GAmera: A Tool for WS-BPEL Composition Testing Using Mutation Analysis

Juan-José Domínguez-Jiménez, Antonia Estero-Botaro, Antonio García-Domínguez, and Inmaculada Medina-Bulo

Dpt. Computer Languages and Systems
University of Cádiz, Escuela Superior de Ingeniería,
C/Chile 1, CP 11003 Cádiz, Spain
{juanjose.dominguez,antonia.estero,antonio.garciadominguez, inmaculada.medina}@uca.es

Abstract. This paper shows a novel tool, GAmera, the first mutant generation tool for testing Web Service compositions written in the WS-BPEL language. After several improvements and the development of a graphical interface, we consider GAmera to be a mature tool that implements an optimization technique to reduce the number of generated mutants without significant loss of testing effectiveness. A genetic algorithm is used for generating and selecting a subset of high-quality mutants. This selection reduces the computational cost of mutation testing. The subset of mutants generated with this tool allows the user to improve the quality of the initial test suite.

1 Introduction

The evolution of software towards Service-Oriented Architectures (SOAs) has led to the definition of a language that facilitates the composition of Web Services (WS): the OASIS Web Services Business Process Execution Language (WS-BPEL) 2.0 [1]. WS-BPEL allows us to develop new WS modeling more complex business processes on top of pre-existing WS. WS-BPEL is an XML-based language which specifies the behavior of a business process as a WS which interacts with other external WS independently of how they are implemented through message exchanges, synchronization and iteration primitives and fault or event handlers, among other constructs.

Mutation analysis has been validated as a powerful technique for testing programs and for the evaluation of the quality of test suites [2]. It generates mutants by applying mutation operators to the program to test. The resulting mutants contain a single syntactic change with respect to the original program. So, in order to apply this technique to programs in any language, we need a language-specific set of mutation operators and a tool for generating and executing the mutants.

Rice [3] lists the ten most important challenges in the automation of the test process. One of them is the lack of appropriate tools, as they are too costly to use or do not fit the tester's intention or the required environment. Several systems to generate mutants for programs written in various languages exist: Mothra [4] for Fortran, MuJava [5] for Java, ... GAmera is the first mutation testing tool for WS-BPEL.

One of the main drawbacks of mutation testing [2] is the high computational cost involved in the execution of the large number of mutants produced for some programs against their test suites. Most existing tools simply generate all the possible mutants. GAmera follows a different approach, generating only a subset of all the possible mutants. To select these mutants, GAmera incorporates a genetic algorithm (GA) that selects only high-quality mutants, reducing the computational cost.

In [6] we presented a set of specific mutation operators for WS-BPEL 2.0, in [7] we proposed a framework for the automatic generation of mutants for WS-BPEL based on GA and in [8] we showed the preliminary results of applying the new technique. The tool described in this paper is a direct consequence of these previous works. This paper shows the functionality and usefulness of GAmera after several improvements and the development of a graphical interface. This tool implements the GA integrated with the mutation operators defined for WS-BPEL in the previous works. This open-source tool is freely available at its official website[1].

2 Tool Design

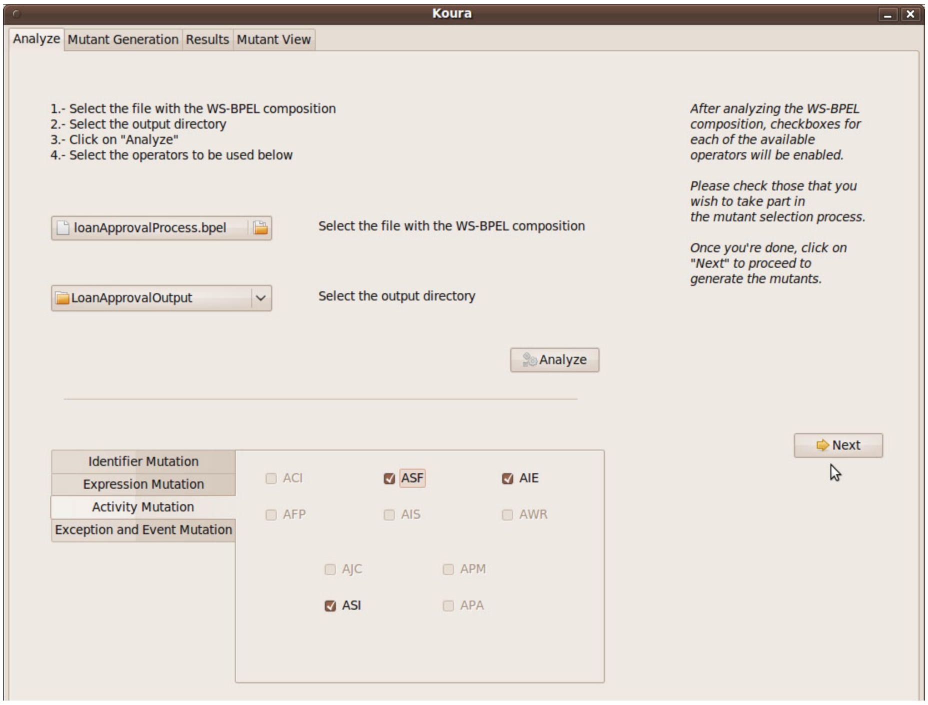
The GAmera tool consists of three main components: the *analyzer*, the *mutant generator* and the *execution engine* that runs and evaluates the mutants.

Analyzer. It starts off the GAmera workflow by receiving as input the WS-BPEL composition under test and listing the mutation operators that can be applied to it. Figure 1 shows a screenshot of the application that interacts with the analyzer. The operators are displayed in separate tabs, depending on the category of the operator. The user can determine the set of mutation operators to use among all the available operators.

Mutant generator. Mutants are generated using the information received from the analyzer. The tool gives us the possibility of generating all the possible mutants, or selecting only a subset of them with the genetic algorithm.
The selection process uses two components. The first, called *mutant genetic search*, is a GA in which each individual represents a mutant. The GA is capable of automatically generating and selecting a set of mutants, using a fitness function that measures the quality of a mutant, depending on if there are or not test cases that kill it [7].
The second element is the *converter*, that transforms an individual of the GA into a WS-BPEL mutant. To perform this conversion, the tool uses a different XSLT stylesheet for each mutation operator.

Execution engine. The system executes the mutants generated against a test suite. Mutants are classified into three categories depending on their output:

[1] `http://neptuno.uca.es/~gamera`


Koura

Analyze | Mutant Generation | Results | Mutant View

1.- Select the file with the WS-BPEL composition
2.- Select the output directory
3.- Click on "Analyze"
4.- Select the operators to be used below

After analyzing the WS-BPEL composition, checkboxes for each of the available operators will be enabled.

Please check those that you wish to take part in the mutant selection process.

Once you're done, click on "Next" to proceed to generate the mutants.

loanApprovalProcess.bpel — Select the file with the WS-BPEL composition

LoanApprovalOutput — Select the output directory

Analyze

Next

Identifier Mutation
Expression Mutation
Activity Mutation
Exception and Event Mutation

☐ ACI ☑ ASF ☑ AIE
☐ AFP ☐ AIS ☐ AWR
☐ AJC ☐ APM
☑ ASI ☐ APA

Fig. 1. Interaction with the analyzer

Killed. The output from the mutant differs from that of the original program for at least one test case.

Surviving. The output from the mutant and the original program are the same for all test cases.

Stillborn. The mutant had a deployment error and it could not be executed. Mutants in this state hint at how the design and implementation of the mutation operators should be revised.

For the execution of the original program and their mutants, GAmera uses the ActiveBPEL 4.1 [9] open-source WS-BPEL 2.0 engine and the BPELUnit [10], an open-source WS-BPEL unit test library which uses XML files to describe test suites.

2.1 Results

GAmera shows the results obtained in the execution of the mutants. It displays the total number of generated, killed, surviving and stillborn mutants. From these values, we can measure the quality of the initial test suite and let the user improve it by examining the surviving mutants and adding new test cases to kill them. For this purpose the tool includes a viewer which shows the differences between the original composition and the mutant.

3 Conclusions and Future Work

We have presented the first tool that automatically generates a set of high-quality mutants for WS-BPEL compositions using a genetic algorithm. This tool automates the test of WS-BPEL compositions and provides information that allows the user to improve the quality of the test suites.

The tool is useful both for developers of WS-BPEL compositions, since it automates the testing process, and for researchers in testing of WS-BPEL compositions that are intended to evaluate the quality of test suites.

We are currently working on the design of new mutation operators for providing various coverage criteria. A future line of work is adding a test case generator to the current tool. This generator will allow the user to enhance the quality of the test suite by generating automatically new appropriate test cases.

Acknowledgments

This paper has been funded by the Ministry of Education and Science (Spain) and FEDER (Europe) under the National Program for Research, Development and Innovation, Project SOAQSim (TIN2007-67843-C06-04).

References

1. OASIS: Web Services Business Process Execution Language 2.0 (2007), `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`
2. Offutt, A.J., Untch, R.H.: Mutation 2000: uniting the orthogonal. In: Mutation testing for the new century, pp. 34–44. Kluwer Academic Publishers, Dordrecht (2001)
3. Rice, R.: Surviving the top 10 challenges of software test automation. CrossTalk: The Journal of Defense Software Engineering, 26–29 (mayo 2002)
4. King, K.N., Offutt, A.J.: A Fortran Language System for Mutation-based Software Testing. Software - Practice and Experience 21(7), 685–718 (1991)
5. Ma, Y.S., Offutt, J., Kwon, Y.R.: MuJava: an automated class mutation system. Software Testing, Verification & Reliability 15(2), 97–133 (2005)
6. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Mutation operators for WS-BPEL 2.0. In: Proceedings of the 21th International Conference on Software & Systems Engineering and their Applications (2008)
7. Domínguez-Jiménez, J.J., Estero-Botaro, A., Medina-Bulo, I.: A framework for mutant genetic generation for WS-BPEL. In: Nielsen, M., Kucera, A., Miltersen, P.B., Palamidessi, C., Tuma, P., Valencia, F.D. (eds.) SOFSEM 2009. LNCS, vol. 5404, pp. 229–240. Springer, Heidelberg (2009)
8. Domínguez-Jiménez, J.J., Estero-Botaro, A., García-Domínguez, A., Medina-Bulo, I.: GAmera: An automatic mutant generation system for WS-BPEL. In: Proceedings of the 7th IEEE European Conference on Web Services, pp. 97–106. IEEE Computer Society Press, Los Alamitos (2009)
9. ActiveVOS: ActiveBPEL WS-BPEL and BPEL4WS engine (2008), `http://sourceforge.net/projects/activebpel`
10. Mayer, P., Lübke, D.: Towards a BPEL unit testing framework. In: TAV-WEB'06: Proceedings of the workshop on Testing, analysis, and verification of web services and applications, pp. 33–42. ACM, New York (2006)