A Study of Usage-Based Navigation Models and Generated Abstract Test Cases for Web Applications
Sara Sprenkle, Lori Pollock*, Lucy Simko

**World Runs on Web Applications**

- Google
- Amazon
- Yahoo!
- Myspace
- Facebook
- CNN
- eBay
- Travelocity
- YouTube
- Wikipedia
- Slashdot

**Today’s Focus**

How do we automatically generate test cases that accurately emulate users and reveal faults at low cost?

**What are Web Applications?**

- Dynamically generated web pages
  - Based on user input, server state, databases, etc.
- Used daily by businesses, consumers, governments

**One TC Generation Approach**

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Web Application</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>User access</td>
<td>User access</td>
<td>User access</td>
</tr>
<tr>
<td>Beta Web Application (v.0.9)</td>
<td>Beta Web Application (v.0.9)</td>
<td>Beta Web Application (v.0.9)</td>
</tr>
<tr>
<td>User access</td>
<td>User access</td>
<td>User access</td>
</tr>
<tr>
<td>Test v.1.0 with collected data</td>
<td>Test v.1.0 with collected data</td>
<td>Test v.1.0 with collected data</td>
</tr>
</tbody>
</table>

**User Session:**

Series of requests from a user to the web application server

- home search?author=meyer
- view?id=w7gx12fca
- purchase?id=w7gx12fca
- ...
One TC Generation Approach

Benefits:
- cheap collect, generate
- represents what users do
- effective at revealing faults
  [Elbaum, ICSE03; Sprenkle, ASE 2005]

Limitations:
- requires storing all accesses
- highly redundant test cases
- only tests what users do

Web Application

Test Case

Output

User-session-based testing

Data Web Application (v.0.9)

Deployment

Record accesses

User accesses

Test v.1.0 with collected data

Web Application (v.1.0)

Navigating Applications

- Developer may implement restrictions
- Browser navigation
  - Back, forward buttons
  - Location bar
  - Multiple windows
- In a 2004 study, 17% of user sessions included an "infeasible" navigation

Example Simplified Intra-session Navigation Model

Given the previous request, what is the probability of next request being X?

Issues in Generating Test Cases for Web Applications

- Need to represent
  - Application navigation, e.g., sequence of requested pages
  - Data entered, e.g., through forms

- Goals
  - Expose faults (i.e., bugs)
  - Low generation and execution cost

State of the Art: Navigation Models
- Wang et al., ICSM 2009
  - Spider application from a start page
  - Traverse links, try out forms
  - Input values into the application’s forms using a combinatorial approach
- Tonella and Ricca, JSME 2004
  - Spider application from a start page
  - Input values from equivalence classes into forms
  - Augment navigation model with usage information, adding usage-based probabilities to the edges

Augmented Tonella and Ricca Model

- Given the previous request, what is the probability of next request being X?
State of the Art: Navigation Models

- Wang et al., ICSM 2009
  - Spider application from a start page
  - Input values into the application's forms using a combinatorial approach
- Tonella and Ricca, 2004
  - Spider application from a start page
  - Input values from equivalence classes into forms
  - Augment navigation model with usage information, adding usage-based probabilities to the edges

Limitations
- Not completely automated
- Require form values to be known beforehand

Sant et al., Model, n=2

Given the previous request, what is the probability of next request being X?

- Given the previous request, what is the probability of next request being X?

Summary: Usage-based Navigation Models

- Problem:
  - Web application navigation can't be restricted
  - Need to test both valid and invalid sequences
- Challenges
  - Can't test all possible sequences → must prioritize
  - Need “good” navigation sequences → effective test cases
- Approach: Test what users’ access—valid and invalid
  - Allows prioritizing what users do
  - Drawback: model is not complete unless users access all parts of the application
  - Goal is not completeness but to focus on actual usage

Test Case Generation Process

- Data model specification
- User session parameter analyzer
- Intra-session navigation analyzer
- Test-case generator
- Data Model
- Test-case suite

Abstract Test Case:
- Sequence of pages that is accessed; independent of parameter values

Sant et al., 2005

- Statistical navigation model generated from user sessions
- n-gram Markov model of the user sessions’ requests
  - n is one more than the number of previous requests used to predict the next request
  - 1-gram: randomly select next request
  - 3-gram: select next request based on previous 2 requests
- Several “start” states since a user’s session may start from any page

**Open Question: Factors in Constructing Navigation Model**

- How should requests be represented?
- How much history should be used in the n-gram Markov model to predict the next request?
- How should transitions between user requests be approximated?
- How should edges in navigation model be labeled?
- How many user sessions should be used to build the model?
- How does model size vary with different configurations for building the model?

**Open Question: Representing User Requests**

- How should requests be represented?
- How much history should be used in the n-gram Markov model to predict the next request?
- How should transitions between user requests be approximated?
- How should edges in navigation model be labeled?
- How many user sessions should be used to build the model?
- How does model size vary with different configurations for building the model?

**Open Question: Amount of History (n)**

- In an n-gram model, n is one more than the # of previous requests used to predict the next request.
- Intuition: as n increases
  - navigation model better represents users’ navigation
  - size of model increases because possibilities for n−1 sequences increases
- Previous results: Sant et al.’s experiments on one small application showed no benefit to using n > 1 with respect to code coverage

**Open Question: Representing User Requests**

- An HTTP request consists of
  - request type, typically GET or POST
  - resource, the ‘R’ in ‘URL’
  - optional parameter name-value pairs

![HTTP Request Example](GET\ petsearch\ ?animal=cat&location=90210)

**Open Question: Required Number of User Sessions**

- How many user sessions are needed to generate an accurate model?
- How does the model change as more user sessions are incorporated into the model?
- Impact: When can a tester stop collecting user sessions?
Open Questions: Generated Abstract Test Cases

- How representative are the abstract test cases of the original user sessions?
- Do the abstract test cases enable testing of usage not found in the user sessions?

Subjects

- 5 publicly deployed web applications written in Java using servlets and JSPs
- 9 sets of subject user sessions

<table>
<thead>
<tr>
<th>Application</th>
<th>LOC</th>
<th>User Session Set</th>
<th>User Sessions</th>
<th>Requests</th>
<th>% Lines Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masplas</td>
<td>609</td>
<td>169</td>
<td>1107</td>
<td>89%</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>5279</td>
<td>125</td>
<td>3164</td>
<td>61%</td>
<td></td>
</tr>
<tr>
<td>CPM</td>
<td>7430</td>
<td>58</td>
<td>1326</td>
<td>47%</td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>10718</td>
<td>203</td>
<td>2183</td>
<td>66%</td>
<td></td>
</tr>
<tr>
<td>DSpace</td>
<td>30847</td>
<td>105</td>
<td>1528</td>
<td>50%</td>
<td></td>
</tr>
</tbody>
</table>

Answering Question: How should requests be represented?

- Methodology: Generated navigation models using 3 representations of user requests:
  - requestType+resource (RR)
  - requestType+resource+parameter names (RRN)
  - requestType+resource+parameter names + parameter values (RRNV)
- Measures: # of states and edges in navigation model

Results: How should requests be represented?

- Larger growth from 2-RR→2-RRN means more combinations of parameter names with the same resources
- Larger growth from 2-RRN→2-RRNV means many different values
- Not shown: as n increases, smaller growth in general

Implications: How should requests be represented?

- Separate model of user behavior into 2 models:
  1. usage-based navigation model using RRN to generate abstract test cases
  2. develop a separate data model for parameter values
- But using RRNV may not require too many additional resources when using a larger n and does not require data model

Growth in 2-gram Model with Various Request Representations
Answering Question:
How many user sessions should be used to build the model?

- Methodology: Generated navigation models, incrementally adding user sessions in order of decreasing # of requests
  - Estimate of best case
    - Larger user sessions more likely contain requests not seen before
  - Order does not affect the final resulting model
- Metrics: # of states and edges in navigation model

Implications: How many user sessions should be used to build the model?

- Stop gathering USs after growth declines to desired threshold
- However, tester may use as many USs as resources permit

Results: How many user sessions should be used to build the model?

- Growth rate eventually tapers off
  - Not surprising because user sessions are redundant; applications and user sessions are finite

Answering Question:
How well do the abstract test cases represent the user sessions?

- Methodology: Generated 500 non-duplicate abstract test cases for each navigation model
  - Varied n from 1 to 10
  - For n > 1, weighted random walk of navigation model
    - 1-gram model: 15 requests/test case
  - 1-gram model: 15 requests/test case
    - Based on average lengths of collected user sessions
    - Generated fixed number instead of using a stopping criteria to aggregate results without introducing bias towards a user session set
- Measure: % of user sessions’ RRN sequences of various lengths that are represented by the abstract test case suite

Results: How well do the abstract test cases represent the user sessions?

- Achieves high coverage without explicit coverage goals

% of RRN Sequences Represented in Logic’s Abstract Test Cases

<table>
<thead>
<tr>
<th>RRN Sequence Length</th>
<th>% of Sequences Represented</th>
</tr>
</thead>
</table>
Answering Question:
Do the abstract test cases enable testing of usage not found in the user sessions?

- Methodology: Generated 500 non-duplicate abstract test cases for each navigation model
  - Varied n from 1 to 10
  - For n > 1, weighted random walk of navigation model
  - 1-gram model: 15 requests/test case
    - Based on average lengths of collected user sessions
  - Generated fixed number instead of using a stopping criteria to aggregate results without introducing bias towards a user session set
- Measure: # of new, unique RRN sequences of various lengths that the abstract test cases explore

Results: Do the abstract test cases enable testing of usage not found in the user sessions?

- As n ↑, # of new sequences of all lengths ↑, with a few exceptions
- Generated ATCs represent a significant # of new navigation paths of various lengths
  - Maximum # of new sequences: on the order of thousands

Conclusions:

- Use request_type+resource+parameter names to represent user requests
  - Balances scalability and representation requirements
  - May include parameter values if using a larger n
- Can tune amount of history used (i.e., n) based on testing and representation goals
- Can construct a navigation model that generates abstract test cases representative of observed user behavior as well as new behavior with only a small number of user sessions
  - Add more user sessions for better transition probabilities
- Redundancy of user sessions prevents exponential model growth with a large number of user sessions

Future Work:

- Use request_type+resource+parameter names to represent user requests
- Can tune amount of history used (i.e., n) based on testing and representation goals
- Can construct a navigation model that generates abstract test cases representative of observed user behavior as well as new behavior with only a small number of user sessions
  - Add more user sessions for better transition probabilities
- Redundancy of user sessions prevents exponential model growth with a large number of user sessions