A Scalable Approach to User-session-based Testing of Web Applications through Concept Analysis

Sreedevi Sampath, University of Delaware
Valentin Mihaylov, Drexel University
Amie Souter, Drexel University
Lori Pollock, University of Delaware

Automated Software Engineering Conference (ASE), 2004
September 22, 2004
Web-based Applications

• Web pages on server, clients access pages
  – Example: e-commerce applications

• Characteristics
  – Short time to market
  – Extensive use
  – High reliability, continuous availability
  – Changing user profiles

• User-session data
  – Client URL requests and name-value pairs (data)
Challenges of User-session-based Testing

• Leverage user-session data as test cases applicable in beta/maintenance testing phases
  – Elbaum et al. (2003)

• Manage/replay large set of user sessions

• Existing test suite reduction non-scalable
  – Harrold et al. (1993)
Contributions

• View user sessions as use-cases
• Apply concept analysis for test suite reduction
• Perform incremental test suite update
• Automate the testing framework [ICSM 04]
• Evaluate cost effectiveness
  – Test suite size
  – Program coverage
  – Fault detection

Scalaable user-session-based testing
Concept Analysis

Mathematical technique for clustering objects that have common discrete attributes

Input – theory
Set of objects, $O$
Set of attributes, $A$
Binary relation, $R \subseteq O \times A$
Example Concept Analysis – Input

**Objects:** \{airplane, boat, rollerskates, unicycle\}

**Attributes:** \{wheel(s), over80mph, passengers, wear, engine\}

**Binary Relation Table:**

<table>
<thead>
<tr>
<th>objects</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wheel(s)</td>
</tr>
<tr>
<td>airplane</td>
<td>●</td>
</tr>
<tr>
<td>boat</td>
<td>●</td>
</tr>
<tr>
<td>rollerskates</td>
<td>●</td>
</tr>
<tr>
<td>unicycle</td>
<td>●</td>
</tr>
</tbody>
</table>
Concept Analysis - Output

• Identifies concepts for a given tuple \((O, A, R)\)

• A \textit{concept} is a tuple

\[ t = (O_i, A_j) \]

• Concepts form a partial order defined as

\((O_1, A_1) \leq (O_2, A_2) \iff O_1 \subseteq O_2\)

\((O_1, A_1) \leq (O_2, A_2) \iff A_1 \supseteq A_2\)
Example Concept Analysis Output

<table>
<thead>
<tr>
<th>objects</th>
<th>wheels</th>
<th>over80mph</th>
<th>passengers</th>
<th>wear</th>
<th>engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>boat</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>rollerskates</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
</tr>
<tr>
<td>unicycle</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

({airplane, boat, rollerskates, unicycle}, null)

({unicycle, rollerskates, airplane}, {wheels})

({boat, airplane}, {over80mph, passengers, engine})

({rollerskates}, {wheels, wear})

({airplane}, {wheels, over80mph, passengers, engine})

(null, {wheels, over80mph, passengers, wear, engine})
Set of objects
- User sessions, \( s \), denoted by IP addresses

Set of attributes
- URLs, \( u \)

Binary relation table
- A pair \( (s, u) \) is in the relation table if \( s \) requests \( u \)
### RELATION TABLE

<table>
<thead>
<tr>
<th>Objects</th>
<th>GD</th>
<th>GR</th>
<th>GL</th>
<th>PL</th>
<th>GS</th>
<th>GB</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>us1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>us2</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>us3</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>us4</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>us5</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>us6</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\{\text{us1, us2, us3, us4, us5, us6}\}, \{\text{GD,GR,GL}\} \\
\{\text{us2, us3, us4, us6}\}, \{\text{GD,GR,GL,GS}\} \\
\{\text{us3}\}, \{\text{GD,GR,GL,PL,GS}\} \\
\{\text{us4}\}, \{\text{GD,GR,GL,GS,GB}\} \\
\{\text{us2}\}, \{\text{GD,GR,GL,GS,GM}\} \\
\{\text{us6}\}, \{\text{GD,GR,GL,PL,GS,GB}\} \\
\text{null}, \{\text{GD,GR,GL,PL,GS,GB,GM}\}
\end{align*}
\]

### CONCEPT LATTICE
Test Suite Reduction

• Exploit lattice’s hierarchical use-case clustering [WODA 04]

• Heuristic for test suite reduction
  – Smallest set of user sessions that will cover all URLs of the application executed by original suite

\[
\begin{align*}
\text{({us1, us2, us3, us4, us5, us6}, \{GD, GR, GL\})} \\
\text{(us2, us3, us4, us6), \{GD, GR, GL, GS\})} \\
\text{(us3), \{GD, GR, GL, PL, GS\})} & \quad \text{(us4), \{GD, GR, GL, GS, GB\})} \\
\text{(us2), \{GD, GR, GL, GS, GM\})} & \quad \text{(us6), \{GD, GR, GL, PL, GS, GB\})} \\
\text{(null, \{GD, GR, GL, PL, GS, GB, GM\})}
\end{align*}
\]
Batch Test Suite Update

- Initial user-session data set
- Apply Concept Analysis
- Concept Lattice
- Reduced Test Suite
- Incremental Concept Analysis
- Updated Lattice
- Updated Reduced Suite
- Additional User Sessions
Incremental Test Suite Update

• Utilize Godin et al.’s incremental algorithm
  – Create new nodes/edges
  – Modify existing nodes/edges

• Key insight: Scalability
  Existing internal nodes do not sink to bottom
  – Test cases not maintained for internal nodes
Incremental Test Suite Update

RELATION TABLE

<table>
<thead>
<tr>
<th>objects</th>
<th>GD</th>
<th>GR</th>
<th>GL</th>
<th>PL</th>
<th>GS</th>
<th>GB</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>us1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>us2</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>us3</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>us4</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>us5</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>us6</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>us7</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

attributes (URLs)

SPARSE UPDATED LATTICE

GD GR GL

us1 us5

GS

GM PL GB

us2 us3 us4 us6

us7
Web Testing Framework

Fault Detection Framework

- Fault Detection Report
- Faults

Access Log

Coverage Analysis Tool

- original
- instrumented
- Coverage Report
- Access Log
- New Access Log

Application Code

Web server

Client

Access Log Parser

Relation Table

Concept Analysis Tool

Lattice

Tester

Incremental Test Suite Updater

Test Suite Reducer

Reduced Test Suite

Replay Tool
Web Testing Framework

Access Log
Access Log Parser
Relation Table
Concept Analysis Tool
Lattice

Coverage Analysis Tool
Tester
Incremental Test Suite Updater
Test Suite Reducer

Fault Detection Report
Fault Detection Framework
Faults

original instrumented
Coverage Report
Access Log
New Access Log
Reduced Test Suite

Application Code
Web server
Client

u
Experimental Evaluation

• Evaluate test suite reduction achieved through concept analysis
  – Test suite size
  – Replay and oracle time
• Cost-effectiveness of incremental vs. batch
• Program coverage analysis of reduced suite
• Fault detection capability of reduced suite
Study Setup

• Bookstore application
  – 9748 lines of code
  – 385 methods
  – 11 classes
• JSP front-end, MySQL backend
• Resin web server
• 123 user sessions
• 40 faults
Test Suite Reduction

Methodology

Initial user-session data set

Apply Concept Analysis

Concept Lattice

Reduced Test Suite

Metrics

• Test suite size
• Replay time
• Oracle time
## Test Suite Reduction Results

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Original Suite</th>
<th>Reduced Suite</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test suite size</td>
<td>123</td>
<td>15</td>
<td>87.8%</td>
</tr>
<tr>
<td>Replay time</td>
<td>16m56s</td>
<td>4m22s</td>
<td>74.2%</td>
</tr>
<tr>
<td>Oracle time</td>
<td>25m30s</td>
<td>5m17s</td>
<td>79.3%</td>
</tr>
</tbody>
</table>

- The large percent reduction in test suite size results in reduction in replay and oracle time.
- Considerable savings in replay and oracle when compared to time for suite reduction, 19s [ICSM 04]
Incremental versus Batch

Methodology

Initial user-session data set
100 user sessions

Apply Concept Analysis

Concept Lattice

Reduced Test Suite

Incremental Concept Analysis

Updated Lattice

Updated Reduced Suite

Metric

Space costs

Additional User Sessions
23 sessions
Incremental versus Batch Results

<table>
<thead>
<tr>
<th>Metric</th>
<th>Original Suite</th>
<th>Reduced Suite</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>4.3MB</td>
<td>1MB</td>
<td>76.7%</td>
</tr>
</tbody>
</table>

- **Scalability**: Incremental test suite update saves space by not maintaining original suite of sessions
- Perform incremental update overnight with day’s collection of user sessions to produce updated suite
Program Coverage

Methodology
• Use *Clover* for coverage analysis
• Restore database state before replay
• Use *wget* for replaying user sessions
• Pass cookies and post-data information

Metrics
• Statement and method coverage
Program Coverage Results

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Original Suite</th>
<th>Reduced Suite</th>
<th>Preserved Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement Coverage</td>
<td>60.3%</td>
<td>58%</td>
<td>96.2%</td>
</tr>
<tr>
<td>Method Coverage</td>
<td>53.2%</td>
<td>53.2%</td>
<td>100%</td>
</tr>
</tbody>
</table>

- Reduced suite preserves program coverage obtained from original suite

- Low loss of program coverage due to
  - Heuristic that covers all URLs of the application
  - Reduction with low loss of use case representation [WODA 04]
Fault Detection Study

Methodology

• Manually seed 40 faults in application
• Replay user sessions through
  – Correct version of application to generate expected result
  – Faulty versions of application to generate actual result
• \textit{Diff} the expected and actual results

Metric

• Number of faults detected
Fault Detection Study Results

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Original Suite</th>
<th>Reduced Suite</th>
<th>Preserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faults Detected</td>
<td>87.5%</td>
<td>70%</td>
<td>80%</td>
</tr>
</tbody>
</table>

- Reduced suite maintains fault detection capability
- Investigating additional test suite reduction heuristics
Related Work

- **Concept Analysis in Software Engineering**
  - Recovering components, Eisenbarth et al. (2003)
  - Debugging temporal specifications, Ammons et al. (2003)

- **Reducing Test Suites**
  - Harrold et al. (1993)
  - Offutt et al. (1995)

- **Web Testing**
  - Link and form testers
  - Liu et al. (2000)
  - Ricca and Tonella (2001)
  - Di Lucca et al. (2002)
  - Elbaum et al. (2003)
Conclusions

• Test suite reduction by concept analysis
  – Achieves large reduction in test suite size
  – Saves oracle and replay time

• Incremental test suite update presents scalable approach to user-session-based testing

• Reduced test suite preserves program coverage and fault detection capability
Future Work

- More significant empirical evaluation
- Extend incremental concept analysis to handle program evolution
- Alternate heuristics for test suite reduction
Relation Table and Concept Lattice

<table>
<thead>
<tr>
<th></th>
<th>index</th>
<th>login</th>
<th>reg</th>
<th>books</th>
<th>myinfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>3</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>4</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

\[
(\{45,4,3,25\}, \{\text{myinfo}\})
\]

\[
(\{25\}, \{\text{books, myinfo}\})
\]

\[
(\{45,4,3\}, \{\text{login, myinfo}\})
\]

\[
(\{45, 4\}, \{\text{index, login, myinfo}\})
\]

\[
(\{45,3\}, \{\text{login, myinfo, reg}\})
\]

\[
(\emptyset, \{\text{books, index, login, myinfo, reg}\})
\]
(\{45,4,3,25\}, \{myinfo\})

(\{25\}, \{books\})

(\{4\}, \{login\})

(\{3\}, \{reg\})

(\{4\}, \{index, login\})

(\{45\}, \{index\})

(\emptyset, \{books, index, login, myinfo, reg\})