Research Journal of Applied Sciences, Engineering and Technology 12(3): 361-365, 2016
DOI: 10.19026/rjaset.12.2344
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: August 28, 2015 Accepted: October 11, 2015 Published: February 05, 2016

Research Article

GATALSS: A Generic Automated Tool for Analysing the Legacy Software Systems

Malladi Srinivas, G. Rama Krishna, K. Rajasekhara Rao and E. Suresh Babu
Department of CSE, K L University, India

Abstract: The software development is primarily more important for the business organization due to its dynamic changes that has to be carried out in a less span of time. Consequently, software reuse will improves the quality of the product, increases the productivity and reduces the costs in an organization. This study presents generic automated tools for analyzing the legacy software systems, which consists of loader, set of parsers and dependency mapper that visualize and transforms different legacy codes. Moreover, we have adopted ANTLR tool to analyze the Legacy Software Systems using multi-threaded environment.

Keywords: Legacy software systems, multi-threaded environment, ANTLR Parser, performance analysis

INTRODUCTION

Legacy system are significant in the operations of most enterprises. Most legacy systems were developed utilizing programming paradigms and languages that lack adequate designates for modularization. Consequently, there is diminutive explicit structure for a software engineer to hold on. This makes effective maintenance or extension of such a system a vigorous task. The modernization is often a sizably voluminous, multiyear project, because deploying the modernized system all at once introduces an unacceptable level of operational peril. As a result, legacy systems are typically modernized incrementally. A migration strategy must ascertain that the system remains functional in all aspects, during the modernization effort. Several approaches to transmute legacy systems have been proposed which can be relegated as redevelopment, wrapping and migration (Bronius and Aurimas, 2006).

Refactoring existing legacy systems to new platform (such as refactoring to web services) first and foremost thing is to understand the existing system in all aspects. In order to analyse the existing system, many researchers afforded their work in this domain to propose different techniques/methodologies. Understanding the legacy system needs expertise on the key functional areas of the domain as well as the legacy platform, Technology and its architecture. Human effort and time will directly impact the cost of the whole process. It is very difficult to get resources who has the knowledge on the legacy system. Hence it is necessary to automate the whole process of analysing and understanding the legacy system.

The scope of this study will help to understand the analysis of legacy system, enhancements with the existing techniques. Hence, the proposed method provides an effective tool for legacy system analysis (software archaeology), extending some of the open source techniques available in the market such as ANTL (Another Tool for Language Recognition).

LITERATURE REVIEW

Analysis of the legacy systems helps to understand the exact situation of the system and its entire process. It lays a great foundation for the further development, advancement or shift to the advanced technology. This section gives literature review already proposed by research committee. Few of them are listed below.

Cuadrado *et al.* (2008) proposed a technique to the recovery of the legacy system architecture which ensures proper documentation of the legacy system. For better understanding of the legacy system, the Author has extracted low-level details from configuration options, users' manual and source code. In addition to that high-level Details acquired from requirements or design documents. He used Different tools for recovery process, the selection of the tool depend on technologies, size, documentation, etc. The proposed recovery technique uses the QAR tool for documentation analysis, static analysis and dynamic analysis. For Information extraction, the prime resource used is user manual. They used a profiling tool for extracting runtime information and reverse-engineering tools like Jude and Omondo UML Studio to extract the static view of the system, which will automatically analyze Java source code, generating UML class diagrams at class and package levels. The above tools detects inheritance and dependency relations between elements. The Java Profiler of TPTP 4.2 tool was used for Dynamic analysis, which captures run time

Corresponding Author: Malladi Srinivas, Department of CSE, K L University, India

information-method invocation, execution time and number of instances in memory. Higher level architecture was obtained from the abstraction process which consists of a series of filtering actions. The analysis was executed combining both the static and dynamic system views.

Salama and Aly (2008) proposed a mechanism called Modernization Strategy Selection Framework (MSSF). And Decision Making Tool, which analyses the source code to measure its quality. The source code selected was measured using the CCCC API. The author suggested a hybrid modernization approach, for the organizations to evaluate each component vs. service in a separate thread to focus on its criteria's values alone.

Alahmari *et al*. (2010) proposed methodology and effective guidelines for the identification of precise services from legacy code. They introduced Meta model that defines the characteristics of business processes and service types. The approach focuses on identifying these services based on a Model-Driven Architecture approach supported by guidelines over a wide range of possible service types. They emphasized the importance of the classification of service types to define service properties correctly. They used UML activity diagrams that identifies coarse-grained services, BPMN business process diagrams were used to identify fine grained services, as well as coarse-grained services (as composite services).

Lewis *et al*. (2006) used SMART tool to analyze the legacy components for determining changes to be made to enable migration of legacy system. This SMART tool uses three sources of information to support the analysis activity-First Information related to the issues, problems that were noted by the team in discovery process, second information provided by a Service Migration Inventory (SMI) that extract the many desired behavior of services executing within SOAs into a set of topics and third source of information entails the use of code analysis and architecture reconstruction tools to analyze the existing source code for legacy components.

Lewis and Smith (2007) proposed a SMART technique with different behavior which assists organizations in analyzing legacy capabilities for use as services in an SOA environment. The goal of this activity is to obtain descriptive data like the name, function, size, language, operating platform and age of the legacy components that are considered for migration. In the recovery process, Architecture code, complexity design paradigms, level of documentation, coupling, interfaces and dependencies between components extracted from Technical personnel. In addition, the tool proposes to extract information related to quality, maturity of legacy components, problems, change history, user satisfaction, longer term needs, cost for maintenance and finally effort needed for these analyses.

Marchetto and Ricca (2008) an approach consists in analyzing Legacy system application to understand the system architecture and recognize the functionalities. They proposed to recover functionalities from textual use cases, functional requirements in natural language and user manuals. They have chosen to represent the application functionalities and their relationships (e.g., extend and include) using an UML Use Case diagram. When the size of the application is considerable, documentation not up to date, the Author suggested Reverse engineering techniques and tools to recover and represent the architecture of the application.

Nguyen *et al*. (2009) proposed a novel business service engineering methodology -GAMBUSE that identifies and conceptualizes business services in a business domain, is based on a stratified reference Service Meta-Model (SMM) that specifies and correlates all modeling constructs for business processes. During this step, the Service Schema Specifications (SSS) of the as-is and to-be process that contain their activities, business entities, attributes, constraints, business rules, etc., are instantiated from the SMM.

Huang *et al*. (2008) proposed approach to identify the component (s) to be reused, analyze the dependencies on other components, Identify the depended components and generate legacy surrogates for depended components.

Idu *et al*. (2012) proposed two mechanisms. One of them is reverse engineering which performs its actions by understanding and analyzing the source code. Other is knowledge based which performs its action by understanding the knowledge on domain and by analyzing the experience of the initial promoters or developers. By understanding the end user experience with the help of the documents available/interview process, one can understand the applications of the legacy systems.

Vemuri (2008) used theory of Feature analysis to locate the features from a legacy system which identifies the key features of the legacy system from end-users, the domain knowledge and expertise that exists in the legacy system team, Identify test cases from the regression test suite that represent a particular feature or a group of features., They proposed code-profiling tools which create internal product scripts that would trace the code traversal paths and frequency via various modules and packages of the legacy system., Execute the test cases identified for each feature and collect the metrics generated by the profiling tools and finally the metrics provide an insight into the un-used areas of legacy code, level of code entangling that exists in the legacy modules per every feature.

OVERVIEW OF ANTLR PARSER

Generally, there are several open source tools to understand the legacy systems ANTLR is the powerful tool to perform better analysis of software system. ANTLR (ANother Tool for Language Recognition) dominant parser generator for studying, analyzing, executing or translation of binary or text files. One of the main advantages of this parser is building languages, frame works and tools. Moreover, ANTLR is capable of building parse trees with the help of the input grammar. ANTLR can also used to process abstract syntax trees, which are generated automatically by the parsers. These tree parsers are unique to ANTLR, which simplifies the processing phase of abstract syntax trees.

This study analyses the legacy system. The analysis of the legacy code helps us to solve the issues such as upgrading the legacy system, effective understanding the business rules, proper documentation for the existing undocumented code and finally managing the ripple effect. However to migrate or refactor, maintaining and rewriting the existing system requires good understanding of the structure and its functionality.

PARSER ARCHITECTURE

The following Fig. 1 shows the parser architecture, whose components are described in subsequent sections. This tool takes the legacy systems source as the input, tokenizes the code, does the dependency mapping and rationalization and present the information needed to the target state in the form of process flows, code flows, rules, test cases and code artifacts.

Various components used in parser architecture:

Configuration loader: Loader will first load the configurations, pattern specific/application specific transaction extractor and then loads the project, which needs to be analyzed.

Parsing engine: Parsing engine uses the grammar to parse the source files and extract the information and persist the data into data base for further analysis. The sample information include List of functions, Global variables, Function variables, Function calls, Class Name with package

Database manager: Database manager is responsible to persist all the parsed data tokens to the database and get the required data from the database for further analysis.

Dependency mapper: Once parser completes parsing all the source files, the dependency manager will start to identify the complete hierarchy of each of the service operations, while finding the call hierarchy of each of the operations, it identifies the type of the call based on the given search patterns and gets the transaction name with the help of Transaction Extractor.

Extractor: Extractors will extract the transaction name/External Service Name from the search patterns that are configured. There can be two types of extractors mainly Generic transaction extractors to capture generic types like Http Calls, SOAP Calls,

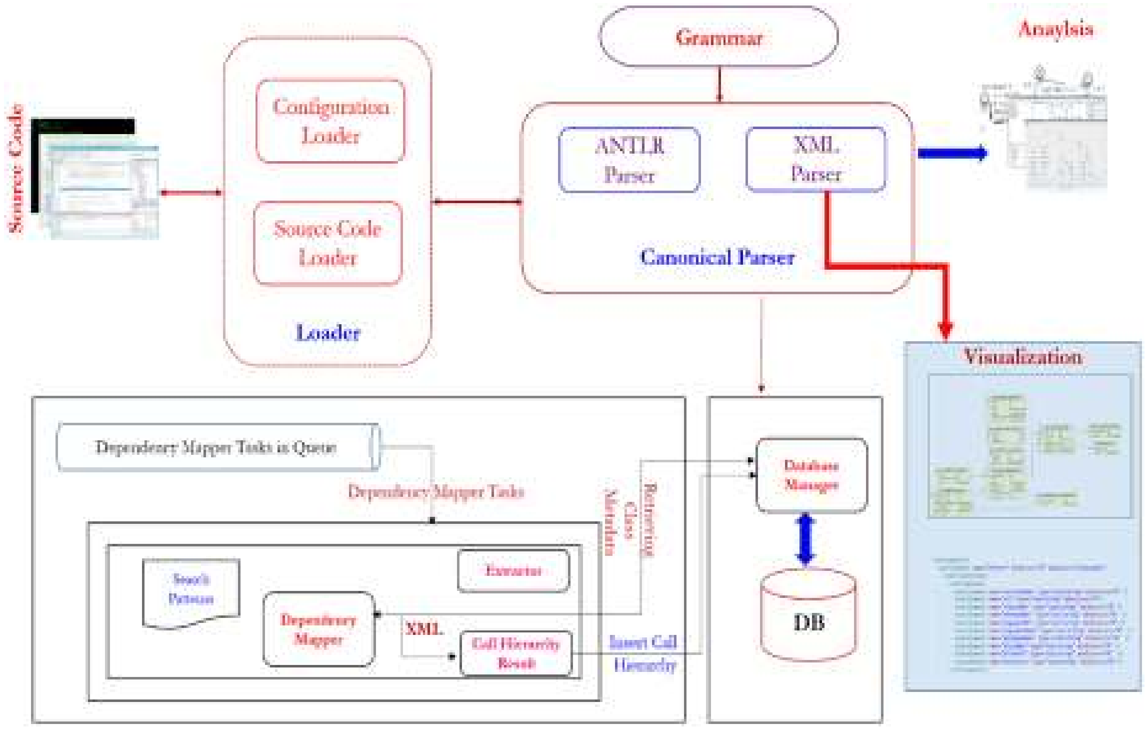

Fig. 1: Parser architecture

Stored procedure calls and Application specific transaction extractor to identify the service name of other applications in the enterprise.

GATALSS Mechanism: We proposed a novel method for analyzing the legacy software systems that makes better scope in the business processes. Many of the software systems become inadequate in terms of capacity and functionality. However, these organizations mainly rely on 'legacy systems' for critical information and business operations support. Hence, they cannot be simply scrapped but need to be enhanced or integrated into the organizational information infrastructure. This mechanism is implemented using ANTLR Parser that provides effective analysis, which improves business information system.

The key aspects of ANTLR, accepts large class of grammars and builds the parse trees for a given input program. Eventually, it is more worth full if ANTLR is integrated with multi-threading environment that creates multiple instances which takes multiple files as input and provide effective outputs for better analysis of legacy systems.

Proposed GATALSS algorithm:

1. Loader Loads the configurations, analyses the project base path and identifies sourcefiles to parse and adds the file path into parser queue.
   Parser starts the threads configured. Each thread will do the below task
   a. It picks the task from parser queue
   b. Parse source and extract the required information and add it into Data Persist queue.
2. One dedicated queue will poll the parsed information from *Data Persist* queue and update the database.
   Note: Both parsing the Data persist will happen parallel.
3. Once the parser completes parsing all the source files then, Dependency manager will start analyse the call hierarchy of the each service operations. Parser will load the task in the Dependency queue. The below steps will be done by Dependency Thread in parallel.
   a. It picks the service from the Dependency queue.
   b. Find out the call hierarchy, type of the call and get the transaction with help of Transaction Extractor.
   c. Writes the Complete call hierarchy report of the current service operations.
4. Generates the final report of the complete service operation details with transactions.
   The output of the tool is represented as an XML file for visualization, Abstract Syntax Tree are effective analysis for source code which can be persisted to the database that are used for later stage.

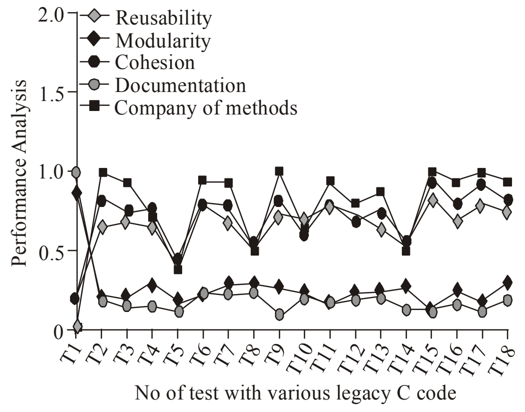

Fig. 2: Performance analysis of legacy C code

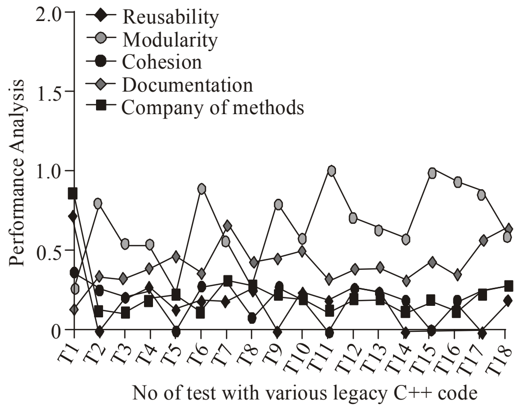

Fig. 3: Performance analysis of legacy C++ code

RESULTS AND DISCUSSION

We have analyzed various Legacy Applications implemented using C, C++ and Java using proposed mechanism. Various tests conducted on application code ranging from simple to complex for the specific language C, C++ and Java. Further we analyzed various performance metrics like modularity, documentation, Cohesion Complexity of method, reusability by taking different applications.

Figure 2 gives the performance analysis of C legacy code, it is observed that the complexity of the methods, cohesion reusability are fair enough compared to application software implemented in C++ and JAVA. Further it is also observed that the documentation and modularity is poor in C code compared to others.

Figure 3 illustrates the performance analysis of C++ legacy code, It is observed that the modularity, documentation are high compared to application software implemented in c and java. Further it is also observed that the Complexity of method, reusability fair and cohesion is low in C++ code.

The performance analysis of JAVA legacy code (Fig. 4) depicts that Complexity of methods, reusability, cohesion are high and Documentation, modularity are fair compared to other legacy codes used for analysis.

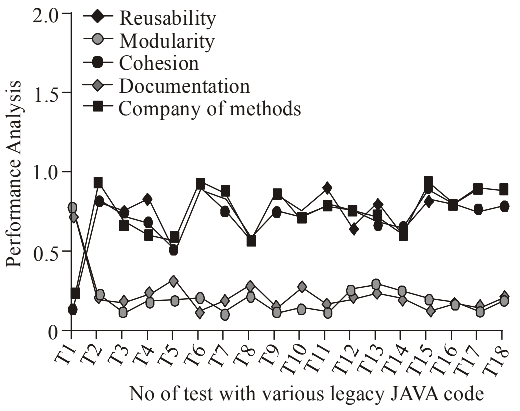

Fig. 4: Performance analysis of legacy JAVA code

CONCLUSION AND RECOMMENDATIONS

This study proposed generic automated tools for analyzing the legacy software systems, which consists of loader, set of parsers and dependency map per that visualize and transforms different legacy codes. Moreover, we have adopted ANTLR tool to analyse the Legacy Software Systems such as c, c++ and java using multi-threaded environment. It is observed that, it is not sufficient to analyse legacy code for processing the business application there is necessity to implement a generic frame work to refactor the legacy code for effective and efficient implementation of business logic.

REFERENCES

Alahmari, S., D.D. Roure and E. Zaluska, 2010. A model-driven architecture approach to the efficient identification of services on service-oriented enterprise architecture. Proceeding of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops. Washington, DC, USA.

Bronius, P. and L. Aurimas, 2006. Business knowledge extraction from legacy information systems. Inf. Technol. Control, 35(3): 214-221.

Cuadrado, F., B. García, J. Duenas and H.A. Parada, 2008. A case study on software evolution towards service-oriented architecture. Proceeding of the 22nd International Conference on Advanced Information Networking and Applications-Workshops. Okinawa, pp: 1399-1404.

Huang, H.Y., H.F. Tan, J. Zhu and W. Zhao, 2008. A lightweight approach to partially reuse existing component-based system in service-oriented environment. In: Mei, H. (Ed.), ICSR, 2008. LNCS 5030, Springer-Verlag, Berlin, Heidelberg, pp: 245-256.

Idu, R.K.A., J. Hage and S. Jansen, 2012. Legacy to SOA Evolution: A Systematic Literature Review. In: Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments. IGI Global, Hershey, PA, pp: 419, ISBN: 1466624892.

Lewis, G. and D.B. Smith, 2007. Developing realistic approaches for the migration of legacy components to service-oriented architecture environments. Proceeding of the 2nd International Conference on Trends in Enterprise Application Architecture, pp: 226-240.

Lewis, G., E. Morris and D. Smith, 2006. Analyzing the reuse potential of migrating legacy components to a service-oriented architecture. Proceeding of the 10th European Conference on Software Maintenance and Reengineering (CMSR, 2006). Bari, pp: 23.

Marchetto, A. and F. Ricca, 2008. Transforming a java application in an equivalent web-services based application: Toward a tool supported stepwise approach. Proceeding of the 10th International Symposium on Web Site Evolution. Beijing, pp: 27-36.

Nguyen, D.K., W.J. van den Heuvel, M.P. Papazoglou, V. de Castro and E. Marcos, 2009. GAMBUSE: A gap analysis methodology for engineering SOA-based applications. In: Borgida, A.T. *et al.* (Eds.), Mylopoulos Festschrift. LNCS 5600, Springer-Verlag, Berlin, Heidelberg, pp: 293-318.

Salama, R. and S.G. Aly, 2008. A decision making tool for the selection of service oriented-based legacy systems modernization strategies. Proceeding of the International Conference on Software Engineering Research and Practice, Las Vegas, USA.

Vemuri, P., 2008. IEEE TENCON-2008 Modernizing a legacy system to SOA-Feature analysis approach. Proceedings of the IEEE Region 10 Conference (TENCON, 2008). Hyderabad, pp: 1-6.