Developing Artifact with Concept Relationship Oriented Methodology: A Progress Report

Bayu Tenoyo, Petrus Mursanto, Ade Azurat,
Hisar Maruli Manurung

Faculty of Computer Science, Universitas Indonesia
Depok Indonesia
{bayu.tenoyo11|santo|ade|maruli}@ui.ac.id

Abstract. In this paper we report partial progress of our research about system comprehension. We used levels of abstraction and decomposition strategy to reduce system complexity. Concept Relationship Oriented methodology is one of its result, the methodology can be used, to develop new artifact for reverse engineering. There are two significant reported activities: searching Concept Representation, and searching Structure of Concept. Searching Concept Representation was a process, to find a representation, proper or might exist less depend on artifact types (for example: unique concepts). In this stage dependency on level of abstraction still tight, but at the second stage we found the representation, Structure of Concept.

Keywords: reverse engineering; complex system, level of abstraction, decomposition, knowledge representation, diagram, comprehension.

1 Introduction

Improving comprehension process can be achieved by providing system behavior, providing system properties, and providing system features description.Those are through with abstraction and decomposition [1,2]. Abstraction as a view of an object, focuses on the information, it has been defined relevant to a particular purpose, it ignores the reminder of the information, and it has clear relationship between its boundary [2,3,4]. Decomposition can be applied to a complex system, to identify its features, or its components, with or without deconstruct the system. The process can be automated [6,7], and can be complied with certain criteria, such as: design, functionality, modular, or hierarchycal relationship[3].

Abstraction and decomposition may produce Set of Concepts, to produce missing artifacts, and to update current artifacts as result of reverse engineering [8]. Set of Concepts member has relationship between each other, the relationship inputs for Structure of Concept construction [9]. The structures may explain the traceability between artifacts, sometimes the traceability can not be found, due missing artifact, or

it was not created as part of project policy (such as Open Source). New member learns with difficulty, a complex system with incomplete set of artifacts .

Several examples of artifact types are: Requirements Documents using Natural Language, Analysis Diagrams, Design Diagrams, Source Code using C language, and data (such as data testing). Natural Language helps reader comprehend the system, but ambiguity sometimes created. Diagrams may help concept structures visualization, and improve their knowledge, or improve comprehension [9] as foundation: before architecture evaluation [11, 12, 13, 14], before and after maintenance [15], architecture modification [2, 11, 16, 17,18], performance evaluation [19, 20], testing specification [21, 22], components improvements [23, 24, 25, 26, 27], reengineering [11, 18, 28], concept recognition and its relationship [29, 30], and data exchange [31].

Reverse engineering process, to construct or reconstruct those artifact types, due missing artifacts, or not created artifacts, on a complex existing system, is required. Its process quality, or its artifact quality, may depend on its development method, and its tools. From our experience in stage one[8], the method of translation from one artifact type to another is unique, or different from others. As an example, translation method is difference between Analysis Diagram - Requirements Document, and Source Code - Design Diagram.

We found Concept Representation may exist in the artifacts (first stage inputs were User Guide and Source Code), and may improve the quality. Second stage, we improve our approach by creating, develop new method, and develop tool based on previous result. We introduce a new definition of a concept, the definiton can be used as a tool, to develop a comprehension artifacts, to develop new artifact such as UML diagram [9].

Chapter 2 in this paper, we present scope of the project, how we conducted, and results, of our first stage research. Chapter 3 in this paper, we present scope of the project, methodology, and results, of our second stage research. Chapter 4 is summary, and we add several Appendices, example of comprehension artifacts.

2 Concept Representation

First stage project is a reverse engineering, the application is netstat , an Open Source, is developed using C language. We use Source Code, and User Guide as an inital artifact, or activities inputs. The application has several features, our reverse engineering activities used the features as scope of the project, but there is a limitation, its features number. The scope or features number are defined using Behavior Tree Diagram [32]. Targets of this project are: creating artifacts at design level, analysis level, and requirement document (Natural Language document), and searching concept representation for each levels of abstraction.

The reverse engineering process consists of several activities:

1. Inputs were Source Code and User Guide. Outputs are Behavior Tree Diagram as scope of project, Set of Concept or Knowledge Representation, and team member comprehension improvement. There are several concepts types: Library Name, Source Code File, Header File, Structure, Variable Type, Variable Name, Function

Name, Function Name, and its parameters, Branch, Branch Conditional, Loop, Loop Conditional, Body of Block, Features (from User Guide).
2. Inputs were source code file, Set of Concepts from previous step, and team member comprehension. Outputs are Header File diagram, Message Flow Between Function diagram, Message Flow Between Files diagram, Control Flow Inside of a Function diagram, new members for Set of Concept, or Knowledge Representation, and team member comprehension improvement. Type of concepts at design level can be categorized as: Application Name, Variable Name, Variable Type, and Function.
3. Inputs were Set of Concepts from previous, and its comprehension. Outputs are Behavior Tree, and Behavior Tree Integration Diagram [32] as analysis level diagram, new members for Set of Concepts, and team member comprehension improvement. Type of concepts at Behavior Tree Diagram analysis level are: node and tree.
4. Inputs were Set of Concepts from previous step, and its comprehension. Outputs are Requirements Document in Natural Language as final artifact and team member comprehension improvement.

Each of those processes has its own translation methodology. To have more than one methodologies, increased complexity of reverse engineering. We want to reduce complexity, less depend artifact types representation of a concept is searched. We hope it may support automation (at certain degree). Second stage of our research, found less depend artifact type representation, it is the meta data of a concept [9].

3 Comprehension Artifact

From our first research stage, we could finish the project, and we found the problem from our approach. In this stage of project, we wanted to achieve less depend artifact type, to develop a new artifact. We decide to construct a tool, or Structure of Concept [9], is based on the problem (improving artifact quality, improving process quality, and achieving less depend artifact types). As a start, we constructed new definition of a concept [9], concept can be defined as follow:

- Concept can be built from others and it has unique structure.
- The definition of a concept may depend on the domain.
- Definition of a concept is a concept .
- A concept can be instantiated, the instantiation based on a definition.
- Two types of concept: are definition, and instance.
- Structure of Concept is difference between Concept Definition Structure, and Concept Instance Structure.
- Instance always complies with definiton. The definition structure leads how to build the Instance Structure.This Concept Definition Structure and Concept Instance Structure are a tool, and are comprehension artifacts, are difference between both of them, can be recognized.

For the second stage, inputs were Set of Concepts from previous stage, and new diagram artifacts, but members of set, are limited based on new artifacts. The artifacts provide Information, relationship between a Source Code file with its libraries. Targets of this project were: creating artifacts at design level, the artifact explains user query or need, and were going to find Structure of Concept.

Second stage process was as follow:

1. Inputs were Set of Concepts based on previous stage. Outputs of this process are: defined need, and queires, those supported using Concept Definition Structure, and its Concept Instance Stucture. There are three queries: Source Code files - libraries, Source Code files - Header Files, refer to libraries, and function inside of Source Code files - Header Files, refer to libraries. Those queries created, to support a need, explanation of relationship between a function with its libraries. When function is changed, what libraries need to be learnt.
2. Processing each queries, was constructing a Concept Structure Definition. The result may be shared to others.
3. Procesing each Concept Structures Definition, were concstructing Concept Structers Instance. The instances may explain queries.
4. Processing the answer, as an example a project manager needs an experienced programmer, to modify a function called bpf_stat inside of bpf.c files. Current Concept Structures Instance may answer the need. From Query results, the project manager may know, the programmer must have experience on stdlib, and sys libraries.
5. (An optional) Friendly comprehension artifact may be created, same diagram notation with existing artifacts, or a new diagram notation.

4 Summary

From two stages project, we found Structure of Concept, the structure is a meta data for a concept. There are two types of structures: Concept Structure Definition and Concept Structure Instance.

Both structure may reduce dependency on artifact type, we may use them as a tool , to develop new artifact, or an instance of different artifact type.

Appendices[1] show partial of Concept Sructures Instance, presented on table format. Concept Structure Definition is not presented yet.

[1] http://facebook.com/conceptrelationship

References

1. Wing, Jeannette M. “Computational Thinking.” Communication of The ACM, March 2006.
2. Biggerstaff, Tedd J. “Design Recovery for Maintenance and Reuse.” Computer IEEE, 1989.
3. IEEE Standard Board, “IEEE Standard Glossaryof Software Engineering Terminology. IEEE”. IEEE Standard Board, 1990. Std. 610.12-1990.
4. Tilley, Scott. “A Reverse Engineering Environment Framework.” Carnegie Mellon. Software Engineering Institute, April 1998. Technical Report CMU/SEI-98-TR-005.
5. Floridi, Luciano. “The Method of Level of Abstraction.” Minds and Machines. 2008, Vol. 18, pp. 303-329.
6. Lakhotia, Arun and Gravley, John M. “Toward Experimental Evaluation of Subsystem Classification Recovery Technique.” IEEE International Symposium on Circuits and System, 2007.
7. Oca, Carlos Montes de and Carver, Doris L. “A Visual Representation Model for Software Subsystem Decomposition.” Proceedings of the Working Conference on Reverse Engineering (WCRE'98), 1998.
8. Tenoyo, Bayu and Mursanto, Petrus. “From C to System Requirements”. Jakarta : Computer Science Faculty Universitas Indonesia, 2012. TR-CSUI-002-2012.
9. Tenoyo, Bayu and Mursanto, Petrus. “Notasi Concept Relationship Approach untuk Memahami Sistem”. Jakarta : Computer Science Faculty Universitas Indonesia, 2013. TR-CSUI-003-2013.
10. Wang, Qin, et al. “EVOLVE: An Open Extensible Software Visualization Framework. 2002.” Sable Technical Report No. 2002-12.
11. Gorton, Ian and Zhu, Liming. “Tool Support for Just in Time Architecture Reconstruction and Evaluation: An Experience Report.”. Proceedings. 27th International Conference on Software Engineering. ICSE, 2005
12. Müller, Hausi A., et al. “Reverse Engineering: A Roadmap”. 22nd International Conference on Software Engineering (ICSE), Future on Software Engineering Track, 2000.
13. Dufour, Bruno. “Objective Quantification of Program Behaviour Using Dynamic Metrics”. Thesis Master Degree. Montreal : School of Computer Science, Mc Gill University, 2004.
14. Demeyer, Serge, Ducasse, Stephane and Lanza, Michele. “A Hybrid Reverse Engineering Approach Combining Metrics and Program Visualisation”. Proceedings. 6th Working Conference on Reverse Engineering, 1999.
15. Abowd, Gregory, et al. “MORALE – Mission Oriented Architectural Legacy Evolution”. Software Maintenance IEEE, 1997.
16. Tzerpos, Vassilios, Holt, R.C. and Charmichael, Ian. “Design Maintenance: Unexpected Architectural Interactions (Experience Report)”. International Conference on Software Maintenance. 1995.
17. Armstrong, M.N. and Trudeau, C. “Evaluating Architectural Extractors”. Proceedings. 5th Working Conference on Reverse Engineering. IEEE, 1998.
18. Byrne, Eric J. “A Case Study, Software-Practice and Experience”. Software Reverse Engineering, Vol. Vol. 21 (12), December 1991.
19. Mendelzon, Alberto and Sametinger, Johannes. “Reverse Engineering by Visualizing and Querying”. Software Concepts and Tools, Springer Verlag, 1995.

20. Systa, T. “On the Relationship Between Static and Dynamic Models in Reverse Engineering Java Software”. Proceedings. 6th Working Conference on Reverse Engineering. IEEE Computer Society Press, 1999.
21. Gannod, Gerald C. and Cheng, Betty H. C. “A Suite of Tools for Facilitating Reverse Engineering Using Formal Methods”. Proceedings. 9th International Workshop on Program Comprehension. 2001.
22. Berger, Bernhard J. and Bunke, Michaela. “An Android Security Case Study Bauhaus”. Proceedings. 18th. Working Conference. Center for Computing Technologies, Universitat Bremen - Germany, 2011.
23. Mishra, S. K., Kuswahana, D. S. and Misra, A. K. “Creating Reusable Software Component from Object-Oriented Legacy System through Reverse Engineering”. Journal of Object Technology. 2009.
24. Sudhakar, P. and Sakthivel, P. “Predicting Source Code Irregularities in Automated Code Conversion System Using SRASG”. European Journal of Scientifice Research. EuroJournal Publishing, 2012.
25. Wang, Zhongjie, Xu, Xiaofei and Zhan, Dechen. “A Survey of Business Component Identification Methods and Related Techniques”.International Journal of Information Techology. 2005.
26. Canfora, Gerardo, et al. “Decomposing Legacy Programs: a First Step Towards Migrating to Client-Server Platforms”. The Journal of Systems and Software, 1999.
27. Grisworld, William G. “Program Restructuring as an Aid to Software Maintenance”. Thesis. University of Washington, 1991.
28. Murphy, Gail C. and Notkin, David. “Reengineering with Reflexion Models: A Case Study”. IEEE, 1997.
29. Harman, Mark, et al. “Code Extraction Algorithms which Unify Slicing and Concept Assignment”.Proceedings. 9th. Working Conference on Reverse Engineering, IEEE. 2002.
30. Weiser, Mark. **“**Program Slicing”. IEEE Transaction on Software Engineering. 1984.
31. Johnson, Michael and Rosebrugh, Robert. “Reverse Engineering Legacy Information Systems for Internet Based Interoperation”. Florence : In press for the International Conference on Software Maintenance, 2001.
32. Myers, Toby. “The Foundations for A Scalable Methodology for System Design”. School of Information and Communication Technology Science, Environment, Engineering, and Technology, Griffith University. 2010.