Ontology-Based Knowledge Management System Model for R3ST Software Maintenance Environment

Eko K. Budiardjo
Universitas Indonesia
Depok, Indonesia
eko@cs.ui.ac.id

Elviawaty M. Zamzami
Universitas Sumatera Utara
Medan, Indonesia
elvi_zamzami@usu.ac.id

Ganjar Ramadhan
Universitas Indonesia
Depok, Indonesia
ganjaramadhan41@ui.ac.id

Muh. Nur Musa
Universitas Indonesia
Depok, Indonesia
muh.nur@ui.ac.id

ABSTRACT
Research shows that approximately 40% to 60% of the maintenance effort, devoted to understand the modified software. One of the obstacles found in software maintenance is the lack of contained knowledge in software. This research proposes a conceptual framework in software maintenance by utilizing knowledge management process and software comprehension using ontology as formal modeling analysis to explicit the tacit knowledge in software. The topic of software comprehension is an important part for software engineer’s effort. This framework is realized by means the Requirements Recovery and Reconstruction (R3) method based Software Tools (R3ST) environment. R3ST bind to domain ontology for increasing the contextual result. It is expected that R3ST would be able to facilitate software developers in software maintenance activities.

CCS Concepts
• Software and its engineering—Software maintenance tools

Keywords
Software Comprehension; Software Maintenance; Requirements Recovery; Knowledge Management; Knowledge Representation; Ontology.

1. INTRODUCTION
Software maintenance is defined in the IEEE 1219 Standard for Software Maintenance, as a modification of a software product after release in order to correct errors, improve performance or other attributes, or adapt to environmental changes [1]. One of the obstacles found in software maintenance is limited understanding of possessed by a software engineer. Limited understanding refers to how quickly a software engineer can understand which sections need to be changed or corrected while the individual does not participate in the previous development. Research shows that approximately 40% to 60% of the maintenance effort, devoted to understanding the software to be modified [1]. Thus, the topic of software comprehension (understanding the software) is an important part for software engineers.

On the other hand, there is some study that defined the software to be viewed as knowledge. This definition came from the nature of knowledge that software as one of knowledge medium [2][3]. Several study also describe that software is embedded knowledge [4], and the characteristic of embedded knowledge is tacit knowledge residing in organizational routines, practices and shared norms, that are not codified, not available as text, and non-transferable as public good. This is defined in [5], [6], and [7]. Ontology can represent captured knowledge in which contained in software to facilitate software developer in software understanding. There are several researches that study the relation between software and domain knowledge by means ontology. This research was described in [8]. It was applying knowledge management in software maintenance, which use ontology in knowledge extraction of SECI model. The formal model as another initiative was presented by [9], that provide ontological representation across different knowledge resource.

Learning from those three previous researches in managing knowledge to facilitate software comprehension activity, this paper presents a conceptual framework in software comprehension by utilizing knowledge management process and using ontology as formal modeling analysis to express the tacit knowledge contained in the software. The expected goal with this framework is that it can facilitate software developer in software comprehension activities.

2. MOTIVATION
The history of knowledge describe that knowledge medium consist of five medium, that is deoxyribonucleic acid (DNA), brain, hardware design, books, and software [2], as we can see the comparison of each medium in Table 1. Software as one of those five medium have a nature insight that software development activities is transcribing the human knowledge into software product that contained that human knowledge [2]. When the software viewed as knowledge, then the software development activities consist of knowledge acquiring / creating activities and translating the knowledge into specific language form that also known as “coding” [2][3].

Different term used by [4], described that software is embedded knowledge, changeable, invisible, embedded in matrix of applications, users, laws, and machines, that still remain inherently
not visualize able. It means that software itself is knowledge that embedded in applications or users. It is also give the challenge for software modelling field to provide comprehensive tools to visualize the software.

**Table 1. Comparison of knowledge medium characteristic [2]**

<table>
<thead>
<tr>
<th>Storage Medium</th>
<th>Persistence in Medium</th>
<th>Update Frequency</th>
<th>Intentionality</th>
<th>Ability to Self-Modify</th>
<th>Ability to Modify Outside World</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA</td>
<td>Very persistence</td>
<td>Very slow</td>
<td>Low</td>
<td>Moderate</td>
<td>Quite limited</td>
</tr>
<tr>
<td>Brain</td>
<td>Very Volatile</td>
<td>Very fast</td>
<td>High</td>
<td>High</td>
<td>Limited to specific design</td>
</tr>
<tr>
<td>Hardware Design</td>
<td>Very persistence</td>
<td>Slow</td>
<td>High</td>
<td>Low</td>
<td>Limited</td>
</tr>
<tr>
<td>Book</td>
<td>Quite persistent</td>
<td>Quite slow</td>
<td>High</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Software</td>
<td>Quite persistent</td>
<td>Fast</td>
<td>Quite high</td>
<td>High</td>
<td>Relative unlimited</td>
</tr>
</tbody>
</table>

With regards to software as embedded knowledge term, different studies have been describing the relation of embedded knowledge and tacit knowledge. There are four type of knowledge as describe in [5], that are embreded knowledge (individual-explicit), embodied knowledge (tacit-individual), encoded knowledge (collective-explicit), and embedded knowledge (tacit-collective). Definition of embedded knowledge from [5] is “the collective form of tacit knowledge residing in organizational routines, practices and shared norms”. The definition of embedded knowledge is also described in [6] as tacit and resides within systematic routines, that tacit means non codified and non-transferable as public good [7]. Knowledge in software product that has been described above, of course, needs to be managed properly. One solution that can be used is to apply knowledge management concepts. One of the concept of knowledge management is the process as explained in [10] with a process consisting of four stages: capturing, organizing, refining, and transfer of knowledge. Knowledge management process described in [10], briefly stated in Table 2.

### 3. THE CONCEPTUAL FRAMEWORK

Based on literature that described in section 2, if we view the software as a knowledge, it is important to use knowledge management concept, to manage the knowledge in which embedded in software itself. These researches aim to propose a conceptual framework on managing knowledge of software to facilitate software comprehension process. This framework as illustrated in Figure 1, combining cyclic series activities, starting from the problem domain ontology creation, software maintenance of existing runtime code by means of software comprehension and requirements reconstruction, and finally result a ontology-based knowledge management system for further enhancement and/or reengineering. The main objective of the framework is to facilitate software developer in software maintenance by manage the knowledge contained on software that would be comprehend.

![Figure 2: Conceptual model for software comprehension in knowledge management process](image)

**Figure 2: Conceptual model for software comprehension in knowledge management process**

In this research, The Software Maintenance Framework adopt knowledge management (KM) process regarding to [10], that consist of four phase: capturing, organizing, refining, and transfer. This framework use an existing information system (business application) software as input and at the end phase of process provides three knowledge representations: concept map, UML document, and SRS document, as illustrated in Figure 2.

Performing analysis across all phase in this framework, including extraction and synthesize using ontology aims to collect information about domain concept and this framework as conceptual is try to manage the tacit knowledge in information system software and make it explicit by generating concept map, UML model document, and SRS document as knowledge representation.

Besides performing domain analysis, this framework also includes software comprehension technique by utilizing ontology and relation of domain concept to provide insight about software on maintenance task. After those all three representations have been constructed, then it is could be transferred to all member of software development via web application tools to facilitate software maintenance activities.

As seen on Figure 3, the Requirements Recovery and Reconstruction (R3) method [14] based Software Tools (R3ST) environment has some purpose, one of those is for requirement
recovery of legacy software by identifying the end-to-end Interaction within application [11, 12] and visualization technique for software comprehension [13].

Look back to the KM process; knowledge capturing is supported by R3ST End-to-end Interaction Capturing Tools and R3ST Ontology-Based Knowledge Management Tools for organizing its knowledge. Along the knowledge capturing and organizing, R3ST Software Comprehension Tools and R3ST Requirements Reconstruction Tools serve as analysis tools by visualizing the organized knowledge.

Domain knowledge representations use the available ontology or developed by using OWL Protégé and then bind to R3ST as term references and consistency checking. This binding uses ontology merge technique in such a way so that it create a tightly couple ontology. This approach works if we create as a unity platform. The Current challenges to face that more and more domain ontology only available in cloud environment, and what if it is implemented in other then OWL.

In R3ST, R2UC ontology is built using Protégé OWL developers environment. OWL Protégé Visualization’s features are very useful for obtaining an overall figure and description of the ontology. All of the end-to-end interaction that have been captured are constructed as ontology, based on use case modeling concept, by means OWL Protégé. Then, the ontology is visualized by OWL Protégé Visualization to comprehending the software. A use case is represented as an ontology concept that has some properties which indicate the pre-condition, post-condition, basic flow, alternate flow, special requirement, and extension point of the use case.

Figure 3: R3ST Building Blocks [11]

4. ONTOLOGY MODEL

R2UC (Requirement Representation using Use Case) ontology is a representation form of ontology which interprets the user interaction through the interface or display software and features in a software. By using R2UC ontology, SRS can be represented in a semi-formal. The advantage of the semi-formal representation is the representation can still be understood by humans (developers and stakeholders) or by the computer. This is due to relationship between concepts on ontology that express a semantic meaning. Therefore, information that provided by R2UC ontology can be shared across software developer in semantic-wiki application. Besides that, ontology is Interlingua [15]. It is a good representation tool to avoid ambiguity or the use of different terminologies but have same meaning. In addition, R2UC can also provide the ontology structure of SRS which refers to the RUP (Relational Unified Process) as SRS document [16].

R2UC ontology modeled into three levels as illustrated in Figure 4. The first level presents software that interprets the requirement of features to meet the needs as the solution of the problem.

From the first level can then be expanded to R2UC second level ontology. At the second level specified by supplementary software requirement specification and use case. Both of these are part of the SRS document format RMUC. The third level obtained from the expansion of the second level. The third level shows in detail how the software requirements specified in the elements of the SRS.

Figure 4: Requirement Representation using Use Case ontology

If observed carefully, class hierarchy contained in R2UC ontology models in Figure 5, represent the two types of documents in accordance with the method Requirements Management with Use Cases (RMUC), namely (a) the documents give overview of functionality aspects known as a use case specification document, and (b) Supplementary specification document that describes non-functional aspects.

Figure 5: Class hierarchy contained in R2UC ontology models
5. EXPERIMENT DESIGN AND RESULT

This experiment aims as partial experiment of variety testing in order to support the validation of this framework. The objective of this experiment is to prove that software maintenance activity by contracting an ontology-based knowledge representation of software requirements, via indentifying end-to-end interaction is sound.

There are three existing software owned by Universitas Indonesia serve as the experimental object for this research. The university name those software as SiMitra, SIAK-NG, and EDOM. Among those three objects, SiMitra is considered as the least complicated among the others. It is a software in which developed to support relationship management activities between industry and the university. Initially, this software was developed by the Faculty of Computer Science student as a software engineering projects.

SIAK-NG (Academic Information System - New Generation), the most complicated among others, is software developed by the University team’s including some expert from the Faculty of Computer Science. This application supports the overall academic activities management. The user includes enrolled students, faculty members, and university administrators. For the purpose of this experiment, due to its complexity, it is considered as three experimental objects.

The last but not least is EDOM, an application for lecturer evaluation by students as part of faculty member performance assessment. For each lecturer of taken courses, the students fill in an electronic questioner provide by system. EDOM will compile the questioneres and provide calculated score for each lecture-course to all individual questioning items. The student may also give feedback to the lecturers in a free format entry. Electronically, the EDOM sends the result, including the feedback to each lecturer via e-mail.

The experiment assume that we would like to conduct software maintenance to those five objects, in which that we have limited information of it, no such development documentation, and have a marginal user manual. By applying the R3 Method [14], The observers of each individual object determine any single interaction that consist of end-to-end interaction (uninterrupted series on interaction between user and computer, to perform a single interaction).

R3ST has R2UC ontology template in OWL Protégé. As the initial step the observers have to be familiar with the template. The observers include all parts of end-to-end interaction ontology into OWL Protégé in which has the R2UC template. Then, analyze the R2UC Ontology of the object, by means evaluating end-to-end interaction of the object in order to find unified use case. This use case view as an use case recovered from the object of the experiment.

To validate the results of obtained recovered requirements, we are distributing a questionnaire that contains those recovered requirements. The respondents was selected among the user of the software. The results are in accordance this experiment are shown in table 3.

Table 3 shows two important parts, recovered software features and constructed classes of R2UC ontology of those recovered features, for each experimental objects. Refering to the R2UC ontology models, the experiment that has been conducted to fulfill the functional part of the object, whereas the non-functional part are not yet tested. It must be another way to recover the non-functional part, because addressing the end-to-end interaction does not directly reflecting the non-functional requirements aspect. Analyzing the recovered functional requirements by means use case approach; it may give the indication on the implementation constraint, in which part of non functional requirements.

<table>
<thead>
<tr>
<th>Classes of ontology model</th>
<th>Feature</th>
<th>The number of SiMitra features</th>
<th>The number of SIAK-NG1 features</th>
<th>The number of SIAK-NG2 features</th>
<th>The number of SIAK-NG3 features</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiMitra</td>
<td>Total of classes: 869</td>
<td>Averages of sibling classes: 16</td>
<td>Maximals of sibling class: 70</td>
<td>Total of restrictions: 2065</td>
<td></td>
</tr>
<tr>
<td>SIAK-NG1</td>
<td>Total of classes: 474</td>
<td>Averages of sibling classes: 12</td>
<td>Maximals of sibling class: 49</td>
<td>Total of restrictions: 1345</td>
<td></td>
</tr>
<tr>
<td>SIAK-NG2</td>
<td>Total of classes: 336</td>
<td>Averages of sibling classes: 8</td>
<td>Maximals of sibling class: 35</td>
<td>Total of restrictions: 811</td>
<td></td>
</tr>
<tr>
<td>SIAK-NG3</td>
<td>Total of classes: 494</td>
<td>Averages of sibling classes: 15</td>
<td>Maximals of sibling class: 53</td>
<td>Total of restrictions: 1407</td>
<td></td>
</tr>
<tr>
<td>EDOM</td>
<td>Total of classes: 98</td>
<td>Averages of sibling classes: 5</td>
<td>Maximals of sibling class: 8</td>
<td>Total of restrictions: 199</td>
<td></td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

This research proposes a conceptual framework for software maintenance by utilizing knowledge management process and supported by R3ST environment. The framework consists of four phase: 1) capturing, that use end-to-end interaction concept; 2) organizing, that perform ontology codification; 3) refining, that consist of ontology analysis and comprehension analysis; and 4) transfer, that provide ontology, UML profiler and SRS model as knowledge representation, which could be transferred to all team on software development organization.

Functional requirements of maintained software have been tested to five objects, following the phase one up to three of the framework. Other fruitful research is for the non functional requirements part of phase one to three. For phase four, knowledge transfer, R3ST has tools for generating the explicit knowledge of R2UC ontology in which based on specified document template.

Currently, the authors work recovering non functional software requirement, the critical theory that is formalization on each phase of this framework, and domain ontology merging in cloud environment in order to get insight of cloud based software product line concept.

7. ACKNOWLEDGMENT

Initially, R3ST Project is three-year research project, starting in 2013 under BOPTN-PUPT grant, which is funded by the Ministry for Research, Technology, and Higher Education, Republic of Indonesia. Currently, the authors work on further development of R3ST to address cloud computing environment.

8. REFERENCES


