Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 2, February 2014, pg.659 – 665

RESEARCH ARTICLE

SAT4BSC: A Static Analysis Tool for BPEL Source Codes

Esubalew Alemneh[1], Abdul Azim Abd Ghani[2], Rodziah Atan[2]

[1] Bahir Dar University, P.O.Box 26, Bahir Dar, Ethiopia
[2] Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM)
Serdang, Selangor 43400, Malaysia
[1] esubalew@bdu.edu.et, [2]{azim, rodziah}@fsktm.upm.edu.my

Abstract — Business Process Execution Language (BPEL) is Extensible Markup Language (XML) based language for describing the logic to orchestrate the interaction between Web services in a business process. Even though it is fairly new language it is getting popularity in various software industries and research environments. The emphasis of recent researches and developments on web services and on BPEL has been in their architecture and interface. However, the work regarding to tool support especially to compute the metrics and to draw control flow graph (CFG) is in its infant stage. Provision of tools to reckon measures has multitude of benefits. CFG is essential tool to analyze various properties of a source code and it is also useful for software testing, software measure, and software maintenance. In this research we have developed a static analysis tool which is dedicated to compute all available BPEL 2.0 metrics and draw CFG of its source code. The tool has been evaluated by various BPEL process source codes obtained from the languages specifications and from other research papers. The test shows that the tool can compute the metrics and draw the CFG effectively and efficiently.

Keywords— Static analysis; BPEL 2.0; BPEL Metrics; CFG

I. INTRODUCTION

BPEL, which is also called Business Process Execution language for Web Service (BPEL4WS), is an XML based web service composition and coordination language for business processes. Web service composition refers to the creation of new web services by combination of functionalities provided by existing ones [15]. Coordination in other hand refers to coordination protocol or business protocol which is used to denote the specification of the set of valid sequence of invocations that occurs between the client and the server during a web service interaction [7].

Web service is a software system designed to support interoperable machine-to-machine interaction over a network [15]. A business process is a collection of related, structured activities or tasks that produce a specific service or product [15].

BPEL has become the de-facto standard for the description of Web Service compositions. Moreover, BPEL is found to be so important that it has many extensions. Some examples of BPEL extensions are, Business Process Execution Language for People (BPEL4people), Business Process Execution Language for Service Oriented Architecture (BPEL4SOA), Business Process Execution Language and Java (BPELJ) and Aspect Oriented Business Process Execution Language (AO4BPEL).

Despite its wide acceptance in industries, and research areas only a little effort has been made to develop tools that compute metrics and generate CFG of BPEL process automatically. Of course, a lot is done on static analysis in general and plenty of

static analysis tools are available for metrics calculation and CFG drawing of various programming languages. Currently no tool is available for BPEL metrics calculation and CFG visualization. Since BPEL has unique syntax and semantics, we cannot use the existing tools for BPEL. However, at the moment there is no tool support for either metrics computation or CFG visualization for BPEL source code. Therefore, there is a need to define conceptions and foundations as well as to provide a concrete tool that can efficiently construct the CFG for BPEL. In this research, a static analysis tool is developed which performs computation of BPEL metrics and drawing of a CFG for its source code.

The availability of such tools is vital for many software engineering tasks. First of all it provides the benefits of software measure. Software measurement is important for prioritizing remediation efforts, allocating resources among multiple projects, quantify the amount of risk associated with a piece of code and getting feedback on the effectiveness of the security process [12]. Tool support for the computation of the metrics supports the achievement of aforementioned advantages of software measurement.

Secondly, the tool promotes the understandability of BPEL source code. BPEL process source code is very complex which makes its comprehensibility difficult. CFG tries to overcome this barrier as it is said "pictures worth thousands of words" simplifying the work of software reusability and maintenance. Moreover, CFG has an advantage in BPEL structural testing. The tool may also be integrated with other tools in order to facilitate software development with BPEL.

This paper is organized as follows. The next section, section two, is literature review on current studies about static analysis, BPEL metrics, BPEL CFG and tools for BPEL. The succeeding two sections are about analysis of BPEL source code metrics and control flow graph respectively. Architecture of the tool is discussed at section five. Finally, conclusion and future work are illustrated at section six.

II. LITERATURE REVIEW

BPEL comes from two programming languages; eXtensible Language (XLANG) and Web Service Flow Language (WSFL). BPEL was first released on July 2, 2002, as BPEL4WS v1.0. Organization for the Advancement of Structured Information Standards organization (OASIS) standardized next version, WS-BPEL 2.0, and it was sanctioned in April of 2007. Salient changes have been made as BPEL grows from BPEL 1.1 to BPEL 2.0. Some of these changes are new activities are added, existing activities are rename, fault handling is enriched, data manipulation is improved and data access is improved [4, 5 and 15]

Static code analysis is the analysis of computer software that is performed without actually executing programs [15]. The analysis is performed on either source code or object code and it can be manual inspection or automated one. Static analysis is essential for testing and for software metrics calculation and particularly it is very important to find low-level errors [15, 11]. Our static analysis tool is used to compute metrics and draw CFG for BPEL source code.

Despite a wide acceptance of BPEL in research and industry environments much hasn't been done on tool support for calculation and CFG drawing. However, certain tools are developed for monitoring and debugging as well as for analysis of BPEL. For instance WofBPEL is a tool which performs static analysis on BPEL processes by translating BPEL processes to Petri nets and applying existing Petri net analysis techniques [6]. The tool is used for detection of unreachable actions and conflicting messages as well as for metadata generation for garbage collection. Execution Analysis for BPEL is another tool which is used for monitoring and debugging of services specified in BPEL [17]. The tool defines execution log for the language.

CFG is one of the most basic information of a program to analyze various properties of a program which in turn would be useful for software testing, software measure or metrics, and software maintenance [20]. CFG promotes the understandability of the source code. This is particularly important for complex programming languages like BPEL.

Ample CFG constructing approaches have been proposed for many programming languages like procedural programming, Object-Oriented programming and Aspect oriented programming. As BPEL has unique features in both syntax and semantics it needs special approach to draw control flow graph. There are only a few works done regarding CFG of BPEL and those works are done for other purposes like test case generation.

Reachability graph is a graph for concurrent programs in order to generate test cases [21]. Since BPEL is a type of concurrent programming it can be applied to BPEL. However, reachability graph is not yet applied for BPEL.

Yuan et al. in [9], has explained the conversion of BPEL code to a CFG and named it as Business Flow Graph (BFG). It is a customization of CFG to BPEL. They have used the graph for graph-search based approaches to BPEL source code test case generation. We borrowed some ideas from this work to draw the CFG using our tool. However, as the CFG drawn by the above group is solely for test case generation and ours is for visualization of the BPEL process. Moreover, in case of BFG dynamic nature of the source code is considered to draw it. This involves the transformation of multiple choices to exclusive choice and dead path elimination. However, in our case we are concerned on static nature of the business process.

No tool has been developed to calculate metrics for BPEL source code. In this research we have developed a static analysis tool which we have named SAT4BSC that has the capability to calculate all known BPEL 2.0 metrics which are either proposed or validated, from its source code.

III. ANALYSIS OF BPEL METRICS

Software metric is a measure of some property of a piece of software or its specifications [15]. Effective management of any process requires quantification, measurement, and modeling. Because BPEL is quite a new language much hasn't been done on its measurement. Only few metrics has been developed for BPEL process and most of this metrics are obtained from neighboring disciplines, especially from traditional software metrics. There is strong analogy between conventional programs and BPEL processes. A business process modeled in BPEL can be seen as a traditional software program that has been portioned into modules or functions (i.e. activities) that take in a group of inputs and provide some output [2]. BPEL metrics can be divided into two categories: size metrics and Complexity metric.

For business process, activities of a process can be seen as statement of a software program and a simple size metrics can be derived that merely counts the number of activities in a business process [2]. The second metrics proposed for business process counts the number of activities, splits, and joins of a process. Considering processes as well-structured the third size metrics was proposed which accounts control-flow elements of a process in addition to activities. Table 1 contains list of size metrics and how to calculate these metrics from BPEL source code.

TABLE 1

SIZE METRICS FOR BPEL

No	Metrics	Description	How to calculate
1	Number of Activities (NOA)	BPEL has eight basic activities and nine container activities in total	count the number of basic and container activities in the process
2	Number of activities, joints and splits of a process (NOAJS)	BPEL has XOR-joins and splits, OR-joins and splits and AND-joints and splits	count the number of activities, joints and splits
3	Number of activities and control-flow elements (NOAC)	There are four control flow elements; sequencing, branching, iteration, and concurrent execution.	number of activities and control-flow elements in a process

Process complexity is defined by Jorge Cardoso on [13] as "The degree to which a process is difficult to analyze, understand or explain. It may be characterized by the number and intricacy of activity interfaces, transitions, conditional and parallel branches, the existence of loops, data-flow, control flow, roles, activity categories, the types of data structures, and other process characteristics." BPEL by its nature is a very complex language and complexity measure is crucial to determine the level of complexity of a program. Because business process are high-level notions made up of many different elements (splits, joins, resources, data, activities, etc.), there can never be a single measure of process complexity [2]. So, several process complexity metrics have been design.

There are two independent approaches to develop a control flow complexity metrics (CFC) metric to analyze business processes. The first metric proposes that the CFC of a BPEL process is the cumulative effect of the complexity of all activities in a process. Formulae are defined for each activity to calculate its complexity in [10] and these formulae are used to calculate CFC metrics using this tool.

The second CFC metrics is calculated based on the splits and joins of a process [12, 13, and 19]. Though this metrics is assumed to be design time metrics it can also be calculated from control flow graph of BPEL process. This version of CFC metrics is validated using an empirical investigation [12] as well is mathematically [13].

The study of data-flow complexity involves the analysis of XML Schema data types and the analysis of the input and output messages of web services (i.e. operations) [19]. There are three individual metrics for data-flow complexity; data complexity, interface complexity, and interface integration complexity. The first two are related to static data aspects (data declarations), while the third metrics is more dynamic in nature [10].

McCabe derived a software complexity measure from graph theory using the definition of the cyclomatic number which corresponds to the number of linearly independent paths in a program [2]. The metrics is empirically investigated and widely accepted for software programs. McCabe's cyclomatic complexity (MCC) is also proposed for business process by Jorge Cardoso et al in [2].

Halstead complexity metrics were developed by the late Maurice Halstead as a means of determining a quantitative measure of complexity directly from the operators and operands in the module to measure a program module's complexity directly from source code [17]. Among the earliest software metrics, they are strong indicators of code complexity. Because they are applied to code, they are most often used as maintenance metric. We can adapt Halstead's metrics for BPEL [2].

Graph theory provides a rich set of graph metrics that can be adapted for calculation of the complexity of BPEL process [2]. Among these metrics our SAT4BSC calculates those indicated in Table 2.

TABLE 2:

COMPLEXITY METRICS THAT ARE DERIVED FROM GRAPH THEORY

Metrics	**Description**
Coefficient of network Complexity (CNC)	CNC = number of arcs / (number of activities, joins, and splits)
Structuredness of process graph	Number of structured activities

Figure 1 shows the metrics computed by SAT4BSC for loan approval process BPEL source code obtained in [8].

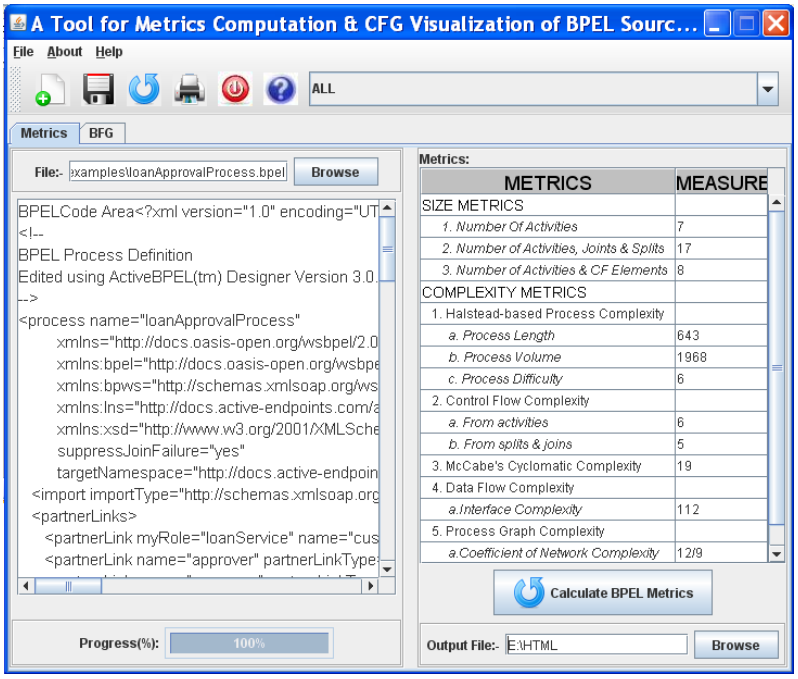

Figure 1: Metrics calculated for loan approval process

IV. ANALYSIS OF CFG FOR BPEL

BPEL is a highly compact and expressive language that makes it hard to analyze. So the transformation of the BPEL source code to a CFG is important for comprehensibility. BPEL can be viewed as a graph composed of nodes (activities) and edges (implicit sequence concatenations, and explicit links) [9]. CFG of BPEL promotes the understandability of the process by reducing the quantity of control structure types. That is more than one activity are mapped to the same designation in CFG. For instance if and pick have the same semantics and so they are mapped to the same type of node in the graph. The formal definition of the BPEL CFG as given in [9] is;

BFG = <N, E, S, F>, where

- N is set of nodes. A node can be normal node, decision node, merge node, fork node, and join node. The type of nodes depends on the type of BPEL activities.
- E is a set of implicit and explicit
- S is the start node.
- F is set of final nodes.

We have used different designations (shape and color) to represent different types of nodes and edges. Figure 2 shows CFG generated by SAT4BSC for purchase order process BPEL source code obtained in [8].

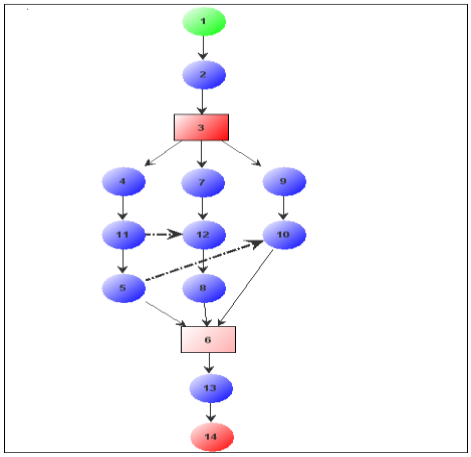

Figure 2: CFG of Purchase order BPEL source code

V. IMPLEMENTATION OF THE TOOL

Because of its high portability and richness in string manipulation we have opted Java for implementation of the tool. The tool has graphical user interface which enables the user view the metrics calculated and the CFG of the source code chosen.

The architecture of the tool is depicted in Figure 3. Each component is briefly described here.

- ***Metrics Reckoner Unit:*** is responsible to compute all BPEL metrics that are mention at section 5. Some of the metrics can be calculated directly from the source code while other metrics depends on CFG elements, edges and nodes.
- ***CFG Determiner unit:*** This unit determines all implicit and explicit edges and nodes of CFG from BPEL source code.
- ***CFG Visualizer unit:*** Displays the CFG of BPEL source code using the predetermined edges and nodes in CFG determiner unit, shape factory and components from JGraph.
- ***Shape Factory:*** Is used to create different kinds of nodes and edges of the CFG.
- ***Components from JGraph*** - is used to create different shapes and for visualization of CFG
- ***Metrics Output:*** After computation of the metrics the output is displayed in the user interface and this metrics can be viewed in text format or saved in XML format

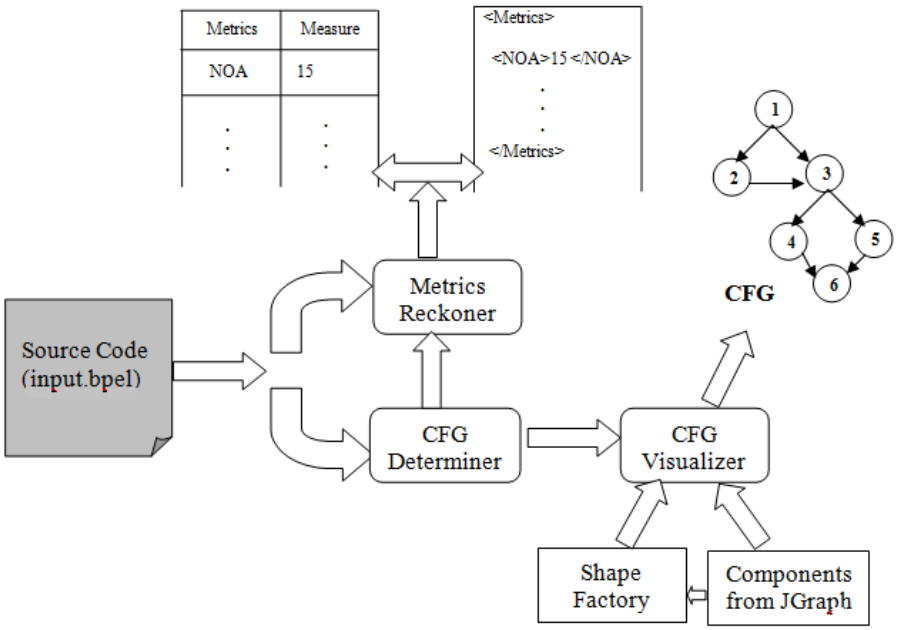

Figure 3: Architecture of the Tool

SAT4BSC is tested on various examples in order to assess its feasibility, correctness and efficiency. The examples ranges from simplest hello world to complex process like travel reservation process and loan approval process. Five of the example BPEL source codes are obtained from BPEL 2.0 specification [8]. Some of examples have been used as a benchmark by other researchers [9, 10]. More examples are obtained from internet [23] and others come together with NetBeans Enterprise 5.5 BPEL engine. The tool can calculate all metrics and draw CFG efficiently and effectively.

VI. CONCLUSION AND FUTURE TREND

In this research we have developed a static analysis tool that computes metrics BPEL source code. The tool can also transform BPEL source code to its CFG for visualization purpose. Prior to the implementation of the tool we have made an analysis on the BPEL source codes and BPEL metrics. This was crucial task because most the metrics are proposed for earlier version of BPEL designs and our concern is on BPEL 2.0 source code. Consequently we have defined CFG metrics formulas for newly added activities in BPEL 2.0. The tool has been tested with sufficient BPEL examples. The result depicts that the tool can effectively and efficiently calculate the metrics and draw CFG of BPEL process source codes.

Further studies are required which involves surveying of more BPEL process metrics and integrating new metrics that are currently in researches progress. Furthermore, we will improve CFG drawing process of the tool by taking into consideration event and fault handling in the graph as well as the appearance of explicit edge between any activities in a process other than solely in activities enclosed in flow activity.

REFERENCES

[1]. Active Endpoints, inc. BPEL Fundamentals. BPEL 2007.
Available via: *http://www.activevos.com/cec/training/content/a_selfPacedTraining/*

[2]. J. Cardoso, J. Mendling, G. Neuwmann and H.A. Reijers. A Discourse on Complexity of Process Models. BPM 2006 workshops, LNCs 4103. 2006.

[3]. Michael Havey. Essential Business Process Modeling, Publisher: O'Reilly**,** e-book, 2005.

[4]. M. Das, A. Yiu, K. Kand. A close look at BPEL. SOA World Magazine. 2007.
Availble Via: *http://soa.sys-con.com/node/434430*

[5]. M. Das**,** D. Shaffer**.** BPEL's Growing Up. SOA World Magazine. 2007.
Availble Via: *http://websphere.sys-con.com/node/346372*

[6]. C Ouyang, E Verbeek, Wil M.P. van der Aalst, S Breutel, M Dumas, and A.H.M. ter Hofstede. WofBPEL: A Tool for Automated Analysis of BPEL Processes. 2005.

[7]. Frederic Servais. Verifying and Testing BPEL Process, masters thesis, 2007.

[8]. Web Services Business Process Execution Language Version 2.0 Specification
Available via: *http://www.oasis-open.org/committees/download.php/12791/.*

[9]. Y Yuan, Z Li, Wei Sun. A Graph-search Based Approach to BPEL4WS Test Generation, 2006.

[10]. J. Cardoso, Complexity Analysis of BPEL web Process, Journal of Software Process: Improvement and Practice, 2006.

[11]. *http://www.testingfaqs.org/t-static.html.*

[12]. J Cardoso. Control-flow Complexity Measurement of Process and Weyuker's Properties. Proceedings of world academy of science, engineering and technology. 2005.

[13]. J Cardoso. Process control-flow complexity metric: An imperical validation. IEEE international conference on services computing, 2006.

[14]. A. Gravel, X. Fu, and J. Su. An Analysis tool for Execution of BPEL Services. The 9th international conference on E-commerce. 2007

[15]. *http://www.en.wikipedia.com*

[16]. Y Qianz, Y Xuz, Z Wangz, GPuz, H Zhuz, and C Caiy. Tool Support for BPEL Verification in ActiveBPEL Engine , 2007.

[17]. *http://www.verifysoft.com/en_halstead_metrics.html*

[18]. P. Mayer. Design and Implementation of a Framework for Testing BPEL Compositions, Masters thesis, 2006.

[19]. R.M. Pariizi and A.A.A Ghani. An Ensemble of Complexity Metrics for BPEL web Process. Ninth ACIS International Conference on Software Engineering, Artificial intellegince, Networking, and Parallel/Distributed Computing. 2008.

[20]. R.M. Pariizi and A.A.A Ghani. AJcFGraph – AspectJ Control Flow Graph Builder for Aspect-Oriented Software, International journal of Computer Science, 2008

[21]. R. N. Taylor, D. L. Levine, and C. D. Kelly. Structural Testing of Concurrent Programs. IEEE transactions on software engineering, 1992.
[22]. C. K. Yee, Design and Implementation of Test Case Generation Tool for BPEL Unit Testing, thesis, 2007.
[23]. Antony Miguel, WS-BPEL 2.0 Tutorial, 2005.
Available via: *http://charltonb.typepad.com/weblog/2003/08/what_is_bpel4ws.html*
[24]. M. Juric, BPEL and Java, article, 2005.
Available via: *http://www.theserverside.com/articles/content/BPELJava.pdf*

