TCML – An XML-Based Test Case Format
Mohd Farid Jaafar, Mohd Hasan Selamat, Abdul Azim Abdul Ghani
Faculty of Computer Science and Information Technology
University Putra Malaysia
43400 UPM Serdang Selangor, MALAYSIA
{mfarid, hasan, azim}@fsktm.upm.edu.my

Abstract - Testing is important and gives the confidence to the developer and user; but it is expensive and may consume at least 50% of the total costs involved in developing software. This cost is just for a normal application but if it involves extreme and critical software, the number might increases. As we know, designing test suite for testing is a tedious and meticulous work. We need to have a deep understanding of the program before we could design a good test suite and even an expert tends to miss some test cases. Manual testing for large-scale program is very expensive and frustrating for humans. The same scenario applies in Visual Programming Language (VPL). Visual Programming (VP) is a much complex program than conventional one. Testing VP program give users the confidence on what they visualize is correctly transformed according to their perception. Most studies in designing test suites are using requirement specification, complex technique (i.e. Z) and third-party tools that require novices and non-expert user to supply information that is not familiar to them. It is certain that what they supplied might be inaccurate and thus affects the test suite produced. Furthermore, there are no standard structures in exchanging the data about a program.

The objective of this paper is to propose an XML-based test case language called Test Case Markup Language (TCML).

1. INTRODUCTION

Testing is a process of examining the functionality and correctness of a program by running it with a certain condition. It is also known as any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

Software testing is a process used to identify the correctness, completeness and quality of the developed software. It is an important process in supporting quality assurance [1, 2] and represents the ultimate review of specification, design and coding [3]. It impacts all phases of the development, including the requirements and specifications preparation [4]. Testing usually performs for one or two reasons: (1) defect detection, and (2) reliability estimation.

Testing is important and give confidence to the developer and user; it is expensive and may consume at least 50% of the total costs involved in developing software [2, 3]. This cost is just for a normal application but if it involves extreme and critical software, the number might increases. The main concern in testing is the effectiveness of finding an error. If it doesn’t find any error, it can be considered as failed. This can be avoided by careful design of test case and good selection of test suite.

Most studies in designing test cases are using requirement specification, complex technique i.e. Z and third-party tools that require novices and non-expert user to supply information that are not familiar to them. It is certain that what they supplied might be inaccurate and thus affects the test suite produced. Furthermore, there are no standard structures in exchanging the data about a program.

2. XML

The eXtensible Markup Language (XML) is an industry-standard format for representing structured documents, storing and transferring data. Data stored in XML is all text, identified by tags, similar to HTML tags. The tags in XML are not predefined as they are in HTML. The tags can identify fields by name. In addition to an XML file, there are also XML schema file. The schema describes the fields, data types, and any constraints, such as required fields. The schema is also defined with XML tags. The format of XML data offers several advantages for programming [5]. Some of XML most important features are:

Simplicity. Structure information obtained by named opening and closing tags. As a text format, XML can be easily read, produced, and transmitted.

Generality. From its general structure any desirable data format can be instantiated.

Standardization. The XML standard is developed and controlled by the World Wide Web Consortium. Many large and important companies are supporting XML.

Altogether, these characteristics make XML a first choice data format for data exchange and representing data. Many applications for XML have
been created for structuring data in various fields includes:

- Mathematics (MathML),
- Chemistry (CML),
- Speech (SpeechML) and

There exist quite a few graph languages for XML by now. The flavor of formulating graph differs, though, and there are also differences in their expressiveness. There have been number of research on graph representation using XML, among them are GraphXML, GraphML, XGMML and GXL.

2.1 GraphXML

GraphXML is a graph description language in XML that can be used as an interchange format for graph drawing and visualization packages. The generality and rich features of XML make it possible to define an interchange format that not only supports the pure, mathematical description of a graph, but also the needs of information visualization applications that view graph-based data structures. A list of features supported by GraphXML includes: embedded and hyperlinked data references, hierarchical (nested) graphs, graph metadata such as graph theoretic properties, presentation styles such as color and geometry, and edit actions for graph transformations and history capture.

The goal of GraphXML is to provide a general interchange format for graph drawing and visualization systems, and to connect those systems to other applications. What we are trying here is to map the source code of the VPL using graph. The requirements of information visualization have greatly influenced design decisions during the development of GraphXML. Although GraphXML can be used for the description of purely mathematical graphs, restricted to the set of nodes and edges, information visualization applications require more features. The simplicity of GraphXML is great but when in come to the complex source code, it seems complex and not suitable for test case representation. Furthermore, we do not care about the visual appearance of the graphs, such as the color of the edges, images used in a window containing a specific graph, only the value and operation of the node.

2.2 GraphML

GraphML (Graph Markup Language) is an XML-based format for the description of graph structures, designed to improve tool interoperability and reduce communication overhead [6]. A key feature of GraphML is the separation into structural and data layer, both conceptually and syntactically; this enables applications to extend the standard GraphML vocabulary by custom data labels that are transparent to other applications not aware of the extension. With the use of XML syntax, GraphML can be used in combination with other XML based formats. GraphML uses a SQL-like language mainly extended by path expressions.

Furthermore, applications are free to ignore unknown concepts appearing in the structural layer, such as <port>s, <hyperedge>s or nested <graph>s. They are even powerful enough to specify advanced transformations that go beyond mapping XML elements directly to other XML elements or other simple text units. However, advanced transformations may result in long-winded style sheets that are intricate to maintain, and most likely to be inefficient. Extension functions appear to be the natural way out of such difficulties. We found that, as rule-of-thumb, XSLT should be used primarily to do the structural parts of a transformation, such as creating new elements or attributes, whereas specialized extensions are better for complex computations that are difficult to express or inefficient to run using pure XSLT.

2.3 XGMML

XGMML (eXtensible Graph Markup and Modeling Language) is an XML application based on GML, which is used for graph description. XGMML uses tags to describe nodes and edges of a graph. The purpose of XGMML is to make possible the exchange of graphs between different authoring and browsing tools for graphs. The conversion of graphs written in GML to XGMML is trivial. Using XSL with XGMML allows the translation of graphs to different formats. XGMML was created to be use for the WWWPAL System that visualizes web sites as a graph. Web Robots can navigate through a web site and save the graph information as an XGMML file. XGMML, as any other XML application, can be mixed with other markup languages to describe additional graph, node and/or edge information.

An XGMML document describes a graph structure. The root element is graph and it can contain node, edge and att elements. The node element describes a node of a graph and the edge element describes an edge of a graph. Additional information for graphs, nodes and edges can be attached using the att element. A graph element can be contained in an att element and this graph will be considered as sub graph of the main graph. The graphics element can be included in a node or edge element, and it describes the graphic representation either of a node or an edge.

XGMML documents can be validated against the DTD, xgml.dtd. XGMML well formed documents
can be part of other XML documents. XGMML is based on Graph Modeling Language (GML), but uses XML to express the information rather than the text format used by GML. Because XGMML is an XML application, it can take advantage of XML tools, notably XSLT. XGMML is useful for general graphing functionality and it can be used for Web site mapping as well. XGMML can also be used to represent RDF information or other structures like PowerPoint documents.

3. STRUCTURE of TCML

Test Case Markup Language (TCML) is derived from GraphML but with modification to suit the test requirement. Appendix A shows a simple graph by TCML. The TCML document consists of a graphml element and a variety of sub elements: graph, node, edge. In the remainder of this section we will discuss these elements in detail and show how they define a graph.

3.1 Header

In this section we discuss the header parts of the document, which are common to all TCML documents, basically the graphml element. Fig. 3.1 shows the header of TCML.

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<graphml xmlns="urn:tsg-schema">
    ...
</graphml>
```

Fig. 3.1. Header of TCML

The first line of the document is an XML process instruction, which defines that the document adheres to the XML 1.0 standard, and that the encoding of the document is UTF-8, the standard encoding for XML documents. Standalone means that this document is standalone document, not attach to other document.

The second line contains the root-element element of a GraphML document: the graphml element. The XML Attributes are needed to specify the XML Schema for this document. In the example, we use the standard schema for TCML documents. The attribute, xmlns="urn:tsg-schema", defines xmlns as the XML Schema. The XML Schema reference is required because it provides means to validate the document.

3.2 The Graph

A graph is denoted by a graph element. A node is declared with a node element, and an edge with an edge element. Fig. 3.2 shows the element of TCML.

```xml
<graph>
    <node key="id1" type="2">
        ...
    </node>
    <edge source="id5" target="id2"/>
</graph>
```

Fig. 3.2. Element of TCML

All nodes must be declared first, followed by the edges. This is done to ensure that the data are in order. The value of a TCML-Attribute for a graph element is defined by a data element nested inside the element for the graph element. The data element has an XML-Attribute key, which refers to the identifier of the TCML-Attribute.

3.3 The Node

Nodes in the graph are declared by the node element. Each node has an identifier, which must be unique within the entire document, i.e., in a document there must be no two nodes with the same identifier. The identifier of a node is defined by the XML-Attribute, key. Nodes also contain another identifier, which indicate the type of the node. The type is not unique and there can be more than one node with the same type. The type of the node is defined by the XML-Attribute, type. Fig. 3.3 shows how the node in TCML is declared.

```xml
<node key="id1" type="2">
    ...
</node>
```

Fig. 3.3. Node of TCML

Within the node, there are three elements that explain the detail about the node. The elements are op, p1 and p2. The op elements represent the operation of the node. It can be a start node, end node or arithmetic operation that is done by the node. p1 and p2, represent the variables of the node. Fig. 3.4 shows the example of these elements.

```xml
<node key="id1" type="2">
    <op>START</op>
    <p1></p1>
    <p2></p2>
</node>
```

Fig. 3.4. Element inside Node TCML

3.4 An Edge

Edges in the graph are declared by the edge element. Each edge must define its two endpoints with the XML-Attributes, source and target. The value of the source and target must be the identifier of a node in the same document. The source represents the node that points to the next node. Target represents the
node that is pointed by the source node. Fig. 3.5 shows how an edge is defined.

In TCML there is an order defined for the appearance of target and source attributes. It is important to have a source and target for each node because from there we will see how the program is related to each other. Another important aspect here is to make sure that the source and target are written correctly. The source must be specified before the target.

Example 3.5. Edge of TCML

4. SCHEMAS of TCML

In validating the language, we used “XML Schema Definition” (XSD). With XSD, the structure of an XML document can be constrained, which specifies all possible elements, attributes, and nesting. XSD represents a kind of type information for XML values. An XML document conforming to a XSD is said to be valid.

We stressed XSD for two reasons: first, certain information that is currently available on VPL will be transformed into TCML without conforming to or even having a particular XSD; this will be particularly true for large amounts of unstructured data. It should be possible to query all these information sources with the proposed language; however, relying on the presence of XSD could be in many cases prohibitive. Secondly, we do not want to burden the users to have knowledge of the concept of a “schema”. Nevertheless, there are ways to nicely exploit XSD when they are given. In particular, the user interface can greatly benefit from XSD.

5. CONCLUSION

The focus of this paper is to create an XML-based language as an input for the test case generator. Several researches have been done but not in the area of generating test case; the researches focus on using XML as data visualization. Although the theory is applicable, but there are some adjustment and refinement that need to be done in order to choose the attribute and element that fit in the testing context.

The current implementation of TCML is at the empirical stage. For the language to be used in representing all programming languages, more work and studies need to be done to strengthen this language and make full use of the XML technologies.

In term of the language, the design only applies to VPL. To make full use of the language and its capability, it should be tested on structural programming and object oriented programming. The language should also have the capability to allow new attribute that is relevant than the one that is pre-specified.

REFERENCES