Towards End-to-end Interaction Ontology Development for Software Comprehension

Eko K. Budiardjo¹, Elviawaty M. Zamzami², and Wahyudianto³
¹³ University of Indonesia, Faculty of Computer Science, Depok, Indonesia
² Universitas Sumatera Utara, Faculty of Computer Science and Information Technology, Medan, Indonesia
¹ eko@cs.ui.ac.id, ² elvi_zamzami@usu.ac.id, ³ wahyudianto@ui.ac.id

Abstract. Most of program comprehension works over source code. In reality we often found that proven and workable software, exist without source code. Identifying end-to-end interaction within existing software is a promising way for run-time software comprehension. 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 gives us a better understanding of software behavior. The procedures to identify the end-to-end interaction has been formulated and tested for university information systems.

Keywords: End-to-end Interaction, Use Case, Ontology, Software Comprehension, Requirements Recovery, Requirements Reconstruction.

1 Introduction

The concept of program comprehension is adopted by research community and it is beneficial for practitioners in software maintenance field. Researchers have contributed in methods, techniques, and tools development, which assist reverse engineering and reengineering task. Understanding the software is the objective of this field. Many approaches have been introduced in this field, and most of them work over source code [1]. The purpose of source code based approaches is used to understand the structure of code, which obtained then through an information retrieval [2] [3], or visualization [4], or machine learning [5], etc.

In the real world, we often found that some proven and workable software, exist without source code. For this case, we need a different approach to handle this runtime code. In order to comprehend some running, proven, and workable software, we can use runtime code based comprehension approach. With this runtime code, we can understand the software behavior. Several contribution on runtime code based approach, has been introduced, some of them are using object instance in runtime software [6], and profiling the runtime behavior of software [7].

This work proposes a promising idea to comprehend the runtime code by identify the end-to-end interaction within existing software. This idea is part of the authors work on research project to develop Requirements Recovery and Reconstruction.
Software Tools (R3ST) as tools to assist and automate the end-to-end interaction identification procedures. These tools can be used in software comprehension without required the source code, nor software architecture and design document, just the runtime code. R3ST based on ontology development using OWL as platform. The ontology created from each finding interaction in order to check for completeness and consistency. By using the ontology, we can have a better understanding of software behavior.

2 Related Work

This paper focused on two main issues: (a) runtime code based comprehension; and (b) software behavior understanding using ontology. This related work, covering those two aspects. Runtime code based software comprehension intents 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 [6], 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 [7], 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 [7], 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 knowledge or real world concepts. This approach has long been recognized, 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 [8]. 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 grep search to locating a concept on source code [9]. Ontology fragment approach can be used to understand very large software. Our work is different, in which 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. Once the ontology has been created, we can get a better software behavior understanding.

Beside those two related aspect, this work is extension of Requirement Recovery and Reconstruction (R3) method [10], that also propose end-to-end interaction procedure [11]. We use end-to-end interaction procedure for software comprehension by finding the flow of interactions between user and software, based on use case modeling concept, started with trigger made by user until some goal of software found by user. We propose to use and extend the end-to-end interaction [11], as most related paper by the author, to be used in software comprehension, with purpose that
we can better understand of the software using knowledge that provided in created ontology.

3 End-to-end Interaction Ontology

In our previous work, we developed an ontology that represents end-to-end interaction in the runtime code. *End-to-end interaction* 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 software is called as an *action*, and, the software does something that 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. 1.

![Fig. 1. End-to-end interaction.](image)

The *actions*, drive by the user, include user commands such as, *create, read, update, delete, click, link, point,* and *push*. Among others, the *responses are display, print, receive, save,* and *load*. In order to obtain the end-to-end interaction, the observer/engineer will run the software, step by step, and record those interactions.

End-to-end interaction ontology consist of merging of two ontologies, WIMP-UI (Window Icon Menu Pointer – User Interface) and USI (User Software Interaction) ontologies. WIMP-UI ontology represents user interface that is involved in interaction between user and software. USI ontology represents interaction between user and software. Thus, end-to-end interaction ontology represents an end-to-end interaction between user and software for achieving the certain goal through WIMP interface. One of the benefit, the end-to-end interaction ontology gives us a better understanding of software behavior from existing software that has no anything, except the running application software itself, for more detailed discuss on [10].

4 OWL Protégé Visualization as Software Comprehension Tool

In the ontology domain, there is a tool that has proven resilient in the development of ontology, i.e. Protégé, developed by Stanford University. As ontology tool, Protégé is providing OWL-API, a service interface that can be used to develop ontology in programmatically. Protégé has also provided a web-based environment in addition to
the desktop version, which is named as Web Protégé. In order to provide convenience to the user on having functionality without have to install the application so that it can be accessed online via the internet. The other feature which provided by Protégé is the OWL Visualization, which can very useful to get an overall picture of the ontology.

The end-to-end interaction that has been captured will be generated into ontology, based on use case modeling concept, using OWL Protégé. When the generation of the ontology has been performed, we can use the OWL Protégé Visualization to comprehending the software. The example of OWL Protégé Visualization of captured end-to-end interaction can be seen in Fig. 2.

![Fig. 2. User Authentication Ontology Using OWL Protégé Visualization.](image)

In the Fig. 2 as seen above, it is describe the use case User Authentication that written as “UC-1-User_authentication”. UC is the prefix of use case with 1 is the number of the use case. The UC-1-User_authentication has a several properties based on use case specification template that describe as follow.

1. Name: The use case name has prefix UCName, i.e. “UCName-1-User_authentication”.
2. Pre-condition: The pre-condition of the use case has prefix Pre, i.e. “Pre-Regular_user”.
3. Post-condition: The post-condition of the use case has prefixed Post, i.e. “Post-View_user_page”.
4. Basic Flow: The use case has a basic flow with prefix BF, i.e. “BF-1-1-Input_username”.
5. Alternate Flow: The use case has an alternate flow with prefix AF, i.e. “AF-1-3-1-1-View_wrong_message”.
6. Special Requirements: The use case has a special requirements with prefix SR, i.e. “SR-none”, in this case the use case does not has SR.
7. Extension Points: The use case has an extension points with prefix EP, i.e. “EP-none”, in this case the use case does not has EP.

The visualization of User Authentication end-to-end interaction ontology using OWL Protégé Visualization in Fig. 2, could give us another way to comprehending the software, and better understanding of software behavior.

5 Conclusion

From the experimenting on visualizing the end-to-end interaction ontology, based on use case modeling concept, we found that is an effective way for software comprehension as part of software maintenance activities. We expect feedback from the conference forum on discussion on the role of runtime program comprehension. This discussion will be in term of user computer interaction of existing software, in which covered comments, critics, and suggestions of OWL Protégé as software comprehension tool, on how to improve it in order to become completes tools for software comprehension, by creating ontology from end-to-end interaction of the existing software.

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

References