Generating Test Cases for Web Services Using Data Perturbation

Wuzhi XU  
wxu2@gmu.edu  
Department of Information and Software Engineering  
George Mason University  
Fairfax, VA 22030-4444 USA  

Joint Research with Dr. Jeff Offutt

Supported by NASA
Introduction

- Black box testing of web services
- Focus on peer to peer interactions
- Data perturbation of existing XML message
- Based on XML schema & type rules
Web Services

- *Internet-based, modular applications that perform a specific business task and conform to a particular technical format.* – IBM

- An Internet-based, modular application that uses Simple Object Access Protocol for communication and transfers data in XML through the Internet.
Technologies in Web Services

- Common communication infrastructure
  - Widely supported
  - Internet-based
  - Implements standardized technical formats

- Advantages of this technology
  - Independent of hardware / software platform
  - Reduces the complexities and cost of software integration

- Web services technologies
  - Extensible Markup Language (XML)
  - Universal Description, Discovery and Integration (UDDI)
  - Web Services Description Language (WSDL)
  - Simple Object Access Protocol (SOAP)
Testing Web Services

- Challenges of testing web services
  - Will web services interactions be handled acceptably well?
  - Will requests be handled for all types of data?

- Characteristics with testing web services
  - No access to source
  - Low testability
  - Dynamic and loosely coupled
  - Heterogeneous
  - Independent platforms
  - No UI

- Testing capabilities involved in testing web services
  - Testing SOAP messages
  - Testing WSDL files and using WSDL files to generate test plans
  - Testing the publish, find, and bind capabilities of web services
Our Solution

1. Modify the request messages
2. Send the modified messages
3. Analyze the responses
Solution Characterization

- Formal model for XML
- Rules for perturbing values
- Peer-to-peer, not multilateral
- Testing SOAP messages
- Black box testing

Multilateral Web Services Interaction

Peer-to-peer Web services interaction
The Formal Model for XML Schema

The Regular Tree Grammar (RTG) is a 6-tuple

\(<E, D, N, A, P, n_s>\)

XML message and schema for the books example

The simplified XML message

\(<books>
  <book>
    <price>59.99</price>
    <year>2002</year>
  </book>
  <book>
    <price>69.99</price>
    <year>2003</year>
  </book>
</books>

The simplified XML schema

\(<xs:element name="books">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="book" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ISBN" type="xs:string"/>
            <xs:element name="price" type="xs:double"/>
            <xs:element name="year" type="xs:int"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>\)
The RTG Model for the example

- The RTG for the books
  
  \[ E = \{ \text{books, book, ISBN, price, year} \} \]

  \[ D = \{ \text{string, double, int} \} \]

  \[ N = \{ n_s, n_b, n_i, n_r, n_y \} \]

  \[ P = \{ n_s \rightarrow \text{books }<(n_b)^*> , \]
  
  \[ \quad n_b \rightarrow \text{book }<n_in_rny>, \]

  \[ \quad n_i \rightarrow \text{ISBN }<\text{string}>, \]

  \[ \quad n_r \rightarrow \text{price }<\text{double}>, \]

  \[ \quad n_y \rightarrow \text{year }<\text{int}> \} \]

- The derived tree for the books

```
books
    ↓
   book
   /   \
 ISBN  price year
```
Our Approach

- Data Perturbation (DP): modify the request messages and analyze the response messages

Two steps in data perturbation:
1. Data value perturbation: modifying values in messages based on their data type
2. Interaction perturbation: Interaction is classified into RPC communication and data communication
   1. RPC communication perturbation
   2. Data communication perturbation
Data Value Perturbation (DVP)

- Three data types are currently considered: string, numeric, and enumeration

- Boundary value test cases for these data types:
  - **String**: maximum length, minimum length, zero length, upper case, lower case
  - **Numeric**: maximum value, minimum value, zero
  - **Enumeration**: boolean set, finite set of values

- One example for numeric data type:

<table>
<thead>
<tr>
<th>Original value</th>
<th>Perturbed value</th>
<th>Test case</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;price&gt;59.99&lt;/price&gt;</td>
<td>2(^{63})-1</td>
<td>Maximum value</td>
</tr>
<tr>
<td></td>
<td>- 2(^{63})</td>
<td>Minimum value</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Zero</td>
</tr>
</tbody>
</table>
RPC Communication Perturbation (RCP)

- Test cases are generated in term of data uses
- Two kinds of data uses: normal use and SQL use
- Mutation analysis is extended to testing normal uses
- Traditional mutation operators are redefined
- SQL injection is extended to testing SQL uses
RCP Example

## Message before modification

```xml
<adminLogin>
  <arg0 xsi:type="xsd:string">turing</arg0>
  <arg1 xsi:type="xsd:string">enigma</arg1>
</adminLogin>
```

## Resulting SQL query

```sql
SELECT username FROM adminuser WHERE username='turing' AND password='enigma'
```

## Message after modification

```xml
<adminLogin>
  <arg0 xsi:type="xsd:string">turing' OR '1'='1</arg0>
  <arg1 xsi:type="xsd:string">enigma' OR '1'='1</arg1>
</adminLogin>
```

## Modified SQL query

```sql
SELECT username FROM adminuser WHERE username='turing'
  OR '1'='1'
  AND password='enigma'
  OR '1'='1'
```
Data Communication Perturbation (DCP)

- Test relationships and constraints in data communication messages

- Definitions of relationship and constraint:
  - Relationship is the parent-child association between two non-terminal elements
    It reflects the cardinality constraints in a XML schema
  - Constraint is the association between a non-terminal element and a terminal element
    It reflects the expression of element values
DCP Example

## Duplicate one book instance
<books>
  <book>
    <price>59.99</price>
    <year>2002</year>
  </book>
  <book>
    <price>69.99</price>
    <year>2003</year>
  </book>
</books>

## Delete one book instance
<books>
  <book>
    <price>59.99</price>
    <year>2002</year>
  </book>
</books>
Preliminary Case Study

- Used web services developed for the study
- Inserted faults by hand
- Evaluated tests in terms of numbers of faults found
The Web Services for the Experiment

- The Mars Robot Communication System
  - Five web services
  - Six servlets
  - Two Java beans

```
<table>
<thead>
<tr>
<th>The Space Station</th>
<th>Robots On the Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station (RPC)</td>
<td>Mars (RPC)</td>
</tr>
<tr>
<td>RequestStation (servlet)</td>
<td>RequestMars (servlet)</td>
</tr>
<tr>
<td></td>
<td>Robot (beans)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Houston</th>
<th>The Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scientist (servlet)</td>
</tr>
</tbody>
</table>

| Application server      | Database Server                          |
| (web-based)             |                                        |
| Administer (servlet)    | Database (RPC)                           |
| Login (RPC)             | Requestdatabase (servlet)                |
| ComputeData (RPC)       | RequestDataSave (servlet)                |
| RequestHouston (servlet)| DBAccess (beans)                         |
```

SOAP XML
Faults and Test Results

- 18 faults were inserted into the application

- 14 seeded faults found plus two natural faults found

<table>
<thead>
<tr>
<th>Number of faults</th>
<th>DVP</th>
<th>RCP</th>
<th>DCP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of test cases</td>
<td>100</td>
<td>15</td>
<td>27</td>
<td>142</td>
</tr>
<tr>
<td>Faults found (seeded)</td>
<td>14</td>
<td>7</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>Faults found (natural)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Conclusion

- Effective and practical way to generate tests for web services without requiring access to the source
- RPC communication perturbation: syntax-based modification on values in messages
- Data communication perturbation:
  - Little work on testing data communication message
  - Focus on testing the semantics
- Future work:
  - Relative usefulness of DVP, RCP, and DCP
  - Testing those applications that need no data inputs
  - Multilateral communication
  - Determining output correctness