International Journal of Software Engineering and Its Applications
Vol.7, No.6 (2013), pp.425-434
http://dx.doi.org/10.14257/ijseia.2013.7.6.36

Requirements Recovery Using Ontology Model for Capturing End-to-End Interaction of Proven Application Software

Elviawaty M. Zamzami, Eko K. Budiardjo and Heru Suhartanto

Faculty of Computer Science, Universitas Indonesia,
Depok,16424, Indonesia
elviawaty.m@ui.ac.id, eko@cs.ui.ac.id, heru@cs.ui.ac.id

Abstract

An existing application software perhaps has no requirements document or requirements document does not represent the application. The situation creates a problem in application software maintenance or reengineering. Thus, requirements document should be reconstructed from the existing application software. Effort on reconstructing a requirements document from the existing application software is similar to requirements document construction for a new application software, needs requirements elicitation. Requirements elicitation can be done using an ontology model that captures end-to-end interaction between users and system. Using the ontology model does not need software documentation and source code at all, the only one is the existing application software itself. In this case, the existing application software should be executed and then observed its behavior in term of end-to-end interaction on each starting point in which the interface identified. This set of interaction flows is known as a Use Case Realization (UCR) in Requirements Management with Use Case (RMUC) approach. The appropriate ontology model provides an ease to understand the software requirements semantic meaning. The model consists of two models includes, WIMP-UI (Window Icon Menu Pointer – User Interface) and USI (User - Interaction) ontology models.

Keywords: *Requirements, Requirements Reconstruction, Requirements Elicitation, End-to-end interaction, WIMP-UI ontology, USI ontology*

1. Introduction

Requirements are one of key success factors in software engineering or software reengineering. 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" [5]. Ideally, requirements are documented in Software Requirements Specification (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. Thus, recovering the requirements from existing software is a relevant effort.

Recovering the requirements of existing application software can be obtained from several sources, *i.e.*, stakeholders, developers, source code, software documents, *etc*. If an existing application software is the only one available, then observing the existing software by executing is the way to recover the software feature [3]. While executing the software, we can observe software features. This features are functional requirements that is implemented in the existing software [6].

The features are as representation software characteristics and will be expressed as requirements in detail. If software characteristics are captured as source of requirements recovery, it may result in difference between recovered requirements and implemented requirements in existing application software. So, we will capture the features or software

ISSN: 1738-9984 IJSEIA

behavior and trace of interaction between the user and the software in detail. In this work, we use a term end-to-end interaction to describes interaction between user (actor) and system.

In this work, we propose an ontology model for capturing the end-to-end interactions. The ontology model is used because it is closely related to modern object-oriented software design that adopt software development for ontology development [10]. The ontology model provides an ease to understand semantic meaning of the end-to-end interaction that will be used in requirements recovery process. That is why we captured it using the ontology model. This ontology model will merge two ontology models, an ontology that models user interface and an ontology that models user interaction.

2. Sources of Requirements Recovery on Existing Application Software

There are several researches on existing application software where the researchers used various source of requirements recovery. These sources are as follows:

- **Source code**; Yijun Yu *et al.*, and Feng Chen *et al.*, use the source code as source of requirements. Yijun Yu *et al.*, [12] perform a research that propose refactoring to source code. The other research, Feng Chen *et al.*, [2] propose two ontologies for requirements recovery that one of them sourced from source code or program.
- **User interaction**; El-Ramly *et al.*, [4] perform a research that involves user interaction on requirements recovery. They propose the interaction-pattern mining.
- **Documents**; Variety of documents as source of requirements, *i.e.*, software requirements documents, user manual, or other documents that support the existing software. Rayson *et al.*, [9] perform semantic parsing, Liu *et al.*, [7] perform context analysis and Luiz and Gave [8] perform document and log analysis.

This work proposed end-to-end interaction as a source of requirements. El-Ramly *et al.*, [4] and our work uses interaction between user and software. However, there are few differences among these two works. El-Ramly *et al.*, focuses on sequences of screen snapshoots that used in user interaction, while we concentrate more on user's actions and software's responses. Role of user interface on our work is to describe about the actions and responses more clearly. We consider a condition where an existing application software have no software documents and source code, only software itself. Also, end-to-end interaction is an interaction of software feature in detail, where the features are the requirements type that implemented in software. Thus, end-to-end interaction is a use case realization and use case is a software requirements.

3. End-to-End Interaction

End-to-end interaction is an interaction between user (actor) and system (software) to achieve a certain goal. In end-to-end interaction, what the user does something to interact with system is called as an *action*. And, the system does something that called as *response*. Thus, end-to-end interaction consists of user actions to the system and some responses that caused by the user actions.

In end-to-end interaction, initiator of the interaction is an user action, ending on last software sequence response. The end-to-end interaction can be a round trip interaction, there may contain some actions and responses that take places alternately. End of an end-to-end interaction will show a goal. The end-to-end interaction can be illustrated as in Figure 1.

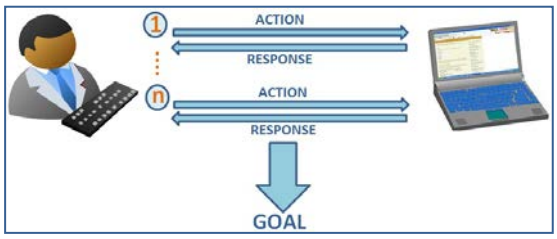

Figure 1. End-to-end Interaction

The actions in end-to-end interaction are the user commands to software. The user actions are *create*, *read*, *update*, *delete*, *click*, *link*, *point*, and *push*. The commands as software responses are *display*, *print*, *receive*, *save*, and *load*. The responses taken by the software are caused by user actions. For obtaining the end-to-end interaction is done by running the existing software.

4. End-to-end Interaction Ontology Model

This work proposes an ontology model for capturing end-to-end interaction as a source of requirements. In Zong-yong's paper conveys an ontology model that consists of 4 tuples [14]: $O = < C, R, H, X >$

where O = name ontology
C = concepts set or class
R = relation set
H = hierarchies set which stands for hyperspace consisting of concepts and relations.
X = Axiom set.

Our ontology model refers to their model and adds a set of restriction as a tuple, thus be:

$O = < C, P, R, H, X >$

where O = name ontology
C = concepts set or class
P = properties or relations set
R = restrictions set, *all values from* (∀), *some values from* (∃), *has value* (∋), *cardinality* (=), *min cardinality* (≥), and *max cardinality* (≤)
H = hierarchies set which stands for hyperspace consisting of concepts and relations.
X = Axiom set.

For example USI = $< C, P, R, H, X >$

where C = {Interaction, Action, Response}
P = {involve, take, consistof}
R = {some values from}
(Action, Push) ∈ H means Action is the superclass of Push.
Action and Response are disjoint class is an axiom.

The ontology model merges two ontology models, WIMP-UI and USI ontology models.

4.1. WIMP-UI Ontology Model

Interactions between user and software are done through the user interface. There are some user interfaces, *e.g.*, command line interface, natural language interface, and 3D

interface. In this work, we use WIMP (Window Icon Menu Pointer) interface which mostly used by the software users.

In previous work, we developed an ontology model that represents WIMP interface, called as WIMP-UI (Window Icon Menu Pointer – User Interface) ontology model [13]. The WIMP-UI ontology model can be seen on Figure 2. Using the WIMP-UI ontology model can be captured the semantic meaning of user interface that used by users of application software. Using the WIMP-UI ontology model, we can capture the semantic meaning on user interface presence that used in interaction.

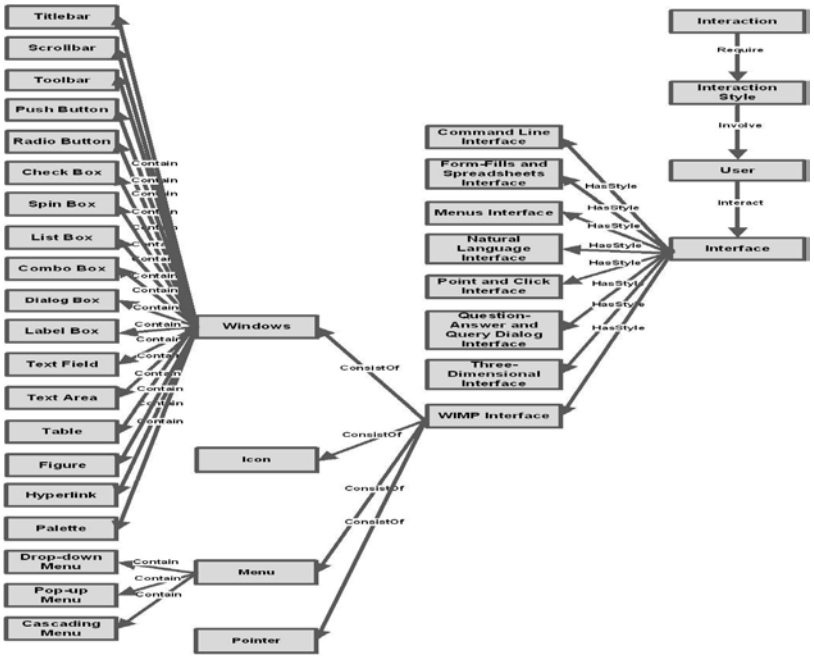

Figure 2. WIMP-UI Ontology Model [13]

4.2. USI Ontology Model

In this work, we propose an ontology model that represents interaction between user and software. The ontology model is named as USI (User - Interaction), and it is modeled as in Figure 3. The USI ontology model itself contains some elements as depicted in Table 1.

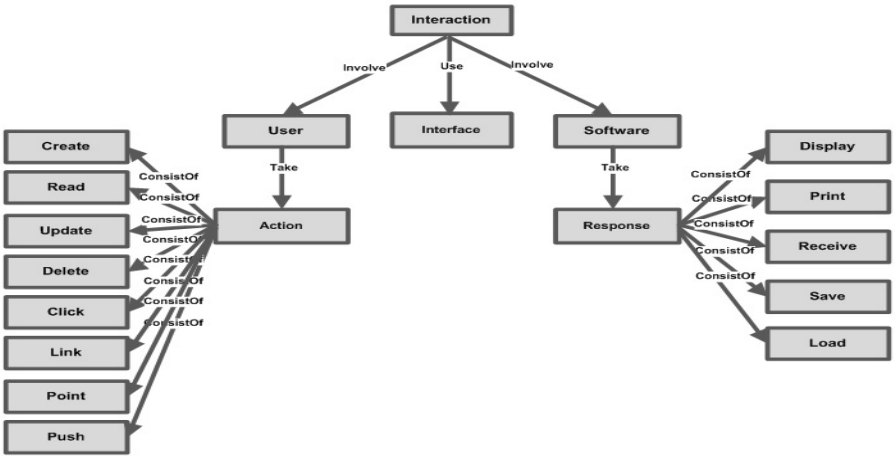

Figure 3. USI Ontology Model

Table 1. Elements of Interaction

Element		Description
Interaction		Communication between user and software.
User		Someone that interact with software for achieving their goals.
Software		Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
Interface		Part of an interactive computer system that sends messages to user and receive instruction from the user.
Action		Something that is done by the user to interact with the software.
Action	*Create*	*Create*, means as make new data; including processes with keywords: *create*, *input*, *receive*, *import*, and *record*.
	Read	*Read*, means as data retrieval; including processes with keywords: *read*, *view*, *report*, *bring up*, *print*, *find*, and *look up*.
	Update	*Update*, means as the data changed or data modification; *Read*, means as data retrieval; including processes with keywords: *update*, *change*, *modify*, and *alter*.
	Delete	*Delete*, means delete or move the data; including the process with keywords: *delete*, *discard*, *remove*, *trash*, and *purge*.
Action	*Click*	*Click*, means to press option on *menu*, *list box*, *check box*, *radio button*, and *icon.*
	Link	*Link*, means link to something that contained in the *hyperlink.*
	Point	*Point*, means point to an object.
	Push	*Push*, means push a button (*push button*).
Response		Something that is done by software to interact with the user.
Response	*Display*	*Display*, means displays something on monitor screen.
	Print	*Print*, means print something using the printer.
	Receive	*Receive*, means receives something (*text*, *numeric*, *command*, *click*, *push*, etc.) as input from the user.
	Save	*Save*, means save something to storage media.
	Load	*Load*, means load something from storage media.

4.3. Merging WMP-UI and USI Ontology Models

Capturing end-to-end interaction will involve user, interface, and interaction. Thus, we merges two ontology models, WIMP-UI and USI ontologies. The merging is done to the same classes, *interaction*, *user*, and *interface* classes. Using merge of the ontology models can be captured the interaction between user and software through the user interface. The merging of WIMP-UI and USI ontologies is illustrated in Figure 4 as follow:

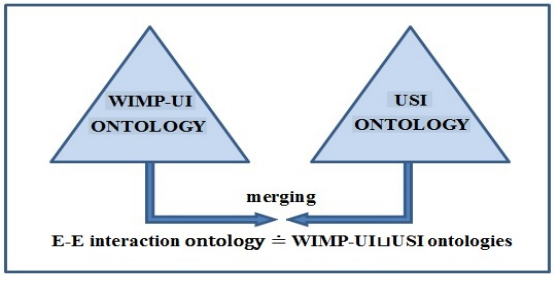

Figure 4. E-E interaction Ontology Model

5. Requirements Elicitation vs Requirements Recovery

Requirements elicitation is an important activity in forward engineering in order to get requirements from sources of requirements. Sources of requirements probably from clients, competitors' application, or technical standards and legislation. In the previous session, sources of requirements from existing application software were conveyed, and we use end-to-end interactions as source of requirements.

5.1. Elicitation Techniques

There are many elicitation tehniques, for example [1]:

- **Interviewing**; Interviews are planned but not detail and flexible.
- **Questionnaires**; Questionnaires are the structured interview, have no possibility of elaboration or deviation.
- **Task observation**; Observing the task being performed.
- **Paper prototype**; Eliciting the requirements using paper prototype [11].

In a related paper, Vijayan and Raju have proposed requirements elicitation using paper prototype, where the paper prototype is a visual representation of what software will look like [11]. They proposed their technique for a new software development. The paper prototype will be implemented as user interface. We perform requirements elicitation from the existing application software, and user interface is implemented in the software.

5.2. Requirements Recovery Using WIMP-UI⊔USI Ontology Model

We work on the other direction of foward engineering, coin as reverse engineering, we recover the requirements using the merging of WIMP-UI and USI ontology models. We observe end-to-end interaction through user interface and put it into the merge of WIMP-UI and USI ontology models that which is provided as a template. The ontology model merges two ontology models, WIMP-UI and USI ontology models. We use Portégé as ontology editor, OWL-DL as ontology language, and Pellet reasoner.

Eliciting the requirements consists of some procedure as follow:

- **Capture User Interface**; User interface that used in interaction should be captured in detail. User interface will be grouped into window, icon, menu, and pointer [13].
- **Transform to Ontology Model**; All of the captured user interface should be transformed into WIMP-UI ontology model as part of ontology model merged [13].
- **Observe User**; Observing all of users who interact with existing application software.
- **Transform to 'user' in Ontology Model**; Transforming all of software users into ontology are performed as follow:
  - Create sub class of User.
  - Create restriction like on template.

  Transforming all of software users into ontology are performed. For example, student is a software user, so result of the transformation as shown in Figure 5.

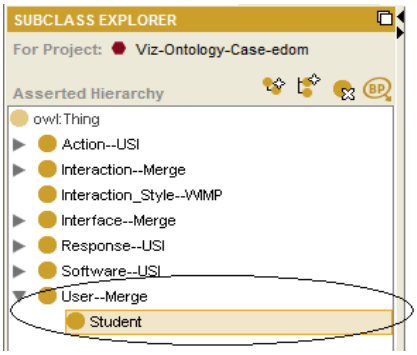

Figure 5. 'Student' in WIMP-UI⊔USI Ontology Model

- **Observe End-to-end Interaction**; Observing end-to-end interaction starts on the first action by user and response by software, it take places alternately. Ending of observation on last response that means user goal is achieved.
- **Transform to 'Action' and 'Response' in Ontology Model**; Transforming all of actions by user will be sub classes of Action and responses by software will be sub classes of Response.
  - **Action**
    - Specify the action by user, what it includes read, update, delete, click, link, point, or push.
    - Check the existence of classes that describe the action.
    - If not found, should be created a subclass of Read, Update, Delete, Click, Link, Point, or Push; subclass name refers to the format <Read|Update|Delete|Click|Link|Point| Push>-<text|label|that appear on user interface>. The example of actions that taken by user (Create-Password and Create-Username) are shown in Figure 6.

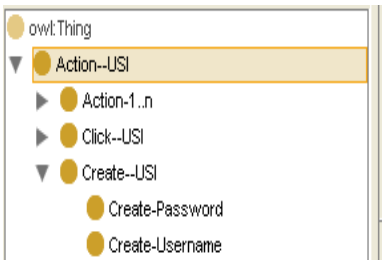

Figure 6. Action by User

    - Create restriction like on template.
  - **Response**
    - Specify the response by software, what it includes display, print, receive, save, or load.
    - Check the existence of classes that describe the response.
    - If not found, should be created a subclass of Display, Print, Receive, Save, or Load; subclass name refers to the format
    - <Display|Print|Receive|Save|Load>-<text|label| that appear on user interface>. An example of response that taken by software (Save-Form) is shown in Figure 7.
    - Create restriction like on template.

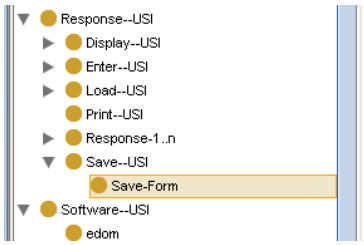

Figure 7. Response by Software

6. Conclusion

Based on this work, we may conclude that (a) End-to-end interaction can be used as a source of requirements for requirements recovery and reconstruction from existing application software; (b) Proposed ontology model can be used for capturing the end-to-end interaction; (c) The end-to-end interaction ontology model uses WIMP-UI ontology model developed in our previous works [13]. Further, WIMP-UI ontology model is merged with the USI ontology model that developed in this work; (d) End-to-end interaction ontology model represents an end-to-end interaction between user and software for achieving the certain goal through WIMP interface; and (e) Reconstructing the requirements document using ontology that captures end-to-end interaction does not need anything, except the application software itself.

7. Future Research

Following this work, there are some possible future researches, such as (a) Performing experiments using existing software to obtain the end-to-end interaction as a source of requirements; (b) Creating procedures to transform end-to-end interaction be elements of the end-to-end interaction ontology model; (c) Building an ontology model that represents requirements recovery, including the end-to-end interaction ontology model.

References

[1] I. K. Bray, "An Introduction to Requirements Engineering", Pearson Educated Limited, **(2002)**.
[2] F. Chen, H. Zhou, H. Yang and M. Wardet, "Requirements Recovery by Matching Domain Ontology and Program Ontology", 35th IEEE Annual Computer Software and Applications Conference, available at http://ieeexplore.ieee.org/stamp/ stamp.jsp?tp=&arnumber=6032405, **(2011)**.
[3] S. Demeyer, S. Ducasse and O. Nierstrasz, "Object-Oriented Reengineering Pattern", http://www.iam.unibe.ch/~scg/OORP/, **(2008)**.
[4] M. El-Ramly, E. Stroulia and P. Sorenson, "Recovering Software Requirements from System-user Interaction Traces", ACM, **(2002)**.
[5] IEEE: IEEE Standard Glossary of Software Engineering Terminology, lEEE Std 610.121990, http://ieeexplore.ieee.org/, **(1990)**.
[6] D. Kulak and E. Guiney, "Use Cases: Requirements in Context", Second Edition, Addison Wesley, **(2003)**.
[7] K. Liu, A. Alderson and Z. Qureshi, "Requirements Recovery from Legacy Systems by Analysing and Modelling Behaviour", IEEE International Conference on Software Maintenance, (ICSM '99) Proceedings, **(1999)**.
[8] F. V. S. Luiz and V. L. Gava, "Recovery Software Requirements in the Reverse Engineering of Legacy Systems", 9th CONTECSI International Conference on Information Systems and Technology Management, São Paulo, Brazil, http://www.tecsi.fea.usp.br/envio/contecsi/index.php/envio/ article/viewFile/343/4, **(2012)** May 30 - June 1.
[9] P. Rayson, R. Garside and P. Sawyer, "Recovering Requirements from Legacy Documents", http://www.comp.lancs.ac.uk/computing/research/soft_eng/projects/ revere/ papers/rgs_icsm99.pdf, **(1999)**.
[10] W. V. Siricharoen, "Ontology Modeling and Object Modeling in Software Engineering", IJSEIA International Journal of Software Engineering and Its Applications, http://www.sersc.org/journals/IJSEIA/vol3_no1_2009/5.pdf, vol. 3, no. 1, **(2009)** January, pp. 43-60.

[11] J. Vijayan and G. Raju, "A New Approach to Requirement Elicitation Using Paper Prototype", IJAST International Journal of Advanced Science and Technology, http://www.sersc.org/journals/IJAST/vol28/2.pdf, vol. 28, **(2011)** March, pp.9-16.
[12] Y. Yu, "Reverse Engineering Goal Models from Legacy Code", http://www.cs.utoronto.ca/~alexei/pub/re05re.pdf, **(2005)**.
[13] E. M. Zamzami and E. K. Budiardjo, "Capturing Semantic Meaning on User Interface Presence By Creating Its Ontology", IJCSI International Journal of Computer Science Issue, http://www.IJCSI.org, vol. 9, Issue 4, no. 1, **(2012)** July, pp. 6-12.
[14] L. Zong-yong, "The Domain Ontology and Domain Rules Based Requirements Model Checking", IJSEIA International Journal of Software Engineering and Its Applications, http://journal.sersc.org/IJSEIA/vol1_no1_2007/IJSEIA-2007-01-01-07.pdf, vol. 1, no. 1, **(2007)** July, pp. 89-100.

