System testing from UML diagrams

Eliane Martins
Institute of Computer – State University of Campinas (Unicamp)
eliane@ic.unicamp.br

52nd. Ifip WG 10.4 Meeting – Uphall, Scotland  Jul 2, 2007
Harpia

- **Goal:**
  - Development of web applications for the Tax Department

- **Goal of the testing group:**
  - Model-based system testing
  - Performability testing
    - Performance
    - Availability
  - Fault injection
Harpia

- **Goal:**
  - Development of web applications for the Tax Department

- **Goal of the testing group:**
  - **Model-based system testing**
  - Performability testing
    - Performance
    - Availability
  - Fault injection
Motivation

- In Brazil: system developers commonly use UML notations for specification and design
- Scenarios are popular as part of requirements specification
  - Scenarios describe how users and system interact to provide some service
  - Many scenarios are needed to describe a system
- How to generate test cases based on these scenarios?
Since many scenarios are needed to describe a system → how to combine these scenarios?

Approaches:
- Sequence Diagram or Message Sequence Chart
- Finite State Machines
- Combination: Activity Diagrams and Sequence Diagrams

Our approach: hierarchy of Activity Diagrams
Modelling the system

- Business description
- Scenarios
- Use cases
  - Use case flows
  - Activity diagrams
    - Intra-use case flow
    - Inter-use case flow
# Use case description

<table>
<thead>
<tr>
<th>Use case name</th>
<th>Insert Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The goal of this use case is to provide a solution for the creation of new products in the database.</td>
</tr>
<tr>
<td>Actors</td>
<td>Manager</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>The user must be logged in and selected the “Insertion” option.</td>
</tr>
<tr>
<td>Invariants</td>
<td>None</td>
</tr>
</tbody>
</table>

### Main Flow

- **P1. Check user identification**
  - If user is not valid then throw E1.
- **P2. Get product information**
  - The system shows a form to be filled with the information concerning the product.
- **P3. Validate product**
  - Includes use case Validate Product.
  - If the product is not valid execute alternate flow A1.
- **P4. Confirm product insertion**
  - The system exhibits a message asking the user to confirm the product insertion into the database.
  - If the user confirms, then go to P7 else the system cancels the insertion.
- **P7. Insert new product in the database**
  - If problems with the insertion in the database then generate exception E2 else the use case terminates successfully.

### Alternate Flow

- **A1. Mensagens de Alerta do Produto**
  - The system exhibits error messages, showing the fields in the form that are wrong. Go to P2 to allow the user to correct the errors.

### Exception Flow

- **E1. User not authenticated**
  - a. An exception corresponding to the error is thrown.
- **E2. Exception generated by the DB**
  - a. The exception is captured by a exception handler, that must guarantee the data is inserted.

### Post-conditions

- The product is inserted into the database or the database remains unchanged.
Activity diagram – Inter use cases

Scenarios

Product Management

Identify User

Insert product

Update product

Remove product

Search product

Control flow dependencies among use cases
(Manager view)

initial node

guard

[option=SEARCH]

[else]

decision node

InvalidUser

«user»

Select option

«use case»

Identify user

«use case»

Search product

«use case»

Update product

«use case»

Remove product

«use case»

Insert product

final node

Business Description

«include»

«include»

«include»
Activity diagrams – intra use cases

Activity: Insert Product

- «system» Check user authentication
  - InvalidUser

- «system» Exhibit product form
  - «user» Enter product information

- «system» Get product information

- «use case» Validate product
  - [invalid]
  - [else]

- «user» Confirm insertion
  - [no]
  - [yes]

- «system» Save product in the database

- «pre-condition» User authentified and Insertion selected
- «post-condition» Product inserted or database unchanged

- «exception handler» HandleDBException
Test Case Generation

1. Build a Flow Graph
2. Select paths from the graph
3. Generate script (XML)
4. Build executable model
5. Generate test data
6. Complete script

UML Activity Diagram
The test model
Path-oriented test selection

Problems:
- How to select paths?
  - Control flow based criteria (e.g. all edges, all nodes)
- How to select realizable paths?
  - Various call-return in a path
  - *Realizable* path: each call edge is matched with its return edge
  - Context sensitive search for a path
- How to deal with loops to avoid infinite number of paths?
  - Limit number of repetitions
  - Loop testing
The tools

Use case specifications
Use case flows
Business rules
Activity diagrams

UML Modeling Tool
XMI
XMI Graph Builder
ICFG
Context Sensitive Path Selector
Paths
Test Case Constructor
Traceability Matrix
Test sequence

Antares Test Generation Subsystem

Legend
Tool
Document
Tool Module
Some results

<table>
<thead>
<tr>
<th>Test case design</th>
<th>Test case execution</th>
</tr>
</thead>
<tbody>
<tr>
<td># UC</td>
<td>27</td>
</tr>
<tr>
<td># ICFG edges</td>
<td>441</td>
</tr>
<tr>
<td># ICFG nodes</td>
<td>530</td>
</tr>
<tr>
<td># test cases (TC)</td>
<td>142</td>
</tr>
</tbody>
</table>
Conclusions

- **On-going work**
  - Regression testing selection based on Activity Diagram
  - Testing process still in use → more measurements are being performed
  - Systematic creation of surrogates (or proxies) for exception handling testing (to be obtained from test cases)

- **Future work:**
  - Implementation of other test criteria
  - Model validation (e.g. simulation)
  - Data flow and test data generation
  - Considering concurrency
    - various actors using the system
Thanks!