Model Based Software Test Plan Automation

Ravi Gorthi, Ph D
Test Automation Research Lab
Software Engineering and Technology Labs
Infosys Technologies Limited, Bangalore, India
email: ravi_gorthi@infosys.com
Topics

• An overview on Test Case and Regression Test Case Creation and Maintenance
  – Test Plan vs Test Execution
  – Creation and maintenance of test scenarios, cases, data and scripts
  – What are regression test cases? Creation and maintenance of regression test cases

• The relevance of this area to the Total-Cost-of-Ownership (TCO) of medium to large s/w applications
  – Typical number of test scenarios and cases in a medium / large s/w application
  – Efforts and costs involved in the creation and maintenance of test cases and regression test cases
Topics

• The state-of-the-art research and practices of this field
  – Model based (UML Activity Diagram, State Diagram based) manual and automated creation and maintenance of test and regression test cases

• Some interesting work done by the speaker and his team in this area for the last three years
  – The concept of ‘Unit of Behavior’
  – Techniques for automatic generation of test cases and regression test cases from UML Activity Diagrams
  – Results of field trials of these techniques

Q&A
Background

• Software Testing
  – Functional and non-functional testing
  – Planning phase and Execution phase

• Planning phase of functional testing: creation and maintenance of,
  – Test scenarios: correspond to UML use-cases
  – Test cases: correspond to ‘input, expected output’ patterns
  – Test data: correspond to actual input and expected output values
  – Test scripts: correspond to code that the execution phase teams can use

• Execution phase of functional testing:
  – Creation and maintenance of required test execution environments
  – Execution of required test scenarios (include test cases, data and scripts)
What are Test Scenarios, Cases and Data?

• Test Scenarios
  – Testing needs of a logical unit of SRS (e.g. use-case / a logical path in a use-case)

• Test Cases
  – Test requirement descriptions related to each unit of stimuli-response
  – Typically described in terms of <Inputs to software system, A set of conditions, Outputs from software system>
    • E.g. <Input: user-id and password; Condition: incorrect password; Output: error message-1>

• Test Data (corresponding to each test case)
  – Specifications of test inputs, conditions and expected output
    • E.g. <Inputs: ravi_gorthi and abc2$password; Condition: abc2$password is an incorrect password; Output: “Invalid Password - enter password again”>
Background

• Regression Testing
  – Arises out of changes made to a software application
  – Functional and non-functional testing
  – Planning phase and Execution phase

• Planning phase of regression testing: creation and maintenance of,
  – Find changes (additions, modifications and deletions) to functional test scenarios, cases, data and scripts
  – Effect the required changes

• Execution phase of functional testing:
  – Find out the changes required to the test execution environments
  – Execution of required test scenarios (include test cases, data and scripts)
Relevance of Test Planning

• NIST report 2002
  – Software Errors Cost U.S. Economy $59.5 Billion Annually
  – More than a third of these costs ($22.2 b) could be eliminated by an improved testing infrastructure
  – An undetected defect, post software deployment, costs $14K to fix

• Medium to large s/w applications are observed to have anywhere between 5K to 100K test cases

• Typically consumes 30 to 50% of total testing efforts (testing typically consumes 40 to 60% of total software TCO)

• Every 10% improvement in productivity and quality of testing can lead to saving of millions of dollars
Current state of affairs

• Planning phase of functional testing:
  – Functional and regression test planning is manual

• Execution phase of functional and regression testing:
  – Tools exist that aid creation, maintenance and execution of test scenarios leading to productivity and/or quality improvements
Model Transformations in Test Plan Automation
Unified Modeling Language

- UML is a popular modeling language widely used by software engineering professionals, especially suited to engineer MIS applications

- A semi-formal language
  - Structured but NOT a language to unambiguously specify the expected behavior of a software application
  - Offers methodologies and models to perform analysis and design of software applications
Model Based Approach to Test Plan Automation

• Generation of
  – system test, regression test and integration test

  • scenarios, cases, data and scripts

from

– Analysis models
  • UML Use-Case Activity Diagrams, State Diagrams, Communication, Collaboration Diagrams

– Design Models
  • Class Diagrams, Sequence Diagrams
Test Case Generation

• From UML Use-Case Activity Diagrams (UCAD)
  
  – UCAD: Directed Cyclic Graph
  
  – Automatic Generation of Test Scenarios
    • One can automatically generate test scenarios through a DFS with restrictions on traversal of cyclic paths
  
  – Automatic Generation of Test Cases
    • One can automatically generate test cases by slicing each scenario into tuples of <Inputs, Processing, Conditions, Expected Output>
Sample UML Use Case Activity Diagram (UCAD)
Sample UML Use Case Activity Diagram (UCAD)
Test Case Generation from UCADs: Methodology

1. Develop one or more use-case diagrams from requirements.
2. Develop one or more use-case activity diagrams based on a predefined set of rules.
3. Automatically check consistency of the use-case activity diagrams.
4. Validate the consistent use-case activity diagrams with one or more users.
5. Generate one or more functional test cases from the validated use-case activity diagrams.
Consider the following use case from 'Automatic Teller Machine (ATM) System': With-draw Cash using a debit-card from ATM

- System displays the msg: “Enter the debit card number (Swipe the card)”
- User inputs the debit-card number (swipes the card)
- System validates the card (assume the card number is valid)
- System displays the msg: “Enter the PIN”
- User enters the PIN
- System Validates the PIN (assume the PIN is valid)
- System displays the menu of choices and the msg: “Choose from the Menu”
- User selects the Menu option: “With-draw Cash from SB Account”
- System displays the msg: “Enter the amount”
- User enters the amount
- System validates the amount (assume the amount is correct)
- System displays the msg: “Take the amount; Thank you!”
Test Case Generation from UCADs: Case Study

- Add the following exceptions to the ATM Use Case

  • Debit-card swiped by the user is invalid

  • PIN entered by the user is invalid (the 1st time error)

  • PIN entered by the user is invalid (the 3rd consecutive time error)

  • Amount entered by the user is invalid (amount > current balance)

  • Amount entered by the user is invalid (amount > daily upper-limit)
### Scenario #1

**Precondition(s):**
- none

<table>
<thead>
<tr>
<th># User Input</th>
<th>Conditions</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter Product ID and tabs out</td>
<td>Does the product ID exist in the product portfolio? = “NO”</td>
<td>1. No fields are populated</td>
</tr>
<tr>
<td>2. Find product in portfolio link with “Product ID” field being empty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Pop-up listing of all products is displayed and select desired product and click on a link.</td>
<td>UserChoice? = n/a</td>
<td>1. Populate shipment details page with information on the product.</td>
</tr>
<tr>
<td>4. Decide whether or not to add the product to the portfolio</td>
<td>Does the user want to add the product to the portfolio? = “YES”</td>
<td>1. &quot;Product details are added to the portfolio&quot;</td>
</tr>
</tbody>
</table>

### Scenario #2

**Precondition(s):**
- none

<table>
<thead>
<tr>
<th># User Input</th>
<th>Conditions</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter Product ID and tabs out</td>
<td>Does the product ID exist in the product portfolio? = “NO”</td>
<td>1. No fields are populated</td>
</tr>
<tr>
<td>2. Find product in portfolio link with</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Results of TCG in a few Real-world Projects

<table>
<thead>
<tr>
<th>PRODUCTIVITY FIGURES FROM FIELD TRIALS</th>
<th>Healthcare-Claims</th>
<th>Internet Banking</th>
<th>Retail – Ecommerce</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of test cases generated</td>
<td>1082</td>
<td>1772</td>
<td>23366</td>
</tr>
<tr>
<td>No of task flows involved</td>
<td>32</td>
<td>48</td>
<td>586</td>
</tr>
<tr>
<td>Effort estimated for manual procedure (Person Days)</td>
<td>23</td>
<td>38</td>
<td>465.9</td>
</tr>
<tr>
<td>Effort spent using our tool (Person Days)</td>
<td>13.5</td>
<td>17.6</td>
<td>132</td>
</tr>
<tr>
<td>% Effort Saving</td>
<td>41%</td>
<td>54%</td>
<td>72%</td>
</tr>
</tbody>
</table>
Test Case Generation

• From UML Use-Case Activity Diagrams (UCAD)

  – Peter Zielczynski [8] and Jim Heumann [9] offer the basics on the analysis of use-cases to manually generate test cases; good to start with.

  – Lionel Briand and Yvan Labiche [12], Clementine et al [10], Chen et al [11], Linzhang et al [13] and Chen et al [14] offer the next level of detail on semi-auto / auto generation of test cases from use-cases.

Test Case Generation

• From System Requirements Specifications (SRS)
  – SRS is typically semi-structured
  – Structured SRS
    • First break a given SRS into a set of Use-Cases
    • Express each Use-Case as ordered sequence of tuples of <Inputs, Conditions, Expected Output>
    • One can automatically generate Use-Case Activity Diagrams (UCADs) from the Structured SRS
  – Automatic Generation of Test Scenarios
    • One can automatically generate test scenarios through a DFS with restrictions on traversal of cyclic paths
  – Automatic Generation of Test Cases
    • One can automatically generate test cases by slicing each scenario into tuples of <Inputs, Conditions, Expected Output>
# Structuring SRS: Use Case Template

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Pre-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC-01 ...</td>
<td>The use case has ...</td>
<td></td>
</tr>
<tr>
<td>Interaction Steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>1</td>
<td>The user ...</td>
</tr>
<tr>
<td>Process (optional)</td>
<td>2</td>
<td>The system ...</td>
</tr>
<tr>
<td>Decision (optional)</td>
<td>3</td>
<td>If (Condition is True) Goto 4 Else Goto 5</td>
</tr>
<tr>
<td>Decision (optional)</td>
<td>5</td>
<td>If (...) Goto ... Else Stop</td>
</tr>
<tr>
<td>Output</td>
<td>4</td>
<td>The system Displays ...</td>
</tr>
<tr>
<td>Post-Condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Non-Functional Requirements (Optional):**

<Keywords such as frequency, performance, priority, fault tolerance>: Requirement Description
ATM Example: Structuring SRS

ATM TA Tool

System Name: [System Name]
User: [User]

Use Case Name: [Use Case Name]
Description: [Description]
Requirement: [Requirement]
Input: [Input]
Output: [Output]
Decision: [Decision]
Action: [Action]

Dependency: Include Validate Pin use case
Precondition: ATM is idle, displaying a Welcome message.
Description:
1. Include Validate Pin use case.
2. Customer selects Withdrawal, enters the amount, and selects the account number.
3. System checks whether customer has enough funds in the account and whether the daily limit will not be exceeded.
4. If all checks are successful, system authorises dispensing of cash.
5. System dispenses the cash amount.
6. System prints a receipt showing transaction number, transaction type, amount withdrawn, and account balance.
7. System ejects card.
8. System displays Welcome message.

Alternatives:
If the system determines that the account number is invalid, it displays an error message and ejects the card.
If the system determines that there are insufficient funds in the customer's account, it displays
Automatic Generation of Use Case Activity Diagrams (UCADs) from Structured SRS
### Test Case Generation from Structured SRS

#### Scenario #1

**Precondition(s):**
- none

<table>
<thead>
<tr>
<th># User Input</th>
<th>Conditions</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter Product ID and tabs out</td>
<td>Does the product ID exist in the product portfolio? = &quot;NO&quot;</td>
<td>1. No fields are populated</td>
</tr>
<tr>
<td>2. Find product in portfolio link with &quot;Product ID&quot; field being empty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Pop-up listing of all products displayed and select desired product</td>
<td>UserChoice? = n/a</td>
<td>1. Populate shipment details page with information on the product.</td>
</tr>
<tr>
<td>and click on a link.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Decide whether or not to add the product to the portfolio</td>
<td>Does the user want to add the product to the portfolio? = &quot;YES&quot;</td>
<td>1. &quot;Product details are added to the portfolio&quot;</td>
</tr>
</tbody>
</table>

#### Scenario #2

**Precondition(s):**
- none

<table>
<thead>
<tr>
<th># User Input</th>
<th>Conditions</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter Product ID and tabs out</td>
<td>Does the product ID exist in the product portfolio? = &quot;NO&quot;</td>
<td>1. No fields are populated</td>
</tr>
<tr>
<td>2. Find product in portfolio link with &quot;Product ID&quot; field being empty</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Test Case Generation

- From UML State Diagrams / Charts
  - Philippe and Pascale [20], Supaporn and Wanchai [21] offer the basics on automatically generating of test cases from UML state diagrams
  - Stefania Gnesi et al [22] and Valdivino Santiago et al [23] offer the next level of details on the automated generation of test cases from UML state diagrams
Some important issues related to regression testing:

- When a software application is modified due to ‘change requests’ received from business users, the application need to be tested to ensure that the modified system meets all the functional (and non-functional) requirements.
- Typically, medium to large software applications are observed to involve, 
  - a few tens of to a few hundreds of ‘o-o classes’
  - A few thousands of to a few tens of thousands of ‘methods’
- In general, each cycle of software maintenance involves changes to around 15 to 30% of methods.
- There will be many classes and methods which remain unchanged across maintenance cycles.
- Do we need to test a modified application, *completely*? Answer is **NO!**
Model Based Regression Test Case Selection

- **Input**
  - Two *consecutive versions* of UML Use-Case Activity Diagrams (UCADs)
    - UCAD-v1 and UCAD-v2

- **Processing**
  - *Identify* changed nodes in UCAD-v2 with respect to UCAD-v1
    - Modified, newly added, deleted or shifted nodes
  - *Select* regression test suite
    - Paths in the UCAD-v2 that are affected due to the above types of node changes
      - *Paths* in UCAD-v2 that are the same as those in UCAD-v1 need *NOT* be tested

- **Output**
  - *Selected regression test suite* to test UCAD-v2
Model Based Regression Test Case Selection

- Original Requirements specifications
  - Structure Activity Diagram (Version 1)
- Changed requirements specifications
  - Structure Activity Diagram (Version 2)

Selection of regression test suite
- Retrieve paths and nodes
- Bucketing of different nodes
- Compare and Identify affected paths
- Select regression tests

Recommended regression test suite
Model Based Regression Test Case Selection

• **Identify** changes to the nodes of UCAD-v2 with respect to UCAD-v1

  – Nodes in a given UCAD are structured as
    • Node type `<Sequence-number.Version-number>`
    • E.g UA `<1.0>`; SP `<2.3>`; CO `<3.1>`; SO `<1.4>`

  – All changes to the nodes of UCAD-v2 can be categorized as
    • Modifications to an existing node
    • Deletion of an existing node
    • Shifting of an existing node
    • Addition of a new node

• Generate *Selected* Regression Test Suite for UCAD-v2
  – Paths that contain one or more *changed* nodes
Model Based Regression Test Case Selection

Examples: UA: <1.0>, SO: <2.2>, CO: <3.0>, FI: <1.0>

Structure of Node Version Number

- **UA**: User Action node
- **SO**: System Output node
- **CO**: Condition node
- **FI**: Final node

**Node Version** — The number after the period
- The version of the node itself — will be incremented after each modification.

**Node Sequence** — The number before the period
- The sequence of the node according to its type
  - Newly added node will always receive the next available node sequence
  - Deleted node sequence will never be used again
Model Based Regression Test Case Selection

• Some nodes are more critical than the others.
• Example: Verification of PIN number and account balance VS. Verification of correct display of welcome message

• Assign criticality to nodes
  – [H] – High criticality
  – [M] – Medium criticality
  – [L] – Low criticality (Default setting)
Original Influx Taskflow Diagram

Newly Added nodes

Modified nodes

New Influx Taskflow Diagram
Deleted nodes

Newly Added
nodes

Modified
nodes

Original Influx Taskflow Diagram

New Influx Taskflow Diagram
# Model Based Regression Test Case Selection

<table>
<thead>
<tr>
<th>Description</th>
<th>New ID</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enters the operation to be performed</td>
<td>N/A</td>
<td>N/A</td>
<td>Deleted</td>
</tr>
<tr>
<td>Presses button to Withdraw cash</td>
<td>N/A</td>
<td>N/A</td>
<td>Deleted</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Steps</th>
<th>Input</th>
<th>Conditions</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter the card</td>
<td>Card inserted properly? = Yes</td>
<td>1. Display a message to enter the PIN</td>
</tr>
<tr>
<td>2</td>
<td>Enter the PIN</td>
<td>Cash withdrawal? = yes</td>
<td>1. Display a message to continue with the transaction</td>
</tr>
<tr>
<td>3</td>
<td>Enter the amount</td>
<td>Cash available? = Yes</td>
<td>1. Display a message to collect the cash</td>
</tr>
</tbody>
</table>
## Experimental Results from a few Real-World Projects

<table>
<thead>
<tr>
<th></th>
<th>Healthcare-Claims</th>
<th>Internet Banking</th>
<th>Retail Order Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of UCAD paths involved</td>
<td>320</td>
<td>730</td>
<td>4300</td>
</tr>
<tr>
<td>No of regression test cases selected</td>
<td>1610</td>
<td>4780</td>
<td>20410</td>
</tr>
<tr>
<td>Effort estimated for manual procedure (Person Days)</td>
<td>23</td>
<td>78</td>
<td>466</td>
</tr>
<tr>
<td>Effort spent using our tool (Person Days)</td>
<td>13.5</td>
<td>37.6</td>
<td>186</td>
</tr>
<tr>
<td><strong>Productivity gain</strong></td>
<td><strong>41%</strong></td>
<td><strong>52%</strong></td>
<td><strong>60%</strong></td>
</tr>
</tbody>
</table>
Selective References


Selective References

Selective References


Selective References


