Automated Software Tool Support for Checking the Inconsistency of Requirements

Massila Kamalrudin
Department of Electrical and Computer Engineering,
University of Auckland,
University of Auckland, Private Bag
92019, Auckland, New Zealand
mkam032@aucklanduni.ac.nz

Abstract—**Handling inconsistency in software requirements is a complicated task which has attracted the interest of many groups of researchers. Formal and semi-formal specifications often have inconsistencies in the depicted requirements that need to be managed and resolved. This is particularly challenging when refining informal to formalized requirements. We propose an automated tool with traceability and consistency checking techniques to support analysis of requirements and traceability between different representations: textual, visual, informal and formal.**

Keywords-Inconsistency management, Requirements Engineering, Traceability, textual and visual requirements representations

I. INTRODUCTION

A Requirement is interpreted as a specification that needs to be implemented during system development [1]. It describes "how the system should behave, constraints on the system's application domain information, constraints on the system operation or specification of a system property or attribute" [1]. Software requirement specifications elaborate the functional and non-functional requirements, design artifacts, business processes and other aspects of a software system. Software requirement specifications that are complete and accepted by developers and clients provide a shared understanding and agreement of what a software system should do and why. Since requirements documents form the basis of development processes and this agreement, they should be correct, complete, and unambiguous [2] and need to be analyzed with respect to Consistency, Completeness and Correctness ("3 Cs") to detect errors such as inconsistency and incompleteness. However Zowghi and Gervasi note that "improving the consistency of requirements can reduce completeness and, thereby again diminishing correctness"[4].

In our work consistency is our key focus in order to ensure that models of requirements are entirely precise and fulfill the needs of a user. In order to make sure requirements are consistent and follow the customers' needs from the beginning we want to apply rigorous consistency checking from early stages of the Requirement Engineering process. We want to support the use of informal natural language requirements and more formalized representations but allow these to be readily related, updated and kept consistent. To this end we are designing and prototyping a tool to support refinement of informal requirements to a set of semi-formal models; analysis of these models for consistency; traceability between informal and semi-formal models; and consistency management between these models.

II. BACKGROUND AND MOTIVATION

The major disadvantages of specifying requirements only in Natural language "are inherent imprecision, such as ambiguity, incompleteness and inaccuracy" [2]. It has also been found that they are often error-prone and this is partially caused by interpretation problems due to the use of Natural language itself [3]. Although the aim of object-oriented analysis e.g. using (semi-)formalized models like UML or formal models like KAOS is to have a better requirement specification, most of the requirements documentation or specification of a software system is still often written in – or at least derived from - free text expressed in Natural language. This is often vague, informal and contradictory and may or may not express the users' needs. Much research has been devoted to the checking of inconsistency of requirements in a formal or semi-formal model. For example, XLinkit uses first order logic, object-Z specifications and utilizes tests of the specification, model abstraction and model checking for their verification. A "formal reasoning approach including the goal elaboration, ordered abduction and morphing of path" [8] is applied together with the use of knowledge base and rule base approach in detecting the inconsistency. Key limitations of using formal specification are the users needing to have deep understanding of the formal modelling language or continually have the formal specification explained to them. Users can not usually directly modify the specifications. Additionally, some of the algorithms "check only the self consistency of each class of a specification which does not guarantee the consistency of a specification"[7].

Much of research has been done using semi-formal specifications especially UML diagrams. Tools like VERIDEV [9], BVUML [5], CDET [10] and VIEWINTEGRA [11] are examples that check requirements consistency using semi formal specifications. They verify the consistency between the user requirements specification and class diagram, or verify consistency between user requirement specification and sequence diagrams. Some verify consistency between sequence diagrams with use

1527-1366/09 $29.00 © 2009 IEEE
DOI 10.1109/ASE.2009.38

cases or state diagrams. Less work has been done in checking the consistency between scenario and textual descriptions of requirements. There is also almost no work done in checking consistency or inconsistency using essential use cases [12]. One of the advantages of models is that they could be broken into smaller parts to allow them to be understood better [6] and this allows the consistency checking process to be easier but maintaining consistency between vastly different models is potentially very difficult and of high cost.

III. APPROACH

Our aim is to better support users and developers to work with informal and semi-formal requirements and keep them consistent. We aim to produce an automated prototype tool providing authoring facilities for textual requirements and checking the inconsistency of these requirements. This tool will assist requirement engineers and business analysts to check whether their requirements that are written or collected in natural language are consistent with other analysis and design representations. We have chosen to use essential use case modeling [16] and high level user interface design as our semi-formal models. This was due to their appeal as representations that developers and end users could work with and the limited research done to date investigating consistency issues with these representations and natural language requirements [12]. It was also to allow us to do complementary work on requirements quality and completeness improvement using characteristics of the essential use case model.

In order to support this concept we need a traceability technique so that the elements of natural language requirements and essential use case requirements can be traced between each other. We believe that supporting this traceability will enable us to better detect and manage inter-specification inconsistencies and also enable developers and users to work more effectively with different models of requirements. We will embed our consistency management and tracing tool within the Eclipse-based Marama [15] meta-tool environment. We plan to support traceability and consistency management with essential use case views, essential use case-based user interface designs, and conventional UML use cases and class diagrams.

To support requirements analysis in order to improve requirements completeness and quality, complementary work will be done in collection and categorization of terminology from different case studies and scenarios. This will provide a set of essential use case interactions and essential use case patterns to assist engineers in finding appropriate abstract interactions for designing the essential use cases for a system.

Figure 1 shows an example of our proposed approach. Grey areas show elements of the work done to date. Natural language requirements (1) are analyzed using a database of essential use case interactions (2) and essential use case models are generated (3). The user may select items in the essential use case and see the originating natural language elements (4). The user, a requirements engineer or end user, may also change elements in the essential use case model or natural language model and see the impact on the other model (4). An analysis tool (5) will use a set of essential use case patterns to determine if an extracted essential use case model is complete, consistent and correct according to acceptable patterns of essential use case interactions in the essential use case pattern library. Further extractors (6) will allow UML use cases, scenarios of use case usage (7), and essential use case-based high level form designs (8) to be derived from the essential use case requirements model OR the essential use case model to be derived or augmented from these other requirements models. Support for traceability and inter-model change management will be done in a similar way as between natural language and essential use case models (4). Conventional techniques to derive OOA/D models from derived use cases or vice-versa will be incorporated (9).

IV. METHODOLOGY

We are using an iterative approach to our work, adding additional extraction, consistency management, traceability and analysis components after evaluation of each stage of the research. An outline of our key steps is shown below. We have:

- Conducted a literature review of consistency and inconsistency checking of requirements in the Requirement engineering domain, compared and evaluated their approaches in checking the inconsistency of requirements;
- Identified from this an initial concept, outlined above, of how to support the checking of requirements inconsistencies, traceability and aspects of completeness and correctness;
- Collected and categorized the natural language terminology which follows the pattern of essential use case from different case studies and scenarios and produced a database of key abstract interactions;
- Developed an initial automated prototype to explore the problems and issues extracting essential use cases and tracing between textual requirements by using our database of abstract interactions;
- Developed a set of consistency rules between the textual requirements and the essential use case model of requirements;
- Identified appropriate usage scenarios and evaluated the result of using our consistency management and tracing tool if changes are made to the requirements;
- Developed an initial prototype of our automated inconsistency checking tool by embedding the tracing tool in Marama and connecting it to Marama Essential Use Case and User interface design tools;
- Evaluated the automated consistency checking tool by using case studies and scenario examples;
- Planned the refinement of our prototype by adding further analysis support for requirements quality checking using essential use case patterns; adding further inconsistency management and traceability support features; and eventually adding traceability

and consistency management support to more requirements and design models.

To date we have prototyped steps (1) to (4) in Figure 1 i.e. developed a database of essential use case interactions; supporting extraction of essential use case requirements models from natural language; and supporting traceability between natural language and essential use case requirements models. Figure 2 shows some voter registration system requirements used as an example for our initial tracing tool.

From the diagram, the requirements are extracted and traced by the extraction engine to provide the list of abstract interaction. The textual requirement is extracted by selecting and comparing particular phrases with the abstract interaction database. The list of abstract interaction also can trace back to the original requirements. This initial prototype has proven that traceability management is needed in supporting the traceability between natural language and essential use case requirements.

V. RESEARCH RESULTS AND PROGRESS

Based on our initial design an automated prototype tracing tool was developed using Java. A collection of essential use case interactions is stored in a database. The database consists of phrases describing abstract interactions to be identified and they are extracted from the natural language requirements. The extracted phrases are compared with the stored abstract interaction terminology in the database. The abstract interaction terminology is gained from a collection of phrase patterns from various scenario domains. The tracing engine is divided into two categories, trace and trace back engine, as shown in Figure 3. The next stage of this research is in progress: integrating with the Marama platform in Eclipse and with Marama Essential Use Case and User Interface Design tools. Any change or modification of a Marama Essential Use case or textual editor is expected to be traceable and the consistency issue between both is to be evaluated. The completeness and correctness in the Essential Use case diagram and textual editor is checked in order to confirm the consistency.

VI. RESEARCH IMPLICATIONS

Evaluation of our initial prototype shows that it is indeed useful to check the traceability between the natural language requirements with an extracted list of abstract interactions. This provides a basic process enabling users and developers to trace requirements elements and modifications which may lead to inconsistencies. Much research shows that traceability is difficult [13] and the number of current supporting tools is limited. Even with the existence of automated traceability and consistency checking tools engineers may still not be able to foresee the results [14]. With our initial work we have shown that the problem might be addressed via an integrated toolset supporting both traceability and consistency management between diverse requirements models. The essential use case abstract interaction list assists engineers in finding the right interaction patterns for designing essential use case and the user interface later. We also plan to use an essential use case pattern library to assist engineers and users in identifying correctness and completeness issues with these requirements. The contribution of our tool is in minimizing the time engineers spend developing consistent interaction models and in providing a framework in which users and developers agree on interaction terminology. Furthermore, our research involves the collection and categorization of terminology for a database of abstract interactions and interaction patterns based on the essential use case model of requirements which assists avoiding textual requirements being vague and error-prone.

ACKNOWLEDGMENT

This PhD research is funded by Ministry of Higher Education Malaysia (KPT) and PReSS Account of University of Auckland. A Special acknowledgement is given to the author's supervisors, Prof John Grundy and John Hosking for their guidance and support throughout this research.

REFERENCES

[1] Gerald Kotonya, I.S., Requirement Engineering Process and Techniques, ed. P.P.W. Professor David Barron. 1998, West Sussex, England: John Wiley & Sons Ltd. 282.

[2] Denger, C., D.M. Berry, and E. Kamsties, Higher Quality Requirements Specifications through Natural Language Patterns, in Proceedings of the IEEE International Conference on Software-Science, Technology \& Engineering. 2003, IEEE Computer Society. p. 80 %@ 0-7695-2047-2.

[3] Fabbrini, F., et al. The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. in Software Engineering Workshop, 2001. Proceedings. 26th Annual NASA Goddard. 2001.

[4] Zowghi, D. and V. Gervasi, *On the interplay between consistency, completeness, and correctness in requirements evolution.* Information and Software Technology, 2003. 45(14): p. 993-1009 %U

[5] Litvak, B., S. Tyszberowicz, and A. Yehudai, *Behavioral consistency validation of UML diagrams*, in Software Engineering and Formal Methods, 2003.Proceedings. First International Conference on. 2003. p. 118-125.

[6] Egyed, A., *Consistent Adaptation and Evolution of Class Diagrams during Refinement in Fundamental Approaches to Software Engineering*. 2004, Springer Berlin / Heidelberg.

[7] Fathi, T., K.D. Jacob, and A. Fouad Mohammed, *On checking the consistency of Object-Z classes.* SIGSOFT Softw. Eng. Notes, 2007. 32(4): p. 11.

[8] Kozlenkov, A. and A. Zisman, *Are their Design Specifications Consistent with our Requirements?*, in Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering. 2002,IEEE Computer Society. p. 145-156 %@ 0-7695-1465-0.

[9] Do Do, K. *Method and Implementation for Consistency Verification of DEVS Model against User Requirement*. in Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on. 2008.

[10] Scheffczyk, J., et al., Pragmatic consistency management in industrial requirements specifications, in Software Engineering and Formal Methods, 2005. SEFM 2005. Third IEEE International Conference on. 2005. p. 272-281.

[11] Egyed, A., Scalable Consistency Checking Between Diagrams-The ViewIntegra Approach, in Proceedings of the 16th IEEE international conference on Automated software engineering. 2001, IEEE Computer Society. p. 387.

[12] Biddle, R., J. Noble, and E. Tempero, *Essential use cases and responsibility in object-oriented development.* Aust. Comput. Sci. Commun., 2002. 24(1): p. 7-16.

[13] ÄÄLINOJA Juho, O.M., Software requirements implementation and management. Software & systems engineering and their applications 2004 vol. vol.1 à 3, : p. pp. 1.1-1.8Note(s)

[14] Egyed, A., Supporting Software Understanding with Automated Requirements Traceability International Journal of Software Engineering and Knowledge Engineering, 1994. 0, No. 0 (1994) 000–000.

[15] Grundy, J.C., Hosking, J.G. Huh, J. and Li, N. Marama: an Eclipse meta-toolset for generating multi-view environments, Formal demonstration paper, 2008 IEEE/ACM International Conference on Software Engineering, Liepzig, Germany, May 2008, ACM Press.

[16] Larry, L.C. and A.D.L. Lucy, *Structure and style in use cases for user interface design*, in Object modeling and user interface design: designing interactive systems. 2001, Addison-Wesley Longman Publishing Co., Inc. p. 245-279.

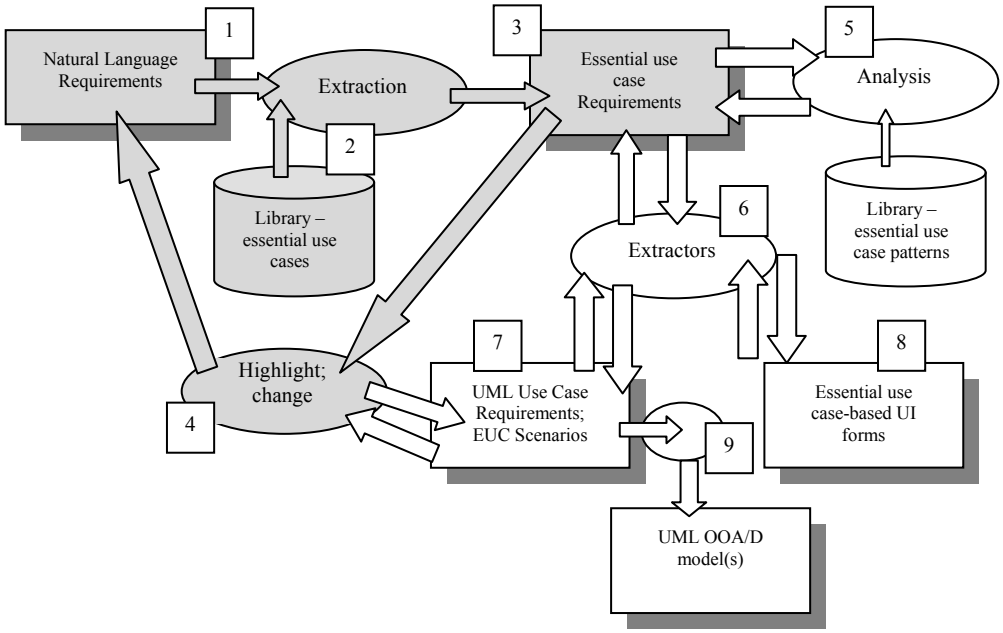

Figure 1. Overview of our requirements consistency and traceability management approach.

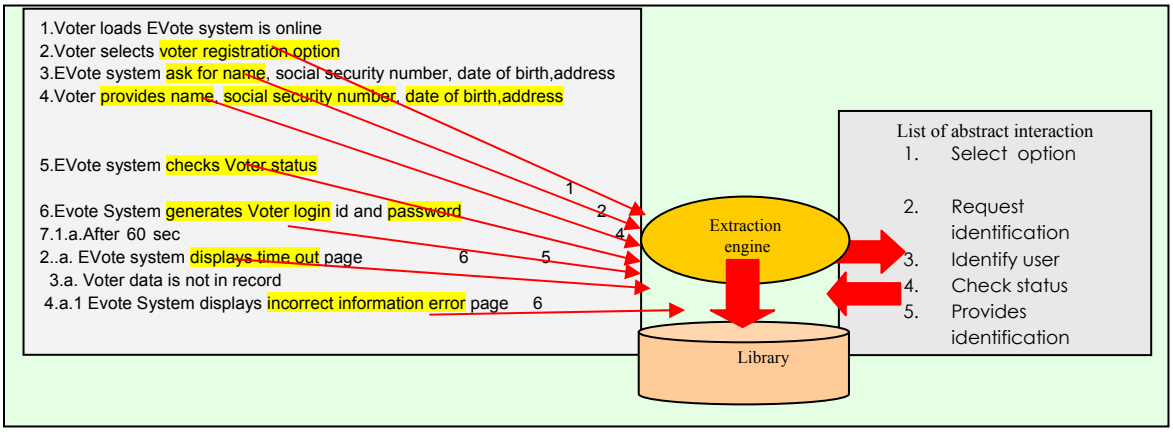

Figure 2. Initial support of extracting essential use cases and tracing between natural language and essential use case requirements models.

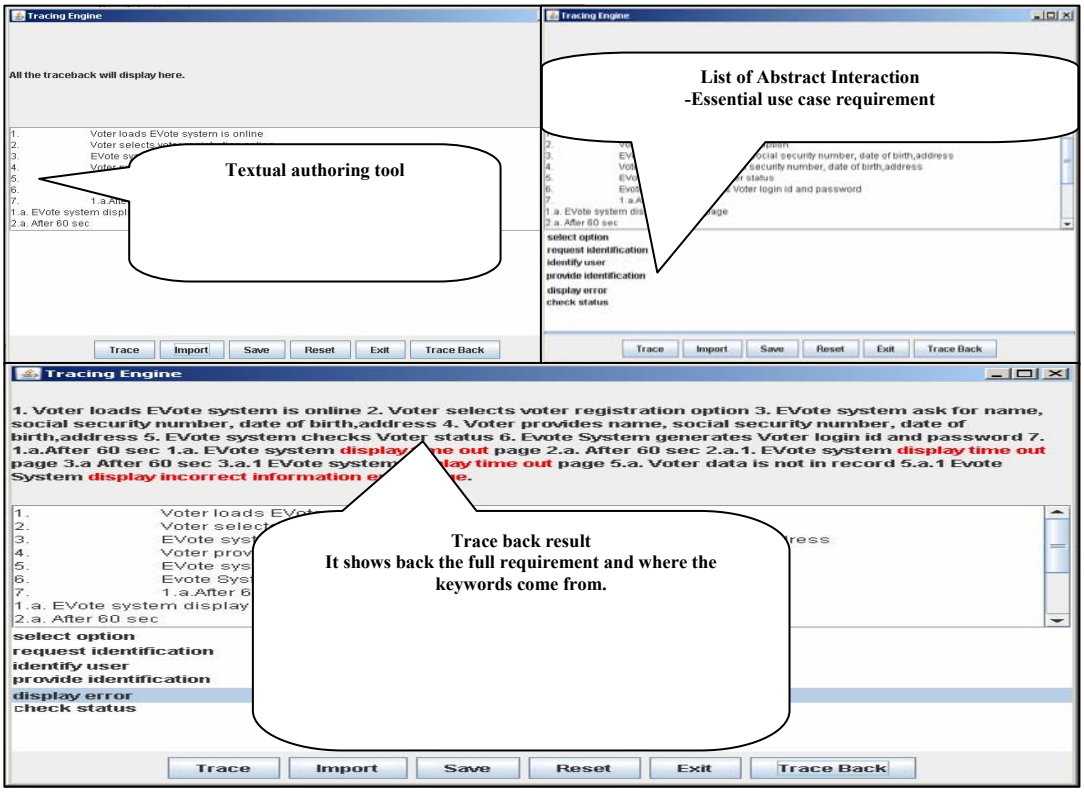

Figure 3. The text authoring tool and Trace and Trace Back Functionality.