



The UML Testing Profile

- Tutorial at the ECOOP 2004 -

www.fokus.fraunhofer.de/u2tp

Ina Schieferdecker, Technical University Berlin/FOKUS
Øystein Haugen, University of Oslo

June 2004

© U2TP Consortium

Agenda

- Motivation
- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks



ECOOP, Oslo, June 2004

2

© U2TP Consortium

Agenda

➤ Motivation

- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks



The Problem

- Software
 - Increases in complexity, concurrency, and dynamics
 - Quality is key
 - Functionality
 - Performance
 - Scalability
 - Reliability
 - Usability
 - Efficiency
 - Maintainability
 - ...
- Testing is
 - Means to obtain objective quality metrics about systems in their target environment
 - Central means to relate requirements and specification to the real system



Testing Today

- **Is**
 - Important
 - Means to obtain approval
 - Time critical
- **But often**
 - Rarely practiced
 - Unsystematic
 - Performed by hand
 - Error-prone
 - Considered being destructive
 - *Uncool*
„If you are a bad programmer you might be a tester“
- **Conjecture:**
There is a lack of appropriate test methods and techniques

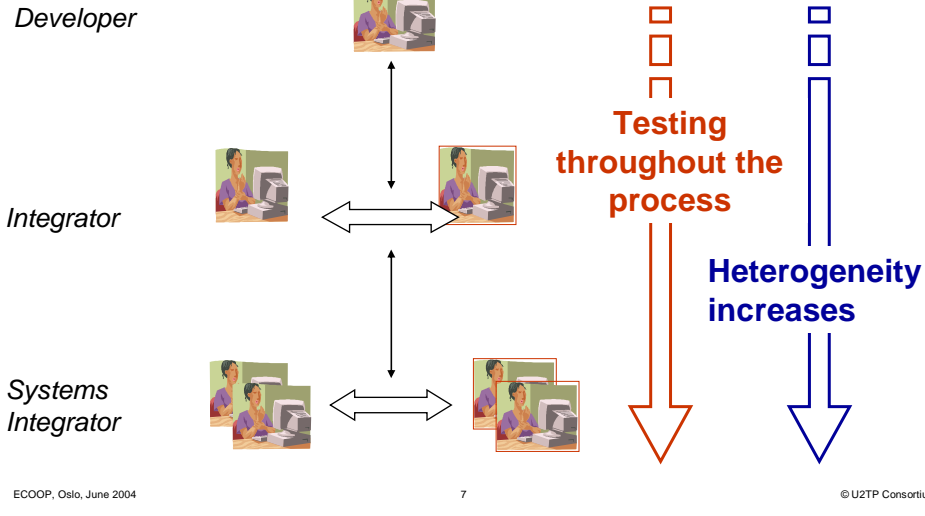


Testing is ...

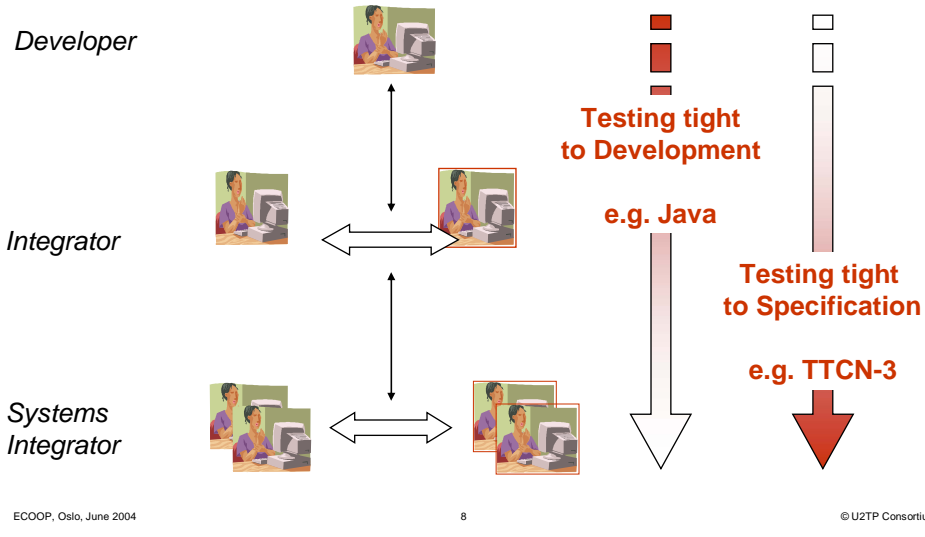
- A **technical process**
- Performed by **experimenting** with a system
- In a **controlled** environment following a **specified** procedure
- With the intent of **observing** one or more **characteristics** of the system
- By demonstrating the **deviation** of the system's **actual** status from the **required** status/specification.

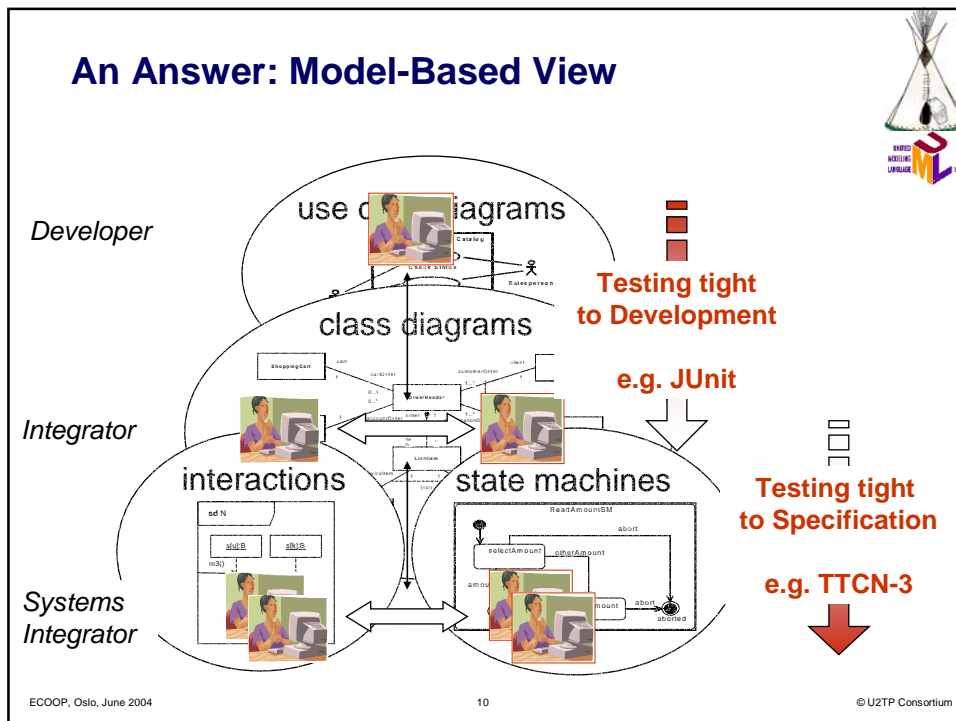
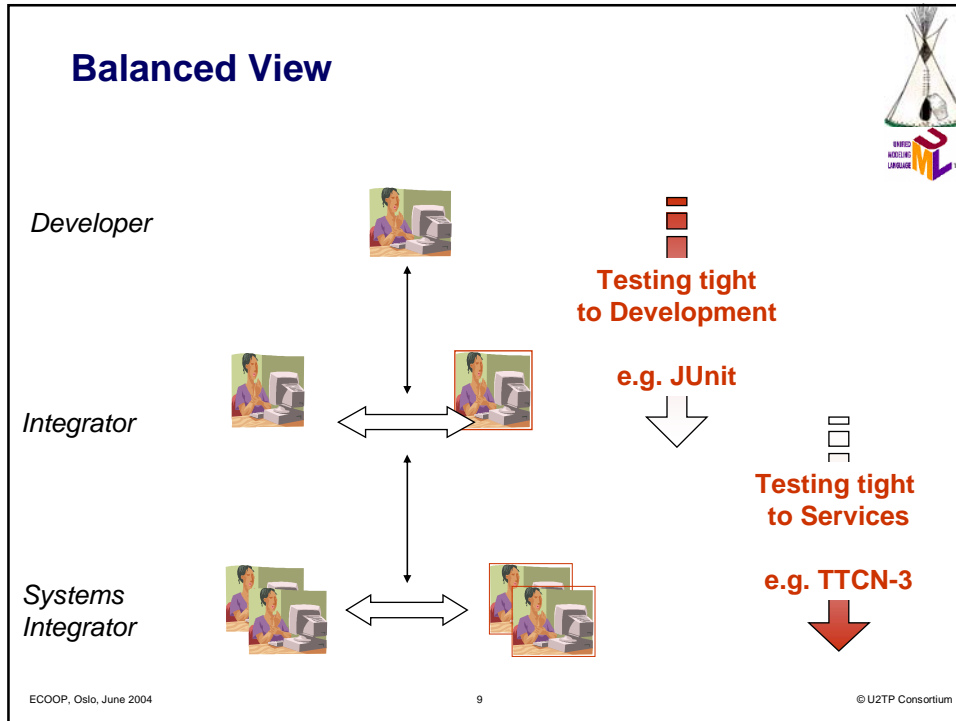


Integrated Development and Testing



Extreme View





UML and Testing

- UML-based test generation, e.g.
 - Integration testing, Siemens, 2000
 - Component test framework (TINA), FOKUS, 2000
 - Component testing (COTE), INRIA, 2001
- UML-based test notation, e.g.
 - Agedis, EC IST project
 - TeLa, COTE project
 - UML Testing Profile, OMG



UML and Testing

- **Model Driven Architecture** as new OMG strategy
 - One objective of UML 2.0 is executable UML meaning
 - Code generation
 - Simulation
 - Validation
 - **Test generation**
 - "..., the expanded role of the OMG must be built on **rock-solid testing**, certification and branding. ..."



Goals of the UML Testing Profile

- Definition of a testing profile to capture all information that would be needed by different test processes
 - To allow **black-box testing** (i.e. at UML interfaces) of computational models in UML
- A testing profile based upon UML 2.0
 - That enables the **test definition and test generation** based on **structural** (static) and **behavioral** (dynamic) **aspects** of UML models, and
 - That is capable of **inter-operation with existing test technologies** for black-box testing
- Define
 - Test purposes for computational UML models, which should be related to relevant system interfaces
 - Test components, test configurations and test system interfaces
 - Test cases in an implementation independent manner



U2TP Partners

- A consortium of **testers, UML vendors and users** dedicated to make UML applicable for software testing
- **Submitters**
 - Ericsson
 - IBM
 - FOKUS
 - Motorola
 - Rational
 - Softeam
 - Telelogic
 - University of Lübeck
- **Supporters**
 - iLogix
 - ScapaTechnologies
 - IRISA

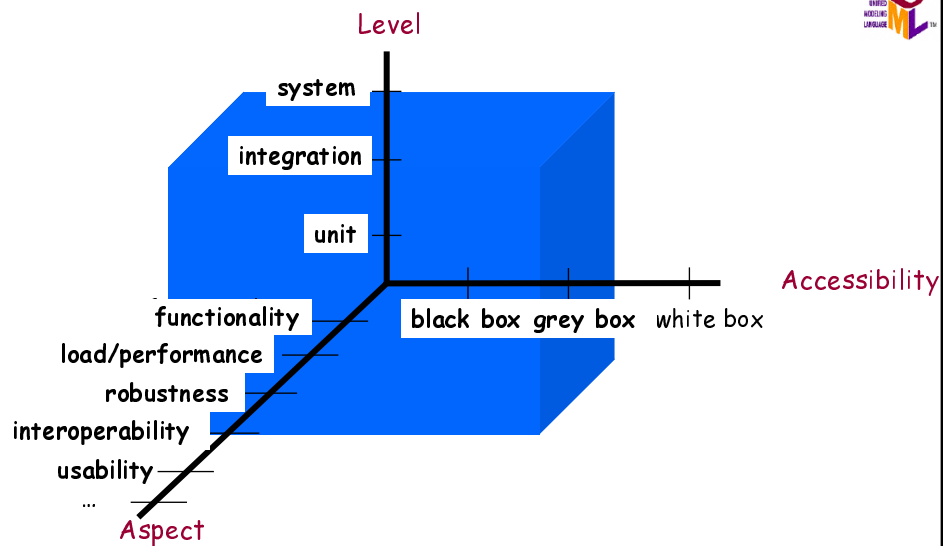


Agenda

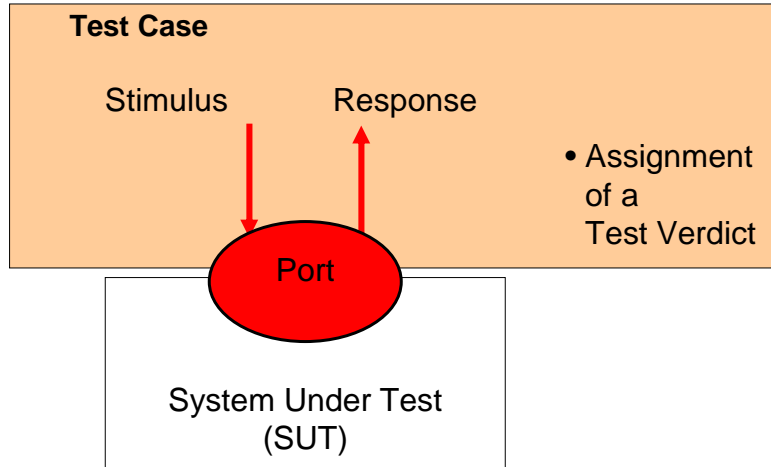
- Motivation
- Basics of Testing
 - Overview of the UML Testing Profile
 - Example Test Specifications
 - The UML Profile and the MOF Model
 - The Mappings
 - Concluding Remarks



Types of Testing



Test Concepts: Black-Box Testing



Test Concepts: Local behavior



- **Stimuli** are given to the system under test (SUT) at well-defined interfaces
- **Observations** are taken and compared to the expected ones – the observations are validated and **arbitrated**
- The **arbitration** decides upon the further testing and/or the assignment of test results – the **verdicts**
- A detailed test procedure consisting of various stimuli and observations is called **test case**
- Several test cases constitute a **test suite**
- The relation between test cases and their execution is defined in a **test control**
- Typically, a test case describes a **tree structure**: one stimuli – several possible observations
- **Alternatives** and **defaults** are used for an efficient description of such trees

Test Concepts: Distributed Behavior

- Typically, distributed SUTs require **distributed test setups**
- Such test setups are realized via **test components** which have to be **coordinated** and **synchronized**
- All test components have a local verdict, which contributes to the **overall verdict**
- The initial **test configuration** defines the test components used initially for a test case – the test configuration can change dynamically
- Test components may use **utility** parts to use/access information outside the test system



Test Concepts: Data

- Both, stimuli and observation are described as **test data**
- Test data for stimuli and observations can be unspecific in their properties and values – **data pools** and **data partitions** are used
- In addition, test data for observations can be unspecific in certain elements – **wildcards** are used
- **Coding rules** describe the encoding/decoding used on interfaces to the SUT

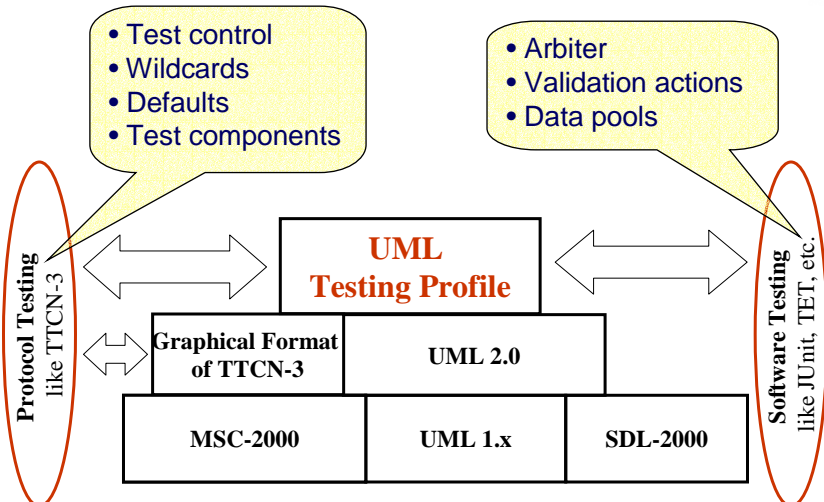


Agenda

- Motivation
- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks



The Testing Profile Roots

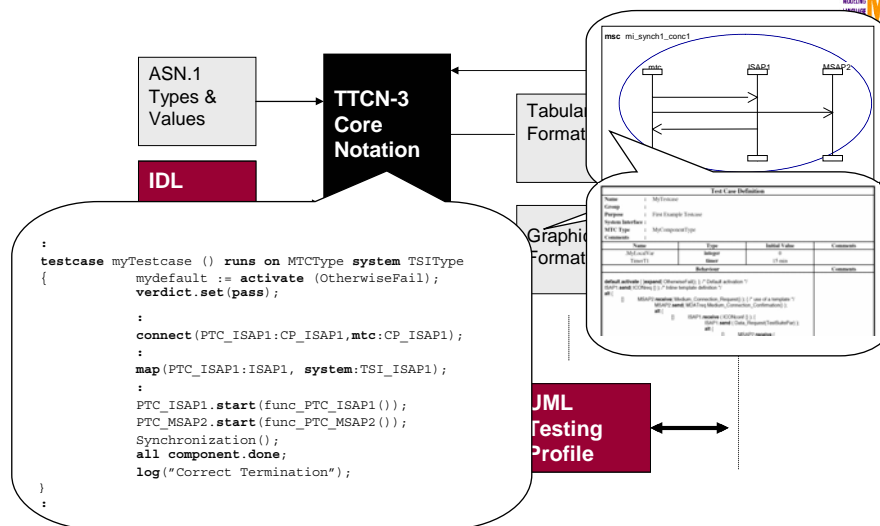


1st Root: TTCN-3



- The new standardised test specification and test implementation language
 - Developed from 1999 – 2002 at the European Telecommunications Standards Institute (ETSI)
- Developed based on experiences from previous TTCN editions
 - Removal of OSI specific concepts; Improvement of concepts; Introduction of new concepts
- Applicable for all kinds of black-box testing for reactive and distributed systems, e.g.,
 - Telecom systems (ISDN, ATM, GSM, UMTS); Internet (IP, IP based protocols and applications); Software systems (Java, XML); Middleware platforms and component-based systems (CORBA, .Net, EJB)

TTCN-3



2nd Root: UML 2.0

- Developed by OMG (Object Management Group)
1999-2004, adoption June 2003, available 2004
 - UML 2.0 Infrastructure RFP
 - metamodel restructuring in order for Core to be reusable by other OMG languages
 - UML 2.0 Superstructure RFP
 - new and improvement/extension of UML concepts
 - UML 2.0 OCL RFP
 - defining an OCL metamodel
 - UML 2.0 Diagram Interchange RFP
 - ensuring diagram interchange between different tools



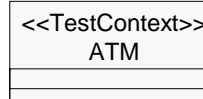
UML 2.0 Improvements

- More unified conceptual base
 - Parts in Internal structure, Collaborations, Use cases and indirectly in Interactions
- More unified semantics
 - Higher precision
- Improved expressiveness
 - Structured Classes, Sequence Diagrams and Statemachines
 - Activities merged with actions
 - Collaborations aligned with structured classes
 - Patterns (templates) and frameworks support
- More powerful and expressive than UML 1.4
- Tighter and more consistent than UML 1.4
- Executable UML becomes possible



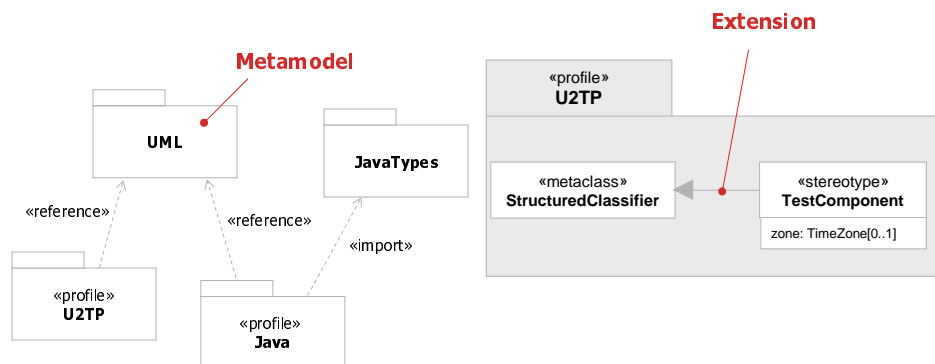
UML 2.0 Profiles

- Use of UML in
 - Analysis
 - Design/implementation
 - Directly executable notation (eg xUML)
 - Architecture description
 - Process engineering, workflow
 - Website structures
 - Data Modeling
- with obviously **different** (and **inconsistent**) semantics
- UML has many “semantic-free zones”, so called “**semantic variation points**”
 - E.g. detailed semantics of state machines, ...
- **Profiles**
 - Specializations of UML by stereotypes, providing special semantics



UML 2.0 Profile Walkthrough (1)

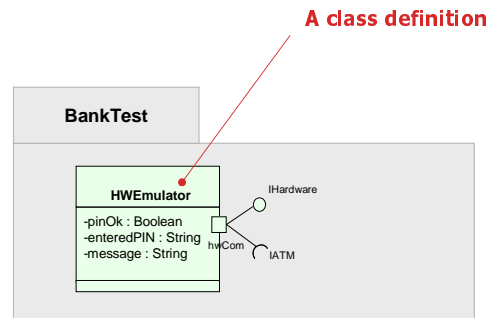
- Define profile(s)
 - based on reference metamodel
 - may use other packages for its definition



UML 2.0 Profile Walkthrough (2)



- Specify model
 - based on UML metamodel



ECOOP, Oslo, June 2004

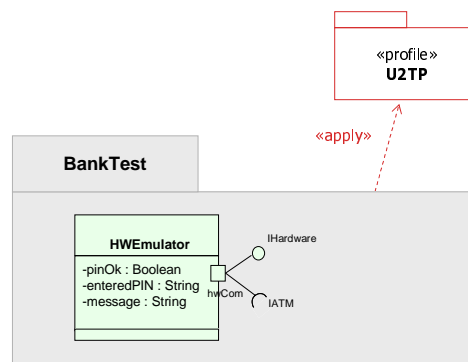
29

© U2TP Consortium

UML 2.0 Profile Walkthrough (3)



- Apply profile(s) to model
 - make it possible to apply stereotypes of the profile to the model elements



ECOOP, Oslo, June 2004

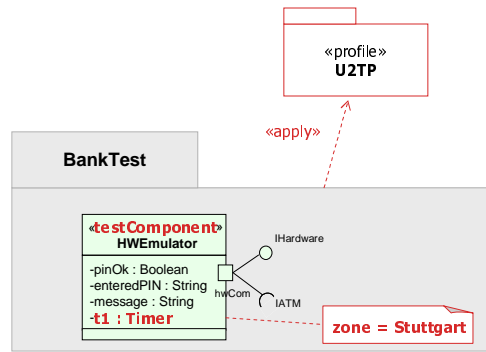
30

© U2TP Consortium

UML 2.0 Profile Walkthrough (4)



- Apply stereotypes to model elements as desired



Concepts of the Testing Profile



- Test architecture
 - Test structure, test components and test configuration
- Test data
 - Data and templates used in test procedures
- Test behavior
 - Dynamic aspects of test procedures
- Test time
 - Time quantified definition of test procedures

Test Architecture Realization

- System Under Test (*SUT*)
- *Test components*
- *Test context* with test configuration and test cases
- Test verdict arbitration with *arbiter*
- Test coordination with *scheduler*

Test Data Realization

- Individual *coding rule* definition
- *Wildcards* * and ?
- Concrete test data with *data pool*, *data partition* and *data selector*



Test Behavior Realization

- *Test objectives*
- *Test cases*
- *Test verdicts*: pass, fail, inconclusive
- *Defaults* behaviors on different levels
- *Utility* part

Test Time Realization

- *Clock*
- *Timezone* definition for synchronizing test components
- *Timer* operations



Concepts beyond TTCN-3



- Unification of test cases:
 - Test case as a **composition of test cases**
 - Test behavior defines the execution of a test case
- Separation of test behavior and verdict handling
 - **Arbiter** is a special component to evaluate the verdict
 - Validation actions are used to set the verdict
- Abstract test cases that work on data partitions rather than individual data
 - **Data partitions** to describe value ranges for observations and stimuli
- Test architecture with test deployment support
 - Part of the test specification is the definition of **deployment** requirements for a test case

Concepts beyond UML



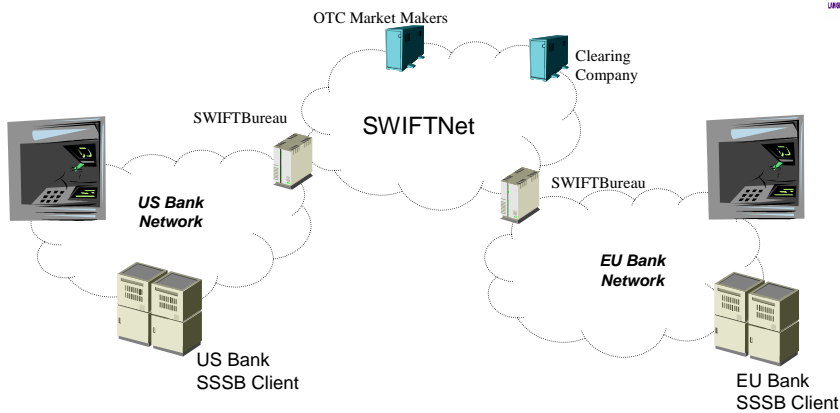
- **Defaults** within test behavior
 - Concentration on main flow of test behavior
 - Default hierarchy to handle different concerns
- **Wildcards** within test data
 - Flexible definition of value sets
- **Timers** and time constraints
 - Time controlled test behavior
- Arbitration and **verdicts**
 - Assessment of test behavior
- **Coding** attributes
 - Encoding/decoding for data exchange with the SUT

Agenda

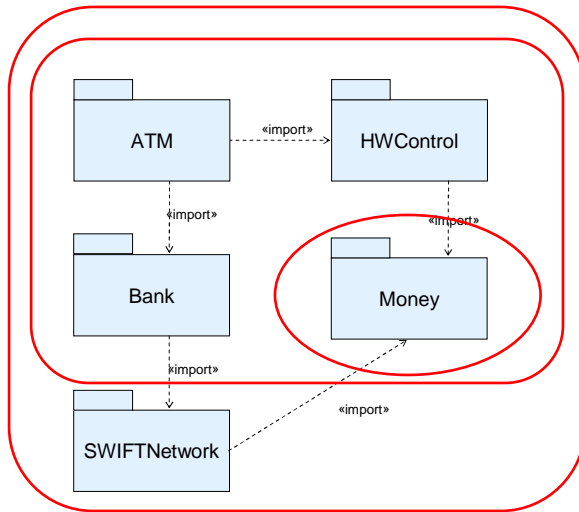
- Motivation
- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks



The Example



UML Model of the System Level Architecture

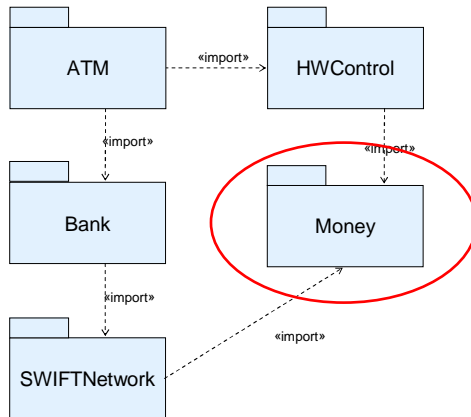


System
Integration Test

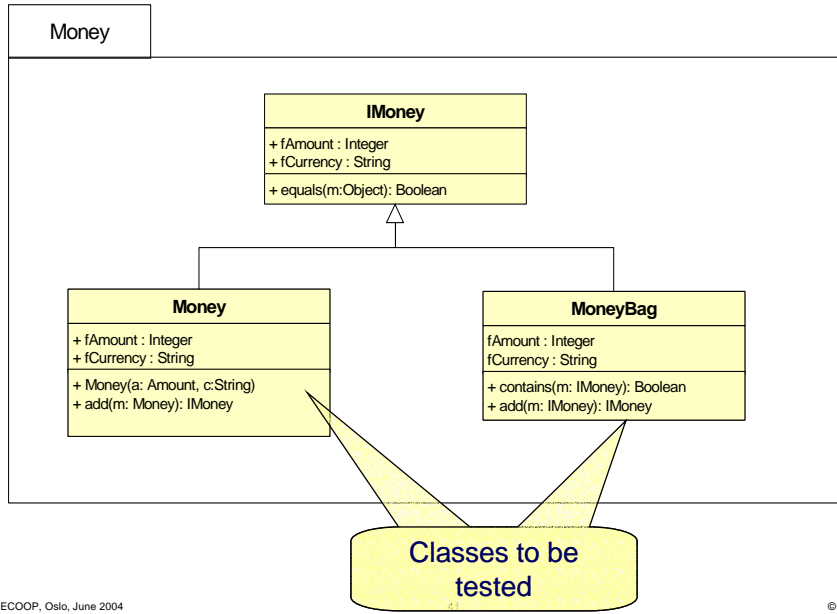
System Test

Unit Test

Unit-Level Test



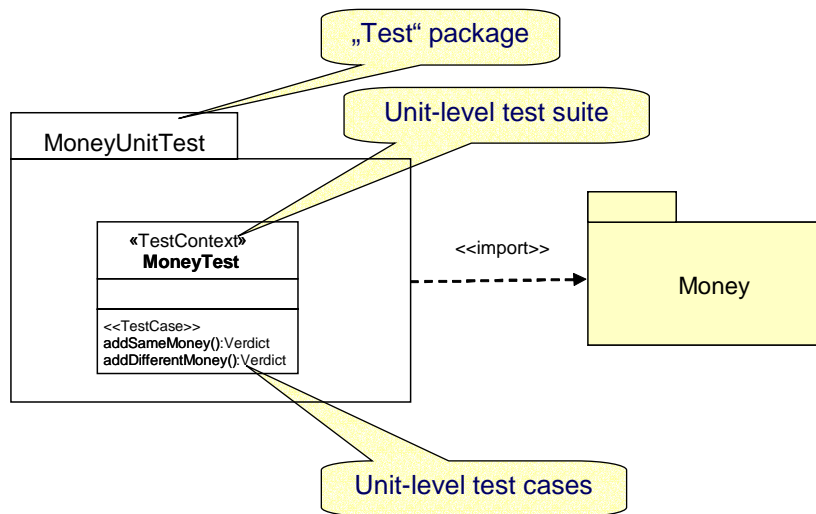
Unit-Level Test



ECOOP, Oslo, June 2004

© U2TP Consortium

A Unit-Level Test Suite

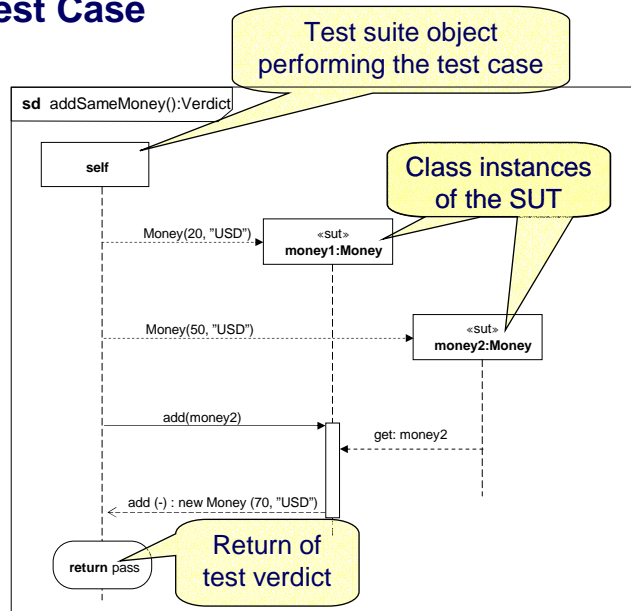


ECOOP, Oslo, June 2004

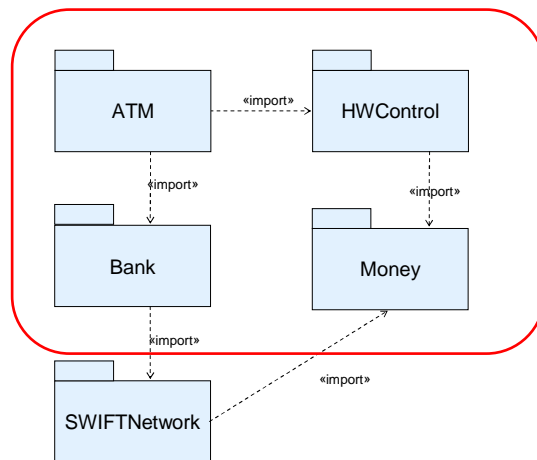
42

© U2TP Consortium

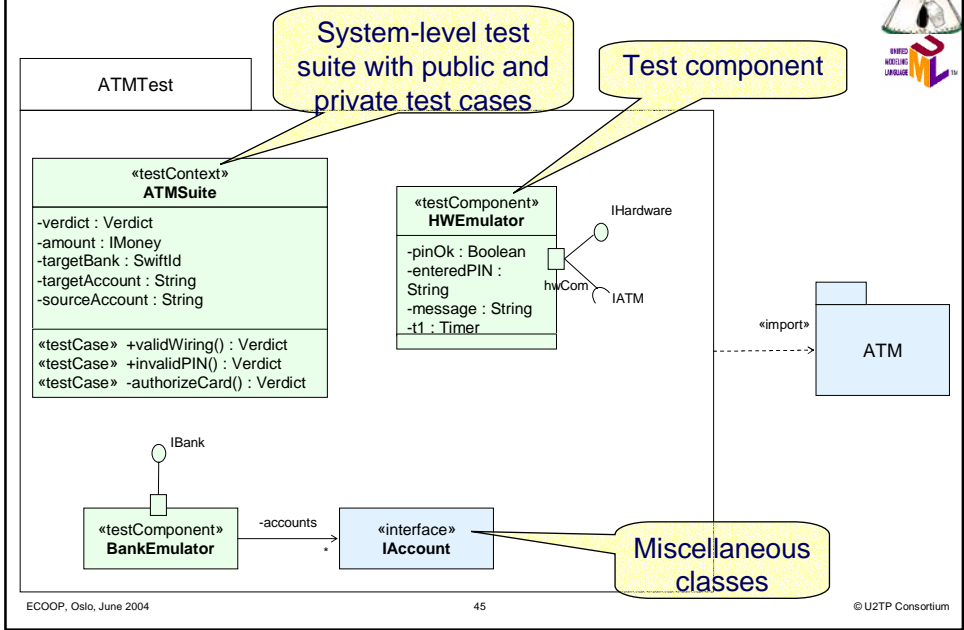
A Test Case



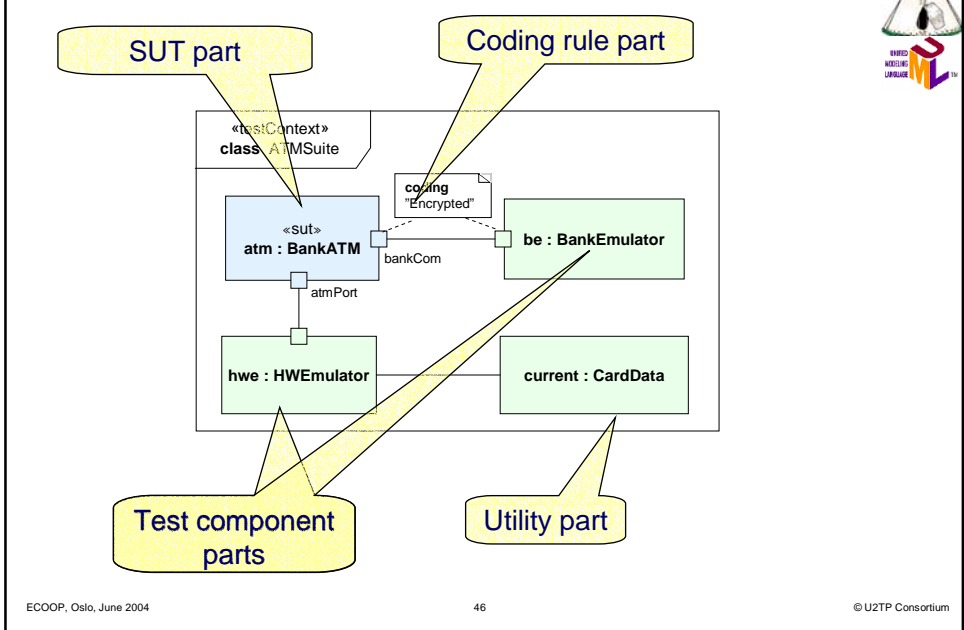
System-Level Test



System Level Test

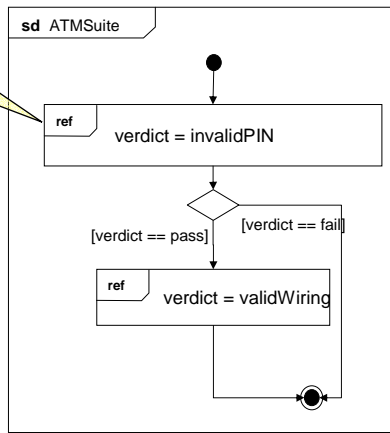


Test Configuration



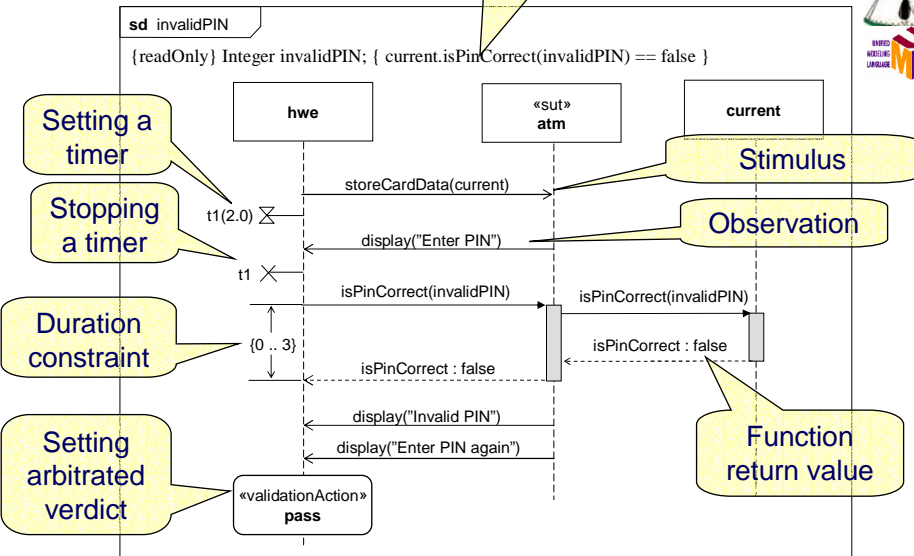
Test Control (execution of test suite)

Referring test case behaviors



A Test Case

Data partition



Setting a timer

Stopping a timer

Duration constraint

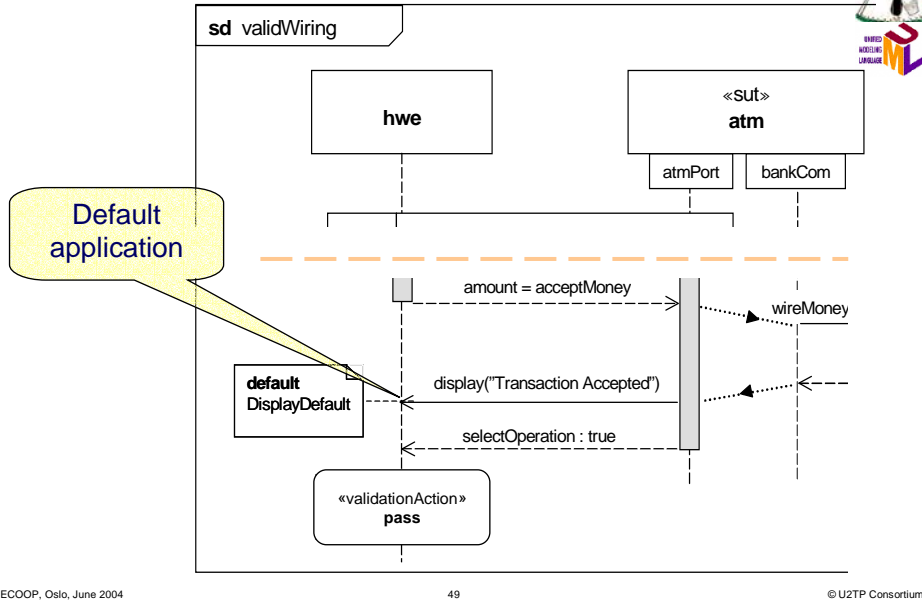
Setting arbitrated verdict

Stimulus

Observation

Function return value

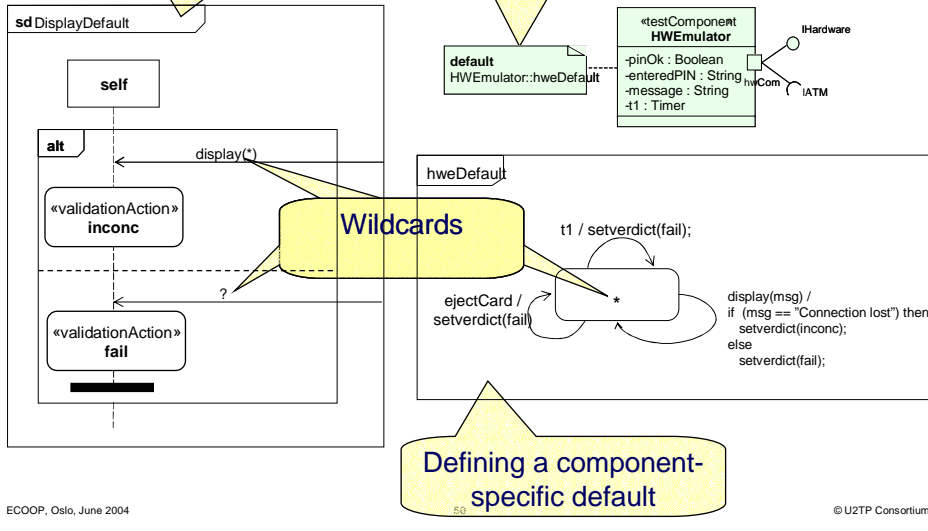
A Test Case with Default (extracts)



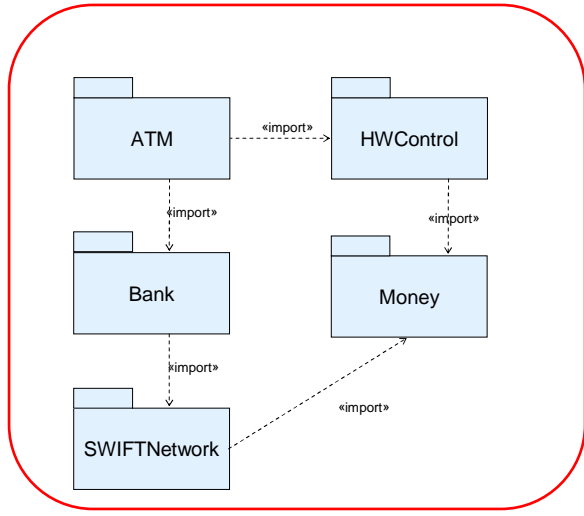
Defaults

Defining an event-specific default

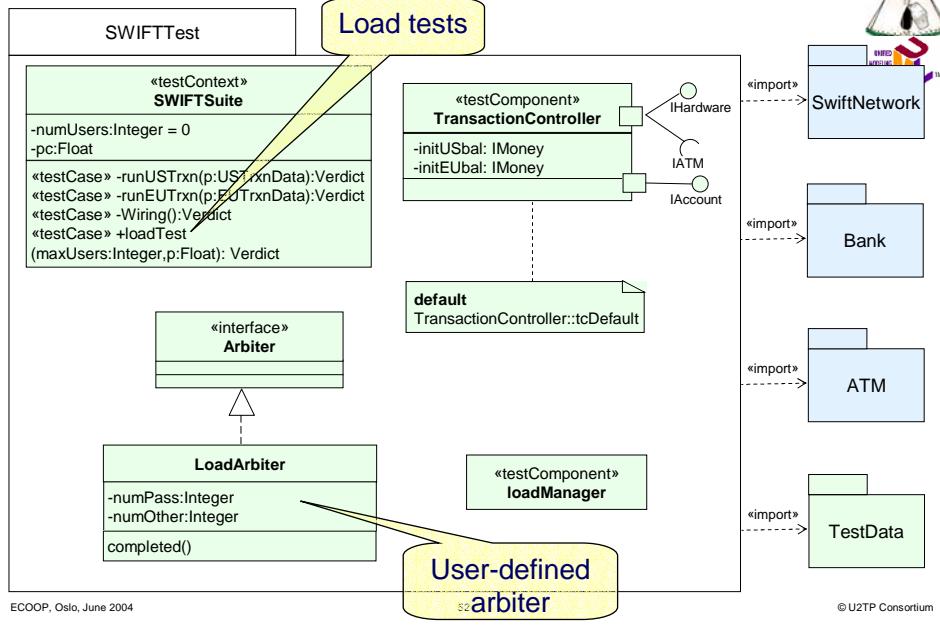
Applying a component-specific default



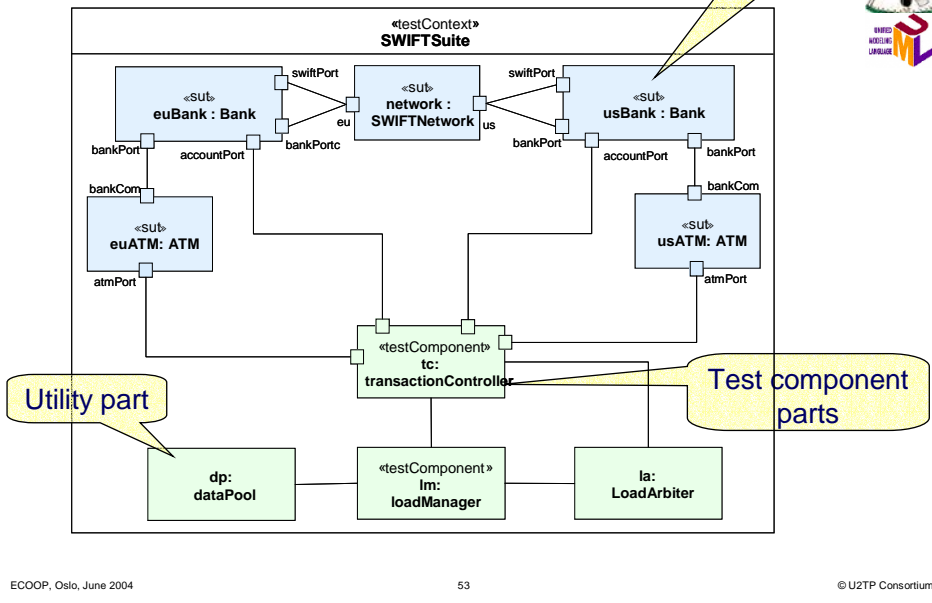
System-Integration-Level Test



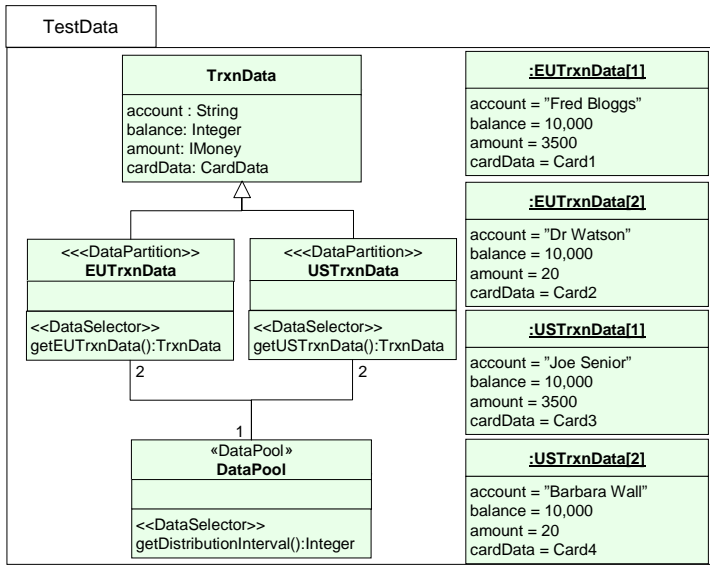
System-Integration-Level Tests



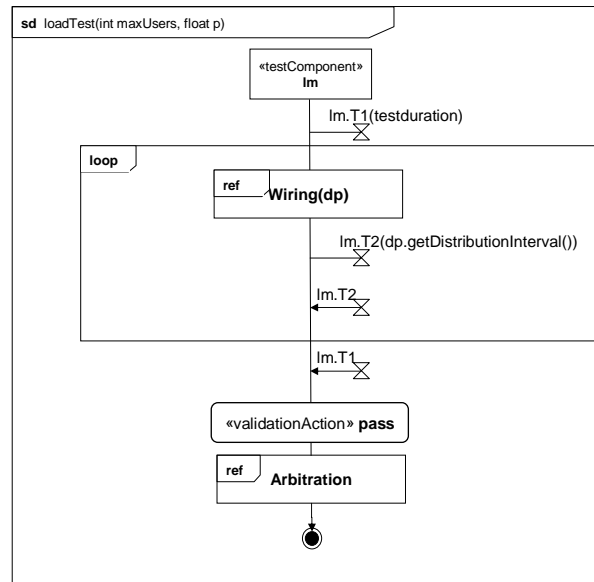
Test Configuration



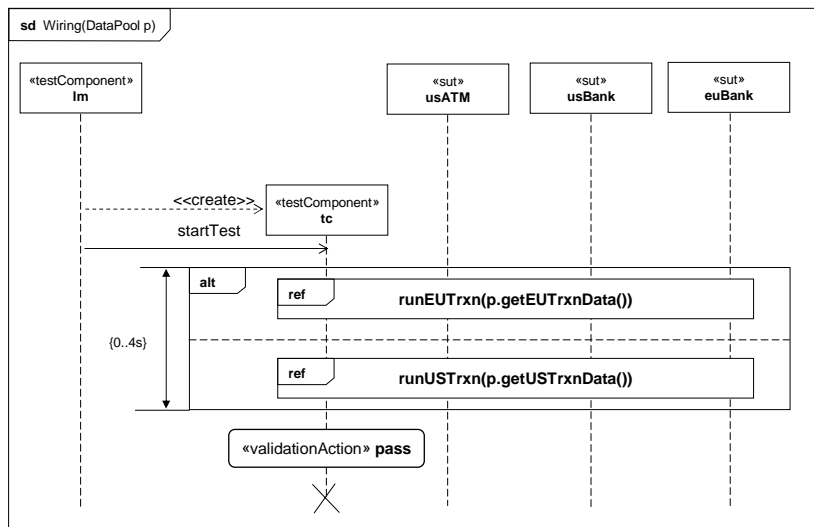
Test Data



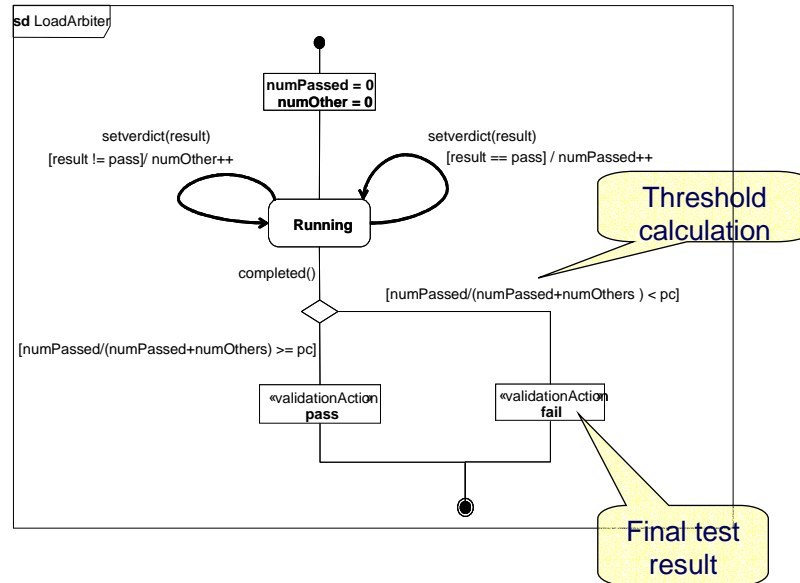
Load Test



Load Test Contd.



User-Defined Arbiter



ECOOP, Oslo, June 2004

57

© U2TP Consortium

Agenda

- Motivation
- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks

ECOOP, Oslo, June 2004

58

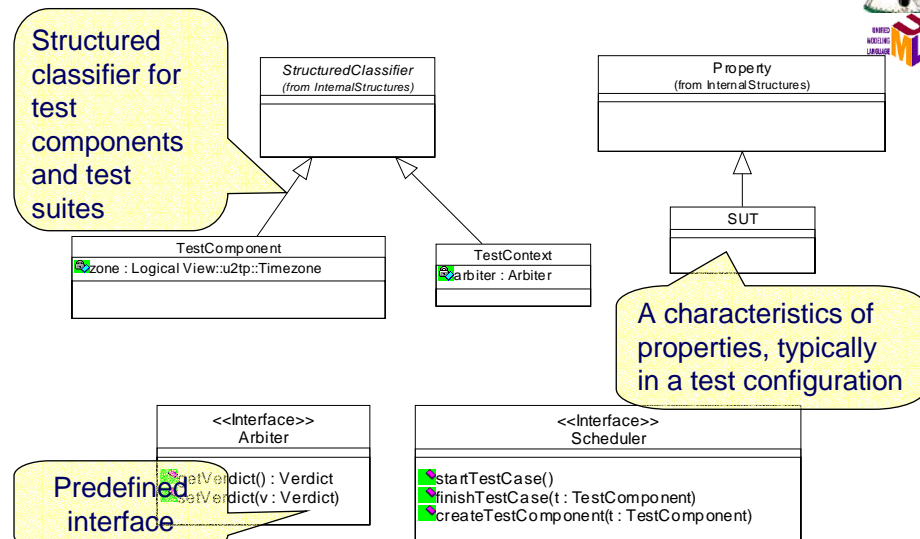
© U2TP Consortium

The U2TP Profile and Standalone Metamodel

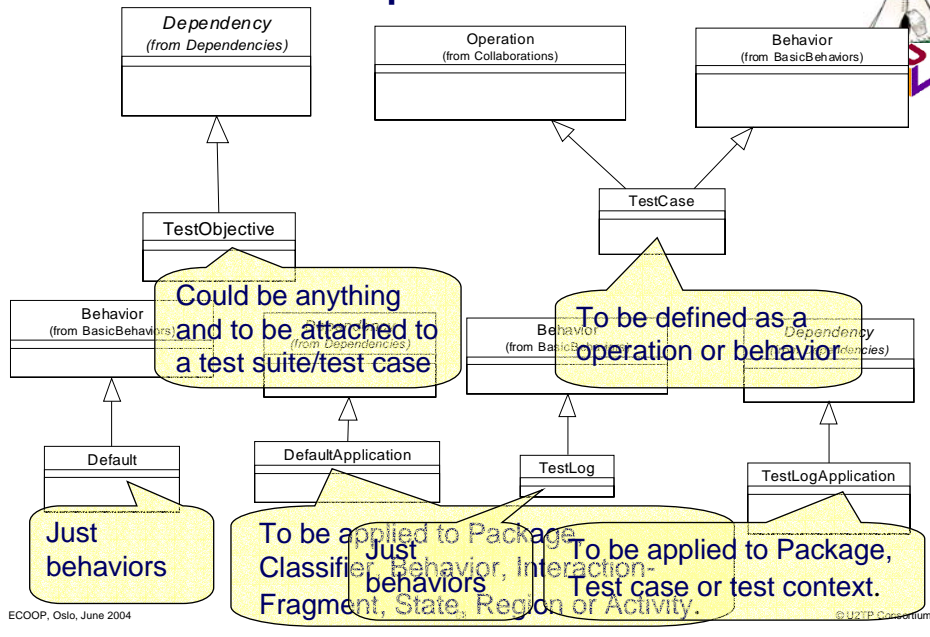
- Organization of concepts
 - Test Architecture
 - Test Behavior
 - Test Data
 - Time
- The UML-based profile
 - Used by UML tools to provide test specification via profiled UML elements.
- The MOF-based standalone metamodel
 - Used by MOF-based tools and repositories to manage and manipulate artifacts created using the profile.



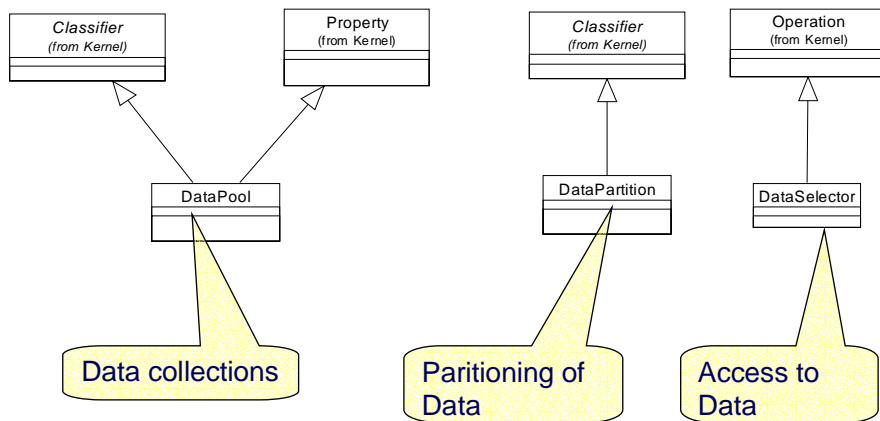
The Profile – Excerpt of Test Architecture



The Profile – Excerpt of Test Behaviour



The Profile – Excerpt of Test Data

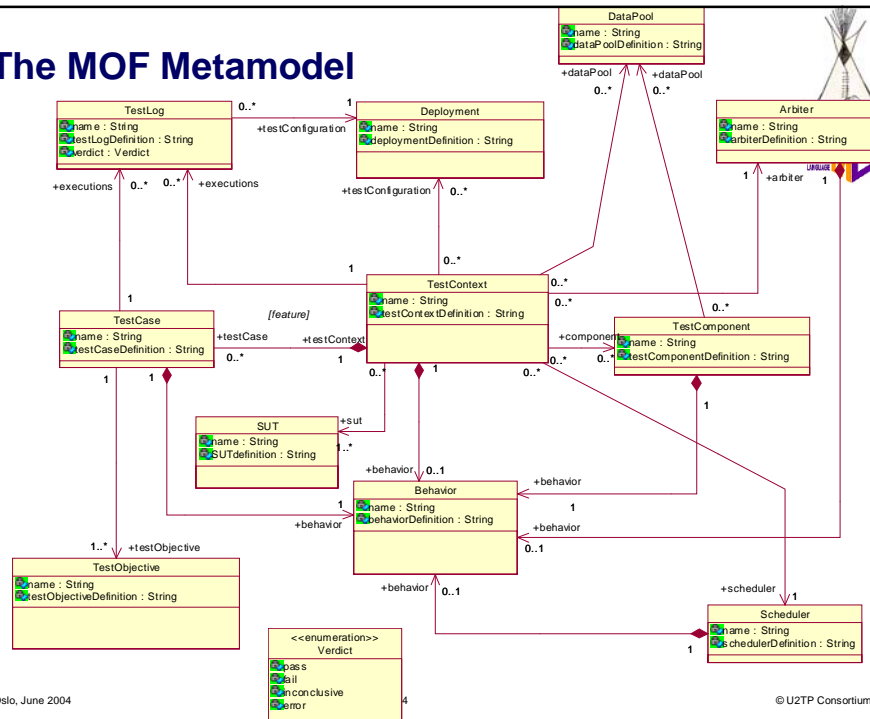


MOF Walkthrough

- Meta Object Facility
- Foundation for the OMG metadata architecture
 - Design and implement metamodels and models
 - Use UML classes for specification
 - Discover and manipulate metadata
 - Find and manage metadata repositories
 - Provides reflective meta-object interfaces for introspection and update of distributed metadata
 - Provides a MOF-to-IDL mapping to automate generation of concrete object interfaces for specific metamodels
 - Provides a MOF-to-XML mapping to automate generation of XML schema and documents



The MOF Metamodel



Agenda

- Motivation
- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks



Mappings

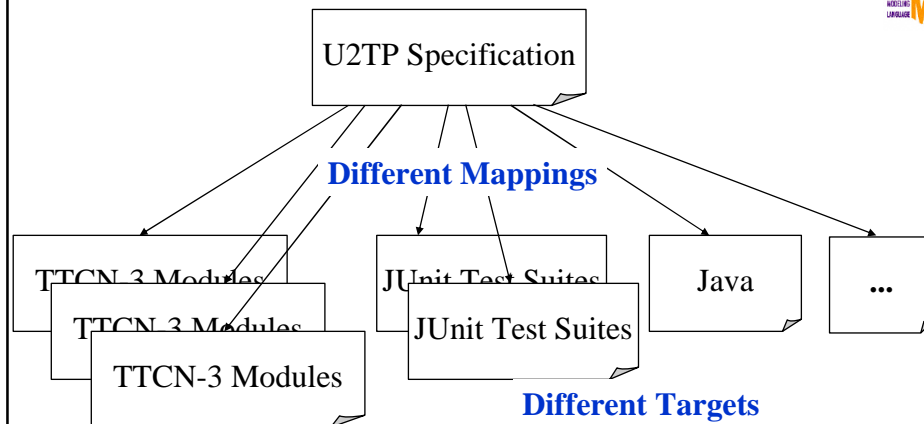
- To enable the direct execution of U2TP specifications
- To reuse existing test infrastructures

➤ Mappings to

- The JUnit test framework
 - An open source test technology for Java
 - Black-box tests on unit level
- The Testing and Test Control Notation, TTCN-3
 - A generic test technology by ETSI/ITU-T
 - Black-box/grey-box tests on unit, component, integration and system level



Mappings



➤ The mappings define possible, but not the only mappings

Principles of Mapping to JUnit



| Direct mapping | |
|-----------------------|--|
| Test control | <code>overload runTest</code> |
| Test case | <code>test methods in a test suite</code> |
| Test objective | <code>setName for test case, test suite</code> |
| Verdict | <code>the same except of inconclusive</code> |
| Arbiter | <code>built-in, can be redefined</code> |
| Validation action | <code>assert operations</code> |
| Data pool | <code>(static) data variables, ...</code> |

Principles of Mapping to JUnit



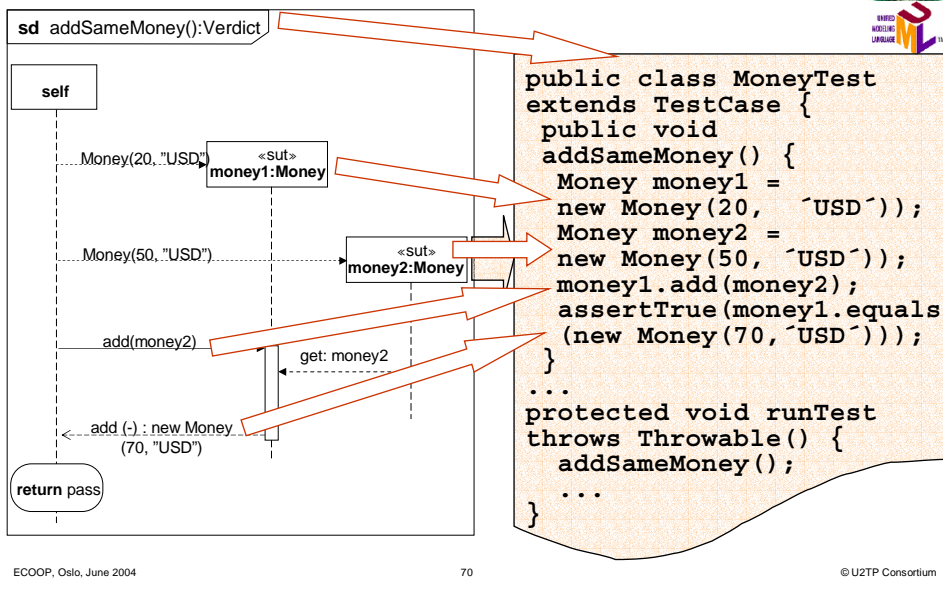
Implicit mapping

| | |
|-------------------------------------|-----------------------------|
| Stimulus, observation, coordination | as part of the test methods |
| SUT and utility part | any accessible Java class |

Not supported

| |
|--|
| Test configuration and test components |
| Wildcards and matching |
| Defaults |
| Test trace and log |
| Time concepts |
| Coding rules |

Example for Mapping to JUnit



Principles of Mapping to TTCN-3



| Direct mapping | |
|-------------------------------------|---|
| Test control | control of a test module |
| Test case | testcase |
| Stimulus, observation, coordination | communication to SUT and to test components |
| Defaults | altsteps and activate/deactivate operations |
| Wildcards and matching | template and pattern mechanisms |
| Test components | component types and instances |
| Test configuration | create, connect, map ... operations |

ECOOP, Oslo, June 2004

71

© U2TP Consortium

Principles of Mapping to TTCN-3



| Direct mapping | |
|----------------------------|--|
| Arbiter | user defined verdict handling (and user defined verdict type) |
| Validation action | (external) data function |
| Utility part and data pool | (external) constants or module parameters |
| Coding rule | encoding attribute |
| Timer | timer |
| Logging | log operation |

ECOOP, Oslo, June 2004

72

© U2TP Consortium

Principles of Mapping to TTCN-3



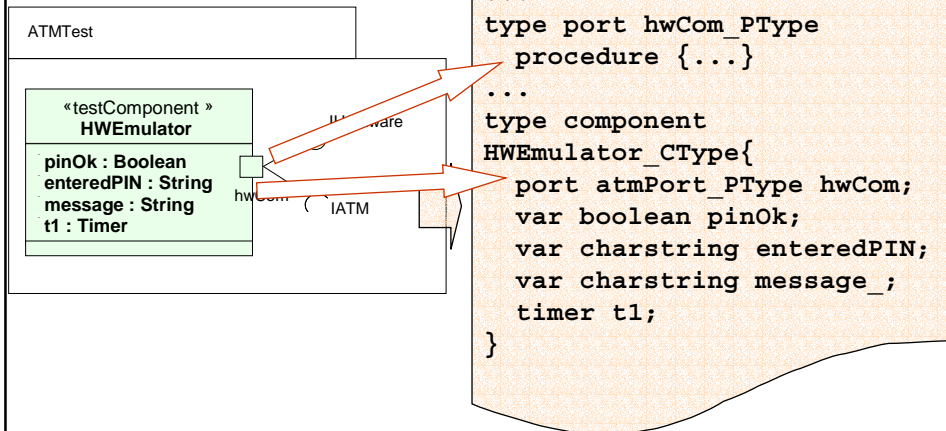
Implicit mapping

| | |
|----------------|--|
| Test objective | comment for a test case in a special attribute |
| SUT | characterized via the test system interface only |

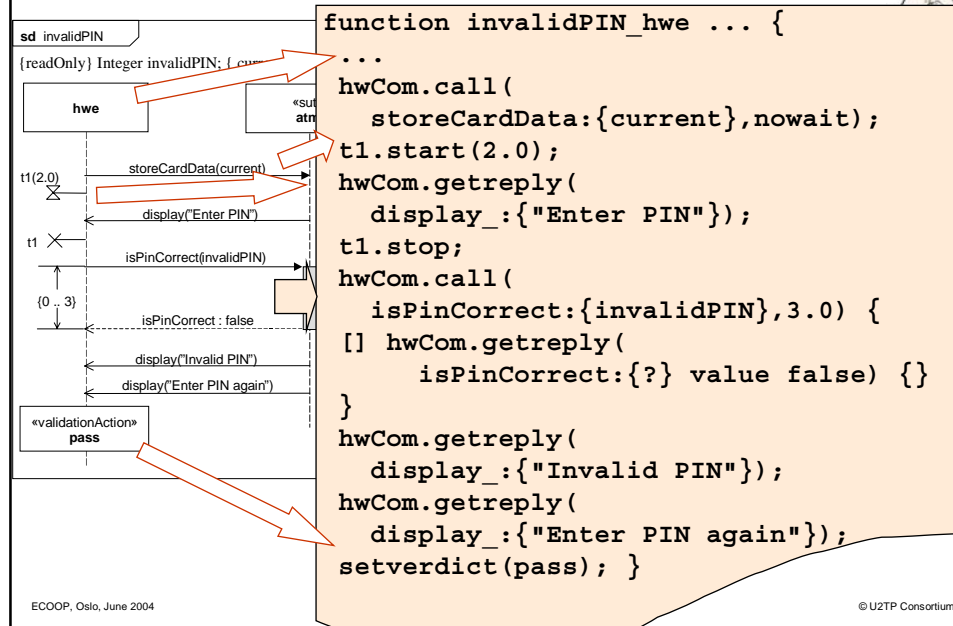
Not supported

Timezone
Test trace

Example for Mapping to TTCN-3



Example for Mapping to TTCN-3



Agenda

- Motivation
- Basics of Testing
- Overview of the UML Testing Profile
- Example Test Specifications
- The UML Profile and the MOF Model
- The Mappings
- Concluding Remarks

Implementations under Development

- Eclipse Project **Hyades** on an Open Source Trace and Test Framework
 - The test part is based on the U2TP standalone metamodel
- Telelogic **Tau G2**
 - The test part is based on the U2TP profile
- Microsoft **Visual Studio**
- ITEA Project on Advanced Test Methods and Tools **TTmedal**



Compliance Points

- 1. UML Profile for Testing:** a compliant implementation supports the UML profiling mechanism, the UML entities extended by the Testing Profile, and the stereotyped entities of the UML Testing Profile.
 - 2. MOF-based Metamodel for Testing:** the compliant implementation supports all of the entities in the MOF-based metamodel.
 - 3. Notation:** If graphical notation is used, the compliant implementation recognizably supports the notation defined by the Testing Profile specification.
 - 4. XMI/DTD:** An XMI compliant implementation of the Testing Profile and/or MOF metamodel provides the UML XMI exchange mechanism.
 - 5. Static Requirements:** The compliant implementation checks the specified constraints automatically.
- **Compliance requires meeting 1 and/or 2. Points 3-5 are optional and can be claimed in any combination.**



At the End: a Standardized Testing Profile



- One test notation for many testing applications
 - Universally understood syntax and operational semantics
 - Off-the-shelf tools
 - Cheaper education and training costs
 - Exchange and reuse of test suites
 - Easier maintenance of test suites
- Transparency for the test process, increase of the objectiveness of tests and comparability of test results
- Direct support for test design
- Integration in the system development process

Summary



- UML Testing Profile provides specification means for test artifacts of systems from various domains
 - Enhances UML with concepts like test configuration, test components, SUT, verdict and default
 - Seamlessly integrates into UML: being based on UML metamodel, using UML syntax
- Adopted at OMG Technical Meeting, Paris, June 2003

<http://www.fokus.fraunhofer.de/u2tp/>



Thank you for your attention !

Any further questions ?

© U2TP Consortium