Towards Efficient Measuring of Web Services API Coverage

Waldemar Hummer\textsuperscript{1}, Orna Raz\textsuperscript{2}, Schahram Dustdar\textsuperscript{1}

\textsuperscript{1} Distributed Systems Group, Vienna University of Technology, Austria
\textsuperscript{2} IBM Haifa Research Lab, Haifa University Campus, Israel

3rd International Workshop on Principles of Engineering Service-Oriented Systems (PESOS), co-located with ICSE 2011
Honolulu (Hawaii), USA, May 23-24, 2011
Talk Date: May 23
Overview

- Introduction
  - Motivation, Scenario
  - Approach Overview
- Related Work
- Exact Definition of Web Service API
- Configurable API Coverage Metrics
  - User-Defined Domain Partitions
- Computation of API Coverage
- Evaluation
- Conclusion and Future Work
Motivation

- **Coverage**: extent to which a system has been tested/used
  1) Important field in Software Testing
  2) Used to derive usage statistics

- Web Services (WS) are *Black Boxes*
  - WSDL defines the Application Programming Interface (API)

- **API Coverage**: \( \frac{\#(\text{distinct invocations})}{\#(\text{possible invocations})} \)
  - Possible WS invocations determined by XML Schema Definitions (XSD)

- **Problem**: Actual Coverage of 100% Often Infeasible
  1) Main Reason: Subject to *combinatorial explosion*
  2) Influencing Factor: Imprecise XSD types (e.g., *integer* without ranges)
  3) Finally: Not all parameter values/combinations equally important to be tested

- **Meaningful API Coverage by Reduction of Value Domains**
  - Exact specification of Web services API and parameter types
  - Define which parameter values and combinations are of interest
Related Work

- **Test and Analysis of Web Services**
  - Vivid research area with various fields (e.g., Baresi et al. [3])
  - Most approaches define coverage criteria and generate test cases
  - Group testing of services [17], Collaborative contract-based testing [2], Execution paths in WS-BPEL processes [10,13], Testing of data-centric service compositions [14,11], ...

- **Interface Based Web Services Testing**
  - WSDL and XML Schema based test case generation (e.g. [1,7])
  - Schema *perturbation* to generate invalid requests (Xu et al. [19])

- **Domain Partitioning**
  - Partition tests into sub-domains (Jorgensen and Whittaker [12])
  - *Category-Partition* test design pattern (Binder [8])

- **Testable Service** (Bertoloni et al. [4])
  - Exposes metadata and aggregated coverage data
  - But: Why not provide coverage calculation as a service?
Approach Overview

1) Prerequisite: Exact Definition Of Web Service APIs
   • Use of XSD Facets (String patterns, min/max values, ...)
   • Framework support for Java Web service (JAX-WS) developers

2) Log and Store Invocation Messages
   • Non-intrusive logging interceptor
   • Messages preprocessed and stored in an optimized format

3) Definition of User-Defined Coverage Metrics
   • Based on Domain Partitioning
   • Customizable, Reusable

4) Computation of API Coverage
   • Based on domain partitions
   • Generation of coverage report
   • Determine service usage statistics
Illustrative Scenario

- **ChartService**: creates charts from numeric inputs
- **Parameters of operation** *generateChart*:

<table>
<thead>
<tr>
<th>Name</th>
<th>XSD Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>type ∈ { 'line', 'bar', 'pie' }</td>
</tr>
<tr>
<td>values</td>
<td>list of integers</td>
<td>values ∈ {−100, ..., 100}ⁿ, 1 ≤ n ≤ 10</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>pattern &quot;[a-z][a-z0-9]{0,4}&quot;</td>
</tr>
<tr>
<td>config</td>
<td>list of key-value pairs</td>
<td>less than 100 entries</td>
</tr>
</tbody>
</table>

- **Input Message Structure**:

```
generateChart
  type
  value [1..10]
  name
  config [0..99]
    key
    value [1..*]
```
JAXB Schema Generation

- Scenario schema not supported by JAX-WS / JAXB
- 3 new annotations: @Facets, @MinOccurs, @MaxOccurs

```java
@XmlElement
public class GenerateChart {
    public static enum ChartType { line, bar, pie }
    public static class Config {
        public String key;
        public List<String> value;
    }
    @XmlElement(required=true)
    public ChartType type;
    @MinOccurs(1) @MaxOccurs(10)
    @Facets(minInclusive=-100, maxInclusive=100)
    public List<Integer> value;
    @Facets(pattern="[a-z][a-z0-9]{0,4}")
    public String name;
    @MaxOccurs(99)
    public List<Config> config;
}
```
<complexType name="generateChart">
    <sequence>
        <element name="type">
            <simpleType>
                <restriction base="xs:string">
                    <enumeration value="line" />
                    <enumeration value="bar" />
                    <enumeration value="pie" />
                </restriction>
            </simpleType>
        </element>
        <element name="value" minOccurs="1" maxOccurs="10">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-100" />
                    <maxInclusive value="100" />
                </restriction>
            </simpleType>
        </element>
        <element name="name">
            <simpleType>
                <restriction base="string">
                    <pattern value="[a-z][a-z0-9]{0,4}" />
                </restriction>
            </simpleType>
        </element>
        <element name="config" type="Config" maxOccurs="99"/>
    </sequence>
</complexType>
API Coverage Metrics (1)

- **Basic Idea: Domain Partitioning**
  - Split value domain $d$ into subsets $P_d = \{d_1, \ldots, d_n\}$, $d_1 \cup \ldots \cup d_n = d$
  - Match messages against partitions and determine domain subset
  - Coverage = $\#(\text{distinct subdomains of logged messages}) / |P_d|$

- **Two Partitioning Types**

- **Example: Partition Integer Into \{Negative, Zero, Positive\}**
  - $m_1(x) := (x<0)$
  - $m_2(x) := (x==0)$
  - $m_3(x) := (x > 0)$
  - $i(x) = \text{sign}(x)$
Example Catalogue of Domain Partitions

| ID | Name         | Type | $|P_d|$ | $m_1(x)$ | $m_2(x)$ | $m_3(x)$ | $i(x)$ |
|----|--------------|------|------|---------|---------|---------|-------|
| i  | ignore       | MT   | 1    | true    | -       | -       | -     |
| n  | negZeroPos   | MT   | 3    | $x<0$   | $x==0$  | $x>0$   | -     |
| z  | zero         | MT   | 1    | $x==0$  | -       | -       | -     |
| t  | extreme      | MT   | 2    | $x==\text{MIN}$ | $x==\text{MAX}$ | - | - |
| b  | blocksOf10   | MI   | int(abs(MAX -MIN)/10)+1 | - | - | - | (int) x/10 |
| o  | outOfRange   | MT   | 1    | $x<\text{MIN} \mid x>\text{MAX}$ | - | - | - |
| p  | pieChart     | MT   | 1    | $x=='\text{pie}'$ | - | - | - |
| d  | default      |      |      | predefined, based on XSD of the Web service | | | |

Additional predefined membership indicator function $m_0$ embraces elements that belong to no other subset:

$$m_0(x) = \left( \bigwedge_{i \in \{1, \ldots, |P_d| \}} \neg m_i(x) \right)$$
API Coverage Metrics (3)

- Apply Partitioning to Logged Messages
  - Based on the message structure in the XSD
  - User selects **two partitions** for each node in the XSD
    - \( P_v \): partition for the **value** (i.e., text content) of this node
    - \( P_0 \): partition for the number of **occurrences** of this node
  - Message gets mapped to domain memberships of all its nodes

- Some Examples (partition IDs refer back to last slide)

<table>
<thead>
<tr>
<th></th>
<th>( P_v )</th>
<th>( P_0 )</th>
<th>( P_v )</th>
<th>( P_0 )</th>
<th>( P_v )</th>
<th>( P_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateChart</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>i</td>
<td>i</td>
<td>i</td>
</tr>
<tr>
<td>name</td>
<td>d</td>
<td>-</td>
<td>i</td>
<td>-</td>
<td>i</td>
<td>-</td>
</tr>
<tr>
<td>config</td>
<td>-</td>
<td>d</td>
<td>-</td>
<td>i</td>
<td>-</td>
<td>z</td>
</tr>
<tr>
<td>key</td>
<td>d</td>
<td>-</td>
<td>i</td>
<td>-</td>
<td>i</td>
<td>-</td>
</tr>
<tr>
<td>value</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>i</td>
<td>i</td>
<td>i</td>
</tr>
</tbody>
</table>

Possibilities:
- default (as in XSD)
- ignore all: 1
- chart types: 3
- zero-config:
  - pie chart: 2
### 2 Sample Messages

#### Selected Partitions:

<table>
<thead>
<tr>
<th>gen.Chart</th>
<th>type</th>
<th>value</th>
<th>name</th>
<th>config</th>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>n</td>
<td>i</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>i</td>
<td>i</td>
<td>i</td>
<td>-</td>
<td>-</td>
<td>i</td>
</tr>
</tbody>
</table>

#### Partition Hash Vectors:

<table>
<thead>
<tr>
<th>2</th>
<th>7</th>
<th>3</th>
<th>3</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>3</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>null</th>
<th>null</th>
<th>null</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### API Coverage Metrics (4)

- **<generateChart>**
  - **<type>bar</type>**
    - **<value>10</value>**
    - **<value>25</value>**
    - **<value>-13</value>**
    - **<value>36</value>**
    - **<value>0</value>**
    - **<value>-13</value>**
    - **<name>ch1</name>**
    - **<config>**
      - **<key>zoom</key>**
        - **<value>1.2</value>**
      - **<name>ch1</name>**
      - **<config>**
        - **<key>zoom</key>**
          - **<value>1.2</value>**
        - **<key>titles</key>**
          - **<value>x=foo</value>**
        - **<config>**
          - **<value>y=bar</value>**
        - **</config>**
      - **</config>**
    - **</generateChart>**

- **<generateChart>**
  - **<type>pie</type>**
    - **<value>34</value>**
    - **<value>12</value>**
    - **<value>29</value>**
    - **<value>25</value>**
    - **<name>ch2</name>**
    - **<config>**
      - **<key>zoom</key>**
        - **<value>1.5</value>**
      - **<key>titles</key>**
        - **<value>x=foo</value>**
      - **<value>y=bar</value>**
      - **</config>**
    - **</generateChart>**

#### Partition Subset:

- **<generateChart>**
  - **<type>bar</type>**
    - **<value>10</value>**
    - **<value>25</value>**
    - **<value>-13</value>**
    - **<value>36</value>**
    - **<value>0</value>**
    - **<value>-13</value>**
    - **<name>ch1</name>**
    - **<config>**
      - **<key>zoom</key>**
        - **<value>1.2</value>**
      - **<name>ch1</name>**
      - **<config>**
        - **<key>zoom</key>**
          - **<value>1.2</value>**
        - **<key>titles</key>**
          - **<value>x=foo</value>**
        - **<config>**
          - **<value>y=bar</value>**
        - **</config>**
      - **</config>**
    - **</generateChart>**
Coverage Computation (1)

- Data Preparation of Logged Messages
  - Parse the XML tree and store all nodes and relationships
  - Determine the partition memberships of all nodes

- Invoked: Model Elements of Web Services (Messages, Operations, Endpoints, ...) and XML Schema
Coverage Computation (2)

- We need: #(possible invocations) and #(distinct invocations)
- Data Stored in a Relational Database
- Queries Performed Using SQL
  - Goal: select distinct partition hash vectors
  - Number of columns depends on message schema
  - → SQL string generated on demand
- Query String for Our Scenario
  - Each “(…)“ represents a sub-query
  - Computationally intensive (DBMS needs to perform many joins)

```
SELECT distinct (...) as gt, (...) as gv_o, (...) as gv_1, (...) as gv_2, ..., (...) as gv_7, (...) as gn, (...) as gc_o, (...) as gck_1, (...) as gck_2, (...) as gcv_o, (...) as gcv_1, (...) as gcv_2, (...) as gcv_3 from ...
```

<table>
<thead>
<tr>
<th>SQL Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 7 3 3 1 3 2 1 3 1 1 1 1 1 null null null</td>
</tr>
<tr>
<td>3 4 3 3 3 3 null null null 1 2 1 1 3 1 1 1</td>
</tr>
</tbody>
</table>
Implementation (1)

Implementation (2)

Coverage of operation `generateChart`

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>generateChart</code></td>
<td></td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>265716</td>
<td>2</td>
<td>0</td>
<td>0.0008</td>
</tr>
<tr>
<td><code>type</code> chartType</td>
<td>Val.: XSD-based</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>66.6667</td>
</tr>
<tr>
<td>``</td>
<td></td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>88572</td>
<td>2</td>
<td>0</td>
<td>0.0023</td>
</tr>
<tr>
<td><code>value</code> int [1..10] {-100,...,100}</td>
<td>Occ.: XSD-based</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td><code>name</code> string /[a-z][a-z0-9]{1-4}/</td>
<td>Val.: ignore</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td><code>config</code> [0..99]</td>
<td>Occ.: ignore</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td><code>key</code> string</td>
<td>Val.: ignore</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td><code>value</code> string [1..*]</td>
<td>Occ.: ignore</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
</tbody>
</table>

Compute Coverage
Evaluation (1)

- Experimentation Based on Scenario Charting Service
  - Generated 10,000 invocations with randomized test data
  - Cvg. Calculation in real time (< 6 seconds for 10,000 invocations)
  - Larger scenarios require distributed storage and computation

![Graph showing calculation time vs. number of invocations]
## Evaluation (2)

### Wide Range of Possible Metrics
- Starting point: \(i\) (ignore) and \(d\) (default) partitions
- Refine the metric using custom partitions

<table>
<thead>
<tr>
<th>#</th>
<th>Example Metrics</th>
<th>(v(gt))</th>
<th>(o(gv))</th>
<th>(v(gv))</th>
<th>(v(gn))</th>
<th>(o(gc))</th>
<th>(v(gck))</th>
<th>(o(gcv))</th>
<th>(v(gcv))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Usage Counter</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
</tr>
<tr>
<td>2</td>
<td>Extreme Values</td>
<td>(i)</td>
<td>(t)</td>
<td>(t)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
<tr>
<td>3</td>
<td>Chart Types</td>
<td>(d)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
<tr>
<td>4</td>
<td>Blocks of 10</td>
<td>(i)</td>
<td>(i)</td>
<td>(b)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
<tr>
<td>5</td>
<td>Pie Charts</td>
<td>(p)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
<td>(d)</td>
</tr>
<tr>
<td>6</td>
<td>Out of Range</td>
<td>(i)</td>
<td>(i)</td>
<td>(o)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
<tr>
<td>7</td>
<td>Config. Settings</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(d)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
<tr>
<td>8</td>
<td>Config. Keys</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
<td>(d)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
<tr>
<td>9</td>
<td>Values of Default Pie Charts</td>
<td>(p)</td>
<td>(i)</td>
<td>(d)</td>
<td>(i)</td>
<td>(z)</td>
<td>(i)</td>
<td>(i)</td>
<td>(i)</td>
</tr>
</tbody>
</table>
Conclusion

- **API Coverage**
  - Extent to which a service has been used/tested
  - Important measure for testing and for usage statistics

- **Model for API Coverage Metrics**
  - Based on domain partitioning
  - Applied to values and occurrences of XML nodes

- **Important Precondition: Exact Interface Definition**
  - Proposed extension for XSD facets support in JAXB

- **Future Work**
  - Extend the scope of API Coverage to invocation sequences
  - Distributed storage and computation of coverage metrics
    - Tailor-Made Implementation and „Divide and Conquer“
    - Apache Hadoop / HBase / Hive
  - Privacy and trust concerns
Discussion
References (1)

References (2)