R3ST for Requirements Recovery of Legacy Runtime Code

Eko K. Budiardjo, Elviawaty M. Zamzami, and Wahyudianto, *Member, IACSIT*

Abstract—In reality, we often find that proven and workable software, exist without source code. Identifying end-to-end interaction within existing legacy runtime code is a promising way for requirements recovery. Those identified interaction forms a set of use cases. Based on use case modeling concept, ontology is created for each finding interaction in order to check for completeness and consistency. OWL serves as platform for ontology development. That ontology reflects the recovered requirements. The procedures to identify the end-to-end interaction has been formulated and tested for university information systems. R3ST, tightly couple to Protégé has been developed with the objective to assist and automate those procedures.

Index Terms—End-to-end interaction, use case, ontology, requirements recovery, software comprehension, software maintenance.

I. INTRODUCTION

In software engineering, being able to define / construct Software Requirements Specification (SRS) is one of key success factors. IEEE Standard Glossary of Software Engineering Terminology defined that “requirement is a condition or capability needed by a user to solve a problem or achieve an objective” [1]. Ideally, requirements are documented in SRS document. Without information about those requirements, one can have a problem in software maintenance and further development iteration. Engineers do not know what must be fulfilled by the software [2].

In reality, we often found that proven and workable software, exist without source code. In this case often found in legacy application, it works well, but hard to maintain because there is no more source code exist. Recovering the requirements from existing software or legacy runtime code is a relevant effort for maintenance effort, especially in reengineering such application.

Recovering the requirements of existing application software can be obtained from several sources, i.e. stakeholders, developers, source code, software documents, etc. If existing application software is the only one available, then observing the existing software by executing is the way to recover the software features [3]. On a legacy runtime code, we can observe software features. These features are

functional requirements that are implemented in the existing software [4], as recovery process result.

The features present some software characteristics and it can be expressed in more detailed requirements. If these software characteristics are captured as source of requirements recovery, may potentially result in some differences between recovered requirements and implemented requirements in existing application software. So, we will capture not only the features or but also trace the software behavior in term of interaction between user and software in detail. In this work, we use a term to describe interaction between user (actor) and system, called as end-to-end interaction.

II. END-TO-END INTERACTION ONTOLOGY

Forward engineering, as seen on Fig. 1, consist of (a) Requirements engineering activity in which results a software requirements specification (SRS) documents with the key ingredients uses case model and its specification; (b) Analysis and Design activities in which produce software architecture document that is consisting of use case realization; and finally (c) Construction which produce code, test plan, test case, user manual.

Fig. 1. Forward vs reverse engineering.

In our previous work, we focus on reverse engineering path (Fig. 1). We developed an ontology model that represents end-to-end interaction [5]. This is an interaction between user and software to achieve a certain goal. In end-to-end interaction, we use terms action and response. The user does something to interact with the software in which is called as an action. The software does something to react that is called as response. Thus, end-to-end interaction consists of user actions to the software and some responses that caused by the user actions. The end-to-end interaction can be illustrated as in Fig. 2.

These actions include the user commands, create, read,

Manuscript received February 22, 2014; revised April 24, 2014. This paper results from R3ST Project. It is a three years research project, which has been funded by the Ministry for Education and Culture, Republic of Indonesia, under BOPTN 2013 grant schema.

E. K. Budiardjo and Wahyudianto are with the Faculty of Computer Science, University of Indonesia, Depok, Indonesia (e-mail: eko@cs.ui.ac.id, wahyudianto@ui.ac.id).

E. M. Zamzami is with Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia (e-mail: elvi_zamzami@usu.ac.id).

update, delete, click, link, point, and push. The responses are display, print, receive, save, and load. The user runs the software to perform end-to-end interactions.

Fig. 2. End-to-end interaction.

End-to-end interaction ontology merges those two ontologies, WIMP-UI and USI ontology [5]. WIMP-UI (Window Icon Menu Pointer – User Interface) ontology represents user interface that involved in interaction between user and software. USI (User Software Interaction) ontology represents interaction between user and software. Thus, end-to-end interaction ontology represents the end-to-end interactions between user and software for achieving a certain goal through WIMP interface. The benefit of having end-to-end interaction ontology is give us a better understanding of software behavior from existing software that has no source code. This implies that for each software has their own / specific end-to-end interaction ontology.

III. R3ST ARCHITECTURE AND ENVIRONMENT

Requirement Recovery and Reconstruction Software Tools (R3ST) is our current research project to assist and automate end-to-end interaction procedures. The main purpose of R3ST is assisting the software comprehension and requirements reconstruction activities, via capturing end-to-end interaction between user and the existing software.

People-Procedures-Tools are three important elements to support software comprehension and requirement reconstruction activities as seen on Fig. 3. Skilled and well train engineer have to master the Requirements Recovery and Reconstruction (R3) method as the foundation. R3 method describes more detailed on [6], as the result of our previous work. This method has set of procedures as companion, in which copy righted under the Indonesian law. The engineers on duty will apply the R3 method and procedures assisted by the R3ST.

As seen on Fig. 5, R3ST consist of four components a) End-to-End Interaction Capturing Tools; b) Ontology Development Tools; c) Software Comprehension Tools; and d) Requirements Reconstruction Tools. The first two components aim to supports end-to-end interaction ontology development, coined as Requirements Recovery. The data obtained from those captured is processed by applying the use case modeling concept in order to generate its ontology. Therefore, the use of ontology becomes an important point of R3ST development itself. The R3ST users may exploit the ontology for the purpose of software comprehension and/or requirements reconstruction.

In the ontology world, there are some tools supporting the development of ontology. Protégé developed by Stanford University, is one of proven resilient tool and it is chosen as

R3ST ontology platform. Protégé as an ontology tools, is also providing OWL-API. A service layer interface makes ontology development in programmatically. Protégé has also provided a web-based environment in addition to the desktop version, which is named Web Protégé to provide a convenience to the user without having to install the application and accessible online via the internet. It is the background of our research project to develop R3ST by utilizing Web Protégé which is already proven in the ontology domain, as the main foundation in the generation of ontology. In other word to say that the main goal of R3ST development can be achieved, which can be used for software comprehension which can gives us a better understanding of software behavior and/or requirements reconstruction,

Fig. 3. R3ST as a unity system.

The end-to-end interaction capturing process is supported by three tool features, which are Mentor, Capture, and Linker (Fig. 4). R3ST Mentor is featuring to assist and give a guide step-by-step process to capture the end-to-end interaction. R3ST Mentor can be used by the user of R3ST to stay on the right track, by follow the step-by-step guide.

Fig. 4. R3ST structure as requirements capturing tool.

R3ST Capture, coined for capturing assistant, is the first step to use this tool. These features will assist one in capturing the end-to-end interaction between user and software. The interaction capture via application user interface component and then store or record the captured data in database. Once the data is stored, then the data will be used by the R3ST Linker. R3ST Linker has the main task is to carry out the development of an ontology of captured data, based on the implementation of use case modeling concept.

R3ST Linker is the bridge of end-to-end interaction captured data and Web Protégé is an ontology tools that have OWL as the main platform used by R3ST to develop the ontology.

Fig. 5. R3ST building blocks.

R3ST is intended to extend the original R3 capturing procedure by adding R3ST Protégé Ontology Framework on the Web Protégé Development is performed on the client side and the server, as seen on Fig. 6. The one that performed on the client side user interface is developed as an adjustment to R3ST user interface and R3ST model, as an adjustment to the model used by R3ST ontology. While on the server side, the development is performed for R3ST Ontology Framework that used as an engine to create ontology based on use case modeling concept, to provide a better understanding of the behavior of software, through visualization and analysis on the ontology

Fig. 6. R3ST architecture on top of web protégé

Fig. 7. R3ST System Infrastructure.

At this time R3ST development, at the end of the first of three years project, results a prototype system, which is serve as the basis for a unified R3ST system, including the infrastructure to facilitate remote access by utilizing R3ST cloud services (see Fig. 7). With this service is expected to facilitate numbers of mobile devices, so that increase R3ST users. R3ST supports group of software engineer work together on remote site. In this situation, a R3ST remote controller serves as local identified end-to-end interactions collector. At the end of a session, the controller will consolidate all identified end-to-end interaction and submit to R3ST sever.

The prototype of R3ST main page for End-to-End Interaction Capturing Tools, the Capturing Assistant, can be seen in Fig. 8.

Fig. 8. End-to-end interaction capturing tools.

IV. SOFTWARE COMPREHENSION AND TOOL

Some related work on software comprehension has been conducted by some other researchers. Runtime code based software comprehension intends to obtain an understanding while the software is running. While study of runtime code based, we found that several approaches is related with this work. The used of runtime program to detect feature dependency is proposed in [7], by tracking the object created at the runtime. It's focused on feature of the software, using dynamic analysis, and visualizes the object oriented software behavior. The different idea proposed in [8], by discussing behavioral profile to understanding the software behavior. This profile uses the UML profiler, and transforms execution traces into sequence diagram. This work have the same objective with [8], as most related paper by the other author, which is to understanding software behavior on runtime code based software, but it has different aspect used to. They use the profiling model, whereas R3ST focus on capturing end-to-end interaction and its ontology.

The ontology term in software comprehension context is commonly used to extracting, mapping, or linking between the software and domain knowledge or real world concepts. The research on this approach has long been recognizing, and most of them work over source code. One of the research has been done is a formal framework that capture relation between concept and program has been introduced [9]. The frameworks use a graph language that similar to RDF graph to represent the ontology. Other initiatives propose partial comprehension using ontology fragment and graph search to locate a concept on source code [10]. Ontology fragment approach can be used to understanding of very large software.

Our work is different used of ontology from two of ontology related paper. The used ontology on this work, is to capture the interactions between user and software, end then extract the interactions in ontology. By this ontology we can get a better understanding of software behavior.

Runtime code based software comprehension intends to understanding while the software is running. While study of runtime code based, we found that several approaches is related with this work. The used runtime program to detect feature dependency is proposed in [7], by tracking the object created at the runtime. It's focused on feature of the software, using dynamic analysis, and visualizes the object oriented software behavior. The different idea proposed in [8], by discussing behavioral profile to understanding software behavior. Behavioral profiles use the UML profiler, and transform execution traces into sequence diagram. This work have the same objective with [8], as most related paper by the other author, which is to understanding software behavior on runtime code based software, but it has different aspect used to. They use the profiling model, whereas we use the end-to-end interaction model.

The term ontology in software comprehension context is commonly used to extracting, mapping, or linking between the software and domain know ledges or real world concepts. The research on this approach has long been recognizing, and most of them work over source code. One of the research has been done is a formal framework that capture relation between concept and program has been introduced [9]. The frameworks use a graph language that similar to RDF graph to represent the ontology. Other initiatives are propose partial comprehension using ontology fragment and graph search to locating a concept on source code [10]. Ontology fragment approach can be used to understanding of very large software. Our work is different used of ontology from two of ontology related paper.

The used ontology on this work, is to capture the interactions between user and software, end then extract the interactions in ontology. By that ontology we can get a better understanding of software behavior. As shown in Fig. 9, it is an example of ontology visualization as part Software Comprehension Tools.

Fig. 9. An example of ontology visualization.

V. REQUIREMENTS RECONSTRUCTION AND TOOL

RMUC is abbreviation of Requirements Management with Use Case [11]. In this method, requirements management is represented using use case. Use case represents requirements with perspective an actor achieve its goal. It is an important part of UML (Unified Modeling Language) notation to explain external behavior of software. It is a sequence of transaction that produce a result that observable of value for an actor. Collection of use cases is complete functionality that has to be fulfilling by the system. With the use case, developers have a way to determine a general understanding with the end users and domain experts.

R2UC (Requirements Representation with Use Case) ontology, is introduced in [12]. It is ontology that contains software requirements that covered user, interaction through interface, and features. The features fulfill the stakeholder's needs as problem solution. The R2UC ontology can be used to documenting requirements that will be implemented in software engineering and forward engineering processes, also documenting requirements from existing software through reverse engineering processes. Using the R2UC ontology, software requirements are presented semi formally. The R2UC ontology can provide structure of Software Requirements Specification (SRS) that refers to Rational Unified Process (RUP) known as SRS document.

In contrast to [13] approach, referring to R2UC ontology [12] as part of R3 method [6], R3ST equipped with Requirements Reconstruction Tool in which consist of two modules: SRS Document Template profilers for customizing SRS document template as basis for Document Generator to generate the expected document. In this case, the template for RUP Requirements Management acts as default.

VI. CONCLUSION

The quality of end-to-end interaction recovery legacy runtime code serves as basis for software comprehension and requirement reconstruction. Recovery Procedure Mentor and Data Verifier modules are umbrella tools to guarantee the correctness of requirements recovery. From the experimenting on visualizing the end-to-end interaction ontology, based on concept of use cases, we found that is an effective way for software comprehension. The results of comprehension give feedback to the engineer on the goodness of requirement recovery. Ones it is make sense, the use Requirement Reconstruction tool become justified in order to produce recovered SRS document.

ACKNOWLEDGMENT

This paper results from R3ST Project. It is a three years research project, which has been funded by the Ministry for Education and Culture, Republic of Indonesia, under BOPTN 2013 grant schema.

REFERENCES

- [1] IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.121990. [Online]. Available: <http://ieeexplore.ieee.org/>
- [2] A. V. Lamsweerde, *Requirements Engineering, From System Goal to UML Models to Software Specification*, John Wiley & Sons Ltd, 2009.

- [3] S. Demeyer, S. Ducasse, and O. Nierstrasz. (2008). Object-Oriented Reengineering Pattern. [Online]. Available: <http://www.iam.unibe.ch/~scg/OORP/>
- [4] D. Kulak and E. Guiney, *Use Cases: Requirements in Context*, 2nd Edition, Addison Wesley, 2003.
- [5] E. M. Zamzami, E. K. Budiardjo, and H. Suhartanto, "Requirements recovery using ontology model for capturing end-to-end interaction of proven application software," *IJSEIA International Journal of Software Engineering and Its Applications*, SERSC, vol. 7, no. 6, p. 425, 2013.
- [6] E. M. Zamzami, E. K. Budiardjo, and H. Suhartanto, "Requirements Recovery and Reconstruction (R3) method from existing information systems," *IJACT International Journal of Advancements in Computing Technology*, AICIT, vol. 5, no. 12, pp. 55-64, 2013.
- [7] A. Lienhard, O. Greevy, and O. Nierstrasz, "Tracking objects to detect feature dependencies," in *Proc. the 15th IEEE International Conference on Program Comprehension*, IEEE, 2007.
- [8] J. Koskinen, M. Kettunen, and T. Systä "Profile-Based approach to support comprehension of software behavior," in *Proc. the 14th IEEE International Conference on Program Comprehension*, IEEE, 2006.
- [9] D. Ratiu, and F. Deissenboeck, "From reality to programs and (not quite) back again," in *Proc. the 15th IEEE International Conference on Program Comprehension*, IEEE, 2007.
- [10] M. Petrenko, V. Rajlich, and R. Vanciu, "Partial domain comprehension in software evolution and maintenance," in *Proc. the 16th IEEE International Conference on Program Comprehension*, IEEE, 2008.
- [11] D. Leffingwell *et al.*, *Managing Software Requirements: A Unified Approach*, USA: Addison-Wesley, 2000.
- [12] E. M. Zamzami and E. K. Budiardjo, "Documenting software requirements specification using R2UC ontology," *Journal of Computer Science and Engineering*, vol. 14, no. 2, August 2012.
- [13] S. A. Fahmi and H. J. Choi, "Software reverse engineering to requirements," in *Proc. International Conference on Convergence Information Technology*, 2007.

Eko K. Budiardjo is a senior member of IACSIT. He has been the faculty member of the Faculty of Computer Science - University of Indonesia since 1985. Teaching, research, and practical services are aligned; give result in a full spectrum of academic achievement. Majoring in Software Engineering as professional track record, he has made some scientific contribution such as Software Requirement Specification (SRS) patterns representation

method, R3 Method, ZEF Framework, and FrontCRM Framework. Graduated from Bandung Institute of Technology (ITB) in 1985, holds Master of Science in Computer Science from the University of New Brunswick – Canada in 1991, and awarded Philosophical Doctor in Computer Science from the University of Indonesia in 2007. Currently he is the Vice Chairman of ICT Technical Committee of The National Research Council (DRN), and Chairman of The Indonesian ICT Profession Society (IPKIN).

Elviawaty Muisa Zamzami graduated from Bandung Institute of Technology (Indonesia), magister of informatics, 2000 and awarded Doctoral in Computer Science from the University of Indonesia in 2013. She is a lecturer at Department of Computer Science, University of Sumatera Utara, Indonesia. Currently her research interests are reverse engineering, requirements recovery, and ontology.

Wahyudianto is a researcher and teaching assistant at the Faculty of Computer Science, University of Indonesia, Indonesia. He was born in 1987. his received bachelor degree in informatics engineering with software engineering major from Islamic State University Syarif Hidayatullah Jakarta, Indonesia in 2010, and held master of computer science from the University of Indonesia at 2014. He worked as a software developer from 2010 until 2011 for one project lifecycle, developing credit rating information system. Currently, he works as a freelance software engineer on web and mobile application. His interests include software engineering, software product line engineering, software verification and validation, automated software engineering, web and mobile engineering, and programming field.