TAXI - A Tool for XML-Based Testing

Antonia Bertolino, Jinghua Gao, Eda Marchetti, Andrea Polini
ISTI-CNR
Via Moruzzi, 1 56124 Pisa, Italy
{antonia.bertolino, jinghua.gao, eda.marchetti, andrea.polini}@isti.cnr.it

Abstract

We present the tool TAXI which implements the XML-based Partition Testing approach for the automated generation of XML Instances conforming to a given XML Schema. In addition it provides a set of weighted test strategies to guide the systematic derivation of instances. TAXI can be used for black-box testing of applications accepting in input XML instances and for benchmarking of database management systems.

1 Introduction

Modern complex software systems are increasingly built according to a modular architecture, within which precise functionalities can be identified. Such modularization allows for the separate development of various components able to interact with each other when integrated into a larger system. Clearly integration requires that the exchanged data are defined into a precise and checkable format. In this regard, fundamental has been the growth of a general consensus towards adoption of standard open formats for data specifications, most notably the recent introduction of the eXtensible Markup Language (XML) and of the XML Schema.

It is our claim here that these standard technologies for data exchange can improve not only the *inter-operability* between (sub)systems adopting such format, but also their *testability*. In fact, we found that the *partition testing* techniques can be quite naturally applied over the XML Schema, since it provides an accurate representation of the input domain into a format suitable for automated processing.

We present TAXI (Testing by Automatically generated XML Instances), which is a tool for the systematic generation of XML instances [1]. The TAXI methodology is largely inspired to the well-known Category Partition (CP) technique [2], which provides a stepwise intuitive approach to functional testing, as follows: identify the relevant input parameters; define the environment conditions; combine their significant values into an effective test suite. In order to reduce the number of combinations generated by CP, TAXI also integrates a set of weighted test strategies. The user can either fix the number of generated instances or distribute them according to defined importance factors (weights). We are currently extending TAXI for also deriving instances containing specified errors (i.e. invalid instances) for checking the robustness of an application reading the XML files.

In the next section we discuss how TAXI derives XML instances from XML Schema and show a simple example of its use.

2 Main Steps of TAXI

In this section we describe briefly the function of TAXI step by step[1].

Input XML Schema. The user interface reads the input XML Schema and lays it down as a tree structure. During this process the default weights are uniformly distributed to the children of each `choice` element. They belong to the [0,1] interval and, are so that the sum of the weights of each `choice` is 1.

Weight Assignment. The user can modify the default weights assigned to the children of `choice` elements, expressing the relative "importance" of a child with respect to the other children of the same `choice`. TAXI automatically checks if the weights of each `choice` sum to 1. Figure 1 shows the TAXI interface of weight assignment.

Database Population. There is a database containing the values for the instance derivation. The user can populate the database by: i) inserting the values manually by means of TAXI; ii) letting TAXI download the values from specific source; iii) letting TAXI recover the values from the definitions of input XML Schema.

[1]For space limitations, we assume knowledge of XML and XML Schema, and in the text we will refer to the elements of XML Schema using this `format`

29th International Conference on Software Engineering (ICSE'07 Companion)
0-7695-2892-9/07 $20.00 © 2007 IEEE

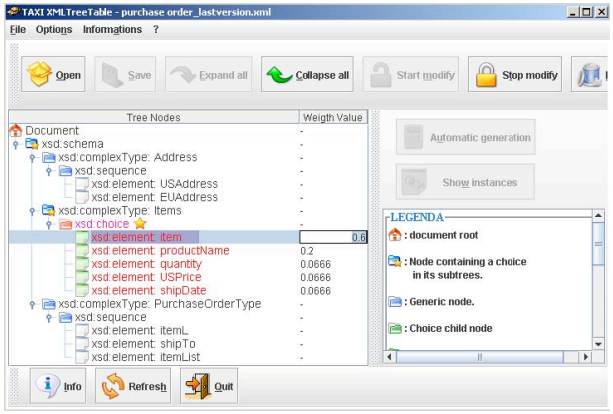

Figure 1. Weight assignment

Choice Analysis. TAXI extracts a set of sub-Schemas, each one containing a different child of `choice` elements. It automatically normalizes the final weights associated to the various sub-Schema elements.

Strategy Selection. TAXI provides three test strategies: *With a fixed number of instances*, that could be in practice the case in which a finite set of test cases must be derived. *With a fixed functional coverage*, when a certain percentage of functional test coverage (e.g. 80%) is established as an exit criterion for instances generation. *With a fixed functional coverage and number of instances*: the above mentioned strategies are combined.

Figure 2 shows the interface for test strategy selection.

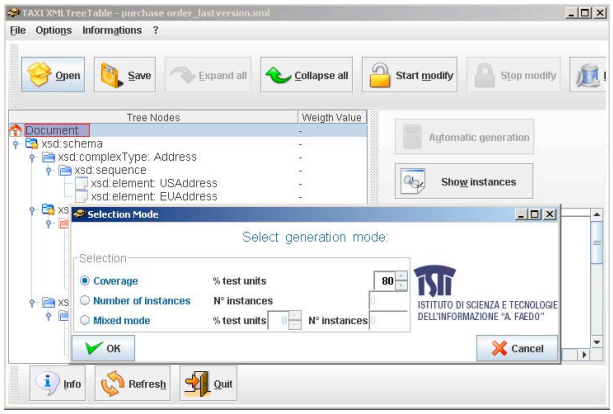

Figure 2. Test strategy selection

Instance Derivation. Accordingly with the test strategy adopted TAXI automatically generates a set of instances. For this TAXI provides specific values to element occurrences and assigns values from the database to the sub-Schema elements. Figure 3 shows a very simple XML Schema used as input with TAXI and two of the instances that can be generated by the tool.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="purchaseForm">
 <xsd:complexType>
 <xsd:all>
  <xsd:element name="Address">
   <xsd:complexType>
    <xsd:choice>
        <xsd:element name="USAddress"  type="xsd:string"/>
        <xsd:element name="EUAddress" type="xsd:string"/>
    </xsd:choice>
   </xsd:complexType>
   </xsd:element>
   <xsd:element name="PurchaseOrdre" type="PurchaseOrderType"/>
 </xsd:all>
 </xsd:complexType>
</xsd:element>
<xsd:complexType name="PurchaseOrderType">
<xsd:sequence>
  <xsd:element name="productName" type="xsd:string"/>
  <xsd:element name="quantity" type="xsd:positiveInteger"/>
  <xsd:element name="USPrice"   type="xsd:decimal"/>
  <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

a)

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseForm>
 <PurchaseOrdre>
   <productName>book</productName >
   <quantity>10</quantity>
   <USPrice>10.5</USPrice>
   <shipDate>2005-09-02</shipDate>
 </PurchaseOrdre>
 <Address>
   <USAddress>
     3269 Sherwood St. Montgomery
   </USAddress>
 </Address>
</purchaseForm>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseForm>
 <PurchaseOrdre>
   <productName>shoes</productName >
   <quantity>2</quantity>
   <USPrice>100</USPrice>
 </PurchaseOrdre>
 <Address>
   <USAddress>
     900 N Walnut Creek Dr. STE 100
   </USAddress>
 </Address>
</purchaseForm>
```

b)

Figure 3. An XML Schema and derived XML instances from TAXI.

3 Conclusions

On the tester's side, TAXI targets the long-standing dream of automating the generation of test cases for black-box testing, which is traditionally done manually by expert testers that analyze specifications of the input domain written in natural or semiformal language. If the input is formalized into XML Schema, then XPT can provide a much more systematic and cheaper strategy.

References

[1] A. Bertolino, J. Gao, E. Marchetti, and A. Polini. Systematic generation of XML instances to test complex software applications. In *Rapid Integration in Software Engineering, (RISE 2006)*. LNCS publication to appear, September 2006. Geneve, Switzerland.

[2] T. Ostrand and M. Balcer. The category-partition method for specifying and generating functional tests. *Communications of ACM*, 31(6), 1988.