Systematic document generation from XML Schema

Antonia Bertolino, Jinghua Gao, Eda Marchetti, Andrea Polini

name.surname@isti.cnr.it

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
(ISTI-CNR), Pisa
Agenda

- XML and XML Schema
- Motivating XML-based Partition Testing (XPT)
- Category Partition(CP) & Mapping from CP to XPT
- XPT Methodology
- TAXI Tool
- Applications
- Conclusion
The eXtensible Markup Language (XML) is a Markup Language which is today a de-facto standard to store information and data.

XML documents are tree structured documents in which data are formatted/organised using tags.

```xml
<?xml version="1.0" encoding="ISO88591"?>
<card>
  <name>John Doe</name>
  <title>CEO, Widget Inc.</title>
  <email>john.doe@widget.com</email>
  <phone>(202) 4561414</phone>
</card>
```
XML & XML Schema

XML Schema provides a means for defining the structure and content of XML documents.

In the open networked world, XML Schema support interoperability between independently developed applications.
Automatic XML-Based Testing and Benchmarking

What we would like to achieve:

Agreed XML Schema

Generally not Finite set!!
Automatic XML-Based Testing and Benchmarking

EASY WAY

Not Structured set, difficult to derive properties!!

Some tools like that exist: XMLSpy, sunXMLGenerator, ...

Agreed XML Schema
Our proposal: A Systematic Automatic Approach

XML-based Partition Testing XPT

The approach has been inspired at-large by the well-known Category Partition methodology for systematic semi-automated test generation...

..or, you can think of it as grammar-based generation, on the XSD syntax, although we have also introduced practical rules.
Mapping CP to XPT

<table>
<thead>
<tr>
<th>CP</th>
<th>XPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze Specifications</td>
<td>Preprocessor</td>
</tr>
<tr>
<td>Identify Functional Units</td>
<td>Identify Sub-Schema Sets</td>
</tr>
<tr>
<td>Partition Categories</td>
<td>Identify Types</td>
</tr>
<tr>
<td>Selecte Choices</td>
<td>Partition Values and Structures</td>
</tr>
<tr>
<td>Determine Constraints</td>
<td>Determine “valid/invalid” constraints</td>
</tr>
<tr>
<td>Generate Test Specification</td>
<td>Generate Intermediate Instances</td>
</tr>
<tr>
<td>Generate Test Cases</td>
<td>Generate Final Instances</td>
</tr>
</tbody>
</table>
Identification of Sub-Schema Sets

<choice> elements partition the XML Schema into distinct sets corresponding to the CP functional units.

XML Schema

- choice A
- choice B

XML Schema

- sequence A
- sequence B

XML Schema

- sequence 1
- sequence 1

XML Schema

- sequence 2
- sequence 2

Mapping from CP to XPT

- Analyze Specifications
- Identify Functional Units
- Partition Categories
- Select Choices
- Determine Constraints
- Generate Test Specification
- preprocessor
- Identify Sub-Schema Sets
- Identify Types
- Partition Values and Structures
- Determine "valid/invalid" Constraints
- Generate Intermediate Instances
- Generate Final Instances

TAXI

SE Day @ Rutgers  4/22/08
Identification of Types

The CP categories in XPT correspond to the occurrence and types of XML elements.

EX: String, sequence, all
Partition of Values and Structures

Values of the elements:
- **Element attributes**: fixed, default ...
- **Restrictions**: “minInclusive”, “maxInclusive”, “minExclusive”, “maxExclusive”, “minLength”, “maxLength”

Information of the structure of the final instances
- **Element**: “minOccurs”, “maxOccurs”
- **Attribute**: “use”
Two types of constraints can be identified

- **Valid**: values in choices conform to the specification of the XML Schema
- **Invalid**: values in choices do not conform to the declaration of XML Schema.
Example of “valid/invalid” constraints

**Sequence:**
- The same sequence of element as the specification of XML Schema [Valid]
- The sequence of element is different from the XML Schema specification [Invalid]

**Numeric:**
- Any digits conform to the specification of XML Schema [Valid]
- Any digits do not conform to the specification of XML schema. [Invalid]

**String:**
- Any strings conform to the specification of XML Schema [Valid]
- Any strings do not conform to the specification of XML schema. [Invalid]

**Occurrence:**
- Occurrence value $\in (-\infty, \text{minOccurs})$ [Invalid]
- Occurrence value $\in [\text{minOccurs}, \text{maxOccurs}]$ [Valid]
- Occurrence value $\in (\text{maxOccurs}, \infty)$ [Invalid]
Intermediate Instances

- Generate intermediate instance by combining the values of “minOccurs” and “maxOccurs”.

- We apply the conventional Boundary Condition test approach to reduce the combinations.

Mapping from CP to XPT

- Analyze Specifications
- Identify Functional Units
- Partition Categories
- Selecte Choices
- Determine Constraints
- Generate Test Specification
- Identify Sub-Schema Sets
- Identify Types
- Partition Values and Structures
- Determine “valid/invalid” Constraints
- Generate Intermediate Instances
- Generate Final Instances

**sub-Schema**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>minOccurs=0</td>
<td>minOccurs=2</td>
</tr>
<tr>
<td>maxOccurs=3</td>
<td>maxOccurs=4</td>
</tr>
</tbody>
</table>

**Intermediate Instance**

- A occurs=0
- B occurs=2
- A occurs=3
- B occurs=2

**Intermediate Instance**

- A occurs=0
- B occurs=4
- A occurs=3
- B occurs=4
### Instance Derivation

- The set of final instances is generated by giving the proper value to each element.

- The values are selected from the choices according to the restrictions expressed in the XML Schema.

---

#### Mapping from CP to XPT

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze Specifications</td>
<td>preprocessor</td>
</tr>
<tr>
<td>Identify Functional Units</td>
<td>Identify Sub-Schema Sets</td>
</tr>
<tr>
<td>Partition Categories</td>
<td>Identify Types</td>
</tr>
<tr>
<td>Select Choices</td>
<td>Partition Values and Structures</td>
</tr>
<tr>
<td>Determine Constraints</td>
<td>Determine “valid/invalid” Constraints</td>
</tr>
<tr>
<td>Generate Test Specification</td>
<td>Generate Intermediate Instances</td>
</tr>
<tr>
<td>Generate Test Specification</td>
<td>Generate Final Instances</td>
</tr>
</tbody>
</table>

---
Instance Derivation (2)

- The problem of CP method: Too many generations!
- Our solution:
  - Apply Pair-wise testing during the occurrence generation
  - Weighted Test Strategies

Mapping from CP to XPT

1. Analyze Specifications → preprocessor
2. Identify Functional Units → Identify Sub-Schema Sets
3. Partition Categories → Identify Types
4. Selecte Choices → Partition Values and Structures
5. Determine Constraints → Determine “valid/invalid” Constraints
6. Generate Test Specification → Generate Intermediate Instances
7. Generate Test Specification → Generate Final Instances

Test Strategy Selection

- Selection
  - Coverage: % test units
  - Number of instances: N instances
  - Mixed mode: % test units (0-80%) and N instances

Test Strategy: OK

Cancel
Main Interface of TAXI
The mapping from the CP to the XML Schema Partition Testing has been partially implemented in a proof-of-concept tool called **TAXI**: Testing by Automatically generated XML Instances

**TAXI** includes four components:

- **Schema Analyzer (XSA)**
  - Expands and preprocesses the XML Schema,
  - Prepares the intermediate instance frames
  - Provides a set of final instances

- **Test Strategy Selector (TSS)**
  - Implements a set of test strategies.
  - Manages the weight assignment for the elements in the identified functional units

- **Values Storage (VS)**
  - Manages a database for occurrences and values assignment

- **User Interface (UI)**
Potential Applications

- For validating database management systems
  - automatically generate valid XML instances for populating database in a systematic
  - evaluate the performance and the quality of the associated management systems
- For testing the inter-operability between applications and for enabling the correct interactions among the interfaces used by remote components in distributed systems.
  - Automatic and controlled generation of valid and invalid instances enables the automated testing of I/O behavior
- For verifying the proper communication protocols between web-services.
  - SOAP-based interaction between services can be reconducted to the corresponding XML Schemas
Black-box testing

XML Schema

Correct XML Instances

Almost Correct XML Instances

Additional Models

Tester

System Under Test

Verdict

are used

defines
100% automatic XSLT testing

XML Schema A (MARC21)  Correct XML Instances  A to B XSLT Stylesheet (SUT)

XML Schema B (DublinCore)  Tester

(Additional Model)  XML instances are (not) compliant with XML Schema B

XML Instances B??
Conclusions

- TAXI tool can automatically derive a set of instances that systematically covers a XSD
- It can be applied for interoperability validation, database benchmarking, black-box testing, ...

Future work
- Invalid instance generation: Robustness testing
- Tool refinement
- Experimental validation
Thank you!

To get TAXI, or for joint experimentation

• Beta Version of TAXI on line at http://labse.isti.cnr.it/
  Or

• send an email to antonia.bertolino@isti.cnr.it