Marama AI: Automated and Visual Approach for Inconsistency Checking of Requirements

Massila Kamalrudin
Department of Electrical & Computer Engineering
University of Auckland
Private Bag 92019 Auckland 1142
New Zealand
Mkam032@aucklanduni.ac.nz

John Hosking
Dept of Computer Science,
University of Auckland
Private bag 92019 Auckland 1142
New Zealand
john@cs.auckland.ac.nz

John Grundy
Swinburne University of Technology
Centre for Complex Software, Systems & Services, PO Box 218, Hawthorn
Victoria 3122, Australia
jgrundy@swin.edu.au

***Abstract*— Requirements are commonly vague and ambiguous. In this paper, we describe an automated Inconsistency Checker called Marama AI for checking for high- level inconsistency between textual requirements, abstract interactions and Essential Use Cases. We use concepts of phrase extraction and essential interaction patterns to carry out these checks. We provide further support for checking of requirements quality attributes such as completeness and correctness using visual differencing.**

Keywords-higher level inconsistency, essential interaction pattern, visual differencing

I. INTRODUCTION

It is acknowledged by many that natural language requirements are very often error prone, imprecise and ambiguous [1], [2]. In order to deal with these issues much research has been aimed at checking of requirements consistency, completeness and correctness either by using heuristic algorithms and formal models [3],[4] or semi-formal models [5],[6]. In our previous work [7, 8], we have introduced the MaramaAI tool for capturing requirements from natural language descriptions and helping to manage requirements inconsistency. The tool captures the essential requirements in the form of abstract interactions from natural language requirements and then transforms it to a semi-formal representation called Essential Use Cases. Besides capturing requirements, it also triggers inconsistency warnings if any inconsistency occurs between the textual requirement, abstract interaction and Essential Use Cases. However, triggering simple inconsistency warnings between these requirement elements is not adequate to make sure the requirements are completely consistent.

II. OUR WORK

Based on the motivations found from previous work, we have enhanced our tool in two key ways. Firstly, to have higher level inconsistency checking of the requirements with the essential interaction pattern library together with the visual help for requirements engineers. This means that each requirements component will be checked for its consistency with an essential interaction pattern library if any changes such as delete, add and change ordering are done. An essential interaction pattern is a sequence of expected essential interactions between user and system [8]. We have developed a library of such patterns to support interaction extraction from natural language [8] and analysis of interaction sequences. Figure 1 shows how the higher level inconsistency checking is performed in MaramaAI when the requirements engineer adds a new essential requirement (abstract interaction). If a new abstract interaction is added, the tool will automatically update the textual requirement based on the correct interaction pattern because the new added abstract interaction is inconsistent with the textual requirement and the interaction pattern library.

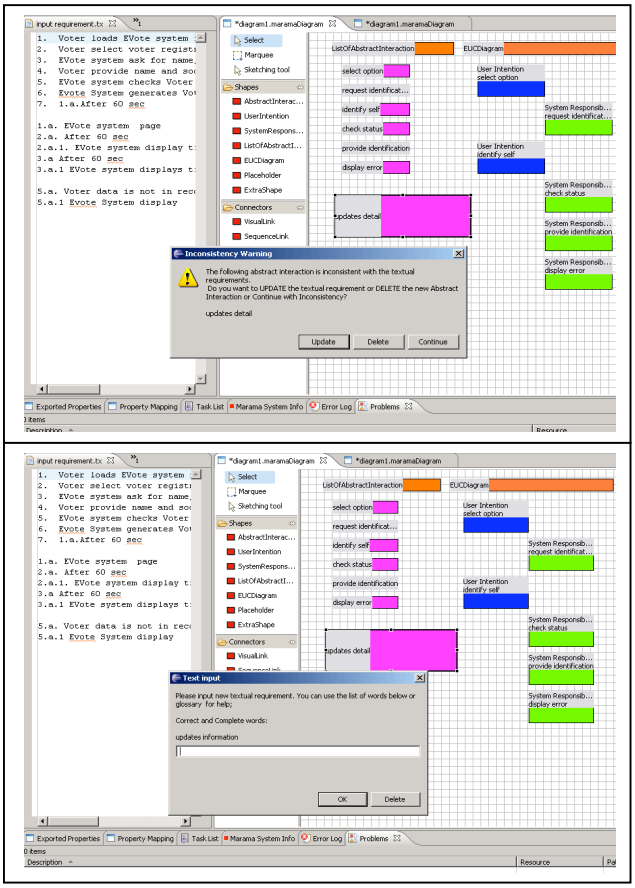

Figure 1. Add New Essential requirement

However, the tool also provides flexibility by allowing user to ignore the addition if they think the addition is needed in the requirements. The inconsistency can be tracked by MaramaAI and later resolved by the requirements engineer. Figure 3. shows inconsistency checking when the ordering of interactions has been changed. The related component change color to red and the textual requirement is highlighted (***) in order to show the user the affected requirement component from the modification. The problem marker will also shows the warning if change is made as the inconsistency will still exist in the textual requirement.

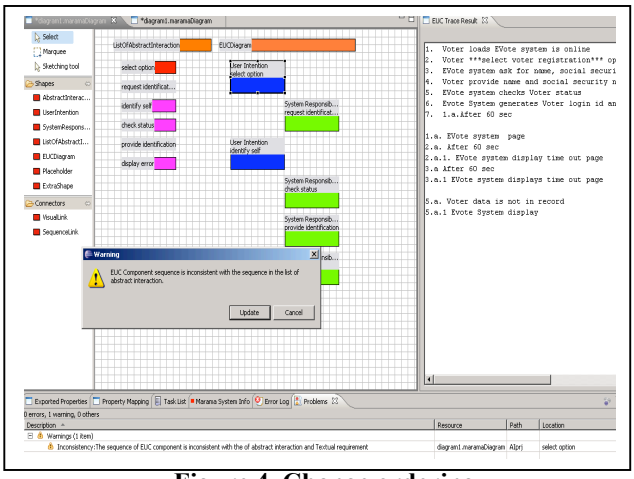

Figure 4. Change ordering

The second part of the tool is to allow requirements engineers to check for requirements completeness and correctness as shown in Figure 4. Users are able to check for the completeness and correctness of the requirements captured by checking the consistency between the modeled Essential Use Cases diagram with the EUC templates that exist in the interaction pattern library. A visual differencing is performed in order to show the difference between the modeled Essential Use Cases and the interaction pattern template. If any parts of the Essential Use cases are missing, extra or in incorrect ordering, the tool will visualize the incompleteness and incorrectness. Users are then able to choose either that they want to keep their requirements as modeled or change their model to follow the suggested template.

III. CONCLUSION AND FUTURE WORK

We have developed an automated inconsistency checker for checking for higher level inconsistencies between the requirements component and essential interaction patterns. Requirements quality checking such as completeness and correctness is also supported by using visual differencing against an interaction pattern library. Key future work is to conduct further evaluation of the tool in term of cognitive dimensions and usability and support consistency management with other models.

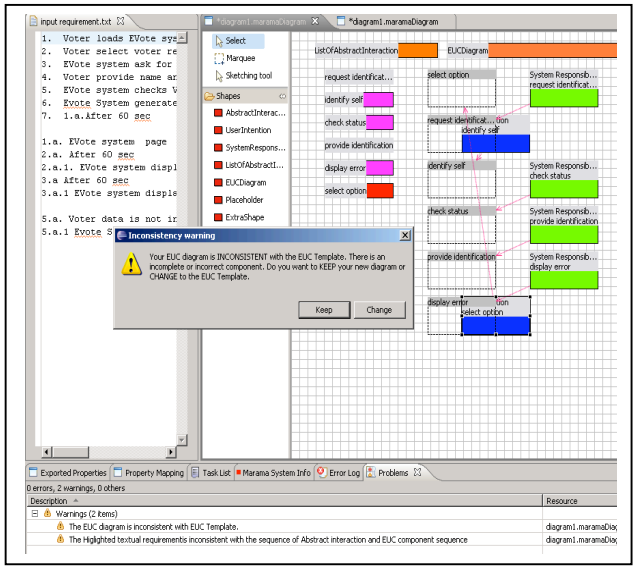

Figure 3. Visual Differencing for completeness and correctness checking

ACKNOWLEDGEMENT

This research is funded by Ministry of Higher Education Malaysia (MOHE) and University of Auckland.

REFERENCES

[1] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "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, pp. 97-105.

[2] C. Denger, 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*: IEEE Computer Society, 2003, pp. 80.

[3] C. Nentwich, W. Emmerich, A. Finkelstein, and E. Ellmer, "Flexible consistency checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 12, pp. 28-63, 2003.

[4] M. Ouimet and K. Lundqvist, "Automated Verification of Completeness and Consistency of Abstract State Machine Specifications using a SAT Solver," *Electronic Notes in Theoretical Computer Science*, vol. 190, pp. 85-97, 2007.

[5] A. Egyed, "Instant consistency checking for the UML," in *Proceedings of the 28th international conference on Software engineering* Shanghai, China: ACM, 2006, pp. 381-390

[6] F. P. Richard, J. B. Phillip, and S. O. Jonathan, "Metamodel-based model conformance and multiview consistency checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, p. 11, 2007.

[7] M.Kamalrudin, J. Hosking, John Grundy, "MaramaAI: Tool support for capturing requirement and checking the inconsistency " in *21st Australian Software Engineering Conference*, Auckland, New Zealand, 2010, pp. 145-149.

[8] M.Kamalrudin, J.Grundy, J.Hosking, , "Managing consistency between textual requirements, abstract interactions and Essential Use Cases," in *34th Annual IEEE International Computer Software& Applications* seoul, Korea, 2010.