Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 4 (2): 137-146, 2010
ISSN 1819-4311

Regression Testing Method Based on XML Schema for GUI Components

Gagandeep Jyotsna Sengupta
Department of Computer Science, Punjabi University, Patiala, India

Abstract: Today XML (eXtensible Mark-up Language) is being accepted as an effective medium in data exchange over the web due to its modeling ability to support multiple inheritance and querying capabilities. In this study, we extend XML Schema based approach for testing software components. Component based software development emphasizes the design and construction of software using reusable components. The reliability and robustness of the software depends on testing the functional correctness of components. However, due to lack of information about the externally developed components, system testers generally can't perform effective testing (especially regression testing). It involves testing the modified program with some test cases in order to re-establish confidence that the program will perform according to the new specifications. The new version should behave exactly as the old except where new behavior is intended. In today's scenario, Graphical User Interface (GUI) is by far the most popular means used to interact with software. An important aspect in GUI testing is specification of system behavior and generation of test scripts. We have proposed XML schema based formalization approach for specifying and testing GUI components; as they combine the great potential of describing input data in open and standard form, with testing activity. We have worked upon specification of GUI component in XML which is validated. A Regression Testing Engine based on XML schema has been designed that generates test sequences based on specific coverage criteria, compares the two schemas for different versions of component and finally generates the regression test suite. Results have been validated for different case studies.

Key words: Component, XML , regression testing, graphical user interface, event sequence

INTRODUCTION

Software testing is critical element of any kind of software quality assurance and is of utmost importance for the credibility of software system. For Component-Based software systems, testing mechanisms and strategies varies from the traditional testing approaches. As components evolve rapidly over time, there is a need for efficient regression testing. Testing a GUI component is challenging, as its characteristics are different from that of conventional software. It is completely event based and hierarchical in nature. GUI-based programs are event driven, where an event is triggered when the user interacts with the program through GUI. Common user interactions include moving or clicking the mouse, selecting a graphic object, typing into a text field, or closing a window. Another important aspect is specification of system behavior and generation of test scripts. With evolution of

XML specification language, XML Schemas are very convenient for definition of interface specifications, definition of data models, protocol specifications etc. XML based stored procedures provide higher degree of independency between database server and the software components. Further it supports existence of heterogeneous platforms and heterogeneous servers. XML has established itself as the de facto standard form for specifying and exchanging data and documents between almost any digital or web applications. Therefore, we have adopted XML Schema Based Testing approach for describing input data in open and standard form to automate test data generation.

This study proposes an approach to automate generation of regression testing for GUI components by incorporating XML based testing support. We believe that schema based testing stands at a testing level above traditional code based testability. We design a Regression Testing engine that incorporates Test case checker to categorize the test suite and test case generator to generate regression test suite for GUI component.

Improved reuse and reduced cost benefits from software components can only be achieved in practice if the components provide reliable services, which makes component analysis and testing a key activity. As most of the components are commercial, testing remains a crucial problem in the absence of enough information. Several different approaches have been proposed for testing Graphical user interfaces. Bertolino *et al.* (2007a) presented Category partition method which provides a stepwise intuitive methodology for the testing of functional units from their specifications written in structured, semiformal language. Further Bertolino *et al.* (2007b) proposed XPT approach for the systematic derivation of XML Instances from a XML Schema. XPT applies to the XML notation, a well-known method for software black-box testing. Haw and Lee (2007) introduced a suitable and compact labeling scheme to identify the relationships in XML documents when modeled as trees. Mao *et al.* (2006) proposed Windows Navigation Networks (WNN) to model the usage of GUI software where vertexes in the model represent windows and arcs represent transitions between windows. Important usage paths of GUI software were extracted WNN.

Bai *et al.* (2005) analyzed generation of web services test cases automatically based on the Web Services Description Language (WSDL). The WSDL file is first parsed and transformed into structured tree. Then test cases are generated from two perspectives: test data generation and test operation generation. Test data are generated by analyzing the message data types according to standard XML schema syntax. Wang *et al.* (2005) introduced XML Schema approach to model polymorphism in object oriented data models. They extend XQuery to support the polymorphic feature.

To evaluate the quality of XML schema McDowell *et al.* (2004) proposed a set of eleven metrics to measure the quality and complexity of XML Schema and conforming XML documents. To provide an easy view of these metrics, two composite indices have been defined. Atif M. Memon has worked a lot in the field of GUI testing. Memon *et al.* (2001) presented new coverage criteria for GUI testing based on GUI events and their interactions. A unit of testing called a GUI component is defined. They identify the events within each component and represented them as event- flow graph. Memon (2001) developed a unified solution to the GUI testing problem with particular emphasis on the integration of tools and techniques to be used in various phases of GUI testing. The main contribution of his work is comprehensive framework for testing GUI's.

The main objective of our study is to visualize the importance of XML in specifying and querying test sequences for GUI and for generating regression test suite.

XML SCHEMA-BASED TESTING

XML schema based Testing is the approach that enables the detection of errors, evaluation and approval of system qualities. As it is independent of the test platform, it improves the transparency of the test process, increases the objectiveness of the tests and make test results comparable. Test specifications are defined in an unambiguous, standardized notation which makes them easier to understand and document. The test process involves generating XML documents with some modifications with respect to the original XML document and using queries to these documents to validate the schema. The XML documents and queries are generated according to a set of fault classes defined for the XML schema. The format of XML schema is defined:

- Defines elements and attributes that can appear in a document.
- Defines which elements are child elements
- Defines the number and order of child elements
- Defines whether an element is empty or can include text
- Defines data types for elements and attributes

Xml schema offers the following advantages:

- **Stronger data typing:** XML schema permits great flexibility in constraining the data type of element content. The XML Schema Recommendation specifies more than 30 data types that can be used to constrain element (and attribute) content
- **Extensible data typing through regular expressions:** XML schema supports regular expressions for defining the type of element content. This capability is particularly useful and powerful for defining and validating customized data types
- **Better content modeling:** XML schema constrains the order and number of child elements in mixed content model
- **Extensibility mechanisms:** It provides the extensibility mechanisms, for extending not only definitions within a schema, but also for extending entire schema instances. . XML Schemas are extensible, because they are written in XML
- **Interoperability:** XML schema provides interoperability using a flexible, open, standards-based format, with new ways of accessing legacy databases and delivering data to web clients

ARCHITECTURE OF REGRESSION TESTING ENGINE

Regression Testing Engine constitutes two important components (1) test case checker and (2) test case repairer. Any GUI Component is represented in XML schema based on its specifications. It includes GUI component, its events, type of event, event coverage like inter coverage and intra coverage. The XML Editor used is XML SPY 2009. The XML instances of the XML Schema of original component are stored in an array. The instances are stored in the form of sequences which are traversed by implementing the all-path coverage criteria in VB.net.

Programs have been written in VB.net to create test cases, to compare the two versions in terms of inter and intra modification of events and to distinguish test cases of different versions. Following are the Algorithms for test instances creation, their comparison and categorization and regression test suite generation.

Algorithm 1

Creation of Test Cases

Step 1: Create XML instances of the XML Schema of original component
Step 2: XML instances are stored in the form of event sequences of the component
Step 3: Find all possible paths in the event sequences
Step 4: Apply all-path and inter- component coverage criteria to generate sequences of test cases
Step 5: Find out if the test case executable or not
Step 6: Count the length of each sequence which is traversed
Step 7: Find out the total number of test cases generated
Step 8: Stop

Test Case Sequence (T) for a GUI component is a pair (S0, e1; e2;...; en), consisting of a state S0_SI, called the initial state for T and a legal event sequence e1; e2;...; en.

If the initial state specified in the test case is not reachable in the GUI and/or its event sequence is illegal, then the test case is not executable.

Algorithm 2

To compare and distinguish event sequences of components.

Step 1: Take XML instances created from XML schemas of both components
Step 2: Compare event sequences of the original and the newer(modified) component
Step 3: Event sequences which are identical are present in both the components
Step 4: Find out the event sequences which are present only in the original component
Step 5: Mark them as deleted sequences.
Step 6: Find out the new event sequences in the modified component and mark them as added
Step 7: Check the parent node of the deleted event sequence in the original component and the added event sequence in the modified component
Step 8: If the parent node is the same, it indicates the original event sequence can be repaired to generate new event sequence otherwise it is anew event sequence
Step 9: Mark the sequences as added , deleted repaired or new event sequences
Step 10: Go to step 1 until there is no more sequence left un-traversed
Step 11: Stop

GUI test cases can be classified as Usable test cases, Unusable test cases and Repairable Test cases. GUI test case (S0, e1; e2; ...; en) is usable if it can execute to completion on a modified GUI. GUI test case (S0, e1; e2;...; en) is unusable if a modification of a GUI causes the state S0 to not be reachable in the GUI or if the sequence e1; e2;...; en can-not execute to completion. Unusable test cases cannot be executed on the GUI and are usually discarded. An unusable test case is repairable if its initial state S0 is reachable and its event sequence can be made legal for the modified GUI, i.e., (ei, ek>i) _ E) or {(ei, ex), (ex, ek>i)}_E for E, for 1 = i = n of the modified GUI.

Algorithm 3

To generate Regression test Suite

Step 1: XML schema of the modified component is evaluated
Step 2: Traverse the schema by adopting all-path coverage criteria

Step 3: Find out the event sequences marked as added. These are the newer event sequences for the modified component. It indicates some new functionality

Step 4: Find out the event sequences marked as repaired. These are the event sequences traced from the original component where parent node remains the same and sequence of events have changed

Step 5: Find out the event sequences which are reusable that is, they remain the same in both the components

Step 6: Find out the complete set of test sequences. It is the final regression test suite generated

CASE STUDY

For practical implementation, Notepad version 4.6 and Notepad version 5.0 as well as Adobe version 5 and version 7 are considered. Based on specification, both the components are represented in XML which is validated by the XML Schema written in the XML SPY 2009. XML instances of the corresponding schema are stored into an array in the form of event sequences as these components are GUI in nature. Further, sequences are traversed for all possible combinations to generate test cases satisfying inter and intra coverage criteria. Test checker component of Regression Testing Engine checks the validity of test instances for the second component and classify them as reusable, repaired, deleted and added events. Test case generator finally creates regression test suite. Figure 1 shows architecture of

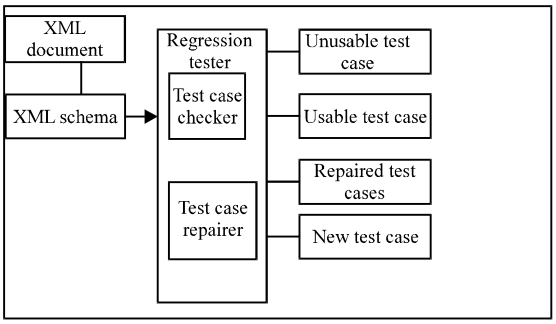

Fig. 1: Architecture of regression testing engine

Fig. 2: XML based GUI regression testing

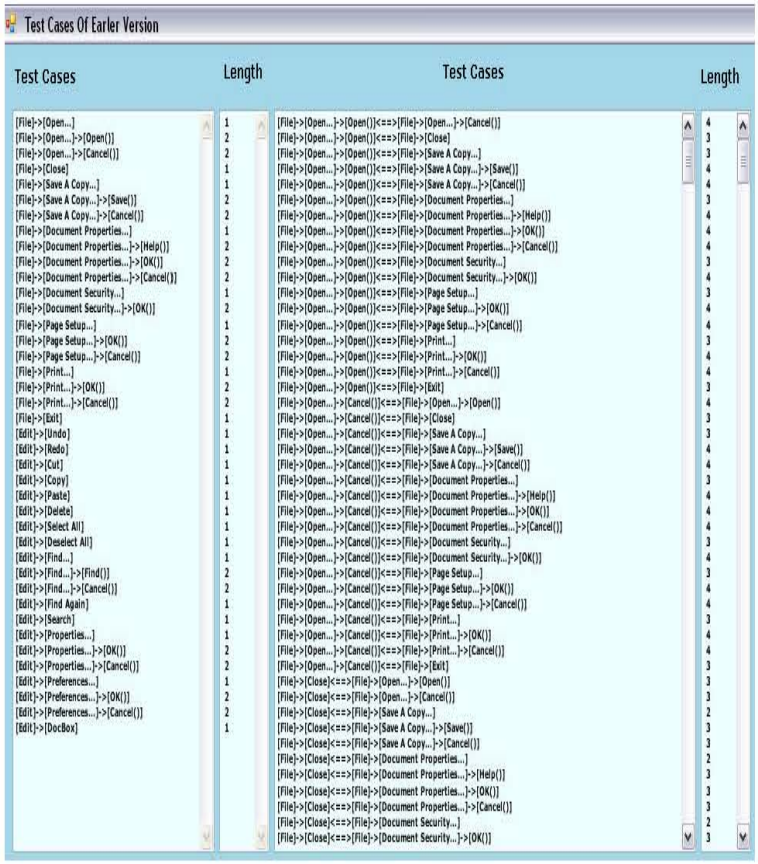

Fig. 3: Event sequences for adobe 5

regression testing engine. It takes as input XML representation of GUI component. GUI's are hierarchical in nature. The state of a GUI is not static; events preformed on the GUI change its state. The events E = {e1, e2... en} associated with a GUI are functions from one state of the GUI to another state of the GUI. A set of states SI is called the valid initial state set for a particular GUI if the GUI may be in any state Si_SI when it is first invoked. Given a GUI in state Si_SI, i.e., in a valid initial state of the GUI, new states may be obtained by performing events on Si. These states are called the reachable states of the GUI. Another important part of regression testing engine is test case checker and repairer that traverses the XML schemas of the components to find out valid event sequences for both as well complete set of regression test suite (i.e., event sequences form the original component that can be used for the newer component).

Figure 2 shows interface designed for XML schema based testing of GUI components. User can interactively generate test case sequences for GUI components. XML Schema needs to be defined initially for the components. Different features of the interface can be used to compare and to generate test sequences.

Figure 3 shows different event sequences generated for Adobe 5. For practical demonstration parent component FILE is taken. Event sequences are generated based on

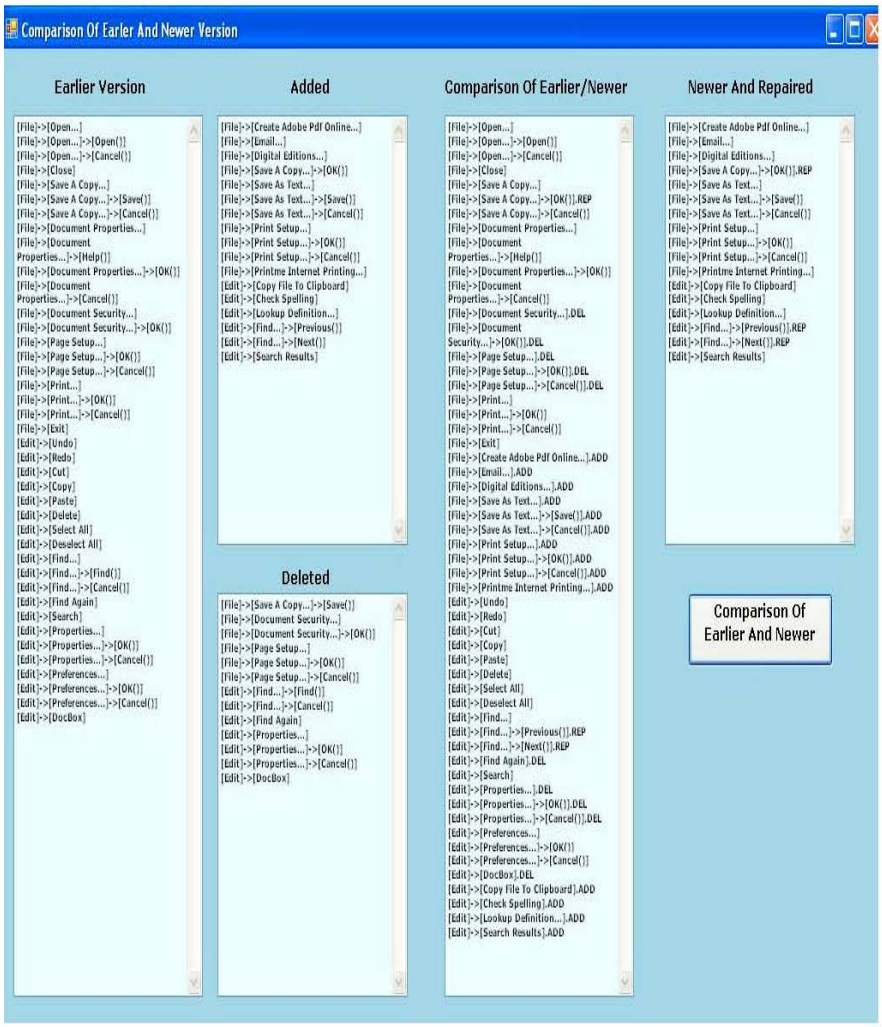

Fig. 4: Comparison of earlier and newer version

event-interaction coverage criteria. We have carried out comparison of earlier and newer version in terms of added and deleted event sequences. The event sequences are suffixed with ADD, DEL or REP to indicate the status. The results are shown in Fig. 4. The complete set of valid test cases that are usable, new and repaired form the earlier test case sequences for the modified component i.e., Adobe 7 are shown in Fig. 5.

For practical demonstration, initially only two main components of Adobe are considered. Figure 6 shows partial XML representation of Adobe 5 component based on XML schema. Test case sequences or event sequences of varying lengths are generated for Adobe 5. Total number of test sequences of Length 1 are 10, Length 2 are 11, Length 3 are 5, Length 4 are 32 and length 5 are 25. Thus in all 83 test cases are generated. The results are shown in Table 1. Out of these 83 event sequence, 33 are found to be reusable for the newer component, 15 are repaired and 92 are new event sequences for the modified component.

Fig. 5: Generation of regression test suite

From the initial set of 83 event sequences, 35 are found to be obsolete. i.e., they are no longer valid for regression test suite. The results of generation of regression test suite from the initial version of component are shown in Table 2.

CONCLUSIONS AND FUTURE WORK

This study continued on previous work in the field however we provide the usage of XML specification language, to test GUI components. An appealing feature of this approach is automation aimed at generating the regression test suite from the XML representation of component. An automated regression test system which is well designed and implemented can efficiently support testing work. The testing engine is proved to have the following

```
<MenuBar>
        <Menu Name="File">
                <MenuItem Name="Open...">
                        <Event Name="Open()"/>
                        <Event Name="Cancel()"/>
                </MenuItem>
            <MenuItem Name="Close">
                                <MenuItem Name="Save A Copy...">
                    <Event Name="Save()"/>
                    <Event Name="Cancel()"/>
              </MenuItem>
            <MenuItem Name="Document Properties...">
                        <Event Name="Help()"/>
                        <Event Name="OK()"/>
                        <Event Name="Cancel()"/>
              </MenuItem>
            <MenuItem Name="Document Security...">
                    <Event Name="OK()"/>
              </MenuItem>
            <MenuItem Name="Page Setup...">
                        <Event Name="OK()"/>
                        <Event Name="Cancel()"/>
              </MenuItem>
      <MenuItem Name="Print...">
                        <Event Name="OK()"/>
                        <Event Name="Cancel()"/>
              </MenuItem>
```

Fig. 6: XML representation of adobe 5 (partial)

Table 1: Test sequences generated for Adobe 5

Length 1	Length 2	Length 3	Length 4	Length 5	Total test cases
10	11	5	32	25	83

Table 2: Regression test suite for Adobe 7

Deleted test cases	Repaired test cases	New test cases	Reusable test cases	Total test cases
35	15	92	33	140

characteristics: (1) The system uses XML to express the test suite and display it. This simplifies the system implementation as well as improves development efficiency of the system. (2) We also provide a visual editor to generate test sequences of different lengths, for classification of test cases. It helps test designer save effort and ensure accuracy of the results.

There are several areas for future work. Currently our approach handles static components, Dynamic objects are other entities in a document that needs to be tested. Moreover, XML schema can be judged for quality by considering quality factors before its usage for test data generation.

REFERENCES

Bai, X., W. Dong, W.T. Tsai and Y. Chen, 2005. WSDL-based automatic test case generation for web services testing. Proceedings of IEEE International Workshop on Service-Oriented System Engineering, Oct. 20-21, IEEE Computer Society, Washington DC, USA., pp: 215-220.

Bertolino, A., J. Gao, E. Marchetti and A. Polini, 2007a. Automatic test data generation for XML schema-based partition testing. Proceedings of the 2nd International Workshop on Automation of Software Test, International Conference on Software Engineering, May 20-26, IEEE Computer Society Washington, DC, USA., pp: 1-4.

Bertolino, A., J. Gao, E. Marchetti and A. Polini, 2007b. Systematic Generation of XML Instances to Test Complex Software Applications. In: Rapid Integration of Software Engineering Techniques, Guelfi, N. and D. Buchs (Eds.). LN CS., 4401, Springer-Verlag, Berlin, Heidelberg, ISBN-13: 978-3-540-71875-8, pp: 114-129.

Haw, S.C. and C.S. Lee, 2007. Structural query optimization in native XML databases: A hybrid approach. J. Applied Sci., 7: 2934-2946.

Mao, Y., F. Boqin, H. Zhenfang and Z. Li, 2006. Important usage paths selection for GUI software testing. Inform. Technol. J., 5: 648-654.

McDowell, A., C. Schmidt and K. Yue, 2004. Analysis and metrics of XML schema. Proceedings of the International Conference on Software Engineering Research and Practice, June 21-24, CSREA Press, pp: 538-544.

Memon, A., 2001. A comprehensive framework for testing graphical user interfaces. Ph.D. Thesis, University of Pittsburgh, USA., 2001.

Memon, A., M.L. Soffa and M.E. Pollack, 2001. Coverage criteria for GUI testing. Proceedings of the 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering, Sept. 10-14, Vienna, Austria, pp: 256-267.

Wang, G., D. Han, B. Qiao and B. Wang, 2005. Extending XML schema with object-oriented features. Inform. Technol. J., 4: 44-54.