International Journal of Current Advanced Research
ISSN: O: 2319-6475, ISSN: P: 2319-6505, Impact Factor: SJIF: 5.995
Available Online at www.journalijcar.org
Volume 6; Issue 12; December 2017; Page No. 8670-8676
DOI: http://dx.doi.org/10.24327/ijcar.2017.8676.1404

Research Article

CONEXT (CONCEPT EXPLORATOR) A FRAMEWORK FOR SYSTEM COMPREHENSION WITH HOLISTIC APPROACH IN REVERSE ENGINEERING

***Tenoyo, B., Mursanto, P and Santoso, H.B**

Faculty of Computer Science, Depok, Indonesia

A R T I C L E I N F O

Article History:
Received 17th September, 2017
Received in revised form 21st October, 2017
Accepted 05th November, 2017
Published online 28th December, 2017

Key words:

Reverse engineering, system comprehension, user model, conceptual model, level of abstraction, decomposition

A B S T R A C T

The purpose of this paper is to introduce a framework of Reverse Engineering to help system comprehension using holistic approach. Our framework consists of several components: conceptual model – to provide model of studied system so it can help user to explore artifacts and understand the relationship between existing concepts, user model – to provide and to record concepts that has been explored and will be explored, methods – how we provide or create concept structures for conceptual model and user model, and the last part is tool – it will help user to understand the studied system. Using this framework we may help user to prevent lost of focus when he explore the concepts he want to know. This framework also provide infrastructure how to quantify a comprehension of a system.

INTRODUCTION

In every organization when an application getting older and its services still require, it will grow to fulfill the business. Source code changes for bug fixing and new features can not be avoided. As the business evolves the application will follow as long as it supports the goal. Sometimes it takes longer time for a system to evolve compare to team member involvement in the development. Thats is why member of development team is hard to maintain. For a new member to understand a new feature or a new module that assigned to him it need time because of a new domain must be understood. Member must explored the concepts as prerequsite of his task that explained from so many artifacts. Regardless unsynchronized artifacts with current source codes or implementation, the exploration itself has problem, for example tracking and decide which document need to visit.

Solving this kind of problems, Reverse Engineering is needed especially to provide the current documentation or artifacts or diagrams in order to help people or development team to finish it tasks.

Reverse Engineering have so many approaches and method types depend on its goal, several goals that has been acquired based on our literature studies are as follow (Figure 1):

- System comprehension (Armstrong, and Trudeau, 1998; Müller *et al*., 2000), regarding architecture comprehension of the application or logical comprehension that can be illustrated using data floww, control flow, Class Diagram, or Sequence Diagram of the source code.
- Architecture changes to increase performance or to produce reuseabillity. Before application's architecture is changed an evaluation should be made first. We can use metrics [Müller *et al*., 2000; Dufour, 2004; Demeyer *et al*., 1999) or diagrams (Gorton and Zhu, 2005) to analyze or to evaluate, such as bottleneck or performance issue (Mendelzon, and Sametinger, 1995) and dependency issues (Systa, 1999).
- To maintain consistency with new requirements or to validate the design. Consistency can be approached using formal method (Gannod and Cheng, 1999; Gannod and Cheng, 2001) and non formal method (Murphy and Notkin, 1997).
- To visualize system properties or behaviors using static diagrams and dynamic diagrams (Systa, 1999). Static diagram such as Class Diagram from UML and Dynamic Diagram such as diagram to visualize program performance and memory leak (Dufour, 2004; Object Management Group, 2006).

Corresponding author:* **Tenoyo, B
Faculty of Computer Science, Depok, Indonesia

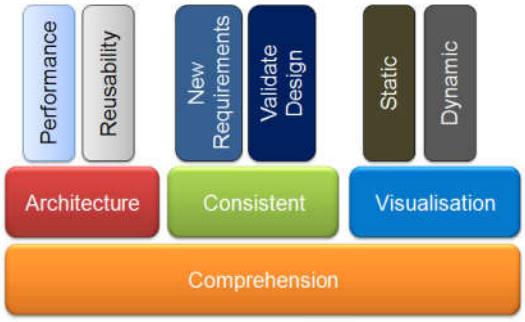

Figure 1 Reverse Engineering Goals

Based on the goals of Reverse Engineering many tools have been developed. They produced static diagrams and dynamic diagrams of source code. Here are some list of tools for Reverse Engineering, organized based on the goal:

System Comprehension and Visualisation

- Simple Hierarchycal Multi Perspective (Storey and Müller, 1995), the tool can produce relationship between concept from source code in term of architecture view, for example it seggregate the view based on sub system of the application. Selection to reduce the diagram complexity also available to help user more focus.
- Rigi (Müller, 1986; Storey *et al*., 1997), it has another advantage than previous tool: subsystem of the application can be define semi automatically based on pattern recognition and domain knowledge.
- Graphical Representation of Algorithms, Structures, and Processes (Cross, 1991), this tool provide Control Structure Diagram that enable user to expand and shrink from a nested block of source code, it may simplify the visualization of source code.
- Hy+: Hygraph based on query and visualization (Consens and Mendelzon, 1993), this tool enable us to query a concept and its relationship to provide relevant information with user needs. It also provide the visualization.
- Architecture Reconstruction and Mining (Kazman *et al*., 2003), its visualisation may help architecture changes to analyze.

Architecture Changes

- Design Maintenance System (Baxter, 1992); the tool is a part of software development tool (forward engineering). DMS provide problems that may exist and the reason why a model has been choosen to solve the problems. The decision is based on pattern recognition of the problem.
- Dr. Jones (Foltz, 2003): it can redesign Java source code based on current refactoring rules. The output of application is design representation based on specific point of view and design representation for a whole source code.

Consistency

- Reflexion Model (Murphy and Notkin, 1997) help user to validate between design decision in the previous phase and design result after the source code is created. It will show which design is consistent from start to implementation and which one is not.
- AutoSpec (Gannod and Cheng, 2001) help user to check consistency between abstraction model and its specification using formal method.
- Bauhaus (Raza, *et al*., 2006), to support user develop critical application and maintenance activities. Other literature shows (Berger and Bunke, 2011) that Bauhaus is able to check consistency between security specification for Android application against Android source code.

Holistic Approach

In this paper we propose a holistic approach for system comprehension has following factors to consider:

Support for evaluation and quantification. A process or Reverse Engineering itself must be able to repeat, define, and optimize (Müller *et al*., 2000) according to Capability Maturity Model Integration (CMMI), there are three factors that must be included in a good software process: people, methods, and tools (Team, 2011). In our approach those three factors can be defined as following requirements:

- People – the framework must support us to quantify the user comprehension.
- Methods – the framework must guide us how we do the Reverse Engineering; what kind of diagrams should be produced, and what kind of process that should be supported.
- Tools – the framework must be supported with tools, and it must have clear requirements or specification to reduce dependency on a specific tool. Tool of Reverse Engineering itself should have user model and domain problem model, for example Air Man-Machine Integration Design and Analysis System (Air MIDAS) (Gore, 2002). Air MIDAS is an application to help user or operator to support their task. In Air MIDAS the user model represents: perception modelling, working memory model, and tasks model, the domain model represents: interruption possibilities, environment, vehicle condition, aerodinamics problem, guidance, and terrain.

Support interaction between user and the system. Interaction when we develop a system is required between developer and its client, same with Reverse Engineering process the tool that we will built must support interaction with people. The interaction should guarantee people (client) understanding same meaning or goal from both parties (Margaret, 1994.): point of view changes from different angle to same angle, to control information changes from both parties, provide feedback such as confirmation, warning, suggestion.

In a complex system, understanding the system itself requires another system (Idiagram, 2012), because it requires so many aspects, actions, and information to digest, to solve and to analyze, to get the comprehension. Idiagram explains their holistic approach for system comprehension consist of following elements: Mental Model (to model people understanding) and Conceptual Model (to model the problem itself or the system). Those two models will help people or team work to focus, predict, agree with, decide, organize, define the priority, and planning.

To support system comprehension from a complex system we can use abstraction leveling and decomposition approach

(Wing, 2006). Abstraction leveling means we see the system from global description into more detail or technical information. The level itself depends on the method that we use. For example in Software Engineering we know Requirement, Analysis and Design, and Implementation. Decomposition see a system from several part or module or package in a software terminology.

Framework in Reverse Engineering

We found that Reverse Engineering frameworks are created depends on its own goal, here are several examples related framework that exist in Reverse Engineering:

- Database Reverse Engineering Framework (Chiang *et al*., 1996) framework to produce database design. It consists of eight steps: choose the right situation when Reverse Engineering can be applied; choose conceptual model as the result; define prerequisites; domain semantic acquisition; create or choose heuristics and rules for Reverse Engineering; performance efficiency of the process itself; completeness and robustness of the results; and the last step is validation.
- EVOLVE (Wang *et al*., 2003) is a framework for visualisation. It has five main components or modules: Data Sources, Data Protocol, Data Platform, Visualisation Protocol, and Visualisation Library.
- FASAR (Kang *et al*., 2009) is a framework for architecture reconstruction. FASAR consists of three steps to reconstruct the architecture. First step is system characteristic extraction, second step is select the proper tools in order with previous result, and the third step reconstruct the architecture using the tool. FASAR also define there are two types of view that should be produced using the tool, dynamic veiw and static view. Static view relates with system architecture, dynamic view relates with system execution properties.
- VizzAnalyzer framework (Panas and Staron, 2005) for customization, the goal of this framework is to answer several performance issues when we do Reverse Engineering. The issues such as: development efficiency such as time and errors that exist, product quality such as parsing time and memory consumption. To answer the issues metrics are required.
- MemBrain (Minhancea, 2008) the goal is to provide representation and framework for data flow Reverse Engineering. The representation consists of instruction sets as low level representation and translator as converter to MemBrain representation. The framework consists of following concepts: BasicBlock, ControlFlowGraph, InOutSet, DataFacts, DataFlowAnalysis, ComputedValue, Instruction, InstructionVisitor, Addition, and InstanceOf.

One of our contribution is a framework that may support system comprehension via Reverse Engineering. We define our holistic approach according to chapter 2. Using our approach, several problems can be handled:

- Provide a navigation diagram to support exploration. User can be more effective to learn a new concept, what concept that should be learned or related with his question.
- We may able to quantify a comprehension of a concept in the system against user exploration. That is why in our holistic approach User Model and Conceptual Model should be developed using same notation.

Conext Framework

Our hollistic approach named ConExt framework. The framework is built based on several components, they are as follow (Figure 2):

- Input: source code and user guide. Source code is main input for diagram or concept representation on each level of abstraction. User guide is used for develop subsystem concept or representation.
- Modelling: User Model and Conceptual Model. User model consists of: Task (list of tasks or goals which concept that will be explored or digest), Scope of comprehension, User types, Task structure, and Memory model. Conceptual model consists of: representation of source codes: file, variable, library, function, etc; Logical view – data flow and control flow (Kruchten, 1995); Development view – architecture or how we organize to source code (Kruchten, 1995); Requirements model/system behavior – system modelling based on system requirements (in natural language) to reduce or to avoid ambiguity and to support decomposition complex problems (Myers, 2010); System requirements represented in natural language – it is more easily to digest than the formal form.
- Methodology: Translation from source code to each level of abstraction, level abstraction refer to conceptual model explain previous; interaction between user and system; how to represent user understanding in system; and decide scope of conceptual model that can be learnt.
- Tool: an application to help user to get comprehension according to their goal. Main features of the tool are as follow: as repository – source codes, artifacts, and concepts that will be used for exploration; visualize the diagrams or artifacts; update or create conceptual model according to the input (source code and user guide); update or create user model according to user interaction.

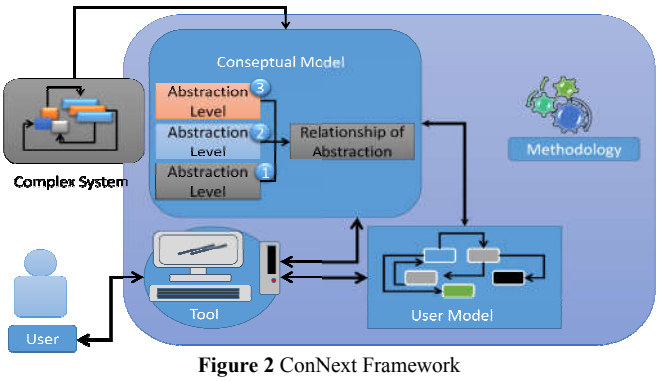

Figure 2 ConNext Framework

Input

Source code file must be organized into repository, it will enable user to see relationship between artifacts or concepts during the exploration process. User guide is required to generate subsystem concepts and system requirements in natural language. Subsystem concepts help user to create clear boundary of exploration or comprehension process.

User Model

User Model consists of several components (Figure 3): Task, User Types, Scope of Comprehension, Task Structure, and Memory Model. Before User Model exists, system must provide Concept Structure that excerpted or created based on Artifacts that exist in every levels of abstraction. The Concept Structure will be copied partially to Task Structure according to a set of constraints, such as: what the User Types and what concept the user want to know.

User explores the concept according to Task Structure that has been created. The exploration is recorded as a Memory Model.

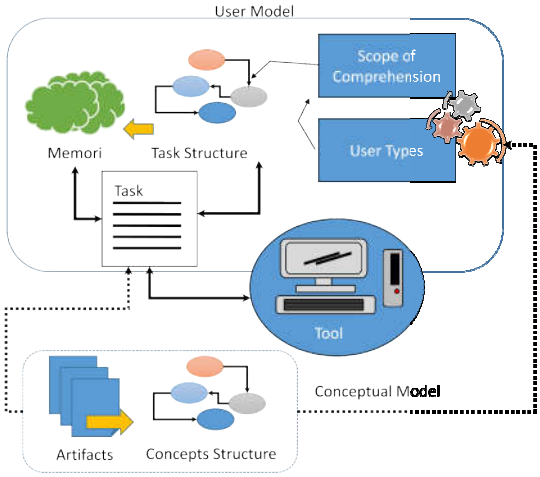

Figure 3 User Model

Here is a brief description for each User Model component:

- Tasks – a task is a set of concepts (exist in the system) that are questioned by user and the user wants to explore relationship with other concepts. A user may have more than one task that recorded in the tool. For example a programmer want to know concepts that may exist in a file main.c and in a file calculate.c. It can be recorded as two different tasks. Another example a requirement engineer want to understand customer concept and product concept that may exist in the system. Those exploration goals can be recorded as two different tasks.
- User Types – every user types has different starting point of level abstraction. We have three kind of user types: Requirement Engineer, System Analyst, and Programmer. For example a programmer start the exploration from source code representation level, but for a requirement engineer – they may start the exploration from system requirement representation level.
- Scope of comprehension is a definition that attached with the user types. For example a requirement engineer may not explore source code representation level. He or she just explore from system requirement representation to logical view representation level. A programmer may not explore system requirements representation level. He or she just explore source code representation until logical or development view level only.
- Task structure is a structure or relationship between concepts that created because user explores the artifacts related to a specific task. Task structure is a partial copy of Concepts structure, as mentioned before it created based on User Type and Task.
- Memory model is a concepts structure that represent explored concepts, in another word as a subset of task structure. Figure 4 show relationship between Artifacts – Concepts Structure – Task Structure – Memory Model. From artifacts of every level of abstraction we can create Concepts Structure as source of Task Structure. Memory Model is created from Taks Structure.

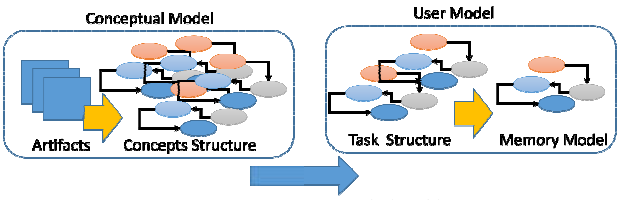

Figure 4 Structures Relationship

Conceptual Model

Figure 5 shows that Conceptual Model consists of system description with several levels of abstraction and conceptual structures that may exist to describe concepts relationship. More higher the level of abstraction means it is more abstract or less detail than lower level. All levels of abstraction are described with its own artifacts: system requirement level (the highest level) describe in requirements document – explains in natural language; system behavior level describe in diagram for example Behavior Tree Diagram [34]; logical view level describes with diagrams such as class diagram, sequence diagram, etc; development view describes with diagram such as package diagram, architecture diagram, etc; and file representation view describes with calls diagram, function and library relationship diagram, etc; the lowest level is source code.

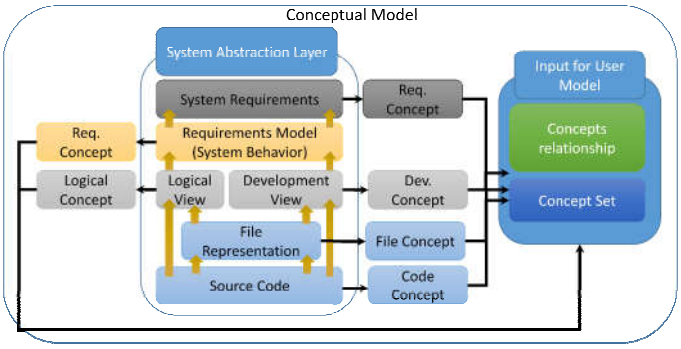

Figure 5 Conceptual Model

After all artifacts are created or available, the translation process from artifacts become concepst structure can be started (Figure 6). Artifacts on the level of abstraction is parsed become a concepts set (for example source code representation: variables, functions, etc). Currently Concepts structure still build using full user interaction – semi automatic tool will develop for further research (at this moment the tool only as concept and diagram container). User build one by one a non complete structure (decomposition diagram) but it will create whole concept structure automatically on each level of abstraction.

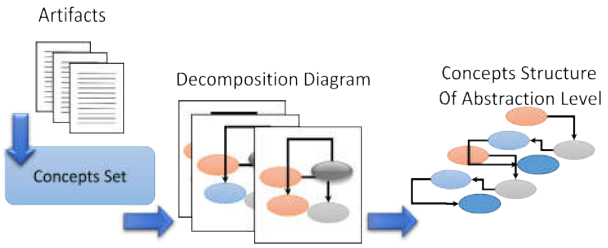

Figure 6 Concepts Level of Abstraction– Concepts Structure

Methodologies

In this framework there are several methodologies introduce, because there are so many processes need to be done:

1. How to use the Application or Tool.
2. How to create Conceptual Model.
3. How to create User Model.

How to Use the Application or Tool

- Preparation: Upload the source code into the application or the tool. It is very important to a user to read the source code when user start exploration.
- Create Conceptual Model as domain knowledge or information as a subject to learn:
- Create file representation view based on source code and file representation concepts set and its structure between the set member.
- Create logical view based on file representation concepts and source code. In this step we also create logical view concepts and its relationship.
- Create development view based on logical view concepts, file representation concepts, user guide and source code. Development view concepts and its relationship has to be created also.
- Create system requirement and system behavior diagram based on all previous views and all concepts that exist in all levels. The requirements concepts also created in this step.
- Create User Model that help user in the process to understand a system or an application:
- Get information of User Type from user, system uses it to decide what kind information or scope of the information as their goal. For example the user type is a programmer, it means he will be restricted only access concepts from Development view, Logical view, and File representation view.
- User record or create a task or a query that will be replied with a Concepts Structure by the application to help them to explore the artifacts. The application or tool allow user to create more than a task in a time depend on his role in a development team.
- Create concepts structure or concepts relationship related with user type and task. The concepts strutcture is use to help user to coupe the comprehension of system.
- Memory is created to record which concepts that have been explored or visited. Each concept may has status that explain which one is already visited and which one is not yet.

How to Create Conceptual Model

The translation process from Artifacts become Concepts Structure has been described on Figure 7. In this section we describe how to create relationship and transformation from concepts in lower level with concepts in higher level. Figure 8 shows the process.

For example we want to create relationship between source code in C Language with Class Diagram on analyst level (or Logical View). In this case source code is the artifact of the lower level and Class Diagram is the artifact of the higher level. Our approach is to create several layer of Concept Structure that can be used to close the gap between those two abstraction layers[1].

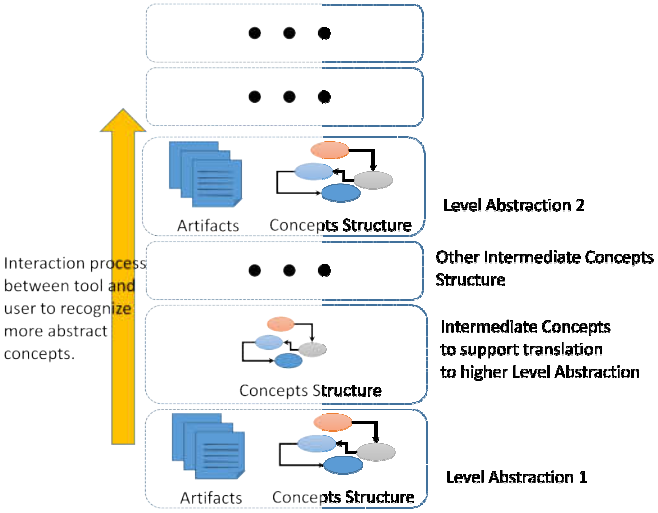

Figure 7 Conceptual Structures Relationship

How to Create User Model

As mentioned previously, User Model components are Task, Task Structure, User Type, Scope Comprehension, and Memory. User Type and Scope Comprehension are something static, but Task Structure and Memory Model are something dynamics. Task Structure and Memory Model will evolve according to how many artifacts that have been visited and how many concepts have been explored.

Figure 8 describes the life cycle of User Model according to a Taks. After user enter a query related artifacts are presented to user. User choose one of the artifacts, than related Concepts Structure are copied and add to Task Structure. User may browse relationship that exist in the new Task Structure Diagram and may choose one or several concepts. Several artifacts will be provided and the process start just like the beginning.

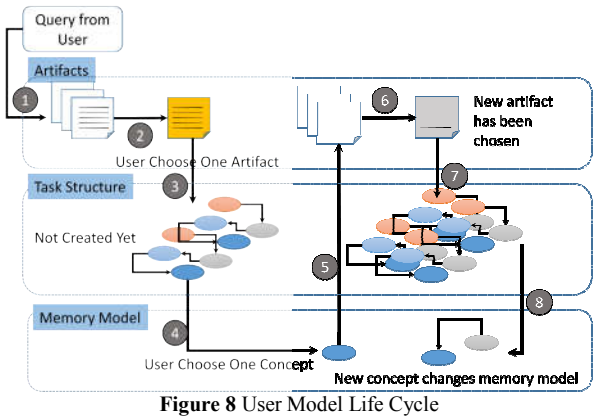

Figure 8 User Model Life Cycle

[1] Notation, diagrams, level of abstraction, and the translation process still in our research.

Same query may produce different result according to User Types, because the authorities of which level of abstraction may be explored must comply with Scope Comprehension that always refer to User Types.

User explores the information with help of a Task Structure, during the exploration user may give status of each concept, which one is already understood, and which is not yet. All concepts that have been visited are recorded in memory regardless the status. The exploration can be saved and can be continued to the next logon.

Tool

The main purpose of the tool is to help user when exploring the domain knowledge. It has to have following minimum features: repository for source code, artifacts, and concepts structure for navigation purpose; it support translation from source code and user guide become artifacts and concepts structure; the tool can create conceptual model and user model; the tool must support all methodologies that have been mentioned above.

Summary

Using our hollistic approach in system comprehension we are able to do following activities:

- People improvement: Scope comprehension indicator – that can be supported using metrics, for example by compared the actual specific related concepts structure with explored concepts. User learning strategies – the tool may provide percentage or number to indicate how many concepts that should be visited in the next exploration. To support that metric concepts suggestion list can be provided: for example by provide another concepts that related or connected between two explored concepts.
- Method improvement: reduce interaction between administrator and tool when develop conceptual structure – until now abstraction from lower level to higher level still requires human intervention or analyst. We may improve the analyst process by providing several similarity structures as input for user to recognize more abstract concept. Provide different learning strategies based on user type. An analyst requirements and a programmer requirements to understand a system have different approach or starting point. Our framework support this requirements.
- Tool improvement: discussion with users based on their experience to get ideal requirements what features that should be exist to support their exploration process or comprehension. For example: feature to provide administrator to add a new concept in the concept structure based on user opinion.
- Interaction between user and tool is supported by using User Model as our dashboard to monitor his understanding or comprehension based on his task.
- User Model and Conceptual Model are different system but may affect each other, using separated model between those domain will help us to understand the studied system.
- Abstraction leveling and decomposing activities as two main activities to support the structures that exists in the studied system.

Our future works will involve several case studies to improve all important parts in our framework. Different types of software and size may help the maturity of our notation and diagrams that use for the structures or models, improve the methodologies to handle many type of cases, and tools to help user work more convinience.

References

Armstrong, M.N. and Trudeau, C. 1998. Evaluating Architectural Extractors, Proceedings IEEE: 5th Working Conference on Reverse Engineering.

Baxter, I.D. 1992. Design Maintenance Systems. Communication of the ACM. April 1992.

Berger, B.J. and Bunke, M. 2011. An Android Security Case Study Bauhaus. In Proceedings 18th. Working Conference. Center for Computing Technologies, Universitat Bremen – Germany.

Chiang, R.H.L., Barron, T.M. and Storey, V.C. 1996. A Framework for the Design and Evaluation of Reverse Engineering Methods for Relational Database. Data & Knowledge Engineering, 21(1), 57-77.

Consens, M., and Mendelzon, A. 1993. Hy+: A hygraph-based query and visualization system. In ACM SIGMOD Record, 22(2), 511-516.

Cross, J. H. 1991. GRASP/Ada: Graphical Representations of Algorithms, Structures, and Processes for Ada. The development of a program analysis environment for Ada: Reverse engineering tools for Ada, task 2, phase 3.

Demeyer, S., Ducasse, S., and Lanza, M. 1999. A Hybrid Reverse Engineering Approach Combining Metrics and Program Visualisation. In Proceedings 6th Working Conference on Reverse Engineering.

Dufour, B. 2004. Objective Quantification of program Behaviour Using Dynamics Metrics. Thesis Montreal: School of Computer Science, Mc Gill University.

Foltz, M.A. 2003. Dr. Jones: A Software Design Explorer's Crystall Ball. Thesis Massachussetts Institute of Technology, Doctor of Phylosophy in Computer Science and Engineering.

Gannod, G.C. and Cheng, B.H. 1999. A Formal Approach for Reverse Engineering: A Case Study. In Proceedings of the 6th Working Conference of Reverse Engineering.

Gannod, G.C. and Cheng, B.H. 2001. A Suite of Tools for Facilitating Reverse Engineering Using Formal Methods. In Proceedings. 9th International Workshop on Program Comprehension.

Gore, B.F. 2002. Human Performance Cognitive-Behavioral Modeling: A Benefit for Occupational Safety. International Journal of Occupational Safety and Ergonomics, 8(3), 339-351.

Gorton, I., and Zhu, L. 2005. Tool Support for Just in Time Architecture Reconstruction and Evaluation: An Exprience Report. In Proceedings. 27th International Conference on Software Engineering.

Idiagram. 2012. http://www.idiagram.com/CP/cpprocess.html.

Kang, S., Lee, S., and Lee, D. 2009. A Framework for Tool-Based Software Architecture Reconstruction. International Journal of Software Engineering and Knowledge Engineering, 19(02), 283-305.

Kazman, R., O'Brien, L., and Verhoef, C. 2003. Architecture Reconstrution Guidelines. Technical

Report CMU/SEI-2002. Carnegie Mellon, Software Engineering Institute.

Kruchten, P. 1995. Architectural Blueprints – The "4+1" View Model of Software Architecture. IEEE Software 12, November 1995.

Margaret, T. 1994. Mutual Understanding in System Design: An Emperical Study. Journal of Management Information Systems, 10(4), 159-182.

Mendelzon, A. and Sametinger, J. 1995. Reverse Engineering by Visualizing and Querying. Software Concepts and Tools, Springer Verlag.

Minhancea, P.F. 2008. Towards a Reverse Engineering Dataflow Analysis Framework for Java and C++. In Proceedings 10th International Symposium on Symbbolic and Numeric Algorithms for Scientific Computing. IEEE Computer Society.

Müller, H.A. 1986. Rigi A Model for Software System Construction, Integration and Evolution Based on Module Interface Specification". Thesis Rice University.

Müller, H. A., Jahnke, J. H., Smith, D. B., Storey, M. A., Tilley, S. R., & Wong, K. 2000. Reverse engineering: A roadmap. In Proceedings of the Conference on the Future of Software Engineering, 47-60.

Murphy, G.C. and Notkin, D. 1997. Reengineering with Reflexion Models: A Case Study. IEEE.

Object Management Group. 2006. Diagram Interchange. Object Management Group.

Myers, T. 2010. The Foundations for a Scalable Methodology for System Design. School of Information and Communication Technology Science, Environment, Engineering, and Technology, Griffith University.

Panas, T. and Staron, M. 2005. Evaluation of a Framework for Reverse Engineering Tool Construction. In Proceedings. of the 21st IEEE International Conference on Software Maintenance. IEEE Computer Society.

Raza, A., Vogel, G. and Plodereder, E. 2006. Bauhaus – a Tool Suite for Program Analysis and Reverse Engineering". In Proceedings 11th Ada-Europe International Conference on Reliable Software Technologies.

Storey, M.A.D. and Müller, H.A. 1995. Manipulating and Documenting Software Structures Using ShriMP Views. In Proceedings. of the Internation Conference on Software Maintenance.

Storey, D., Wong, K. and Müller, H.A. 1997. How Do Program Understanding Tools Affect How Programmer Understand Programs? In Proceedings. of the 4th Working Conference on Reverse Engineering.

Systa, T. 1999. On the Relationship Between Static and Dynamic Models in Reverse Engineering Java Software. In Proceedings. 6th Workng Conference on Reverse Engineering. IEEE Computer Society Press.

Team, C.P. 2011. CMMI for Acquisition Version 1.3. Software Engineering Institute, November 2010. CMU/SEI-2010-TR-032, Lulu.com.

Wang, Q., Wang, W., Brown, R., Driesen, K., Dufour, B., Hendren, L., and Verbrugge, C. 2003. EVolve: an open extensible software visualization framework. In Proceedings of the 2003 ACM symposium on Software visualization.

Wing, J.M. 2006. Computational Thinking. Communication of the ACM.

How to cite this article:

Tenoyo, B., Mursanto, P and Santoso, H.B (2017) 'Conext (Concept Explorator) A Framework For System Comprehension With Holistic Approach In Reverse Engineering', *International Journal of Current Advanced Research,* 06(12), pp. 8670-8676. DOI: http://dx.doi.org/10.24327/ijcar.2017.8676.1404

****