International Journal of Engineering and Applied Computer Science (IJEACS)

Program Comprehension and Analysis

Asfa Praveen
Dept. of Computer Science
Shri Venkateshwara University, Gajraula
Amroha, U.P., India.

***Abstract* - Due to continuous change in business requirements and to implement proactive, adaptive maintenance procedures in application program; it is required to evolve application program time to time. Program comprehension is a process of program understanding and reverse engineering, which supports the analyst to easily undertake the program for further reengineering. This paper highlights the program elements, components, its analytical solutions for understanding, comprehensions and extension.**

Keywords- component; wrapper; comprehesion; application; code.

I. INTRODUCTION

Program comprehension process uses the term component to denote the various elements needed to develop a program. Comprehension is a process of analyzing program components and reverse engineering. Analyze data are recorded in knowledge base. To better comprehend the program, it is core issue to find the elements of program which mostly depends on perspective of understanding and investigation. Subject of concern is to investigate the candidate program, identify program elements, slice it and fed them in to a knowledge base. This knowledge base can be further managed to find the information of program for comprehension purpose. The effectiveness of method depends on program under study, its logic, structure, programming language used, problem solution implementation mechanism, what program does and how the program does. Program comprehension process is a highly cognitive task; conceptual knowledge based on human cognitive efficiency can't be overlooked. Analyst must understand the program structure, flow and environment.

Comprehension process [1] also based on objects identification, class investigation, study of components formation process, elements interactions and flow control. Program elements also include database descriptors, program specification blocks, screens, file definitions and message formatting services. These also represent the components associated with system-level understanding.

II. PROGRAM AND ITS RELATIONSHIP LAYER

Program comprehension layers can be categorized as:

1) Program elements and their relationships
2) Program control, data flow and business modules
3) Program process flows and business rules

In the process of program comprehension, analyst identifies

a) Unknown elements
b) Unknown relationships

The various levels of comprehension identify all level of relationships with application interfaces, this is significant when existing software is needed to be reengineered or subjected to migration towards wrapping technologies. It also establishes the migration opportunities to updating software. This stage of comprehension needs identifying the relationship among program elements. In homogeneous system architecture, identifying and analyzing this type of relationship is easy to obtain and record, while in heterogeneous program, it is very tedious task.

Next step for the comprehension process is to study about control flow, data description, data flow, business logics and modular structure; integration of components and program code. With the analysis of program code and connection of relationships the understanding of control and data elements will take place. This maturity of comprehension is important to develop a big understanding level of program elements integration and deployment. This is also useful when existing systems incorporate new components through wrapping.

Process flows investigation propagates to write line by line description of code components, inner process flow and business rules integration. Modifications in the program code which have occurred time to time generally increases the level of complexity of comprehension and reduce understandability level [2]. Syntax, semantics and statement logic organization are necessary to analyze and document to find a relationship with domain of program classification.

III. PROGRAM COMPLEXITY

75-80% of total software cost dedicated to maintenance and reengineering in a software life time. 40-60% of total reengineering and maintenance effort spent for program comprehension process. Program complexity plays a major role in comprehension of candidate program. The discipline of software measurements has analyzed the levels of complexity with many specified software and conceptual tools such as cyclomatic complexity.

ISBN: 978-0-9957075-0-4

Cyclomatic complexity represents the number of independent paths through a program. The Software Engineering Institute (SEI) provides the following range for identifying complex programs and associated risks.

TABLE I. COMPLEXITY EVALUATION

Cylcomatic Complexity	Risk Evaluation
1 – 10	Simple program which don't have so much risk
11 – 20	This is more complex program which has moderate risk level
21 – 50	This is a complex program with high risk
> 50	This is an un-testable program with very high level of risk

(Source: SEI CMU)

This Program comprehension has become more complex process when it is needed to analyze business rules and migration to new technologies. Study of program, its level of effort, which is needed for comprehension study, is facilitated by cyclomatic complexity analysis. Cyclomatic complexity results play crucial role in program study and understanding process.

IV. MAJOR PROGRAM ELEMENTS

In the abstract level program analysis procedure focuses on the following three major elements of application structure.

a) Program presentation
b) Program logic
c) Program data and flow

Above mentioned elements are very helpful in identifying business process and analysis of rules in candidate application program. Program logic is a cognitive entity which has high level of abstraction of data, control flow and application integration [3]. Other graphical user interface extension capabilities allow more pleasant and accessible performance for many initiatives. Separating the performance logic allows the various initiatives to increase the complexity of their integration capabilities with other program elements. Further separation of business logic compromises the probability for assessment of integrated elements [4] which can be reused during program integration initiatives. It is assumed that understanding the data is not considered as a part of comprehension study but it is required because of data flow and file management [5]. Because of many strict causes data needs profiling, which is a task of filtering and summarizing. Renovation phases need many stages to cover to comprehend code elements. The following headings explain the level of comprehension required for each stage [6].

A. Extensions

Program comprehension process improves with help of component extension approach. Extension if properly documented and implemented with existing program can give a better realization of program defects. Front and back end of database applications need extensions time to time when evolution is required for business rule applications.

B. Integration

Current application and program comprehension knowledge at most level of initiatives are not adequate to apply the increased levels of system evolution. Whether determined by application integration requirements, analysis for program elements replacement activities for identifying wrappers and integration to migrate system program. There is a strong comprehension of application data flow and control analysis is needed for proper logical understanding of element integration.

C. Vendors Trends

Many vendors take application integration initiative to compete the demands of new business objectives. Vendors adopt the generic technologies and multidisciplinary initiatives as some distributed application program integration process need some Java involvements. Comprehension of flows, interface and logics and relationships in old technologies and databases need advanced tools. Vendors trends changes according to development process and practices affects the comprehension and differs the practices.

D. Restructuring

Restructuring of programs is studied to assess interfaces, program insulation from its surroundings, isolation of individual functions from each other, and what are prevented undesired side-effects. Extracting knowledge from program requires practically structured code [7]. Program rationality is hard to measure without objective software metrics. Extraction process uses these metrics to control the extremes and implementation of complex system application. It reduces the cost of reengineering by dropping the program complexity.

For programs with high-complexity metrics, analysis proposes to wrap them in the same condition in spite of extracting them for reusability. Much research effort has been applied into considering the basis of extracting elements or developing wrappers. Covering the presentation layer with business logic and data steering logic originates most analysts to keep away from restructuring the consequences of scenarios. The previous research returns the results with programs or well-structured code understanding. Start with user interface and go to the code base. Analyze sequence and state diagram of application. Need to comprehend the following elements of restructuring for reengineering.

a) Class inheritance
b) Control Flow tree
c) Form show tree
d) External class metric

ISBN: 978-0-9957075-0-4

V. Analytical Solutions

A. System Extensions

System needs to be extended for users those who are external to the organization want to access the system and associated program application; these are categorized as untraditional users. It has been become very difficult to study the character-based data and covert them to more meaningful form. There are some needs to convert a character based interface to visual interface, which has changed variety of sources of business information which need to being made available outside traditional channels. These business process changes possibilities have changed the flow of work and data control; this has become to a source to study further future changes in program [8]. Hyper Text Markup Language based documents based presentation has not limitations which can be presented using data streams application programming interface. If it is required to update existing system in a new ways program and surrounding analysis is needed for systems modifications. Software wrappers can support up to some extents but do not fulfill all demands in all perspectives.

B. Wrappers

Several types of software wrappers are available which can be integrated to application and program, which is a depiction of the process but not the data. Object oriented methods are extended with wrappers to use the existing procedural. These wrappers generate a fulfillment of gap between traditional procedural programs and Object Oriented methods. Procedural programs are purposed to perform some action to solve some particular problem such as processing of database queries. The actions on data are in procedural order. The wrapper exists to hide one method from other code in action. This is done in n-tier architecture to apply separation of concerns. It is accepted widely, program logic represent data to the traditional user, who should not have to check that from where the data is coming and similarly program code who retrieving the data should not care what is the display of this. Wrappers support for some specific features as some tile language does not provide the multiple inheritance but it can be simulated with the help of wrappers.

C. Black-Box Method

Extending the business logic needs the use of analysis or code comprehension tools and summary of reports [9]. The efficacy of a comprehension capability depends on the ease with which the analysis has been performed. Batch and concurrent processes, database transactions, application programs or even methods subroutines can be analyzed for reengineering towards new applications. These approaches minimize the efforts of understanding needed. This reduces internal complexity of comprehension which is time consuming process.

D. White Box Method

Analyst can understand the leveraging of business rules of existing program, but needs to remove from the limitations of the program surrounding. Migration existing business rules and functionalities to new platform requires the understanding with wrapping and comprehension to program translation. Further translation requires the program concepts identification, understanding business functions, rules with identifying the data items. This identification is critical to program redefinition. This method of white box approach is performed for detailed level of understanding, reusable components.

E. Performance and Scalability

The transaction analysis solutions must communicate a session to program interaction. Many results may limit scalability to manage a dynamic session's management for always changing source information and interactions. Other than this the solutions must provide clustering and load balancing capabilities for new application to adapt the new demands and growing new extensions in services.

F. Solution for Data ExtensionUnderstanding and Recovery

Data integration analysis is a problem of assortment such as variety of data sources and uses types. Data analysis tools can be applied to separate the data source from interface, backend and application layers and the client interface layer from the integration of transaction server. This enables enhanced growth in analysis of data sources and clients implementations. The main data sources are virtual machine, recovery files, transactional servers, extension applications, component objects, query processers source optimizers. Various analysis technologies allow the understanding of used data models in the all layers of file management systems. These models can be converted into logical models for further analysis and extension, solution of end term extension of application and program slicing. Many analyst uses a set of data implementation allows the extended use of the actual data for either server reporting or for transporting of data to new application migration.

G. Transaction Anslysis

Extracting information from repository which usually created after analysis is very useful for knowledge base development. Migrating from straightforward interactions of program with candidate application is done by this procedure. This requires more programmatic analysis to ensure integrity of objects. Software analysis enables existing business rule logic to be reused and unchanged functionality. The program interface business logic mainly requires modification to support input validation parameters. Analytical input requirements are major concerns for reengineering when using advanced programming tools develop wrapping solutions. Information systems analysts are experienced to enable increased use of traditional systems components for reuse to new environments. Enlarged complexity by duplicate business logic and maintenance of data inconsistencies become difficult to manage for post analysis procedures.

Program can be observed as the point of integration rather than simply as a means of accessing data using traditional

ISBN: 978-0-9957075-0-4

transactional programs. Analysis procedures have to evolve for transaction server to enable connectivity of lost data identification. In order for the comprehension to get complete use of evolved capabilities of leveraging candidate programs, it must isolate communication logic from the outstanding business logic for ease of implementation. Most analysts have opted to delay this procedure to use the existing logic as a program implementation. The continuing evolution of analysis procedures results for possibilities of technology and business logic should change at specific layer of concern. The loss of control of the flows changes the demands on this program analysis effort for migration. While many tools application offer more refined integration of knowledge to repository but this is not manual tracking solution.

H. Program Analysis Issues

Procedure of program analysis is for automatic extraction of useful information from program and supply for further extraction for knowledge generation. Data and program control flow information are considered as major entity of observation to understand the sequence of action. Some of the major issues are:

a) Session establishment management
b) Performance and scalability
c) Control building

I. Sessions Establishment Management

One of the technical problems related with application analysis solutions is to find out the differences between the session-based management of applications and the session-less management for networked program [10]. The transaction analysis solution must assure session establishment to client program and interactions, to maintain the integrity of the data, application, and sessions.

J. Control Building

Analysis procedure implements methods through a more traditional ways to call for existing procedural program encapsulation and control integrity. Comprehension depending on the current implementation of the program design and logical framework, reengineering the interface may be applied for that. Program modules are mostly easiest to track the flow of control, or encapsulate, because they are probably designed to independent connected modules. Encapsulating program modules is the most difficult, because it normally needs restructuring of existing program to enable external procedures to be internally incorporation with suitable parameters

The following guidelines should follow in building control:

a) Object oriented analysis and implementation should separate the data only to the methods that use them.

b) Program slicing tools are required to extract business logic. Object identification is not clear from procedural implementation.

c) Unstructured code must be corrected, to separate, code slice with methods.

d) Naming validation is essential for more strong association with attributes and methods.

e) Reengineering of complete application should not solve the issue but need to connect the entities with procedural framework as defined in control flow.

f) Deadlock need to be identified and removed from program code structure and execution flow.

g) Traditional procedural programs control in many cases uses the same flow in the same field for different code segments, which results in different methods of analysis. These field usage problems need to be understood to enable proper separation of concern.

VI. CONCLUSION

Program analysis is an important and useful task for program comprehension. Wrapper provides useful extensions to the existing capabilities of the program for functionality enhancement. Program comprehension is a necessary and significant requirement; if the document of application is not available at that time the comprehension procedures are very useful to draw program control and logic design. This paper has presented some useful methods and solutions for program comprehension and elements analysis.

REFERENCES

[1] A. S. Alardawi and A. M. Agil, "Novice comprehension of Object-Oriented OO programs: An empirical study," *2015 World Congress on Information Technology and Computer Applications (WCITCA)*, Hammamet, 2015, pp. 1-4. doi: 10.1109/WCITCA.2015.7367057

[2] A. Shargabi, S. A. Aljunid, M. Annamalai, S. Mohamed Shuhidan and A. Mohd Zin, "Program comprehension levels of abstraction for novices," *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, Kuching, 2015, pp. 211-215. doi: 10.1109/I4CT.2015.7219568

[3] T. Roehm, "Two User Perspectives in Program Comprehension: End Users and Developer Users," *2015 IEEE 23rd International Conference on Program Comprehension*, Florence, 2015, pp. 129-139. doi: 10.1109/ICPC.2015.22

[4] F. Fittkau, S. Finke, W. Hasselbring and J. Waller, "Comparing Trace Visualizations for Program Comprehension through Controlled Experiments," *2015 IEEE 23rd International Conference on Program Comprehension*, Florence, 2015, pp. 266-276. doi: 10.1109/ICPC.2015.37

[5] I. d. M. Lessa, G. d. F. Carneiro, M. J. T. P. Monteiro and F. B. e. Abreu, "A Multiple View Interactive Environment to Support MATLAB and GNU/Octave Program Comprehension," *Information Technology - New Generations (ITNG), 2015 12th International Conference on*, Las Vegas, NV, 2015, pp. 552-557. doi: 10.1109/ITNG.2015.93

[6] *A. Shargabi, S. A. Aljunid, M. Annamalai, S. M. Shuhidan and A. M. Zin, "Tasks that can improve novices' program comprehension,"* 2015 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), *Melaka, 2015, pp. 32-37. doi: 10.1109/IC3e.2015.7403482*

[7] *R. Kadar, S. M. Syed-Mohamad and N. Abdul Rashid, "Semantic-based extraction approach for generating source code summary towards program comprehension,"* 2015 9th Malaysian Software Engineering

Conference (MySEC), *Kuala Lumpur, 2015, pp. 129-134. doi: 10.1109/MySEC.2015.7475208*

[8] *X. Liu, X. Sun, B. Li and J. Zhu, "PFN: A novel program feature network for program comprehension,"* Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on, *Taiyuan, 2014, pp. 349-354. doi: 10.1109/ICIS.2014.6912158*

[9] Y. Liu, X. Sun, X. Liu and Y. Li, "Supporting program comprehension with program summarization," *Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on*, Taiyuan, 2014, pp. 363-368. doi: 10.1109/ICIS.2014.6912159

[10] *N. Saroni, S. A. Aljunid, S. M. Shuhidan and A. Shargabi, "An empirical study on program comprehension task classification of novices,"* 2015 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), *Melaka, 2015, pp. 15-20. doi: 10.1109/IC3e.2015.7403479*

ISBN: 978-0-9957075-0-4