Proposal for Enhancing UTP2 with Test Aspects

Makoto Nakakuki  
TEF-Do
Shizuka Ban  
NPO ASTER
Noriyuki Mizuno  
TEF-Do

Koki Abe  
Tosei Systems Co., Ltd
Hiroki Iseri  
NPO ASTER
Tomohiro Odan  
Mamezou Co., Ltd.

Hitoshi Ando  
TEF-Do
Kumiko Iseri  
TIS Inc.
Akiharu Satoh  
Hitachi Information Academy Inc.
Contents

1. INTRODUCTION
2. PROPOSAL FOR TEST ASPECT MODEL
3. PROPOSAL FOR THE NOTATION
4. CONCLUSION
INTRODUCTION
Introduction

Various models are developed prior to software testing.
Introduction

These incomplete models that have omitted information are usually revised by modeling for software testing.
Introduction

Other aspects of development model are created from test engineer’s concern.
PROPOSAL FOR
TEST ASPECT MODEL
Common Problem

Development Models are usually Incomplete...
To Complement a Development Model

The development model can be created using modeling notations such as UML and SysML.
To Complement a Development Model

The test design process creates a test project model. “Test project model” has test cases.
To Complement a Development Model

Relationship among software, development model and test project model.
Case 1: Development model is perfect

Software

Developing → Developing Model

Testing → Test Project Model

Referring
Case 1: Development model is perfect

Software

Development Model

Test Project Model
Case 1:
Development model is perfect

- Software
- Development Model
- Test Project Model
Case 1: Development model is perfect
Case 1: Development model is perfect
Development Model is Incomplete
Case 2: Development Model is Incomplete
Case 2: Development Model is Incomplete
Case 2: Development Model is Incomplete
Case 2: Development Model is Incomplete

Test cases designed based on test engineer’s concern usually detect these bugs.
Test Depends on the Test Engineers’ Skill

Test engineers sometimes rely on some information such as “bugs” detected in the software components in the past.
Case 3: Test Depends on the Test Engineers’ Skill
Case 3: Test Depends on the Test Engineers’ Skill
Case 3: Test Depends on the Test Engineers’ Skill
Use Aspect Model

Concerns of test engineers are usually tacit knowledge...
Concerns Tend to be Tacit and Subjective

We consider that much of information on testing will be derived from concerns of test engineers.
Define Concerns as Aspects

[Diagram showing the relationship between Other Aspect of Software and Concerns, with elements like Test Engineers’ Experience and Tacit Knowledge indicated.]
Example Define Method

Viewpoint Diagram

Define Concerns as Aspects
Use Test Aspect

For Example...
- Reliability
- Security
- Invalid Case
Proposal for Test Aspect Model

Development Model
Proposal for Test Aspect Model
Proposal for Test Aspect Model
Proposal for Test Aspect Model
Proposal for Test Aspect Model
Proposal for Test Aspect Model

Development Model

Test project Model

Test aspect Model

Test Project Model
Proposal for Test Aspect Model

- Test Engineers' Experience
- Test Engineers' Skill
- Tacit Knowledge

Software

Development Model

Other aspect of software

Testing

Test Project Model

Compensated Model from Other Aspect of Software

Referring
Proposal for Test Aspect Model
Proposal for Test Aspect Model

Software

Developing

Testing

Development Model

Test Project Model

Test Aspect Model

Referring
Proposal for Test Aspect Model

Development Model

Test Aspect Model

Test Project Model

Referring

Information on testing derived by referring to development model only

Referring

Referring
Case 1: Development Model is Perfect
Case 1:
Development Model is Perfect

Development Model = Test Aspect Model ∈ Test project Model

Information on testing derived by referring to development model only
Case 2:
Development Model is Incomplete

Development Model \textless Test Aspect Model \in Test Project Model

\text{Development Model + Other Aspect of Software}

Referring

Information on testing derived by referring to development model only

Referring
Case 2: Development Model is Incomplete
Also Enrich Test Project Model

Test Aspect Model

Test Project Model

Referring
Using the Test Aspect Model

Furthermore, we suggest to share information and agreement among stakeholders, and to grasp the whole picture, using the test aspect model. This approach will smoothen the problems when test designs rely on test engineer's skills.
PROPOSAL FOR THE NOTATION
Next Problem

Complicated system has many test aspects...
Next Problem..

As the software itself is bigger and more complicated now. → Test aspect models also become complicated.
Next Problem..

These test aspects are usually tacit knowledge.
Next Problem → Necessity for the Notation

- to share the information with other test engineers
- to grasp a holistic model
- to find the omitted test aspect information

The unified notation is essential...
Necessity for the Notation.

The requirement for the notation
In UML Testing Profile...
Necessity for the Notation

- to share the information with several test engineers

- to grasp a holistic model
- to find the omitted aspect information.
Requirements for the Notation

- to share the information with several test engineers
  → The notation should be easy to be shared and to be learned by test engineers.

- to grasp a holistic model
- to find the omitted aspect information.
  → The notation should have capacity to organize complicated information.
Example of Test Aspect

Test for ATM “deposit” function. Following items (test aspects) may be considered for designing test cases.

Insert Money (Test)
- Valid Money
- Invalid Money
- ...

Reliability (Test)
- Repeated transaction
- Contingency approach
- ...

56
References of other Notations

Viewpoint Diagram

Test Conglomeration


Requirements for the Notation

Each element has relationship with others.
Requirements for the Notation

Each element has relationship with others. In the notation, “has-a”, “is-a” relationships are useful. → UML metaclass “class” is suitable for describing test aspect.

“is-a” relationship  “has-a” relationship

<<TestAspect>>
Insert Money

<<TestAspect>>
Reliability

<<TestAspect>>
Repeated transaction

<<TestAspect>>
Contingency approach

<<TestAspect>>
Ammount

<<TestAspect>>
Under allowable limit

<<TestAspect>>
Over allowable limit
In UML Testing Profile

http://utp.zen-tools.com/
In UML Testing Profile

Test System Architecture

http://utp.zen-tools.com/
In UML Testing Profile

Test System Architecture

It has no concept to describe an element like the test aspect.
Nor it has no way to describe structure of test cases.

http://utp.zen-tools.com/
In UML Testing Profile

Test System Architecture

Test Suite Architecture

It has no concept to describe an element like the test aspect.
Nor it has no way to describe structure of test cases.

http://utp.zen-tools.com/
We propose for

http://utp.zen-tools.com/
We propose for

Enhancing UTP2 with “Test Aspects”

http://utp.zen-tools.com/
Explanation and Example of the Notation for Test Aspect

The notation of Test Aspect
Example of Test Aspect Model
The Notation for Test Aspect

Metaclass "class" with stereotype "TestAspect"

Metaclass "package" with stereotype "TestAspects"
Example of Test Aspect Model

Example for ATM “deposit” test

ATM

<<Interface>>

IATM

+ withdraw(in amount : iMoney) : boolean
+ deposit(in amount : iMoney) : boolean
+ isPinCorrect(in c : integer) : boolean
+ selectOperation(in op : OpKind) : boolean
+ storeCardData(In c : CardData)

<<Enumeration>>

OpKind

- withdrawMoney
- getBalance
- wireMoney

CardData

- pinCode : integer
- cardNumber : String
+ isPinCorrect() : boolean

described in UTP2.0

https://www.omg.org/spec/UTP/About-UTP/
ATM

IATM

- withdraw(in amount : iMoney) : boolean
- deposit(in amount : iMoney) : boolean
- isPinCorrect(in c : integer) : boolean
- selectOperation(in op : OpKind) : boolean
- storeCardData(in c : CardData)

CardData

- pinCode : integer
- cardNumber : String
- isPinCorrect() : boolean

Example of Test Aspect Model

“Test Aspects” deposit package is extracted.

deposit
ATM

+ storeCardData(In c : CardData)
+ selectOperation(in op : OpKind) : boolean
+ isPinCorrect(in c : integer) : boolean
+ deposit(in amount : iMoney) : boolean
+ withdraw(in amount : iMoney) : boolean

IATM

- cardNumber : String
- pinCode : integer

CardData

- wireMoney
- getBalance
- withdrawMoney
- getBalance

<<Interface>>

OpKind

- withdrawMoney
- getBalance
- wireMoney

<<Enumeration>>

<<TestAspects>>

deposit_TestAspects

Example of Test Aspect Model

“Test Aspects” deposit package is extracted.
Example of Test Aspect Model

Deposit has following test aspects.
Example of Test Aspect Model

Deposit has following test aspects.

[Diagram]

- <<TestAspect>>
  - Insert money
    - <<TestAspect>>
      - Valid money
    - <<TestAspect>>
      - Invalid money
    - <<TestAspect>>
      - Amount
      - <<TestAspect>>
        - Under allowable limit
      - <<TestAspect>>
        - Over allowable limit
Example of Test Aspect Model

Deposit has following test aspects.

- **Performance efficiency**
  - **Response time**
  - **Valid money**
  - **Invalid money**
    - **Insert money**
    - **Ammount**
  - **Under allowable limit**
  - **Over allowable limit**
Example of Test Aspect Model

Deposit has following test aspects.

- **Performance efficiency**
- **Response time**
- **Insert money**
- **Reverted transactions**
- **Reliability**
- **Contingency approach**
- **Valid money**
- **Invalid money**
- **Amount**
- **Under allowable limit**
- **Over allowable limit**
- **Power outage**
- **Cash is packed**
Example of Test Aspect Model

Test cases are designed from the part of these test aspects.

Example of Test Cases...
- Insert a valid money that amount is under allowable limit
  (User can insert money successfully)
- Insert invalid money (Error will occur)
- Insert valid money repeatedly
  (All transactions success normally)
Position of Test Aspect in UTP2

In UTP2.0...

UML Testing Profile (UTP) Version 2.0 – Beta
Publication Date: September 2017
Figure 7.2 Test Requirement and Test Objective Overview

SysML etc.

(Functional & Non Functional) Requirement → System specification item

UTP

Test objective

▲ specifies

Test context

refers to

Test requirement

▼ is designed to meet

▲ verifies

Test case

Test context refers to Test requirement which is designed to meet Test objective.

▲ specifies

Test objective is designed to meet Test requirement which verifies Test case.
Position of Test Aspect in UTP2

In UTP2.0 with Test Aspect...

Example: defects that occurred in the past.
Advantage of Test Aspect

The test aspect can complement development models.
Advantage of Test Aspect

- Test aspects can be used to organize the information in the test design process. This characteristic can improve systemized knowledge of the test design.
- The relationship among test aspects enables to grasp the intention of the test design.
- Test engineers can maintain the structure of test aspects using the test aspect model.

As a result, maintainability of the test suite improves.
CONCLUSION
Conclusions

- The test aspect can complement the development model.
- The test aspect can indicate missing information in the development model.
- Test aspect model can describe relationships among test aspects.
- Test aspects organized appropriately in the test aspect model can complement necessary information that is not found in development models.
- This model enables to reduce test case omissions in test design process.
Thank you!

Any Questions?

We are not good at English. So please speak clearly and slowly.
Special Thanks

Satomi-Juku

What is Satomi-Juku?
"Satomi-juku" is a research group of test architecture. It is a place to disclose & share each advanced test development methods them through discussion.
References
References


References


References


[10] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software,” Addison-Wesley Professional, 1994