Employing User Models for Testing

Punam Agrawal

Why User Models?

- Exhaustive testing is impossible
- Select/Generate test cases intelligently
  - Many ways of doing this
    - Code coverage
    - Find bugs before they are found by users
- Base test cases on user models
  - Test cases will represent actual users

Types of Users

- There are many types of users
  - Based on experience with the software
    - Expert User
    - …….
    - Novice User
  - Other types of Users
    - Malicious users
    - Based on geography, Educational background, Domain….

Outline

- Technique for modeling Novice Users
- Technique for modeling Web Users
- My Work

Toward Automatic Generation of Novice User Test Scripts

David J. Kasik
Harry G. George

Outline of the paper

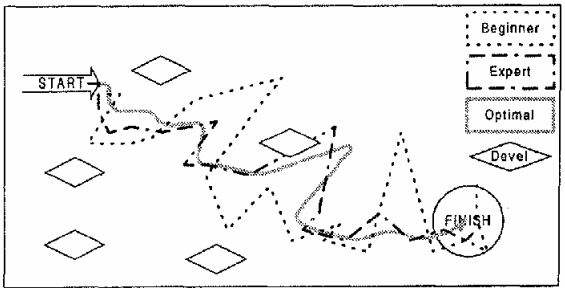
- Motivation
- Challenges
- The technique
- Conclusion/ future work

Motivation

- Applications are accessible to a wide range of users.
- Novice users often exercise application in a different way, that the tester didn't anticipate
- Tester follow path of an expert
- As a result software works fine for predicted user pattern but unstable when given to novice users
- For Novice like testing we launch beta versions of the software
  - Involves many people

What is a Novice User?

- Learning while using
- Different from random
- Different from expert



Challenges

- How to mimic the novice user
  - How to Create a model of the space of all possible interactions.
  - How to traverse this space

The Technique

- Genetic Algorithms
- Key Idea
  - Choose some task to be performed
  - Generate a sequence using an expert to perform the task
  - Use genetic algorithms to derive a “novice sequence” from the expert sequence

Genetic Algorithms

- Used primarily for optimization problems
- Problem solution is represented using genes
  - A gene is a vector
  - Ex:- Shortest path problem, Gene is a vector representing places visited on the path.

Genetic Algorithms Contd.

- Start with random pool of genes
- Associate a fitness value with each gene
  - Computed using a fitness function
    - Length of the path for the previous example
- Choose the top best fit genes (from the pool)
- Generate the next generation of genes using
  - Crossover operator
  - Mutation operator
- Stop when we get the optimal solution

The Technique Contd

- Given a bunch of task
- Generate the expert path
- In each path insert DEVIATE at random points
- Start processing each path using genetic algorithm technique.

The technique Contd

- Traverse each path
- At deviate, select some random event using cross-over operator.
- Use penalty function to get back to the expert path.
- The result will be Gene which will mimic novice users.

Conclusion

- helps identify application failures earlier than beta test/Production
- Genetic algorithm produces novice input events to test an application in an unexpected, but not purely random, way
- It should be combined with automated test tools and/or expert test scripts

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

Sreedevi Sampath

Valetin Mahaylov, Amie Souter

Lori Pollock

Outline of the paper

- Motivation
- Challenges
- The technique
- Conclusion/ future work

Motivation

- One promising approach to testing the functionality of web applications are replaying the collected user-session data.
- Faults detection capability increases with the number of captured user session
  - Impractical test preparation and execution time
- Key approach
  - Clustering user session for test suite reduction

Challenges

- How to reduce number of user-session data that represent usage / operational profile of the application
- Resulting coverage by the reduced test suite should be identical to the original user session suite

The Technique

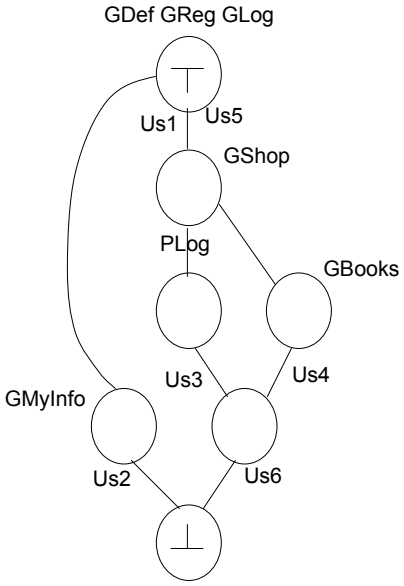
- Concept Analysis
  - Clustering user session for test suite reduction
- Key Idea
  - Collect some user session data
  - Derive the lattice using the concept analysis
  - Incrementally add more user profile
    - Saves the overhead for storing and processing complete set at once
  - Modify the lattice diagram to get the incremental reduced test suite update

Concept Analysis

- Generate Lattices from concept analysis for test suite reduction

	GDef	GReg	GLog	PLog	GShop	GBooks	GMyInfo
Us1	X	X	X				
Us2	X	X	X		X		X
Us3	X	X	X	X	X		
Us4	X	X	X		X	X	
Us5	X	X	X				
Us6	X	X	X		X	X	

Concept Lattice for test suite reduction



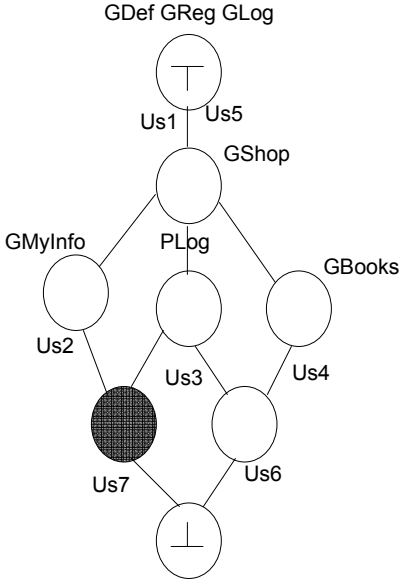
Reduction

- Generate Lattices from concept analysis for Incremental reduction

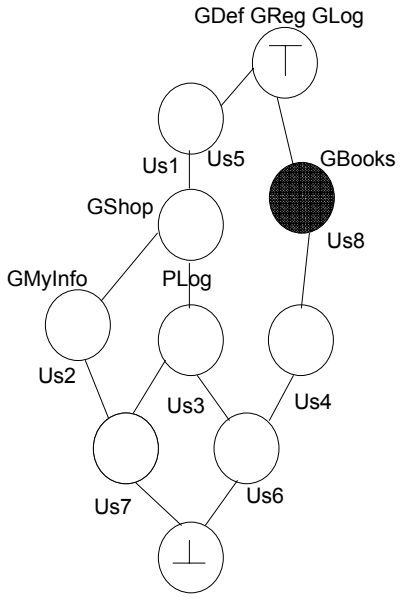
Ex:-

	GDef	GReg	GLog	PLog	GShop	GBooks	GMyInfo
Us1	X	X	X				
Us2	X	X	X		X		X
Us3	X	X	X	X	X		
Us4	X	X	X		X	X	
Us5	X	X	X				
Us6	X	X	X		X	X	
Us7	X	X	X	X	X		X
Us8	X		X			X	

Include Us7 & in the lattice diagram



Include Us8 & in the lattice diagram



Experiment

Objective:

- What test case reduction can be achieved through concept analysis for user session testing?
- How does reducing the test suite affect replay time?
- What is the cost-effectiveness of incremental versus batch concept analysis for reduced test suite update?
- Does concept analysis reduce the test suite while maintaining good coverage?
- How does the reduced test suite fare against the original test suite in terms of fault detection capabilities.

How Do we proceed?

- Collection of User-Session data
- Generate the reduced test cases
  - Using Concept Analysis
- Generate the coverage report
- Continue the Incremental update of reduced test suite

Result

Metrics	Original Suite		Reduced Suite		Reduction/ Loss(%)
	Abs	%	Abs	%	
Test Suite Size	123	-	15	-	87.8%
Replay Time	16m56s	-	4m22s	-	74.2%
Oracle Time	25m30s	-	5m17s	-	79.3%
Statement Coverage	5878	60.3	5654	58	3.8%
Function Coverage	205	53.2	205	53.2	0%
Faults Detected	35	87.5	28	70	20%

Conclusion

- Test Suite Reduction
- Program Coverage
- Fault Detection Capability
- Replay Time

Metrics	Original Suite		Reduced Suite		Reduction/Loss(%)
	Abs	%	Abs	%	
Test Suite Size	123	-	15	-	87.8%
Replay Time	16m56s	-	4m22s	-	74.2%
Oracle Time	25m30s	-	5m17s	-	79.3%
Statement Coverage	5878	60.3	5654	58	3.8%
Function Coverage	205	53.2	205	53.2	0%
Faults Detected	35	87.5	28	70	20%

My Work

- Trying to model test case using an event flow graph

Method

- Generate event flow graph for the application.
- Insert some information in the edges using the user profile
- Generate test cases based on these information