Testing strategy in discovering faults in workflow

By

Ting Cham Ho

A Thesis Submitted to
The Hong Kong University of Science and Technology
In Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
In Computer Science

August 2010, Hong Kong
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Ting Cham Ho
27 August, 2010
Testing strategy in discovering faults in workflow

By

Ting Cham Ho

This is to certify that I have examined the above MPhil thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

PROF. S.C. CHEUNG, SUPERVISOR

PROF. MOUNIR HAMDI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering
27 August 2010
Acknowledgements

It is tough to complete this thesis and I would like to thank sincerely the following people who helped me a lot during my MPhil study.

First, I would like to express my gratitude to my supervisor, Dr. Shing-Chi Cheung, for his guidance and patience. His feedback and advices has driven me to finish the thesis.

I would also like to thank Prof. L. Gu and Prof. L. Chen, who are my thesis exam committee members, to dedicate their valuable time.
TABLE OF CONTENT
Title Page ......................................................................................................................i
Authorization .............................................................................................................ii
Signature Page ......................................................................................................... iii
Acknowledgements ..................................................................................................... iv
TABLE OF CONTENT ................................................................................................. 5
Abstract ......................................................................................................................1
Chapter 1 Introduction ..............................................................................................2
Chapter 2 Related Work ............................................................................................4
Chapter 3 Web Services and BPEL ...........................................................................6
  3.1 Web Services Architecture ...............................................................................6
  3.2 BPEL ................................................................................................................... 8
  3.3 Process Notation ................................................................................................10
  3.4 Schedule Coverage ..........................................................................................11
  3.5 Fault Model .......................................................................................................13
Chapter 4 Empirical Study .......................................................................................15
  4.1 Experimental Setup .........................................................................................15
  4.2 Data Analysis ..................................................................................................17
Chapter 5 Conclusion ...............................................................................................18
Bibliography and References .....................................................................................20
Testing strategy in discovering faults in workflow

By

Ting Cham Ho

Computer Science and Engineering
The Hong Kong University of Science and Technology

Abstract

Web services are extensively used nowadays as a typically application programming interfaces for Business-to-Business process collaboration and BPEL (Business Process Execution Language) is an emerging standard language to orchestrating multiple web services. BPEL also offers advanced features such as concurrency and hierarchy which provides different programming construct from traditional sequential program. There are studies to propose Unit test framework and test cases generation; however, there is lack of studies in evaluating the testing criteria against BPEL. To address this issue, an empirical study is conducted to evaluate some test criteria derived from traditional sequential program testing. Also, a schedule-related criterion is introduced to compare the test case efficiency and fault reviewing power against the tradition test criteria.
Chapter 1 Introduction

Services-oriented architecture (SOA) has become popular for building distributed business applications. These applications' logic is often decomposed into fine-grained services which are software components accessible through Internet. They are further composed into different executable business processes based on the actual business requirements. The services in SOA are designed to provide a flexible and loosely coupled model for distributed, heterogeneous system such that it is applicable for coordinating the services to achieve cross-organizations collaboration. The majority implementation strategy for SOA is to use web services standards [1] and the related technologies, like the Business Process Execution Language (BPEL) [4], have been developed to define a model and grammar for describing the business process behavior according to the interactions between the processes and the partner's processes.

Business Process Execution Language (BPEL) is a kind of orchestration [2] language for coordinating web services into complex Business process which is also exposed as web services. It is increasing in modeling the compositions by BPEL and there are few BPEL development environments currently available such as Oracle BPEL process manager [7], ActiveBPEL Designer [8], Net Beans 5.5 [9].

To define a business process, BPEL supports the programming constructs used in traditional programming languages, e.g. switch, while-loops and fault handling. In addition, BPEL also provides built-in constructs for concurrency and synchronization which can be seen as a kind of concurrent programs. Therefore, the BPEL programs, like the traditional program, are software artifacts that can involve complex logic and it is error-prone. As the BPEL programs are also suffered from the same problems as traditional programs and
mission-critical business solution, it is required to test them in order to provide acceptance quality to the customers. There are studied in creating unit testing frameworks for BPEL [3][5][6].

In this thesis, we consider the runtime ordering of activities and propose schedule testing coverage criteria to construct the test suites. In addition, we carry out empirical studies to compare fault reviewing power between the proposed schedule coverage and the traditional one such as activities coverage. Random coverage criteria are also used as reference.

The thesis is organized as follows: Chapter 2 presents related studies of BPEL testing. Chapter 3 provides the overview of BPEL constructs, also the proposed schedule coverage and the faults to be studied. Chapter 4 provides the experimental setup and result. Finally, we conclude our work in Chapter 5.
Chapter 2 Related Work

BPEL language is relatively new and it becomes more extensively used in inter-enterprise application integration through Web Services. However, the BPEL language is different from traditional programming language which BPEL provides concurrency and hierarchy construct. In addition, XML and related technologies (e.g. XPath and Schema) are heavily used in BPEL which have different behaviors from implementing traditional program. In recent years, BPEL becomes a research focus and the studies can be roughly divided into test cases generation, modeling and verification.

In the sub area of test case generation, Yan [16] proposed to generate test cases for BPEL using concurrent path analysis. They used the extended control-flow graph to model the flow and link constructs and then generated test cases by solving paths’ constraints. It aims at producing message parameters instead of controlling the message sequence for testing. In Fjnjul studies [17], they made use of the transition coverage criterion and use the existing tool SPIN for test cases generation. There are also some studies using model based testing [18][19].

The modeling of service composition is also studied in [22]. They study the interactions between BPEL programs and web services using WS-BPEL as specification. A tool WSAT has been developed to conduct formal analysis of web services.

There are few studied to provide BPEL unit test framework [3][5][6] for facilitating the testing tasks on BPEL.
Rather than the practical approaches to verify the BPEL, there are also theoretical approaches to verify the correctness of the business process for some requirements fulfillment. One of them is verifying the atomicity property for B2B collaboration [20][21]
Chapter 3 Web Services and BPEL

With the advent of Internet, the usage of Internet was first for information sharing. Nowadays, the internet is so mature and it's widespread to the whole world. E-Business or web services are the trend for organizations to offer their services besides traditional businesses through Internet. Intensive online services like eBanking, eAuction, banking, retail purchase, booking system, etc are developed. To allow better heterogeneous system collaborating with each other, many standards have been proposed and commonly used today such as XML, XPath, SOAP, WSDL, WS-BPEL, WSTransaction, HTTP, etc. The Web Services technology becomes the trend to enable inter-enterprise application integration and the BPEL is an emerging standard language to describe web services composition behavior.

3.1 Web Services Architecture

From the W3C's definition, a Web Service is "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web services in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" [10]. Web services can be defined as stack of standards and shown as Figure 3-1. At the bottom of the stack, the standards are responsible for transporting messages between applications/systems. The base technologies at next upper layer are XML, Schema, XPath, WSDL, etc. They are used for manipulating the information throughout the Web Services stack. XML plays the critical role in the Web standardization and all messages communicating among applications are formatted in XML. These XML messages will be
exchanged through SOAP standard and these message protocols will be associated with Interface Definition Languages like Web Service Definition Language (WSDL) which provide definition of the offered services.

![Web Services Architecture Stack](image)

*Figure 3-1. Web Services Architecture Stack*

These layers provide the view of the infrastructure which other higher level standards should build on top of it. Few high level standards have existed such as WS-Transaction [11] for transactional interoperability between Web services domains and provide a means to compose transactional qualities of service into Web services applications, WS-Federation [12] for allowing disparate security realms to broker information on identities, identity attributes and authentication. Business Process Execution Language for Web Services (BPEL4WS) is another high level specification.
3.2 BPEL

BPEL is a language for coordinating and composing web services. It can be viewed as workflow language for Web Services. The standards used by Web Services like WSDL, SOAP, HTTP, UDDI allows implementation and execution of a basic services which is invoked as a single operation request.

The basic activity of BPEL includes invoke, receive, reply, and assign. The invoke activity invokes the execution of partner Web Service. The receive activity is waiting for receiving a message from outside. The reply activity replies a message to outside. The assign activity provides assignment operation which is expressed in XPath syntax.

BPEL also provides several structures such as sequence, flow, switch and while. Sequence defines the containing activities to be executed sequentially. Flow describes a scope of activities that can be executed concurrently and in any order. In the scope of flow structure, a link can be defined for synchronization among the activities with transition condition. Switch and while are the branch and loop structure behaves similarly as those in traditional programming languages.

BPEL as a coordination language for Web Services, it supports the interaction involving several services. The BPEL program defines the invocation sequences of services in order to fit the business rules. For example, an online shop requires the customer to put the items to their shopping cart before the payment. The hotel flight booking system requires the user to provide information for reservation before the payment. Once the payment is confirmed, no cancellation is allowed.
BPEL as a composition language, having the standard of declarative language WSDL to define the interface of a web services, BPEL can make use of it to combine several other Web Services into one Business Process to achieve particular business goal. For a typical loan approval service example, it provides a Web Service port to who want to access the loan service. If the amount is greater than $10K, the process will pass the required information and invoke another service to request for approval. If the amount is less than $10K, the process will invoke another service to access the risk. Finally, the loan approval service will make the final decision based on the result sent from other services. From this example, it invokes the composition of three web services: the Risk Assessment Web Service, the Load Approval service and the Loan Request service. BPEL compositions are themselves exposed as web service and the composition can be recursive to form more complex Web Service.

Figure 3-2. Loan Approval Services
3.3 Process Notation

To generate the process execution sequence, the user is required to transfer the BPEL program to the following notation. This notation is regarded as a textual representation of the BPEL program.

A BPEL program $P$ is a 5-tuple $(s,f,\prec,A,V)$. $A$ is a set of activities e.g. assign operation, invoke operation, switch, etc. For each process, it must contain one and only one start task $s$ and a final task $f$ to represent the single entry and single exit of the process. $\prec$ is the set of transition among activities and it is in the form $(a_0, \text{predicate}, a_1)$ which means the transition only fired from activity $a_0$ to activity $a_1$ only when the predicate is evaluated to true. It is possible that the predicate to be $null$ and it means $a_1$ should execute only when $a_0$ completed. $V$ represents a set of input parameters required by the program.

Based on the Figure 3-1, Loan Approval Process $(s, f, \prec, A)$ can be translated to the following representation.

\[
\begin{align*}
    s &= a_0 \\
    f &= a_4 \\
    A &= \{a_0, a_1, a_2, a_3, a_4, sw1\} \\
    \succ &= \{(a_0, sw1),(sw1, \text{amount} > 10K, a_1),(sw1, \text{amount} < 10K, a_2),(a_2, \text{risk} = \text{high}, a_1), \\
            (a_2, \text{risk} = \text{low}, a_3),(a_3, a_4),(a_1, a_4)\}
\end{align*}
\]
3.4 Schedule Coverage

As stated in previous section, BPEL has provided flow-construct to achieve the concurrent execution. Below is the example:

![Flow-construct illustration](image)

Figure 3-3. Flow-construct illustration

Figure 3-3 shows portion of purchase order process. The path on the left represents the payment flow and the path on the right represents the delivery logistics flow. These two paths can be executed at the same time. The execution order of activities are non-deterministic which it is possible receive payment executed earlier than Invoke delivery or vice versa. However, the company may want the Invoke Delivery to be proceeded only payment from customer is received. For this situation, a link is added which has similar form as the activity transition defined in previous section. The link is represented as (a0, predicate, a1).

For any two activities a0 and a1 defined in two defined sequences, schedule pair is defined as a 2-tuple (a0, a1) of activities to refer the execution associated with these two
activities. **Schedule order** (a0 < a1) represents the runtime execution order of these two activities. In this case, a0 is executed before a1. It should note that it doesn’t mean a0 must be immediately followed by a1 but it just shows the relative execution order of these two activities.

Consider the following simple sample. It has totally 4 schedule pairs which are (A,C), (A,D), (B,C) and (B,D) and it has totally 8 schedule orders which are (A<C), (A<D), (B<C), (B<D), (A>C), (A>D), (B>C) and (B>D).

![Diagram](image)

*Figure 3-4. Illustrative flow example*

Next, a schedule order (a0, a1) is covered in runtime when a0 is executed before a1 during runtime execution. Hence, we define **schedule coverage** as the covered schedule orders during process execution. To cover all 8 schedulers order above, the simplest transition set contains two execution set which are p1>p2 and p2>p1, where p1=(a<b) and p2=(c<d).
3.5 Fault Model

In this thesis, it focuses on the concurrent constructs provided by BPEL specification. They are flow and link constructs. Based on figure 3-4, we consider adding one link from B to C. With added the link, we may have the following 4 faults that can happen. Remind that the link is expressed in the form of (a0, predicate, a1).

Case 1: The link specifies a wrong source or target activities.

Case 2: The link specifies wrong transition predicate.

Case 3: The link has the transition direction reversed.
Case 4: The link is missing.

Link (B, C) is missing
Chapter 4 Empirical Study

4.1 Experimental Setup

In the experiment, we use 2 open-source BPEL applications [7] that are publicly available on the Internet to evaluate our work. Purchase order and Book buying are modified from the sample projects shipped with Oracle BPEL Process Manager Development suite. These two case studies are often used in WS-BPEL studies.

We require the applications to be executed in our desired order but we can’t control the scheduler of the Oracle BPEL process engine. So, we develop a standalone schedule controller tool and create a custom BPEL construct called Control Point. The control point is inserted before any activity and it will consult with schedule controller if the current activity is permissible to proceed.

Next, we create 5 different faulty versions for each application by seeding one fault into the subject program with control point inserted. 4 faulty version will contain one of the faults discussed in section 3.5 and 1 faulty version will seed a predicated fault which is often seeded in tradition program for study. In total, 10 faults were injected in these subject applications.

Based on the process notation discussed in section 3.3, we applied Spec Explorer [21] to create combination of different execution order and 400-500 execution sequences are generated for each application. Next we adopted the following process to generate a test suite for our proposed scheduled coverage criteria, activities coverage criteria, and random
testing [14][15]. The tool adds the test case to the test suite under construction only if the test case increases the desired coverage. The process terminates if either 100% coverage of the criterion is reached or an upper bound of 80 trails is reached. For random testing, we randomly select a test suite whose size is the same as the maximum number of test cases in all test suites for criteria on the same program version.

Once the test suites are generated, we execute these tests against the original programs as test oracle. That is we record the output result and compare it with the output from the faulty version. The test suite construction process is repeated 10 times to obtain 10 test suites for each application.

![Figure 4-1. Experiment procedure flow graph](image)
4.2 Data Analysis

The results are presented in this section. We analyze the fault-detection capability of the testing criterion. From the result, it shows the proposed schedule coverage is more valuable than the activities coverage. The proposed schedule coverage is much higher than the activities. Furthermore, it shows that the test suite generated from randomly selected test cases does also perform better than the activities coverage.

<table>
<thead>
<tr>
<th></th>
<th>Activities Coverage</th>
<th>Schedule Coverage</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book Buying</td>
<td>24%</td>
<td>86%</td>
<td>46%</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>36%</td>
<td>90%</td>
<td>44%</td>
</tr>
</tbody>
</table>
Chapter 5 Conclusion

In traditional test criteria such as activities, it does not consider the execution order of different tasks. Hence, the non-deterministic or race condition may not be reviewed by these criteria effectively. However, BPEL process is long running process and concurrent construct in BPEL is useful feature for parallelizing tasks. In the generation of the schedule-based test pool, although the complexity is large, it can review the concurrent issue effectively against from other traditional testing criteria.

In order to control the execution of the process to meet the generated schedule, we inserted a ‘control point’ before any activities and they are allowed to pass only if the schedule controller has sent the allowance message to these control points. If the control point has not received the message for long time, it will throw an exception to indicate that the process can’t make progress.

To consider the threats to validity of our empirical study, we only use a limited number of programs but they are common subjects used in most BPEL studied. Also, we use certain types of fault in our experiments. Similar to most other empirical studies, the result of our empirical study may not be generalized to cover all cases.

There are many studies in BPEL Unit framework and test cases generation for BPEL process while there is lack of studies in evaluating the testing criteria against BPEL process. Our empirical study is conducted to evaluate some test criteria derived from traditional sequential program testing. In addition, we consider the concurrent construct provided from the BPEL and proposed the schedule-based test criteria. We generated the test suites
from 2 kinds of test criteria adequacy and presented the experimental results to show the fault detection effectiveness for each of the test criteria.

When the BPEL process gets more complex and larger, the generation time of test pool based on schedule order is long and we have to take more considerations to reduce the generation time. Also, we plan to work on automatic test generation based on the proposed schedule-based criteria from BPEL source code. In addition, the fault handling construct is not considered in this study which could be also included in future work.
Bibliography and References


    http://www.w3.org/TR/ws-arch/.

11. WS-Transaction specification.

12. WS-Federation specification.


