The Journal of Systems and Software 000 (2016) 1–15

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Optimal control based regression test selection for service-oriented workflow applications

Hongda Wang[a], Jianchun Xing[a,*], Qiliang Yang[a,b], Ping Wang[a], Xuewei Zhang[a], Deshuai Han[a]

[a] College of Defense Engineering, PLA University of Science and Technology, Nanjing, China
[b] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

ARTICLE INFO

Article history:
Received 14 December 2014
Revised 28 April 2016
Accepted 23 June 2016
Available online xxx

Keywords:
Software cybernetics
Optimal control
Service-oriented workflow applications
Regression test selection
Behavioral difference
BPEL program dependence graph
Safe

ABSTRACT

Regression test selection, which is well known as an effective technology to ensure the quality of modified BPEL applications, is regarded as an optimal control issue. The BPEL applications under test serves as a controlled object and the regression test selection strategy functions as the corresponding controller. The performance index is to select fewest test cases to test modified BPEL applications. In addition, a promising controller (regression test selection approach) should be safe, which means that it can select all test cases in which faults might be exposed in modified versions under controlled regression testing from the original test suite. However, existing safe controllers may rerun some test cases without exposing fault. In addition, the unique features (e.g., dead path elimination semantics, communication mechanism, multi-assignment etc.) of BPEL applications also raise enormous problems in regression test selection. To address these issues, we present in this paper a safe optimal controller for BPEL applications. Firstly, to handle the unique features mentioned above, we transform BPEL applications and their modified versions into universal BPEL forms. Secondly, For our optimal controller, BPEL program dependence graphs corresponding to the two universal BPEL forms are established. Finally, guided by behavioral differences between the two versions, we construct an optimal controller and select test cases to be rerun. By contrast with the previous approaches, our approach can eliminate some unnecessary test cases to be selected. We conducted experiments with 8 BPEL applications to compare our approach with other typical approaches. Experimental results show that the test cases selected using our approach are fewer than other approaches.

1. Introduction

With the emergence of cloud computing, service-oriented workflow, a large-scale programming mode, has gradually become a mainstream technology for developing instant applications in an open environment (Canfora and Di Penta, 2009, Fu et al., 2004). Web Service Business Process Execution Language (WS-BPEL or BPEL) is one of the most popular standards for developing service-oriented workflow applications. BPEL applications can provide value-added services by compositing Web Services or other BPEL applications. However, these applications usually present some faults or defects, particularly during the evolution of service composition. Maintenance of these applications is expensive. On an average, these activities (especially regression testing) account for as much as two-thirds of the overall software life cycle cost (Beizer, 2003, Leung and White, 1989). The cost of regression testing could be reduced if old test cases and results were reused. Therefore, *regression test selection* (also concerned as *selective retest technique*), one of the most important approaches to provide confidence during software evolution, has gained growing concerns in recent years (Epifani et al., 2010, Zhu and Zhang, 2012, Li et al., 2008).

Regression test selection, which is well known as an effective technology to ensure the quality of modified BPEL applications, is regarded as an optimal control issue. The BPEL applications under test serves as a controlled object. The test results servers as feedback and the regression test selection strategy functions as the corresponding controller. By selecting some subsets of the original test cases to be rerun, regression test selection approach attempts to reduce the time and resources required to retest a modified BPEL application. The performance index of this control issue is to select fewest test cases to test modified BPEL applications, which is similar to the famous least fuel control issue in the optimal

* Corresponding author.
E-mail addresses: wanghongda000@126.com (H. Wang), xjc@893.com.cn (J. Xing), yql@893.com.cn (Q. Yang).

http://dx.doi.org/10.1016/j.jss.2016.06.065
0164-1212/

control area (Khalil, 2001). In addition, a promising controller (regression test selection approach) should be *safe* (Rothermel and Harrold, 1996), which means that it can select all test cases in which faults might be exposed in modified versions under *controlled regression testing* from the original test suite. This problem has been extensively studied for a long time (Liu et al., 2007, Li et al., 2012, Lin et al., 2006, Ruth and Tu, 2007). However, most of the existing approaches identify *affected components* to be retested from syntactic perspective rather than semantic (behavioral) perspective. For example, to ensure correctness or to improve efficiency of service composition, it is common to rearrange or parallelize two activities (Song et al., 2011, Ni et al., 2011) in BPEL applications. In this case, some approaches may rerun some unnecessary test cases that without exposing any faults to test the modified BPEL application (e.g., (Liu et al., 2007, Li et al., 2012)) due to different syntaxes presented by the two activities in two versions. Our observation is that the state is consistent after the execution of the two corresponding activities in the two versions. In other words, although the execution orders of the two activities in two versions are different, these two activities may have equivalent behavior. Therefore, the activities in modified BPEL application are not affected components and this application does not need to be retested. Although some approaches (e.g., (Agrawal et al., 1993, Bates and Horwitz, 1993)) have studied this problem from behavioral perspective, they are not safe, i.e., they may neglect some necessary test cases that could detect faults in modified versions. In addition, the unique features (dead path elimination semantics, communication mechanism, multi-assignment, etc.) of BPEL language also bring enormous problems for the issue of regression test selection. For example, two activities of BPEL applications may have different forms (syntaxes) but same behavior due to these unique features. Therefore, the modified BPEL applications may also not need to be retested. However, few studies have focused on this phenomenon. To address these issues, we present in this paper a safe controller or regression test selection approach guided by *behavior difference* of activities, which is computed with *BPEL program dependence graphs* and *program slices*. Our approach is inspired by a heuristic rule, that is, if the activity behavior in the modified BPEL application is different from that of the old version, its corresponding test cases need to be selected.

Program dependence graphs are an intermediate representation of programs which characterizing control and data dependency relations between the statements of programs in software engineering. Program dependence graphs have previously been introduced for its usages in incremental program testing (Bates and Horwitz, 1993), optimizing, and program analyzing (Ferrante et al., 1987). BPEL program dependence graph extends the notion of program dependence graph by introducing some new dependence relation. In this paper, we adopt BPEL program dependence graphs and the program slicing approaches to identify all affected components of the modified BPEL applications that need to be retested. By using a semantic (behavioral) rather than a syntactic definition of "affected components", our approach can identify BPEL components that have been directly or transitively affected by the modifications anywhere in BPEL applications. There are three steps in our approach. Firstly, to facilitate later comparison of program dependence graphs, three rules are given to transform BPEL applications into *universal BPEL forms*. These three rules correspond to the unique features of dead path elimination, communication mechanism and multi-assignment respectively. Secondly, program dependence graphs corresponding to the two universal BPEL forms (BPEL application and its modified version) have been established. To this end, all kinds of BPEL program dependencies between activities should be analyzed. We mainly capture *control dependency, data dependency* and *asyn-invocation dependency* (Song et al., 2011). Finally, we adopt program slicing approach to identify all affected components of the modified BPEL applications and select corresponding tests to be rerun. Our approach is able to select not only test cases that execute new or modified activities, but also those previously executed the activities deleted from the modified BPEL application. Based on its performance, we can prove our controller is safe. Compared with the previous approaches or controllers, our controller can eliminate some unnecessary test cases to be rerun. We conducted experiments using our approach and other typical approaches with 8 BPEL applications. Experimental results demonstrate that the test cases selected using our approach are fewer than other approaches.

This paper mainly makes the following three contributions:

- Regression test selection problem is treated as an optimal control issue, and an optimal control strategy is presented.
- With the unique features of BPEL language, three rules are given to transform any BPEL applications into a universal BPEL form. Using this transformation, our approach proposed in this paper can be applied directly in software engineering.
- With a semantic (behavioral) definition on affected components, our controller can select fewer test cases to test modified BPEL applications effectively. Also, we prove that our controller is safe under controlled regression testing.

The rest of this paper is organized as follows. Section 2 shows a running example to motivate our approach, followed by some preliminaries. Section 3 presents our approach. Section 4 reports the experiments based on our proposal. Section 5 illustrates some discussions about our approach. Section 6 reviews some related works. Section 7 gives a summary and future work.

2. Background

In this section, we show a running example to motivate our approach, followed by some preliminaries about BPEL language and *BPEL program dependency graph*.

2.1. A running example

In this subsection, we use a BPEL application *travel agency*, which is adapted from *Travel*, as a running example to motivate our approach. This is a so well-known and carefully designed application used in many researches (Song et al., 2011, Van Der Aalst et al., 2008). First, we give a brief overview of this application.

For intuitive expression, we use UML activity diagrams instituting of BPEL codes (in XML format) to depict these applications. In each activity diagram, a node represents a BPEL activity and an edge stands for a transition between two activities. In addition, we also annotate the nodes with extracted application information, such as input and output parameters of activities. Fig. 1(a) is the initial version V_1, which depicts a BPEL application of a travel agency. On receiving ordering information from a client, the travel agency application can choose appropriate flight service and hotel service for the client. At the same time, this application records the ordering information received from all clients to establish a database. Finally, this application gives the result (confirmation message of services) back to the client. With the increasing uses of this BPEL application, supposing the developer gradually found some deficiencies in this application. Therefore, the developer made the following incremental modifications. Each corresponds to a BPEL application version.

(1) At the beginning, the developer found that there was no need to use two activities to assign *airlineInput* and *hotelInput* to variable *query* at all. According to the BPEL specification (WS-BPEL 2.0 Specification 2007), they can be merged together into one activity. As a result, the developer modified version V_1 to version V_2 in Fig. 1(b).

Please cite this article as: H. Wang et al., Optimal control based regression test selection for service-oriented workflow applications, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.06.065