Viewpoint-based Test Architecture Design

Yasuharu NISHI
Department of Informatics, Graduate School of Informatics and Engineering
The University of Electro-Communications, Tokyo
Tokyo, Japan
Yasuharu.Nishi@uec.ac.jp

***Abstract*— Software test recently becomes large-scale and complicated artifact as software itself. Research and practices has to be boosted such as test architecture. In this paper first we mention TDLC: Test Development Life Cycle, which includes test requirement design phase and test architecture design phase instead of test planning from engineering view. Second we discuss concepts of test architecture and propose NGT: Notation for Generic Testing, which is a set of concepts or notation for design of software test architecture. Viewpoint is discussed as a key concept of test architecture representing a group of test cases and test objective. And this paper gives an example of test architecture model. Finally this paper shows possibility that viewpoint diagram will be a platform of test architecture design technology such as test design patterns, test architecture style, variability analysis of product line engineering and so on.**

Keywords-component; test architecture; Test Development Life Cycle; test requirement analysis; test suite; viewpoint; UTP; NGT;

I. INTRODUCTION

Software test recently becomes large-scale and complicated artifact as software itself. There can be a test project with over one million test cases or with over ten test levels. Technology of large-scale and complicated software test has just begun advance and has to be boosted.

"Software architecture" technology arose in 1990s for development of large-scale and complicated software based on abstraction, separation of concerns, modeling, patterns and so on. "Software test architecture" technology has just arising in our age, and we have to boost research and practices on software test architecture technology more and more. This paper shows perspective of research and practices on software test architecture.

Architecture of software system has two kinds of scope: system architecture and software architecture. System architecture is for software, platform, peripherals, environment, network et al. Software architecture is only for software inside, which mainly consists of modules (groups of statements) such as classes.

Test architecture also has two kinds of scope: test system architecture and test suite architecture. Test system architecture is for test system, system/software to be tested (SUT), platform where SUT is executed, generator of test cases et al. Test suite architecture is for test suite inside, which mainly consists of groups of test cases such as test levels and test types.

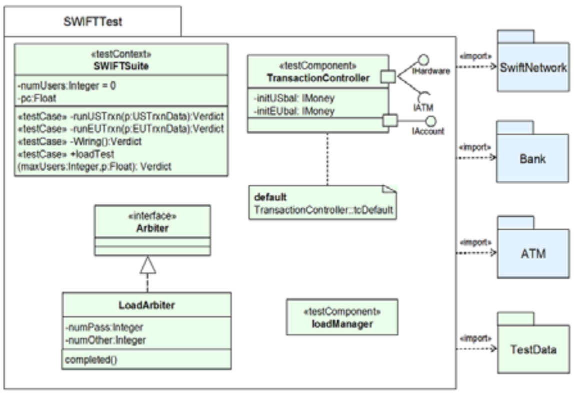

Fig.1 A Test system architecture example on UTP[1]

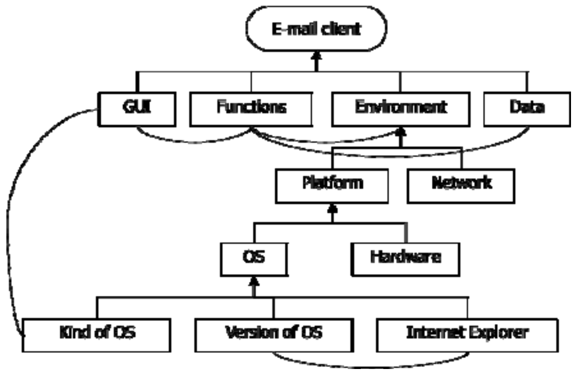

Fig.2 A Test suite architecture example on NGT

II. TEST SYSTEM ARCHITECTURE AND TEST SUITE ARCHITECTURE

There are several research and practices on test system architecture. UML Test Profile[1] is standardized as a notation based on UML for test system architecture. But research and practices of test suite architecture stays just experiences and heuristics. In this paper hereinafter the word "test architecture" means test suite architecture. Fig.1 shows an example of test system architecture according to UTP, UML Test Profile. Fig.2 shows an example of test suite architecture according to NGT, Notation of Generic Testing discussed in chapter V.

III. Test Planning and Test Architecture Design

Test process is recognized roughly by tradition as below: Test planning, test design and test execution. Traditional test design means a phase to derive test cases by test techniques such as control path testing. Traditional test planning means a phase which includes planning test project and drawing big picture of test cases, that is, which includes both tasks of management side and engineering side.

In software development project planning phase includes only tasks of management side and software architecture design phase fills a role of drawing big picture of software, that is, just engineering side. A lot of companies have both positions of project manager and software architect. In software testing tasks of management side and engineering side are traditionally mixed as test planning, test strategy or test approach, because software testing is tight and careful task for budget and effort. Fig.3 shows Heuristic Test Strategy Model by James Bach [2]. Mixture and severe constraint lead test researchers and practitioners to one-sided view. A lot of companies have a position of "test manager" but only a few companies have a position of "test architect".

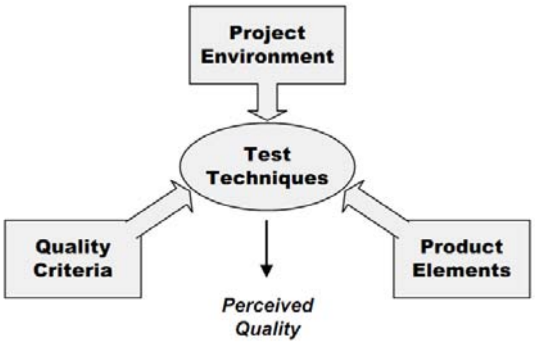

Fig.3 Heuristic Test Strategy Model [2]

To boost research and practices on software test architecture technology, we have to distinguish management side and engineering side. It is necessary to re-define test process only from engineering side named TDLC, Test Development Life Cycle. Fig. 4 shows TDLC, which consists of four phases: test requirement analysis, test architecture design, test detail design and test implementation. TDLC is just to develop test cases or test script. Whole test process needs test execution phase, test result recording phase and several test management tasks.

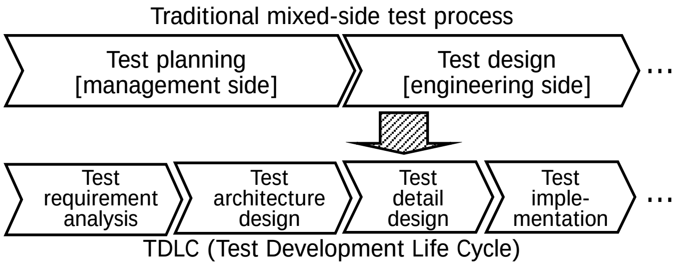
Traditional mixed-side test process
Test planning [management side]
Test design [engineering side]
…
Test requirement analysis
Test architecture design
Test detail design
Test imple-mentation
…
TDLC (Test Development Life Cycle)

Fig.4 TDLC (Test Development Life Cycle)

IV. Concepts for Test Architecture

As there is still no agreement on the precise definition of the term "software architecture", the precise definition of test architecture is impossible for the present. For example IEEE std. 1471-2000[3] defines "architecture" as "The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution". To follow IEEE's definition, we have to clarify what are components and relationships as well as statements in software testing.

It is natural for statements to correspond to test cases or test scripts. This correspondence leads components to be group of test cases such as test types or test levels, which are essentially hierarchical. It should be noted that classes, which are components in OO paradigm, has two angles. The one is group of statements (and data) as an extension of structured programming as a way of OOP. The other one is constituent of the world as a way of OOA. Test types or test levels may be from the former angle. We should deeply discuss which angle is suitable for test architecture just following test requirement analysis and how seamless test requirement analysis model and test architecture model should be.

Relationships are more difficult than statements and components. There may be at least two types of relationships. The one is for combinatorial testing. If a load test type should be tested combinatorially with a configuration test type, they have some relationship. The other one is sequential dependency. As an integration test level should be tested after a unit test level, they have some relationship. We should find various types of relationships.

In addition we have to clarify what is necessary for principles guiding test detail design and implementation. Some principles for software design can be applicable such as abstraction, separation of concerns, modularity. Quality characteristics of test suite can indicate and assist good test design such as maintainability of test suite or test cases. Notation or formulation can make engineers easy to store reusable test assets, test design patterns and test architecture styles. Product line engineering of test suite can arise separately from test design just for software product line.

V. NGT: Notation for test architecture design

For design of test architecture, notation or a set of concepts is necessary. It should consist of concepts of a group of test cases, hierarchical structure, relationship for combinatorial testing, relationship for sequential dependency. It would be better if it can harmonize the principles, abstraction, separation of concerns, modularity, quality characteristics.

We propose notation or a set of concepts named NGT, Notation for Generic Testing. NGT consists of three concepts which are viewpoint, hierarchical relationship and interactive relationship. Viewpoint is a concept of a group of test cases. Hierarchical relationship is used for hierarchical structure of viewpoints. Hierarchical relationship means abstraction (is-a), composition (has-a), cause-effect and object-attribute. Interactive relationship means necessity for combinatorial testing.

Lowest boxes (most detailed viewpoints) usually mean coverage items of groups of test cases or test detail design. Test detail design is a phase to extract test cases by test design technique such as equivalence partitioning, control flow testing and state transition testing. When control flow testing is used, "control flow" is most detailed viewpoints.

In Fig.2 viewpoint diagram of NGT, boxes represent viewpoints. Directional lines represent hierarchical relationships and unidirectional curved lines represent interactive relationships. This diagram is named as "Viewpoint diagram".

Though viewpoint diagram looks similar to classification tree[4], viewpoint diagram is more suitable for drawing big picture of software testing than classification tree. Viewpoint concept doesn't include only equivalence partition but coverage item. Viewpoint diagram can represent combinatorial relationships in the same diagram as viewpoints at higher abstraction level, i.e. coverage item level, although classification tree can do so in different diagrams at lower abstraction level, i.e. parameter level.

Viewpoint diagram has also two angles. Like an angle of OOP, which is lower abstraction level, viewpoint means a group of test cases. Like an angle of OOA, which is higher abstraction level, viewpoint means test objective. In test requirement analysis phase test objectives are listed and refined. Fig.5 shows an example of viewpoint diagram for testing of some mission critical system in test requirement analysis phase. Viewpoints are listed enough but combinatorial relationships are too many and too complicated to test. In test architecture design phase the viewpoints diagram should be well-organized using modeling technique.

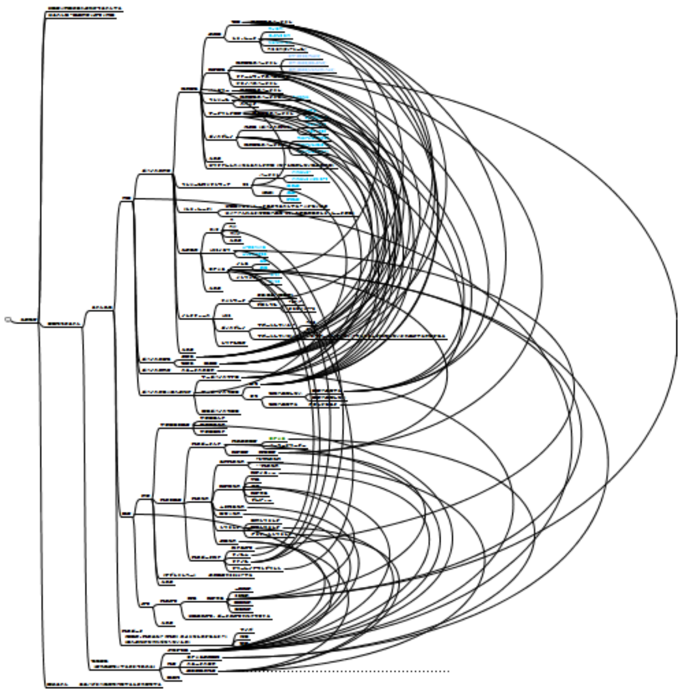

Fig.5 An example of viewpoint diagram in test requirement analysis phase

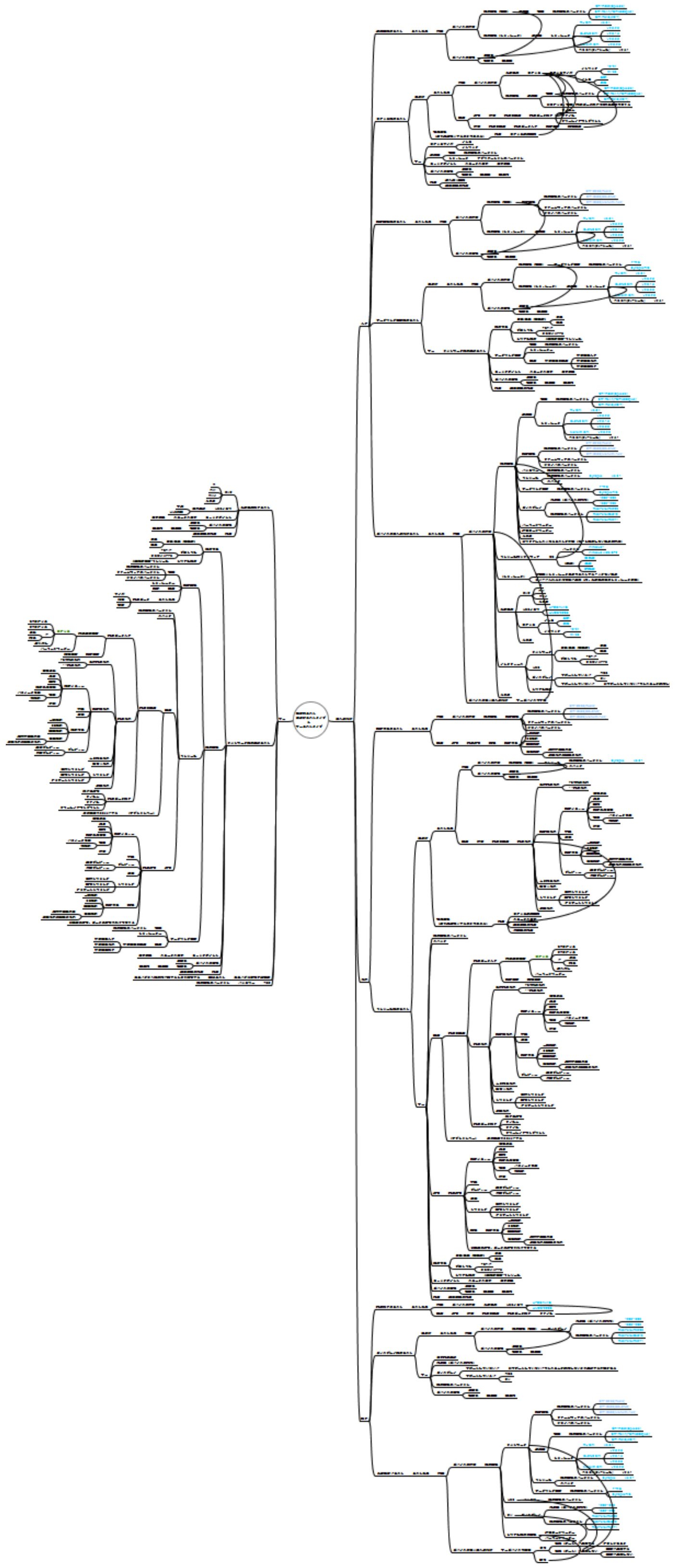

Fig.6 organized viewpoint diagram in test architecture design phase

Fig.6 shows an organized viewpoint diagram in test architecture design phase. Fig 5 includes less viewpoints and more interaction, which means combinatorial relationship. Fig.6 includes more viewpoints and less interaction. Fig.6 is more complicated visually and conducts far more test cases because test cases conducted by interaction are proportional to multiplication among test cases conducted by each viewpoint. Fig 5 is larger visually but conducts less test cases because it has less interaction.

In test architecture design phase, we use some modeling techniques such as unification and re-define of viewpoint, unification and abstraction of interaction, clustering viewpoints, separation of key interaction and so on.

Each modeling technique is usually applied in test planning phase with experiences and heuristics. Viewpoint diagram can makes it easier to develop, accumulate and reuse experiences and heuristics as modeling techniques or test design patterns. In other words, viewpoint diagram will be a platform of test architecture design technology such as test design patterns, test architecture style, variability analysis of product line engineering and so on.

NGT can complement UML Test Profile because research and application of UTP mainly focus on test system architecture such as automation at present and NGT focuses on test suite architecture. NGT should harmonize UTP in future research.

VI. Conclusion

Software test recently becomes large-scale and complicated artifact as software itself. Research and practices has to be boosted such as test architecture. In this paper first we mentioned TDLC: Test Development Life Cycle, which includes test requirement design phase and test architecture design phase instead of test planning from engineering view. Second we discussed concepts of test architecture and propose NGT: Notation for Generic Testing, which is a set of concepts or notation for design of software test architecture. Viewpoint is discussed as a key concept of test architecture representing a group of test cases and test objective. And this paper gave an example of test architecture model. Finally this paper showed possibility that viewpoint diagram will be a platform of test architecture design technology such as test design patterns, test architecture style, variability analysis of product line engineering and so on.

References

[1] OMG, “UML Testing Profile (UTP) Version 1.1 RTF - Beta 1,” http://www.omg.org/spec/UTP/1.1/PDF/, June 2011.

[2] J. Bach, “Heuristic Test Strategy Model,” http://www.satisfice.com /tools/satisfice-tsm-4p.pdf, March 2006.

[3] IEEE, “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems,” IEEE Std 1471-2000, September 2000.

[4] M. Grochtmann, K. Grimm, “Classification trees for partition testing,” Software Testing, Verification and Reliability, Vol. 3, Issue 2, pp. 63–82, June 1993.