requirements Traceability Problem, " *Proc., 1st Int. Conf. on Reqs. Eng.*, 1994. pp. 94-101.

[2] B. Ramesh, "Factors Influencing Requirements Traceability Practice," *Communications of the ACM*, 1998. pp. 37-44.

[3] B. Ramesh and M. Jarke, "Toward Reference Models for Requirements Traceability," *IEEE Trans. Soft. Eng.*, 2001. pp.58-93.

[4] J. Cleland-Huang, R. Settimi and O. Benkhadra, "Goal-Centric Traceability for Managing Non-Functional Requirements," *Proc. 27th Int. Conf. on Software Engineering*, 2005.pp.362-371.

[5] M. Yamin, V. Zuna and M. A. l. Bugami, "Requirements Analysis and Traceability at CIM Level," *Jour. of Soft. Eng. and App.*, 2010. pp. 845-851.

[6] N. Aizenbud-Reshef, B. T. Nolan, J. Rubin and Y. Shaham-Gafni, "Model Traceability," *IBM Sys. Jour.*, 2006. pp. 515-526.

[7] A. João Paulo A, M.E. Iacob and P. van Eck. "Requirements Traceability in Model-Driven Development: Applying Model and Transformation Conformance," *Inf.Sys.Fron.*, 2007. pp. 327-342.

[8] A. van Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective," *Proc., 22nd Int. Conf. on Soft. Eng.*, 2000. pp. 1-15.

[9] E. Yu, P. Giorgini, N. Maiden and J. Mylopoulos, *Social Modeling for Reqs. Eng.*, The MIT Press. 2011.

[10] J. Mylopoulos, L. Chung and B. Nixon, "Representing and Using Nonfunctional Requirements: a Process-Oriented Approach," *IEEE Trans. on Soft. Eng.*, 1992. pp. 483-497.

[11] L. Xing and S. V. Amari, "Fault Tree Analysis," *Handbook of Performability Engineering*, Springer London, 2008.pp.595-620.

[12] K. Ishikawa, *Guide to Quality Control*. No. TS156. I3713, 1982.

[13] S. Supakkul and L. Chung. "Extending Problem Frames to Deal with Stakeholder Problems: An Agent- and Goal-Oriented Approach," *Proc., ACM Sym. on App. Com.*, 2009. pp. 389-394.

[14] http://www.utdallas.edu/~chung/RE/syllabus.htm

[15] T. Clancy, "The Standish Group Report," *Chaos Report*, 2014.