Data Flow Testing of Service Oriented Work Flow Applications

December 9, 2009

Abstract

Service-oriented applications use XPath extensively to integrate loosely-coupled workflow steps. A mismatch among components (e.g., extracting the wrong contents or failing to extract any content from a correct XML message) may cause an application to function incorrectly. XPath should be studied deeply to improve the quality of the applications. In the development of the web services, XPath play a crucial role to capture the messages. Sometimes, XPath can extract error messages, so if XPath can be combined with mutation analysis, the error messages can be avoided. The mutation analysis helps to develop effective tests or locate weaknesses in the test data used for the program or in sections of the code. The time taken to find the error messages and to remove them is critical in the context of XPath. Data flow testing will be done on the test cases generated by the XPath. The effectiveness of the application can be measured by using the fault detection rate. In this paper a technique called the mutation analysis with the use of finite state machine was used to find the mismatch among the components. This approach can be implemented for open source applications. The fault detection rate and the time taken to find the mismatches occured are used to evaluate the approach. We will evaluate this approach by performing the dataflow testing on the open source applications until all the errors in the XML messages are found.

1 Introduction:

To develop a service-oriented workflow application, software engineers often employ a collection of heterogeneous but closely related technologies, such as WS-BPEL. To integrate loosely-coupled workflow steps, XML is used [1].

XML is fundamental to many service-oriented workflow applications, and XPath is the means to query on XML documents. XPath expressions play a central role in both querying XML databases and searching for XML data which are distributed over the web. The extensive usage of XPath poses a demand to study how to test these applications effectively. Most failures in applications involve the execution of an incorrect definition [3]. Mismatches in XML manipulations through XPath (such as to relate incoming and outgoing messages) may also result in failures. Using incompatible (in the sense of semantics) extracted messages to conduct follow-up workflow activities in an application may result in integration errors.

We believe that the use of data-flow testing leads to a richer test suite and helps to deal with these failures in an enhanced way if a control flow graph is used. As an example, by using Xpath expressions in the digital subscriber line application, we can find the zip code of a user. In data-flow testing, the first step is to model the program as a control flow graph. This helps to identify the control flow information in the program. In step 2, the associations between the definitions and uses of the variables that are needed to be covered in a given coverage criterion is established. In step 3, the test suite is created using a finite number of paths from step 2.

The goal of this proposal is to analyze the XML messages received, by doing so, we can know the control flow information and the error messages can be easily captured. We do this by constructing a mu-

tation graph on which the dataflow testing is performed.

2 Background:

A service- oriented application separates the functions of an application into distinct units, or services, which developers make accessible over a network in order to allow the users to combine and reuse them in the production of applications. These services communicate with each other by passing the data from one service to another, or by coordinating an activity between two or more services. Each service implements one action, such as filling out an online application for an account, or to find the zipcode of a user.Web services can implement a service-oriented architecture. Web services make functional building-blocks accessible over standard Internet protocols independent of platforms and programming languages.

WS-BPEL(Work space Business Process Execution Language) applications are a kind of service-oriented application. They use XPath extensively to integrate loosely-coupled workflow steps. Existing testing research does not adequately address the interactions among XPath, XML schema, and XML messages, and their relationships with BPEL.

Modelling a BPEL program as a control flow graph involves two steps:

1. An approach to identify the data flow associations relevant to the conceptual variables in XPath and ordinary variables in the BPEL program.

2. Formulation of a set of test adequacy criteria to measure the quality of test sets.

XPath is a language for navigating an XML document and selecting a set of element nodes. XPath expressions are used to query XML data, describe key constraints, express transformations, and reference elements in remote documents. XPath processing algorithms should be effective in time and space efficiency. XML messages often read error messages. So to avoid the error messages, mutation analysis is used. The mutation analysis uses XPath only for the error messages extracted.

To apply data flow analysis and testing to an XRG, we should, therefore, respect the ordering of nodes; otherwise one may construct illegitimate data flow associations or miss legitimate ones. We do not know whether the legitimate path was taken or not.

2.1 Related work:

Early work in the area was (necessarily) mostly concerned with proposals for data models and query languages for semistructured data. After the introduction of XML and XPath much of the research converged.

Modeling BPEL and web service components using a state model is popular. Mongiello and Castelluccia[8] translate a BPEL program into such a model and apply model checking to verify temporal properties. Apart from using a state model to represent a BPEL program, Foster[3] translated web services into Promela for formal verification using their tool WSAT. Different fragments of XPath are covered and translate an XPath strictly according to the definition of XPath expressions while they translate an XPath into a Promela procedural routine that uses self-proposed variables and codes to simulate XPath operations. Intuitively, a test suite covering the data flow associations in a translated routine would test the implementation rather than the declaration as expressed in the WS-BPEL application.

Benedikt, Fan and Kuper[8]were the first to find error messages for the service-oriented workflow applications, but they were able to find only for some fragments of XPath. They did not use any tool to find these erroroneous messages [8]. XPath 2.0 is an expression language that allows the processing of values conforming to the data model, which provides a tree representation of XML documents. The structure of an XPath is denoted by an XPath Rewriting Graph.

An XPath Rewriting Graph (XRG) [1] forms an explicit artifact to represent different paths conceptually defined in an XPath expression over a schema Ω.

An XRG for an XPath Query is a 5-tuple <q, Ω, Nx, Ex, Vx>.

The integration of heterogeneous sorts of techniques such as program representation, dataflow analysis, and declarative semantics on diverse types of artifact can be done. Tse and Chen[1] developed a data structure known as XPath Rewriting Graph (XRG) to capture how an XPath can be rewritten from one form to another in a stepwise fashion, and proposed an algorithm to construct XRGs. An XRG captures the mathematical variables to support stepwise rewriting of XPath.

Mei [1] has proposed an XPath Rewriting Graph to represent an XPath with a model of XML documents. Here, XRGs are revisited to facilitate the description of the techniques. An XRG is built on XPath syntactic constructs. Their technique treats the definitions as left-to-right rewriting rules and, through a series of rewriting, transforms an XPath into a normal form or a fixed point. Chan applied metamorphic relations to construct test cases for stateless web services. Many existing techniques on data flow testing are based on information obtained from program code without considering artifacts like XPath.

2.2 Limitations:

The previous research techniques used for service-oriented workflow applications did not focus on the error messages found at the deeper level of the control flow graph. The limitations of the current testing methods for service-oriented workflow applications are:

1. Testing of software built on top of the service-oriented architecture (SOA) is tedious[1].
2. There is no automated process for the tasks of the web services[2].
3. No proper standard for specifying and executing workflow specifications[8].
4. Evaluating an XPath query and the document validation problem[4].
5. No tool to convert a XML document into a state-transition model[5].
6. The faults can only be detected at a deeper level of the symbolic execution tree. The path conditions are complex[3].
7. Mismatches may occur in Xpath manipulation and the entire application may fail[7].

3 Challenges and goals:

Stefan Bottcher [5] proposed that mismatches may occur in manipulation, and the testing of the software built on top of the service-oriented architecture is tedious as the integration errors can occur. Tse[1] used XPath technique to reduce the integration techniques, but not all the errors are discovered and reduced by using only XPath.There is no automated process for the tasks of the service oriented workflow applications . The manual testing of the new applications is a difficult and error-prone task. For the service-oriented applications XPath is extensively used to integrate loosely-coupled workflow steps. XPath plays a key role in workflow Integration and may extract wrong data from the XML messages received, resulting in erroneous results in the integrated process. These techniques are proposed by Mei, chen and Tse [1].So, all the faults cannot be detected as the faults can only be detected at a deeper level of the testing. But can we reduce fault detection rate by using an automated tool?

4 Proposed research:

4.1 Proposed novel approaches:

We can model the components of the service-oriented workflow application using a state model. The workflows of the service-oriented workflow application are translated into state model by checking the quality of test cases generated. These test cases are generated for the XML messages received. A data structure known as XPath Rewriting Graph (XRG) can be used to capture how an XPath can be rewritten from one form to another in a stepwise fashion. The mutation graph consists of all the XML messages received as the nodes and dataflow testing is performed

for each message received.

The mutation analysis works in the following manner:Until the effective quality of the test cases is reached, the mutation analysis will be applied to each wrong read XML message. So we can locate weaknesses in the test data used for the application. In mutation analysis, if one of the test fails, then the mutation analysis is applied again. From the state model, a mutation graph can be developed which jumps from one state to another after a wrong data has been read from XML message. If correct data is read from the XML message, a state model is created. Then the dataflow testing will be performed.

This approach is different from others, as more error messages can be removed effectively as we are using a mutation graph. Also, the time taken to remove the error messages will be less.

Apart from using a state model to verify the interactions between the applications, we can treat the application as a finite state machine, and faulty versions can be generated for each state of finite state machine. These faulty versions can be checked by using SPIN[3]. In the finite state machine, we treat each error message extracted from XPath as a state and we try to find the behavior of the error messages. The behaviour of error mesages are recorded and if the same error messages occur again , we do not consider them. After an error message is recorded, we seed faults into that state and apply the dataflow testing for that state. Then we continue for all the states until we achieve 100

If there are same error messages that repeat over a period of time, finite state machine is useful as the behavior is recorded. If there are large number of error messsages, mutation analysis can be used.

So by using the state model along with the mutation analysis we eliminate the error messages extracted. By following the above steps, we can achieve more accurate testing, as more erroroneous XML messages are discovered.

5 Evaluation Plans:

We can use open-source applications and an automated tool can be created that creates test suites and performs the above approach. We generate different faulty versions by seeding one fault into the application before the test suite is created. The minimal, mean and maximal coverage that have been achieved by the test suites can be recorded and can be compared with other methods.

RQ1: Effectiveness: The fault detection rate can be taken as the effectiveness measure in the experimentation, which is defined as the proportion of the number of test suite that can expose the fault(s) in a version to the size of the test suite. Also the time taken to find the faults is considered.

RQ2: Precision: We can find more errors as we test the XML messages received effectively. So we can find more errors in much less time.

6 Foreseen contributions:

This work is very useful for web service applications. These days, most of the data is shared through web applications. So testing of these applications should be done effectively. The fault detection rate with respect to the time is used to measure the quality of test sets. The main contribution of the paper is to model XPath in a cost effective way. The approach is promising as the first set of experiments are used to evaluate the impact of XPath using open-source applications. When a tool is created to automate the approach, much less time is required to find the erroroneous messages. In this proposal we have seen an approach, using mutating analysis. The user can choose any technique depending on the requirement.

References

[1] Data Flow Testing of Service Choreography, L. Mei, W.K. Chan, and T.H. Tse at ACM SIGSOFT in 2009.

[2] Simulation, Verification and Automated Composition of Web Services, S. Narayanan and S. A. Mcllraith at World Wide Web Conference in 2002.

[3] Model-Checking verification for reliable web service, Howard Foster, Sebastian Uchitel, Jeff Magee, Jeff Kramer at OOPSLA in 2003.

[4] The complexity of XPath query evaluation and XML typing, Howard Foster, Sebastian Uchitel, Jeff Magee, Jeff Kramer at JACM in 2005.

[5] Testing Xpath Queries using Model Checking, Stefan Bottcher at World Wide Web Conference in 2006.

[6] Semi-proving: an integrated method for program proving, testing, and debugging, Y. Chen, T.H. Tse, and Z.Q. Zhou at IEEE, Transactions on Software Engineering in 2009

[7] Structural properties of XPath fragments, Michael Benedikt, Wenfei Fan and Floris Geerts at ACM in 2003.

[8] Modelling and verification of BPEL business processes at Third International Workshop on Model-Based Methodologies in 2006.