Testing Service-Oriented Software

Lu Zhang
Agenda

- Background
- Test-Adequacy Criteria
- Test-Case Generation
- Regression Testing
- Collaborative Testing
- Beyond Services
- Conclusion
Service Concept

- Services are **autonomous, platform-independent** computational elements that can be described, published, discovered, orchestrated and programmed using **standard protocols** to build networks of collaborating applications distributed within and across organizational boundaries.
Background (2)

- Interesting Features of Services
  - Development by composition
  - Provider’s control on services
  - Enterprise application integration
Background (3)

- Testing Service-Oriented Software
  - Testing a single service
    - Testing in development phase
    - Testing for service selection
  - Testing software systems composed of services
Background (4)

- Issues in Testing Software Systems Composed of Services
  - Unavailability of service source code
  - Inability to control the execution of services
  - New languages for integrating services (e.g., BPEL)
Agenda

- Background
- Test-Adequacy Criteria
- Test-Case Generation
- Regression Testing
- Collaborative Testing
- Beyond Services
- Conclusion
Importance of Test-Adequacy Criteria

- Test-adequacy criteria play a central role in software testing
  - An indicator for terminating the testing process
  - A measure for evaluating test suites
  - A measure for evaluating various techniques in software testing
    - Test-case generation
    - Test-case prioritization
    - Test-suite reduction
  - A guidance for test-case generation
Motivation for Criteria Based on Dataflow

Goal of service composition (e.g., BPEL)
- Connection of services
- Passing data between services

Dataflow-based criteria for BPEL
- Existing research on dataflow-based criteria
- Focusing on data passing between services
- Dealing with unavailability of service code

Criteria Based on Dataflow

- Importance of XPath
  - XPath plays a central role to extract contents in XML messages
- XPath Rewriting Graph (XRG)
Modeling a BPEL program

- X-WSBPEL model
  - Control Flow Graph (CFG) of a BPEL
  - XRGs in the control flow graph

- Def and Use
  - In either the CFG or an XRG
Dataflow Criteria

- All Queries
  - Execute all XRGs

- All Query-Du
  - Execute all Def-Use pairs

- All Query-Pu
  - Execute all Predicate-Use pairs (if a variable is used in a predicate, then execute all branches)
Empirical Results (1)

- **Subjects**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Applications</th>
<th>Element</th>
<th>LOC</th>
<th>Query</th>
<th>Query-pu</th>
<th>Query-du</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ATM [4]</td>
<td>94</td>
<td>180</td>
<td>3</td>
<td>12</td>
<td>35</td>
</tr>
<tr>
<td>C</td>
<td>DSLService [27]</td>
<td>50</td>
<td>123</td>
<td>3</td>
<td>11</td>
<td>47</td>
</tr>
<tr>
<td>D</td>
<td>GYMLocker [4]</td>
<td>23</td>
<td>52</td>
<td>2</td>
<td>9</td>
<td>23</td>
</tr>
<tr>
<td>E</td>
<td>LoanApproval [4]</td>
<td>41</td>
<td>102</td>
<td>2</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>F</td>
<td>MarketPlace [4]</td>
<td>31</td>
<td>68</td>
<td>2</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>G</td>
<td>Purchase [4]</td>
<td>41</td>
<td>125</td>
<td>2</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>H</td>
<td>TripHandling [4]</td>
<td>94</td>
<td>170</td>
<td>6</td>
<td>14</td>
<td>33</td>
</tr>
</tbody>
</table>

- **Faults**

<table>
<thead>
<tr>
<th>Category</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-BPEL</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>in-XPath</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>in-WSDL</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>
Empirical Results (2)

- Effectiveness

<table>
<thead>
<tr>
<th>Category</th>
<th>Fault-detecting rates</th>
<th>Box-whisker usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Criterion</td>
<td>Min.</td>
</tr>
<tr>
<td>in-BPEL</td>
<td>Random</td>
<td>0.286</td>
</tr>
<tr>
<td></td>
<td>All-queries</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>All-query-pu</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>All-query-du</td>
<td>0.667</td>
</tr>
<tr>
<td>in-XPath</td>
<td>Random</td>
<td>0.095</td>
</tr>
<tr>
<td></td>
<td>All-queries</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>All-query-pu</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>All-query-du</td>
<td>0.571</td>
</tr>
<tr>
<td>in-WSDL</td>
<td>Random</td>
<td>0.167</td>
</tr>
<tr>
<td></td>
<td>All-queries</td>
<td>0.556</td>
</tr>
<tr>
<td></td>
<td>All-query-pu</td>
<td>0.556</td>
</tr>
<tr>
<td></td>
<td>All-query-du</td>
<td>0.778</td>
</tr>
<tr>
<td>Overall</td>
<td>Random</td>
<td>0.183</td>
</tr>
<tr>
<td></td>
<td>All-queries</td>
<td>0.533</td>
</tr>
<tr>
<td></td>
<td>All-query-pu</td>
<td>0.533</td>
</tr>
<tr>
<td></td>
<td>All-query-du</td>
<td>0.667</td>
</tr>
</tbody>
</table>
Empirical Results (3)

- Test suite size

<table>
<thead>
<tr>
<th>Subject</th>
<th>Random / all-queries</th>
<th>Random / all-query-pu</th>
<th>Random / all-query-du</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atm</td>
<td>2.16</td>
<td>2.13</td>
<td>1.81</td>
</tr>
<tr>
<td>BuyBook</td>
<td>9.84</td>
<td>9.66</td>
<td>8.91</td>
</tr>
<tr>
<td>DSLService</td>
<td>3.61</td>
<td>3.59</td>
<td>2.78</td>
</tr>
<tr>
<td>GYMLocker</td>
<td>2.38</td>
<td>2.37</td>
<td>2.18</td>
</tr>
<tr>
<td>LoanApproval</td>
<td>8.13</td>
<td>8.01</td>
<td>7.97</td>
</tr>
<tr>
<td>MarketPlace</td>
<td>13.40</td>
<td>13.13</td>
<td>12.06</td>
</tr>
<tr>
<td>Purchase</td>
<td>3.31</td>
<td>3.18</td>
<td>2.85</td>
</tr>
<tr>
<td>TripHanding</td>
<td>3.47</td>
<td>3.40</td>
<td>2.48</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>5.13</strong></td>
<td><strong>5.05</strong></td>
<td><strong>4.29</strong></td>
</tr>
</tbody>
</table>
Motivation for a New Criterion

- Unavailability of service source code may make existing test-adequacy criteria inadequate.
- Misunderstanding between service providers and service users may impact the quality of service-centric software system.

```cpp
double f (int n)
{
    int a = WS-abs(n);
    int b = WS-log(a);
    return b;
}
```

Criterion Based on Contracts

- Definition of interface contracts
- Mutation operators for interface contracts
- Detecting killed mutants of interface contracts
- Mutation scores for interface contracts
Definition of Interface Contracts

Contracts ::= {
   PreconditionSection
   PostconditionSection
}  
//排序条件
PreconditionSection ::= <precondition [pre-name]> ContractExpression <\precondition>
//后置条件
PostconditionSection ::= <postcondition [post-name]> ContractExpression <\postcondition>
//合约表达式
ContractExpression ::=  
   Expression  
   | forall type variable-declarator in collection Expression  
   | exists type variable-declarator in collection Expression  
   | Expression implies Expression  
//表达式
Expression ::= numeric_expression | testing_expression | logical_expression  
   | string_expression | casting_expression | literal_expression  
   | null | Request | Response | identifier  
   | ( Expression ) | Expression ( Expression ) | [ Expression ] . Expression  
numeric_expression ::= (- | ++ | -- ) Expression | Expression ( ++ | -- )  
   | Expression ( | += | - = | * | *= | / | /= | % | %= ) Expression  
testing_expression ::= Expression ( > | < | >= | <= | == | != ) Expression  
logical_expression ::= Expression | Expression ( & | ^ | && | || ) Expression  
   | Expression ? Expression : Expression | true | false  
string_expression ::= Expression ( | += ) Expression  
casting_expression ::= ( type ) Expression  
literal_expression ::= integer-literal | real-literal | string-literal | character-literal  
collection ::= { element { , element } }  
element ::= integer-literal | real-literal | string-literal | character-literal | literal_expression
Normal Form of Interface Contracts

- Transforming preconditions and post conditions into the conjunctive normal form

\[(a==b) \rightarrow (c==d \land e==f)\]
\[\Downarrow\]
\[(a!=b) \lor (c==d \land e==f)\]
\[\Downarrow\]
\[(a!=b \lor c==d) \land (a!=b \lor e==f)\]
## Mutation Operators for Interface Contracts

<table>
<thead>
<tr>
<th>Name</th>
<th>Fault Model</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreWk</td>
<td>Precondition Weakening</td>
<td>Delete a precondition item</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Weaken an atomic expression</td>
</tr>
<tr>
<td>PreStr</td>
<td>Precondition Strengthening</td>
<td>Strengthen an atomic expression</td>
</tr>
<tr>
<td>PostWk</td>
<td>Postcondition Weakening</td>
<td>Delete a postcondition item</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Weaken an atomic expression</td>
</tr>
<tr>
<td>PostStr</td>
<td>Postcondition Strengthening</td>
<td>Strengthen an atomic expression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Original Atomic Expression</th>
<th>Mutated Atomic Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strengthening</td>
</tr>
<tr>
<td>P==Q</td>
<td>-------------------------</td>
</tr>
<tr>
<td>P!=Q</td>
<td>P&gt;Q, P&lt;Q</td>
</tr>
<tr>
<td>P&gt;Q</td>
<td>P&gt;Q + Constant</td>
</tr>
<tr>
<td>P&lt;Q</td>
<td>P&lt;Q - Constant</td>
</tr>
<tr>
<td>P&gt;=Q</td>
<td>P&gt;Q, P==Q</td>
</tr>
<tr>
<td>P&lt;=Q</td>
<td>P&lt;Q, P==Q</td>
</tr>
</tbody>
</table>
Detecting Killed Mutants

Algorithm DetectMutants
Input $E$: the execution trace of the test case $t$
$M_{all}$: the set of interface-contract mutants of the components invoked in $E$
Output: $M_{killed}$: the set of killed interface-contract mutants

begin
for each invocation of an interface (denoted as $f_i$) of a component invoked in $E$ do
  for each interface-contract mutant (denoted as $m_j$) in $M_{all}$ do
    if $m_j$ is a mutant produced by mutating the precondition of $f_i$
      then if the result of the mutated precondition differs from that of the original precondition
        then Remove $m_j$ from $M_{all}$; Add $m_j$ into $M_{killed}$
      end if
    else if $m_j$ is a mutant produced by mutating the postcondition of $f_i$
      then if the result of the mutated postcondition differs from that of the original postcondition
        then Remove $m_j$ from $M_{all}$; Add $m_j$ into $M_{killed}$
      end if
  end if
end do
end

• Each test case needs to be executed only once, since contracts do not affect the execution.
**Mutation Score**

- $\text{MS} = \frac{|M_K|}{(|M| - |M_E|)}$
  - MS: the mutation score
  - $M_K$: the set of killed mutants
  - $M$: the whole set of mutants
  - $M_E$: the set of equivalent mutants
Empirical Results (1)

- Test-suite reduction
- Subjects
  - Lines of Code
    - Triangle: 192
    - ATM: >4700
    - Finance: >5500
  - # Seeded Faults
    - Triangle: 5
    - ATM: 12
    - Finance: 9
  - # Atomic Expressions (Manually)
    - Triangle: 30
    - ATM: 10
    - Finance: 13
  - # Atomic Expressions (Daikon)
    - Triangle: 18
    - ATM: 61
    - Finance: 20
- Initial test suites
  - Quantity
    - Triangle: 183
    - ATM: 79
    - Finance: 120
  - M-FER© (%)
    - Triangle: 100.00
    - ATM: 100.00
    - Finance: 100.00
  - C-FER® (%)
    - Triangle: 100.00
    - ATM: 100.00
    - Finance: 75.00
  - S-FER® (%)
    - Triangle: 100.00
    - ATM: 100.00
    - Finance: 77.78

1. M-FER: the FER of the faults seeded in the main program
2. C-FER: the FER of the faults seeded in components
3. S-FER: the FER of the faults seeded in the entire system
### Empirical Results (2)

#### Triangle System

<table>
<thead>
<tr>
<th>Criteria</th>
<th>IM</th>
<th>AI</th>
<th>AE</th>
<th>ACD</th>
<th>ICM_D</th>
<th>ICM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adequacy (%)</td>
<td>100.00</td>
<td>100.00</td>
<td>85.77</td>
<td>75.00</td>
<td>100.00</td>
<td>98.53</td>
</tr>
<tr>
<td>Quantity</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M-FER (%)</td>
<td>0.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>C-FER (%)</td>
<td>50.00</td>
<td>50.00</td>
<td>50.00</td>
<td>50.00</td>
<td>75.00</td>
<td>100.00</td>
</tr>
<tr>
<td>S-FER (%)</td>
<td>40.00</td>
<td>60.00</td>
<td>60.00</td>
<td>60.00</td>
<td>80.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>

#### ATM System

<table>
<thead>
<tr>
<th>Criteria</th>
<th>IM</th>
<th>AI</th>
<th>AE</th>
<th>ACD</th>
<th>ICM_D</th>
<th>ICM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adequacy (%)</td>
<td>91.45</td>
<td>100.00</td>
<td>100.00</td>
<td>85.00</td>
<td>71.11</td>
<td>85.56</td>
</tr>
<tr>
<td>Quantity</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>M-FER (%)</td>
<td>66.67</td>
<td>66.67</td>
<td>66.67</td>
<td>66.67</td>
<td>50.00</td>
<td>66.67</td>
</tr>
<tr>
<td>C-FER (%)</td>
<td>33.33</td>
<td>33.33</td>
<td>50.00</td>
<td>50.00</td>
<td>66.67</td>
<td>66.67</td>
</tr>
<tr>
<td>S-FER (%)</td>
<td>50.00</td>
<td>50.00</td>
<td>58.33</td>
<td>58.33</td>
<td>58.33</td>
<td>66.67</td>
</tr>
</tbody>
</table>

#### Finance System

<table>
<thead>
<tr>
<th>Criteria</th>
<th>IM</th>
<th>AI</th>
<th>AE</th>
<th>ACD</th>
<th>ICM_D</th>
<th>ICM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adequacy (%)</td>
<td>83.87</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>60.00</td>
<td>71.43</td>
</tr>
<tr>
<td>Quantity</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>M-FER (%)</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>C-FER (%)</td>
<td>12.50</td>
<td>37.50</td>
<td>50.00</td>
<td>62.50</td>
<td>37.50</td>
<td>75.00</td>
</tr>
<tr>
<td>S-FER (%)</td>
<td>22.22</td>
<td>44.44</td>
<td>55.56</td>
<td>66.67</td>
<td>44.44</td>
<td>77.78</td>
</tr>
</tbody>
</table>

1 Adequacy: Adequacy stands for mutation scores (MS) in IM and ICM; Adequacy stands for coverage in AI, AE, and ACD (see CIG approaches in Section 2.2 and Section 4.1.3.)
Empirical Results (3)

- Test-Case Prioritization
Agenda

- Background
- Test-Adequacy Criteria
- **Test-Case Generation**
- Regression Testing
- Collaborative Testing
- Beyond Services
- Conclusion
Effective Test Generation for BPEL (1)

- Effective Message Sequences
  - Covering paths in BPEL
  - Considering message correlations

- Building Message Sequence Graph
  - Analyzing message receptions in the BPEL program
    - Sequential: sequence, link
    - Parallel: flow
    - Exclusive: switch, if, loop
  - Labeling correlation sets

---

Message Sequence Graph

Sequential

Exclusive

Sequential

Parallel

...<sequence>

<receive variable=Msg_1>
</receive>

<pick>

<onMessage variable=Msg_2>
</onMessage>

<onMessage>

<onMessage variable=Msg_3>
</onMessage>

<onMessage>

<onMessage variable=Msg_4>
</onMessage>

</pick>

<receive variable=Msg_5>
</receive>

<flow>

<receive variable=Msg_6>
</receive>

<receive variable=Msg_7>
</receive>

</flow>

</sequence>

...
Effective Test Generation for BPEL (2)

- Generating Message Sequences
  - Traversing the MSG

- Generating Parameters
  - Message correlations
  - Data pool
    - Setting up a data pool with values of 0, 1, boundary values etc.
    - Probability of $p$ of values from the data pool and $1-p$ of random values
Empirical Results (1)

Subjects

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Subjects</th>
<th>Source</th>
<th>Loc</th>
<th>STR</th>
<th>COR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>LoanApproval-1</td>
<td>ActiveBPEL</td>
<td>125</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>ATM</td>
<td>ActiveBPEL</td>
<td>180</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>MarketPlace</td>
<td>BPWS4J</td>
<td>68</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D</td>
<td>GymLocker</td>
<td>BPWS4J</td>
<td>52</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>E</td>
<td>LoanApproval-2</td>
<td>BPWS4J</td>
<td>102</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>BPEL1-5</td>
<td>Apache ODE</td>
<td>50</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Seeded Faults

<table>
<thead>
<tr>
<th>Ref.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>WSDL</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>XPath</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>34</td>
</tr>
</tbody>
</table>
Empirical Results (2)

- Ratios of ineffective tests

<table>
<thead>
<tr>
<th>Ref.</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3.00</td>
<td>4.00</td>
<td>4.00</td>
<td>4.50</td>
<td>3.60</td>
<td>3.82</td>
</tr>
<tr>
<td>B</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>C</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.75</td>
<td>0.60</td>
<td>0.67</td>
</tr>
<tr>
<td>D</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>E</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>F</td>
<td>7.00</td>
<td>5.00</td>
<td>4.67</td>
<td>5.00</td>
<td>5.20</td>
<td>5.38</td>
</tr>
</tbody>
</table>

- Fault exposure

<table>
<thead>
<tr>
<th>Ref.</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>90.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>98.00</td>
</tr>
<tr>
<td>B</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>C</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>D</td>
<td>50.00</td>
<td>50.00</td>
<td>75.00</td>
<td>100.00</td>
<td>100.00</td>
<td>75.00</td>
</tr>
<tr>
<td>E</td>
<td>80.00</td>
<td>80.00</td>
<td>80.00</td>
<td>80.00</td>
<td>80.00</td>
<td>80.00</td>
</tr>
<tr>
<td>F</td>
<td>50.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>90.00</td>
</tr>
</tbody>
</table>
Extension: Targeting Contract Mutation (1)

- **Goal:** Generating tests to kill contract mutants
  - **Reachability condition**
    - To guide the execution to reach the mutant
  - **Necessity condition**
    - To distinguish the contract from its mutants

- **Target programs**
  - BPEL programs
Extension: Targeting Contract Mutation (2)

- For Reachability Condition
  - Using the same strategy
- For Necessity Condition
  - Adding special values from contracts to the data pool
Concurrent Test Generation for BPEL (1)

- Intrinsic concurrent but no explicit processes
- Necessary to test concurrency
- Analyzing concurrent paths in BPEL programs

J. Yan, Z. Li, Y. Yuan, W. Sun, and J. Zhang, BPEL4WS Unit Testing: Test Case Generation Using a Concurrent Path Analysis Approach, ISSRE2006
Concurrent Test Generation for BPEL (2)

Main steps
- Constructing the eXtended Control flow Graph
- Enumerating sequential
- Combining sequential paths
- Solving constraints
Example XCFG

- E0: receive
- E1: receive-to-approver link
- E2: InvokeAssessor
- E3: InvokeApprover
- E4: assessor-to-approver link
- E5: Assign
- E6: Assign-to-reply link
- E7: approver-to-reply link
- E8: fault thrown
- E9: reply
- E10: fault handler
Concurrent Test Generation for BPEL (3)

- Generating concurrent paths
  - combining sequential paths
  - Evaluating combinability
  - Setting an upper bound
Empirical Results

- Loan processing
- 57 combinations of sequential paths
- 9 valid concurrent paths
Agenda

- Background
- Test-Adequacy Criteria
- Test-Case Generation
- Regression Testing
- Collaborative Testing
- Beyond Services
- Conclusion
Test-Case Prioritization

- Given: T, a test suite; PT, the set of permutations of all subsets of T; f, a function from PT to the real numbers;
- Problem: Find $T' \in PT$, such that $(\forall T'')(T'' \in PT)(T'' \neq T') [f(T') \geq f(T'')]$.

Basic Strategies
- The total strategy
- The additional strategy
Dataflow-Based Test-Case Prioritization

- **Workflow branches**
  - Total (M1), Additional (M2)

- **Workflow branches + XRG branches**
  - Total (M3), Additional (M4)
  - Total-Refine (M5), Additional-Refine (M6)

- **Workflow branches + XRG branches + WSDL elements**
  - Total (M7), Additional (M8)
  - Total-Refine (M9), Additional-Refine (M10)

### Empirical Results (I)

- **Subjects**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Applications</th>
<th>Modified Versions</th>
<th>Element</th>
<th>LOC</th>
<th>XPath</th>
<th>XRG Branch</th>
<th>WSDL Element</th>
<th>Used Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>atm [1]</td>
<td>8</td>
<td>94</td>
<td>180</td>
<td>3</td>
<td>12</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>buybook [16]</td>
<td>7</td>
<td>153</td>
<td>532</td>
<td>3</td>
<td>16</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>dslservice [24]</td>
<td>8</td>
<td>50</td>
<td>123</td>
<td>3</td>
<td>16</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>gymlocker [1]</td>
<td>7</td>
<td>23</td>
<td>52</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>E</td>
<td>loanapproval[1]</td>
<td>8</td>
<td>41</td>
<td>102</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>F</td>
<td>marketplace [1]</td>
<td>6</td>
<td>31</td>
<td>68</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>purchase [1]</td>
<td>7</td>
<td>41</td>
<td>125</td>
<td>2</td>
<td>8</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>H</td>
<td>triphandling [1]</td>
<td>9</td>
<td>94</td>
<td>170</td>
<td>6</td>
<td>36</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>60</td>
<td>527</td>
<td>1352</td>
<td>23</td>
<td>114</td>
<td>106</td>
<td>43</td>
</tr>
</tbody>
</table>
Empirical Results (2)

- Effectiveness

Images show box plots for different levels with various metrics.
Impact of Quota Constraints on Services

- Constituent services need to be intensively invoked during regression testing a service-centric software system.
- Quota constraints on constituent services may postpone the testing process.

<table>
<thead>
<tr>
<th>Web Services</th>
<th>Request Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Historical Pricing Web Service</td>
<td>60000 requests per user per month</td>
</tr>
<tr>
<td>eBay Shopping Web Service</td>
<td>5000 requests per IP per day</td>
</tr>
<tr>
<td>Yahoo! Web Search Web Services</td>
<td>5000 queries per IP per day per API¹</td>
</tr>
<tr>
<td>Google SOAP Search API</td>
<td>1000 queries per license key per day</td>
</tr>
</tbody>
</table>

Test Prioritization for Quota Constraints

- Time-Slot Partition
- Test-Case Selection and Prioritization for Each Time Slot
- Information Refreshing
Time-Slot Partition

- Three services: \{((x,2 \text{ days}), (y, 3 \text{ days}), (z, 5 \text{ days}))\}
- Time-Slot Partition: \([0, 2], [2, 3], [3, 4], [4, 5], [5, 6], [6, 8]\)…

Algorithm \text{TS\_Partition}

Input: \text{WSQ} = \{(q_1, t_1), (q_2, t_2), ..., (q_n, t_n)\}
the number of anticipated time slots \(s\)
Output: \(s\) time slots \(t_{s_1}, t_{s_2}, ..., t_{s_s}\)

Begin
\[ t_{s_1} = [0, e_{p_1}], \text{ where } e_{p_1} = \min(t_1, t_2, ..., t_n) \]
\(p = 1\)
do
for each \(t_j\) in \text{WSQ} do
\[ \text{cand}\_t_j = ((e_{p_1}/t_j) + 1) * t_j \]
end
\[ e_{p+1} = \min(\text{cand}\_t_1, \text{cand}\_t_2, ..., \text{cand}\_t_n) \]
\[ t_{s_{p+1}} = [e_p, e_{p+1}] \]
\(p++\)
until \(p \geq s\)
End
Test-Case Selection and Prioritization (1)

- Quota-Constrained Total Strategy
  - Selection
    - Decision Variables
      \[ x_i = \begin{cases} 
      1, & \text{if } tc_i \text{ is selected to execute in } ts_p, \\
      0, & \text{otherwise} 
      \end{cases} \]
    - Constraints
      \[ \sum_{i=1}^{m} tw_{ij} * x_i \leq awsq_{pj} \]
  - Object Function
    \[ \max \sum_{i=1}^{m} \left( \sum_{k=1}^{l} tr_{ik} \right) * x_i \]
- Prioritization
  - Traditional total strategy
Test-Case Selection and Prioritization (2)

- Quota-Constrained Additional Strategy
  - Selection
    - Decision Variables
      
      \[ x_i = \begin{cases} 
      1, & \text{if } tc_i \text{ is selected to execute in } ts_p, \\
      0, & \text{otherwise} 
      \end{cases} \]

    - Constraints
      \[
      \sum_{i=1}^{m} tw_{ij} * x_i \leq awsq_{pj} \\
      \sum_{i=1}^{m} tr_{ik} * x_i \geq y_k
      \]
  - Object Function
    \[
    \max \sum_{k=1}^{l} y_k
    \]
  - Prioritization
    - Traditional additional strategy
Information Refreshing

- Removing test cases selected in the previous slot
- Calculating available quotas for the next slot
- For the additional strategy, removing testing requirements covered in the previous slot
Illustration

(t_{8}, t_{13}, t_{2}, t_{4}, t_{1}, t_{6}, t_{10}, t_{5}) (t_{11}, t_{3}, t_{16}, t_{14}, t_{9}) (t_{12}, t_{7}, t_{14})

(t_{8}, t_{13}, t_{2}, t_{4}, t_{1}, t_{6}, t_{10}, t_{5}, t_{11}, t_{3}, t_{16}, t_{14}, t_{9}, t_{12}, t_{7}, t_{14})
Empirical Results (1)

- **Subject**
  - A travel agent system using 12 Web services with 17 methods

- **Test Suites**
  - A test suite generated by JUnitFactory
  - A randomly generated test suite

- **Quota Constraints**
  - Five levels: the lower the tighter
Empirical Results (2)
Agenda

- Background
- Test-Adequacy Criteria
- Test-Case Generation
- Regression Testing
- Collaborative Testing
- Beyond Services
- Conclusion
Why we need collaborative testing

- Difficulty in generating test cases
  - Unavailability of service source code
- Difficulty in test execution
  - No control of service execution
- Difficulty in evolution control
  - Independent evolution of services
- Third-party testing
  - Certifying the quality of services
Collaborative Testing Frameworks

• Tsai et al.’s enhanced UDDI server
• Zhu’s twin services
• Bai et al.’s contract-based collaborative testing framework
• Bartolini et al.’s whitening framework
• Tsai et al.’s CV&V framework
Enhanced UDDI Server (1)

W.T. Tsai, R. Paul, Z. Cao, L. Yu, and A. Saimi, Verification of Web Services using an Enhanced UDDI Server, WORDS2003
Enhanced UDDI Server (2)

- Publically ensuring service quality via testing
  - Check-in and Check-out

- Test scripts
  - Method testing scripts
  - Object testing scripts
  - Integration testing scripts
  - System integration testing scripts
  - Domain testing scripts
Enhanced UDDI Server (3)

- **Advantages**
  - Providing certification for quality assurance of services

- **Disadvantages**
  - Centralization
  - Lack of automation support
Twin Services (1)

Twin Services (2)

- Each functional service is accompanied with a testing service
- Ontology-based reasoning
  - Distinguishing testing services from functional services
  - Automating service testing
Twin Services (3)

- **Advantages**
  - Control over test execution
  - Decentralization
  - Flexibility

- **Disadvantages**
  - Difficulty in synchronization between functional services and testing services
Contract-Based Collaborative Testing Framework (1)

Contract-Based Collaborative Testing Framework (2)

- Adding brokers between services and testers
  - Decentralizing the enhanced UDDI server

- Using contracts for collaboration
  - Testing service contracts
  - Test collaboration contracts
Contract-Based Collaborative Testing Framework (3)

- Advantages
  - Actually a combination of the enhanced UDDI server and the twin services

- Disadvantages
  - Complexity
  - Cost of contracts
Whitening Framework (1)

1: Instruments services with TCOV probes
2: Invokes GDS services
3: Logs coverage information
4: Gets test reports

C. Bartolini, A. Bertolino, S. Elbaum, and E. Marchetti, Whitening SOA Testing, ESEC/FSE2009
Whitening Framework (2)

- Instrumenting service code to record coverage information
- Providing coverage services for testers to check coverage information
Whitening Framework (3)

- **Advantages**
  - Targeting at a specific problem in service testing

- **Disadvantages**
  - Limited usage
  - Insufficient for testing composite services
CV&V Framework (1)

CV&V Framework (2)

- Facilitating the testing of composite services
- Targeting at the oracle problem
CV&V Framework (3)

- Advantages
  - Easy to automate

- Disadvantages
  - Requiring a large number of homogeneous services
  - Computationally expensive
Agenda

- Background
- Test-Adequacy Criteria
- Test-Case Generation
- Regression Testing
- Collaborative Testing
- Beyond Services
- Conclusion
Beyond Services

- Software libraries -> Software components -> Services
- Uncontrollable ingredients in software -> Testing in semi-open environment
  - Test generation
  - Test execution
  - Regression testing
  - Test management
Conclusion

- Research topics of testing raised by the emergence of Web services and SOA
- Impact of characteristics of service-centric software systems on testing
  - Unavailability of service source code
  - Impact of service quotas
  - Service composition languages
- Requiring more investigation on testing service-centric software