Viewpoint-based Test Requirement Analysis Modeling and Test Architectural Design

6th World Congress for Software Quality
London, UK
2014/7/3(Thu)
Nishi, Yasuharu
The University of Electro-Communications, Japan

© NISHI, Yasuharu
Profile – Dr. NISHI, Yasuharu

Assistant professor:
the University of Electro-Communications, Japan
(also providing consultancy service to industry on testing and TQM)

President:
Association of Software Test Engineering, Japan (ASTER)

President:
Japan Software Testing Qualifications Board (JSTQB)

National delegate:
ISO/IEC JTC1/SC7/WG26 Software testing

Founder:
Japan Symposium on Software Testing (JaSST)

Founder:
Testing Engineers’ Forum (Japanese community on software testing)

Vice chair:
SQiP/Software Quality Committee of JUSE (promoting organization of TQM)
(SQiP has published the book of “SQuBOK: Software Quality Body of Knowledge”
and is operating engineer certification on software quality)

Research interest:
Software testing, software quality/TQM, embedded software engineering,
software process improvement, software project management, system safety
Software Test Engineering Process

• As software has got huge and complicated, test cases (= test suite) also get huge and complicated
  - such as
    » a test project with over 100,000 test cases
    » over 10 test levels
    » various test types such as load, configuration and security
  - You have to develop huge and complicated test suite systematically

• But technologies on test planning or test strategy are just immature
  - Engineering work and management work for test development are confused

• It is necessary to define software test engineering process to develop huge and complicated test suite systematically
Independent RA is necessary for testing

- Independent requirement analysis for testing is necessary
  - Requirement analysis for software isn’t usually exact or detail enough
  - Of course test requirement analysis depends on SUT and its dev. Process
- Requirement analysis for testing should be more important
  - Test “planning” is just management word and NOT engineering word

**VSTeP: Viewpoint-based Test Process**

Test Planning → Test Design → Test Implementation (Scripting)

Part of typical test process

Test Requirement Analysis → Test Architecture Design → Test Detail Design → Test Implementation

Test Management (including planning for management)
VSTeP

- VSTeP (Viewpoint-based Software Test Engineering Process) is a generic test engineering process model focusing on test viewpoint
  - You can stress upper phase of test engineering process such as test requirement analysis and test architecture design which tend to be negligent
  - VSTeP drives you to good test suite, good review for test design, accumulation of knowledge and experience on testing
  - Reuse and improvement will be easy because you can do reverse-engineering of your past (unorganized) test suite
  - NGT (Notation for Generic Testing) is a made-in-Japan notation for Test Requirement Analysis and Test Architecture Design
    » Modeling skill like object-oriented design is essentially necessary

VSTeP: Viewpoint-based Test Process
Detail phase of VSTeP

• TRA: Test Requirement Analysis
  - To make a test requirement model
    » To extract, organize and understand test requirement
    » To create a test requirement model which consists of test viewpoints,
      i.e. to create a viewpoint diagram

• TAD: Test Architecture Design
  - To make a test architecture model
    » To re-organize test viewpoints into test containers
      as test types, levels and cycles for making test smooth
    » To assemble test viewpoints into test frames which is template for TDD

• TDD: Test Detail Design
  - To make test cases
    » To set values in detail into test frames or test viewpoints

• TI: Test Implementation
  - To make test scripts
    » To add detail information necessary to execution to test cases
    » To combine simple test scripts into a compound test script
      for making execution efficient
Example of part of viewpoint diagram drawn for TRA

E-mail client  \textit{Test Item / SUT}

- GUI
- Functions
- Environment
- Data

- Platform
  - OS
    - Kind of OS
    - Version of OS
    - Internet Explorer
- Network
  - Hardware
What is test viewpoint: abstract test case

- Test cases have test values
  - ex) parameter: Kind of OS, values: Win7, WinXP, Win2000
  - Test parameters are also called test conditions and test values are also called test coverage items
  - Test cases consists of test values

- Viewpoints are abstract test cases
  - Bottom viewpoints mean test parameters
  - Viewpoints don’t express any test values or test cases
  - Viewpoints can have hierarchical structure like classification trees or class diagrams
  - Viewpoints can be extracted from test conditions, test items, and quality characteristics such as load, configuration, and performance
  - Ideally viewpoints should indicate an INTENTION of a test case
    » Viewpoint diagram can be a repository of intentions of TCs
Various test viewpoints

- Test viewpoint is a point where test engineers focus an attention for grasping a big picture of test design
  - Test viewpoint is abstraction and source of test cases
- Types of test viewpoints depend on organizations and/or test engineers
  - What should be exhausted:
    » Specs, functions, data etc.
    » Test conditions
  - Characteristics which should be achieved
    » Quality characteristics, non functional requirements etc.
  - Parts of test items
    » Funcs, Subsystems, modules etc.
  - Bugs
    » Errors and failures, bug patterns, weak points of test items etc.
  - Customer usage
    » Business, lifestyle etc.
  - Other parts of systems than software
    » Hardware units, hardware failures etc.
  - Test types
    » Load test, configuration test etc.
  - Test levels
    » Component test, system test etc.
  - Lists and/or diagrams developed until software testing
    » Use cases, State transition diagrams etc.
Why “viewpoint”? 

- The word “viewpoint” is independent of roles

- Requirement!
- Parameter
- Test purpose
- Test condition
- Test Environment

© NISHI, Yasuharu
Types of Hierarchical relationship

- Test viewpoints have two fundamental relationships
  - Hierarchy relationships and Interaction relationships
  - Types of relationships can be expressed as "<<stereotype>>"
- Hierarchical relationships can bear several meanings
  - is-a relationship: inheritance
  - has-a relationship: possession
  - There may be other hierarchical relationships
    » object-attribute and cause-effect is example
Interactive relationships of viewpoints

- Viewpoints can relate each other with interactive relationships
  - Non-hierarchical relationships are necessary: Interactive relationships
  - They can also bear several meanings: combination, sequential etc.
  - Lines without arrowhead represent “combinatorial relationships”
  - Arrows with an open head represent “sequential relationships”
  - Relationships can represent their meanings with «<<stereotype>>»
  - In this workshop interactive relationships without stereotypes represent combinatorial relationship

```
OS  Web browser  Function  Configuration
<<combination>>  «<<sequence>>»
```
Relationships of viewpoints

- Test viewpoints have two fundamental relationships
  - Hierarchy relationships
    » Detail a viewpoint step by step to reach test coverage item with a straight line
    » Have several types such as is-a, has-a, cause-effect, object-attribute
  - Interaction relationships
    » Connect test viewpoints to test combination of viewpoints with a curved line
    » Have several types such as combination (needs combinatorial testing) etc.

- Types of relationships can be expressed as “<<stereotype>>”
Notation of viewpoint diagram in NGT

Viewpoint

Hierarchical Relationship

Interactive Relationship

Stereotype

Combinatorial Relationship

Sequential Relationship

Test Container

Drawing tools for mindmaps are useful
Viewpoint diagram is simple enough

- Viewpoint diagram is simple enough to make a TRA/TAD model
  - More simple than classification tree

OS
Web browser

7 Vista IE Chrome Firefox

Classification Tree

Viewpoint diagram
TRA: Test requirement analysis

• To extract, organize and understand test requirements
  - Requirements from customers to achieve
    » Functional requirement, non-functional requirement, business goals etc.
  - Constraints to achieve requirement from customers
    » Requirement of test project management such as efforts, costs etc.
    » Test tools and/or methods directly requested by customer especially
  - Information of current quality of the test item
    » Ex) bugs which were detected in prior reviews

• To create a test requirement model on viewpoint diagram
  - Extract test viewpoints from test requirements
  - Detail test viewpoints and connect parent viewpoint and child viewpoints
  - Extract interaction relationships and connect viewpoints
  - Top-level viewpoints are most important for grasping a big picture, called “View”
Refinement of a test requirement model

- You can refine a test requirement model to make it clear and easy to understand
  - To detail viewpoints step by step to exhaust / list all test conditions
  - To move, divide or rename viewpoints if necessary
  - To check non MECE viewpoints in each layer and re-organize them as MECE
    - MECE: Mutually Exclusive and Collectively Exhaustive
  - To check whether brotherhood viewpoints have the same stereotypes of hierarchy connections
  - To check whether interactions would be better to change viewpoints
VSTeP

- VSTeP (Viewpoint-based Software Test Engineering Process) is a generic test engineering process model focusing on test viewpoint
  - You can stress upper phase of test engineering process such as test requirement analysis and test architecture design which tend to be negligent
  - VSTeP drives you to good test suite, good review for test design, accumulation of knowledge and experience on testing
  - Reuse and improvement will be easy because you can do reverse-engineering of your past (unorganized) test suite
  - NGT (Notation for Generic Testing) is a made-in-Japan notation for Test Requirement Analysis and Test Architecture Design
    » Modeling skill like object-oriented design is essentially necessary
TAD: Test Architectural Design

- Test architecture is a big picture of test suite
  - It is easy to grasp a big picture in test container level for large and complicated testing
  - Several viewpoints can be packed into a “test container”
  - Test containers can be test levels, test types and test cycles
Guides for good TAD

• Some characteristics, attributes and patterns for software can be applied as guides for good TAD
  - “Quality Characteristics” for software are already available such as ISO/IEC 25000s
    » Functional Suitability / Performance efficiency / Compatibility / Usability / Reliability / Security / Maintainability / Portability
  - Other characteristics and design patterns for software design are also major
    » Coupling / Cohesion / Encapsulation / Responsibility
    » Design patterns such as MVC, singleton

• Characteristics for TAD is important for good TAD
  - Cohesion and coupling
    » A test container should be packed with viewpoints with similar meanings
    » Test containers should have so few combinatorial relationships as possible
  - Maintainability / Internationalizibility / Portability
    » Test containers which needs modifications should be easily identified in maintenance, internationalization and porting
  - Design patterns on viewpoint level could be another guide for TAD
Quality attributes of test suite

• Test architecture depends on required quality attributes of test suite
  - Test suite can have its own quality attribute if test suite is artifact
  - Ex) Maintainability of test suite
  - It doesn’t mean testing of quality attribute such as ISO/IEC 25000s/9126s
Example of design pattern for TRA/TAD

- Interaction Cluster Partitioning Pattern
  - if you can specify the source of combinatorial bugs is e-mail protocols and accept risks of bugs from other sources, you can reduce combinatorial test cases with ICP design pattern

\[
(3 \times 4 \times 2) \times (2 \times 3 \times 4) = 576 \text{ cases}
\]

\[
(3 \times 4 \times 2) + (2 \times 3 \times 4) + (4 \times 3) = 64 \text{ cases}
\]
Research Question

- Can Test Viewpoint Diagram reduce omission of test conditions?
  - Can test requirement modeling activity based on test viewpoints reduce omission of test conditions more than based on test conditions?
  - Is independent requirement “modeling” for testing is better than “deriving test conditions”?
    » Does “Deriving test conditions” mean just reading test base and writing them?

- We conducted an experiment to compare “deriving test conditions” and “modeling test viewpoints”
  - Lecture from academia and experiment by industry
Experiment for reducing omissions of test conditions

• Overview of Experiment
  - Experiment for reducing omissions of test conditions in test requirement analysis phase by 2 teams using test conditions and test viewpoints
    » Team C: Selected test conditions using test conditions
      · Team C selected test cases by the same way as actual test design using spreadsheet
      · To compare easily, we redrew Team C’s test cases into Test Viewpoint Diagram
    » Team V: Selected test conditions using test viewpoint model
      · We made lectures on NGT/VSTeP and instructed them to model with Hierarchical relationships
      · They drew Test Viewpoint Diagram using a mindmap tool (freemind)
  - Both teams had 3-4 engineers and all engineers had 5-10 years experiences in software testing company mainly for embedded software

• Test Base / SUT
  - User manual of a highly functional rice cooker
  - Manual mainly has 2 parts:
    » Operational instructions / functional descriptions
    » Recommendations of cooking rice, e.g. for Sushi
Condition-based test requirement model

<table>
<thead>
<tr>
<th>Condition-Value list</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy for listing</td>
<td>To derive necessary test conditions and values for the rice cooking functions referring to operational steps of rice cooking</td>
</tr>
<tr>
<td>Precondition</td>
<td>-</td>
</tr>
<tr>
<td>Test Conditions No.</td>
<td>1</td>
</tr>
<tr>
<td>Kind of rice</td>
<td>Polished rice</td>
</tr>
<tr>
<td>Amount of rice</td>
<td>1 cup</td>
</tr>
<tr>
<td>Amount of water for cooking</td>
<td>Same as gauge</td>
</tr>
<tr>
<td>Pre-soak in water</td>
<td>Not to be soaked</td>
</tr>
<tr>
<td>Amount of water for post-cooking</td>
<td>45ml</td>
</tr>
</tbody>
</table>

To compare easily, we redrew the list into Test Viewpoint Diagram
Viewpoint-based test requirement model
Comparison of Condition-based and Viewpoint-based model

Condition-based model

Viewpoint-based model

Software Testing
Result: no. of omissions of test conditions

- **Condition-based model omitted 13 more test conditions than Viewpoint-based model**
  - Team C selected test conditions only which are explicitly written and easily identified as test conditions
    » Model is constructed in spreadsheet style
  - Team V modeled the SUT itself whether viewpoints are test conditions or not
    » Model is constructed in NGT/mindmap style

<table>
<thead>
<tr>
<th></th>
<th>Conditions explicitly written</th>
<th>Omissions of Conditions</th>
<th>Out-of-scope</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Condition-based Model</strong></td>
<td>All</td>
<td>+13</td>
<td>0</td>
</tr>
<tr>
<td><strong>Viewpoint-based Model</strong></td>
<td>All</td>
<td>+0 (baseline)</td>
<td>18</td>
</tr>
</tbody>
</table>
Detail of omitted test conditions out of Condition-based model

• Ambiguously written input parameter: 1 condition
  – “Memory of a past cooking course”

• Ambiguously written usecase: 1 condition
  – “Successive cooking”

• Attributes of physical object explicitly written in Recommendation part of the manual: 6 conditions

• Ambiguously written expected results (behavior): 4 conditions

• expected result of physical object explicitly written in Recommendation part of the manual: 1 condition
  – “Badness of taste”
Consideration on threats to validity

- Lack of empirical study?
  - Single experiment can be biased and consistently extract wrong information
  - We gathered multiple engineers who have almost the same experience into each team
- Lack of assessing the validity of cost and time measures?
  - Team V spend more time (but practically acceptable) on TRA than team C
  - Cost and time for TRA and TAD is often made more ignorable than TDD, TI and Test Execution
- Lack of assessing the validity of domain skills of engineers?
  - All the engineers have almost the same domain skills as rice is the most popular food in Japan
- Lack of evaluations for instances of growing size and scope?
  - Omissions are mainly about domain objects, i.e. physical objects, and ambiguous specifications
  - As size and scope of test grow, domain objects and ambiguous specifications will grow, VP model will be more effective
- Lack of evaluations for integrity and testability of the test base
  - Integrity and testability of the test base grow, description on domain object will get richer and ambiguous specification will decrease
  - Although C model can reduce omissions, integrity and testability of test basis is limited actually. We can estimate VP model will be yet more effective
- Lack of evaluations for refinement of model
  - We didn't measure or limit the number of refinement of model exactly
  - VP model was more refined than the C model
  - As good model written in good notation will be more refined, VP model can be estimated to be more effective.
Conclusion

• Independent test requirement analysis (TRA) is necessary

• It is important to construct a test requirement model in TRA

• In our experiment, test viewpoint model can reduce omission of test conditions more than test condition model

• Scientific measurement of merit of “modeling” is rather difficult or not?
  – Empirical study is necessary
Thank you for your kind attention

NISHI, Yasuharu
http://qualab.jp/
Yasuharu.Nishi@uec.ac.jp

© NISHI, Yasuharu