Xml Schema Based Approach for Testing of Software Components

Manpreet Kaur
Assistant Professor
RIMT-MAEC
Mandi Gobindgarh

Neeshu Sharma
Assistant Professor
RIMT-MAEC
Mandi Gobindgarh

Reet Kamal Kaur
Assistant Professor
RIMT-MAEC
Mandi Gobindgarh

ABSTRACT
The widespread usefulness of Graphical User Interfaces has made GUIs the most important component of software today. As the GUI have characteristics like event driven input, mouse clicks etc., and the testing of conventional software cannot be applied on the GUI's. One of the most important innovations that strongly contribute to solve this issue has been the introduction of the Extensible Markup Language (XML) .The XML Schema based testing is introduced to combine the great potential of XML Schema in describing input data in open and standard form, with testing activity. We have theoretically analyzed different components based testing techniques especially XML based testing and regression testing. We have written the representation or specification of GUI in XML which is validated by XML Schema. Program have been written reads the XML and to generate the test sequences. We have developed XML Regression Test Suite Modeler to perform testing of GUI component. It includes Test Case Generator, GUI Comparer and Regression Test Suite Generator as the main components. A case study applying the proposed approach is described and results are presented.

Keywords
XML, XML Schema, GUI testing, test coverage, test cases, regression testing.

1. INTRODUCTION
Graphical user interface are the most popular means used to interact with today's software. The functional correctness of GUI is of utmost importance and it's required to ensure the safety, robustness and usability of entire software. GUI's are hierarchical in nature.GUI testing for functional correctness is a challenging research area. Testing is, in general, labor and resource intensive, accounting for 50-60% of the total cost of software development. GUI testing is especially difficult today because GUIs have characteristics different from those of traditional software, and thus, techniques typically applied to software testing are not adequate. Current GUI testing techniques are incomplete, ad-hoc, and largely manual**.** The test data required for testing GUI is huge sometimes so manual approach isn't suitable. Manual testing makes regression task more laborious and time consuming and longer times are needed for test coverage.

We propose a XML Schema based representation for GUI testing, which is capable of simulating the hierarchical nature of the GUI and stores the event interaction information. It satisfies a number of testing coverage criteria's for effective test case generation. Testing GUI's for functional correctness is needed to ensure the overall correctness of these applications. The proposed model also promotes reusability. Once created it can be quickly redeployed to generate additional test cases for the same or modified version of GUI, thus providing an effective regression testing facility.

The main objectives of XML Schema Based Testing are as follows:

- To explore the impact of XML Schema based testing on software component.
- To design an improved XML Schema testing technique that works both for built-in and third party component and follows the automation of test data generation
- Study the XML quality factors to optimize the XML Schema.
- Finally to validate the testing technique by using some coverage criteria for XML Schema based testing of components.
- To study the characteristics of GUI components in the software.
- Using the characteristics of GUI component to generate test data for various coverage criteria's.
- Creating a framework for generation of XML Regression Test Suite Modeler for GUI.

2. XML SCHEMA
XML Schema was approved as a W3C Recommendation in May, 2001 and is now being widely used for structuring XML documents for E-commerce and web Services applications. XML Schema Recommendation has significant value for XML based publishing applications. An XML schema describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD). The purpose of an XML Schema is to define the legal building blocks of an XML document. The XML Schema verifies and validates XML document and checks for its well-formedness. XML Schema defines the valid contents of particular elements and attributes as well as an ability to define the data structure and content of XML documents.

```
<?xmlversion="1.0"?>
<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://www.w3schools.com
note.xsd">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Fig1.1 XML Schema

3. XML SHEMA BASED TESTING

XML Schema based testing is the approach for the systematic generation of the XML instances. A comprehensive set of instances is generated by sampling all the possible combinations of elements within the Schema by applying and adapting the well known Category Partition Method. This method provides a systematic and semi-automated method for test data derivation, starting from analysis of specifications until production of the test scripts. The generated instances can then be used for the black-box testing of applications that expect such XML instances as input.

Characteristics of XML Schema Based Testing

- Interoperability.
- Well-formedness of XML Schema.
- Verifying XML Documents against the Schema Structure.
- Automation of test data Generation.
- Systematic Generation of a set of XML Instances.
- Integrating and Matching Schemas.

4. GUI (Graphical User Interface)

A GUI is a graphical user interface to a program. A GUI is a hierarchical, graphical front-end to a software system that accepts as input user-generated and system –generated events from a fixed set of events and produces deterministic graphical output. Most of today's software interacts with a user through a graphical user interface. Objects of a GUI include elements such as windows, pull-down menus, buttons, scroll bars, iconic images, and wizards. The software user performs events to interact with the GUI, manipulating GUI objects as one would real objects. These events cause deterministic changes to the state of the software that may be reflected by a change in the appearance of one or more GUI objects.

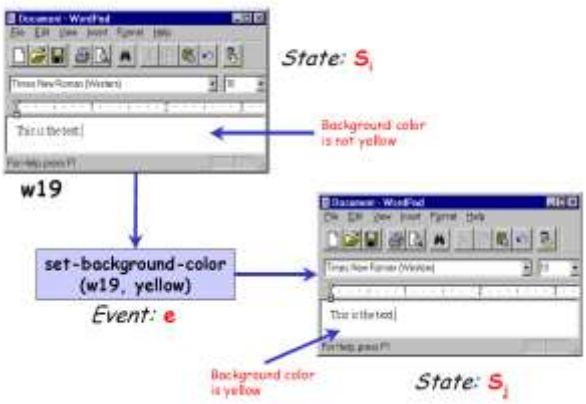

Fig 1.2 An Event changes the state of the GUI .

Thus the important characteristics of GUI's include their graphical orientation, event driven input and hierarchical structure, the objects they contain, and the properties (attributes) of those objects.

5. PROPOSED METHODOLGY

A GUI component can be represented with the help of XML. The XML representation is validated by XML Schema. The XML structure models all possible interaction among the events within a component. The structure is a general structure used for implementing a number of testing technique like regression testing. The steps of the proposed methodology are given below:

Step1: Specification of GUI Components Using XML Schema.

Step2: Creation of test Cases for earlier Version.

Step3: Comparison of Components in terms of inter and intra Modifications of events.

Step4: Distinguish Test Cases of two versions.

Step5: Generation of Regression test Suite.

Configuration Required

- **Operating System**: Windows XP Service Pack 2
- **RAM** :1 GB
- **Hard Disk Space Required** : 3 GB
- **XML Schema Editor** : XML SPY 2009
- **Language Platform:** VB.Net 2005 Version Professional Edition

Our Proposed Methodology is to generate XML based GUI Regression Test Suite. In this the Original GUI component is represented by XML validated by XML Schema. The instances of XML are traversed to generate the test cases by using the coverage criteria. Then the original test cases are used for the generation of Regression Test suite. The XML based GUI Regression test suite is shown.

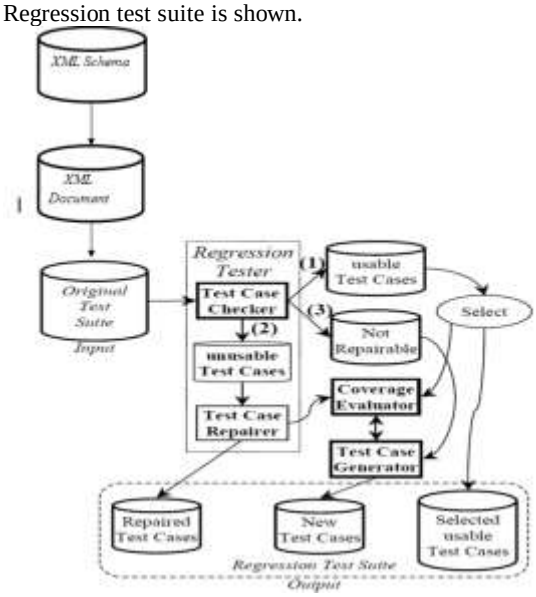

Fig 1.3 XML based GUI Regression Test Suite

Step1: Specification of GUI Components Using XML Schema:

The Specification of the GUI Component is represented in the XML which is validated by creating XML Schema on the basis of information gathered.

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="Employee"
                minOccurs="0"
                maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="SSN="xsd:string>
            <xsd:element name="Name" type="xsd:string"/>
            <xsd:element                    name="DateOfBirth"
type="xsd:date"/>
             <xsd:element                name="EmployeeType"
type="xsd:string"/>
              <xsd:element name="Salary" type="xsd:long"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
```

Fig 1.4 XML Schema

Step2: Creation of test Cases for earlier Version:

Length n Event sequence Coverage: This criterion says that we can generate test sequence up to a limit of n events per test case. Testing all event sequences that start in one event and ends in another event.

Test Case: GUI test case T 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

Input ……XML of Original Component
Output …… Test Cases, Length of Test Cases, Total no. of Test Cases

1. *Store the XML document into an array in the form of sequences.*
2. *Traverse the array by implementing all the coverage criteria.*
3. *Test cases are generated for original component.*
4. *Compute the length of each test case.*
5. *Compute the test cases in each length.*
6. *Compute total no. of test cases.*

Step3: Comparison of Components in terms of inter and intra Modifications of events:

Input……XML of Original Component and Modified Component

Output……..Added and deleted event sequences

1. *Store the XML of Modified Component into an array in the form of sequences.*
2. *Compare the XML of both the Components.*
3. *Compare original component with the modified component.*
4. *If event sequences in both are same, it means the sequences which were in original component are also in modified component.*
5. *If event sequences are not same, then these are deleted event sequences from original component*
6. *Compare modified component with the original component.*
7. *If event sequences in both are same, it means which are in modified component were also in original component.*
8. *If event sequences are not same, then these are added events in modified component*.

INTRA Component Coverage Criteria: It focuses on checking all possible interaction among all possible pair of events in a component. It requires that each event in the component should be executed at least once.

INTER Component Coverage Criteria: The goal of inter component criteria is to ensure that all interactions among the components are tested. In GUI's the interaction takes the form of invocation of components and termination of components and the event sequence that start in one component and ends in another component.

Step4: Distinguish Test Cases of two versions:

COMPONENTS	Length 1	Length 2	Total Test Cases
File and Edit	22	18	40

Usable test cases: GUI test case (S0, e1; e2; . . . ; en) is usable if it can execute to completion on a modified GUI.

Unusable test cases: 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.

Repairable Test cases: 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 $1 \leq i \leq n$ of the modified GUI.

Input……...Test sequences of original component, Deleted and Added event sequences

Output……Repaired and reusable event sequences

1. *Search the deleted sequences in the test sequences of original component.*
2. *Find the positions of deleted sequences in the test sequences of original component.*
3. *Compare the test sequences of respective positions with the added event sequences.*
4. *If the parent event of both the sequences is same then replace deleted sequence with the added sequence.*
5. *Suffix the repaired sequences with .rep, deleted with .del in the original test sequences.*
6. *The added sequences which are not replaced are newer sequences and suffix with .add.*
7. *These newer sequences are now added in the original component test sequences.*
8. *Remaining event sequences in the original component are the reusable test cases.*

6. RESULTS AND DISCUSSIONS.

The GUI testing is done with the help of XML Schema Based Testing. In this the GUI environment is represented through the XML which is validated by the XML Schema. So, the test case for the earlier version is created by reading the XML representation which is done by creating the VB.Net program. The test cases for Modified version are created by using the Regression Test Suite.

CASE STUDY: Adobe Reader 5 and Adobe Reader 7

Results of Adobe Reader 5 Test Cases*:*

Results of Regression Test Suite of Adobe Reader 7 Test Cases:

Adobe Reader7 Test Cases	Reusable	Added	Repaired	Time
44	27	14	3	60 sec

The reusable test cases found as 27 (61.36 %), Repaired test cases were found as 3 (6.81 %), added test cases as 14 (31.81 %). The rest observations are summarized in the above table. Reparability factor as relative to total test cases= (repairable/ test cases) * 100 = 6.81 %

7. FUTURE WORK

While doing some research work there is high probability that you will find more issues or areas which could be worked upon. In this dissertation the area of research was GUI but all the algorithms and techniques discussed apply on static components. Dynamic objects are another thing which keeps on changing like we see on some of the website, text logo's keep on changing in the same area.

Algorithms discussed in dissertation are not extended able for dynamic objects handling so another important area of research is testing of GUI' applications having dynamic Objects and Components.

Traversing GUI's produce large number of event sequences hence producing large test data set so some work can be done for prioritizing, extracting or reducing test data in such a way that all the major paths are covered.

6. REFERENCES

[1] Antonia Bertolino, Jinghua Gao, Eda Marchetti, Andrea Polini, "Automatic Test Data Generation for XML Schema-based Partition Testing", Proceedings of the second international workshop on automation of software test , 2007, DOI 10.1109/AST.2007.6.

[2] Antonia Bertolino, Jinghua Gao, Eda Marchetti, and Andrea Polini "Systematic Generation of XML Instances to Test Complex Software Applications", volume 4400/2007, 2007, pp 114-129.

[3] A. Bertolino, J. Gao, and E. Marchetti "XML every-flavor testing". In Proc. Web Information Systems and Technologies WEBIST 2006, Setbal, Portugal, April 2006.

[4] A McDowell, C Schmidt, K Yue -"Analysis and Metrics of XML Schema" SERP'04, Proceedings of the International Conference on, 2004 - sce.uhcl.edu.

[5] AR Houser "XML Schemas for Publishing Applications" Proceesings of XML, 2001 - idealliance.org.

[6] A Bertolino, A Polini –"A framework for component deployment testing" Software Engineering, 2003. Proceedings. 25th International, 2003, pp221-231, ieeexplore.ieee.org.

[7] B Hasling, H Goetz, K Beetz - " Model Based Testing of System Requirements using UML Use Case Models" Software Testing, Verification, and Validation, 2008 1st …, 2008 – pp 367-376 ieeexplore.ieee.org.

[8] B Sumak, M Hericko, M Pusnik – "Towards a framework for Quality XML Schema Evaluation" Information Technology Interfaces, 2007. ITI 2007. 29th …, 2007 - pp 783-788 ieeexplore.ieee.org.

[9] D Lee, WW Chu – " Comparative Analysis of Six XML Schema Languages" pp 76-87, volume 29, ACM SIGMOD Record, 2000 - portal.acm.org.

[10] DS Rosenblum -"Adequate Testing of Component Based Software" Univ. California, Irvine, TR UCI-ICS-97-34, 1997 - cs.ucl.ac.uk.

[11] J Gao – "Component Testability and Component Testing challenges" Proceedings of International Workshop on Component-based … - diku.dk.

[12] MJ Harrold, D Liang, S Sinha - "An Approach to Analyzing and Testing Component Based Systems" Proceedings of the First International ICSE Workshop on …, 1999 - cc.gatech.edu.

[13] Offutt, W Xu - " Generating Test Cases for Web Services Using data Perturbation" ACM SIGSOFT Software Engineering Notes, 2004 - portal.acm.org, pp1-10.

[14] S Ghosh, AP Mathur - "Issues in testing Distributed Component based Systems"… ICSE Workshop on Testing Distributed Component-Based Systems, 1999 - citeseer.ist.psu.edu

[15] Trace Galloway " Principles of XML Schema design " XML 2002 proceedings by deepX.