



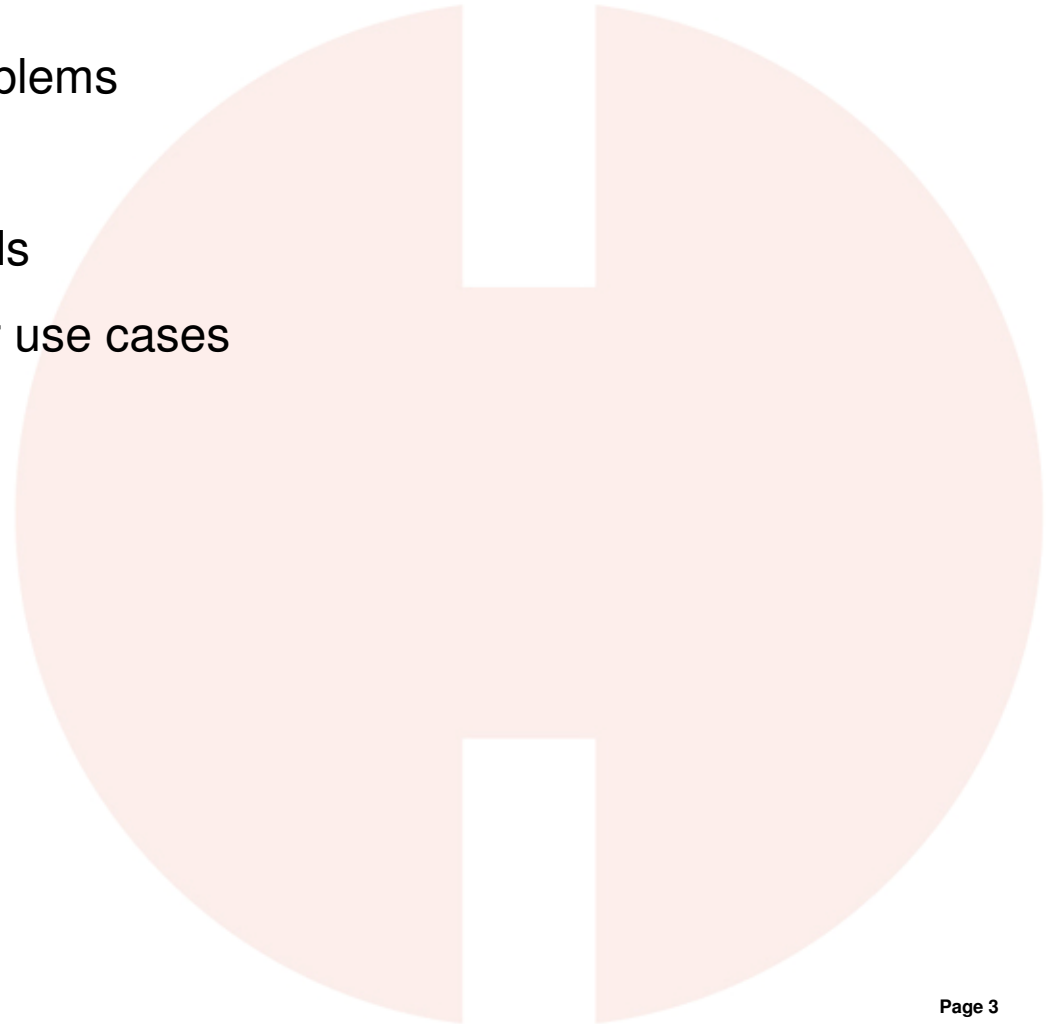
From Use Cases to Test Cases

Step-by-step approach to ensure the quality of specifications and to derive test cases based on a use case model

- HOOD Group
 - Experts in Requirements
 - Keltenring 7, D-82041 Oberhaching
 - <http://www.HOOD-Group.com>
 - 45 employees
 - Customer industries: Automotive, Health, Defense,..
- Dr. Rudolf Hauber
 - UML/SysML and technology consultant
 - Rudolf.Hauber@HOOD-Group.com

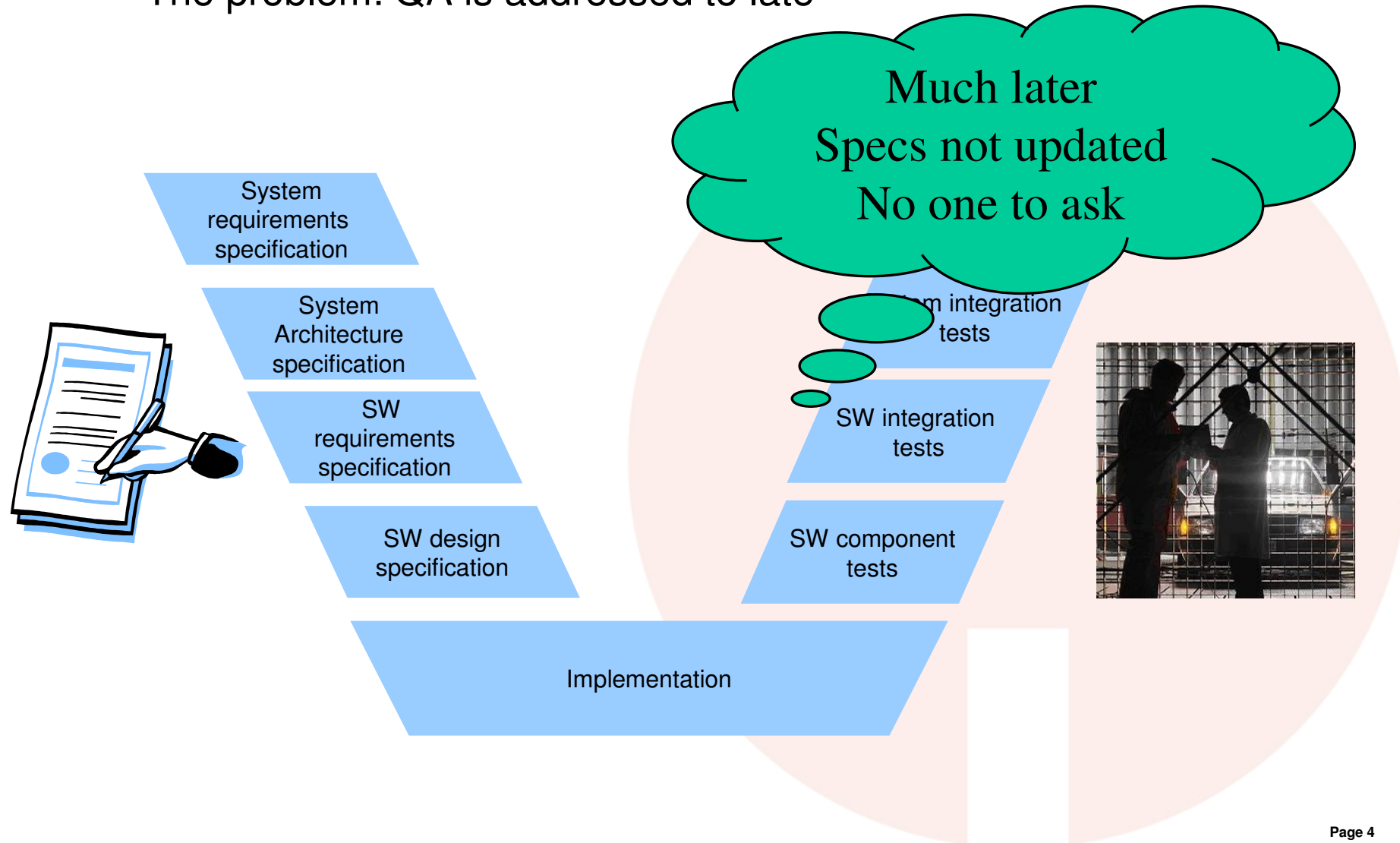
Content

- Why so much testing problems
- Use Case concept
- Simulation aims and levels
- Simulation techniques for use cases
- Testing use cases



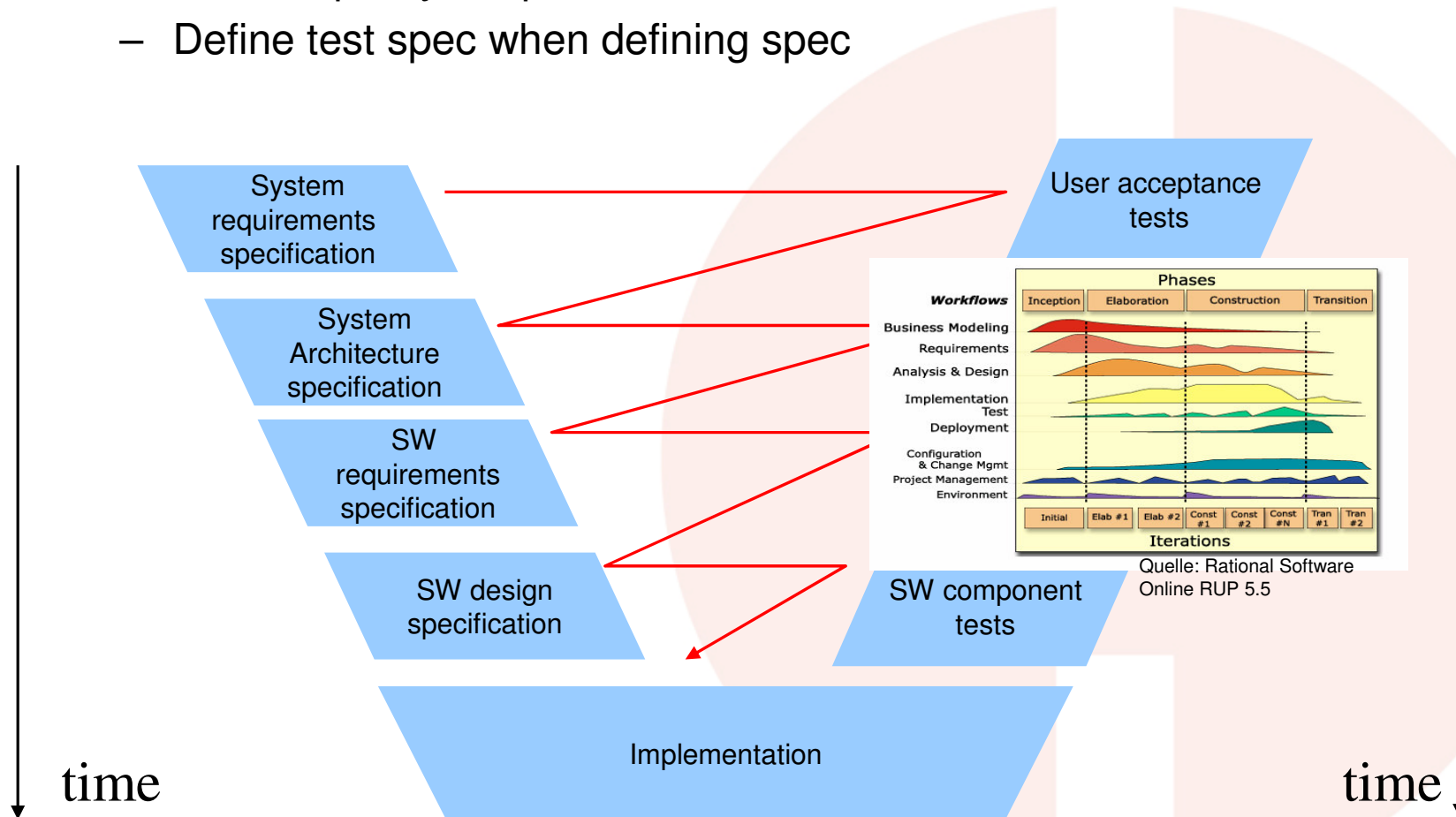
How to get test requirements?

- The problem: QA is addressed to late



Getting test requirements

- How it should be
 - Ensure quality of spec
 - Define test spec when defining spec

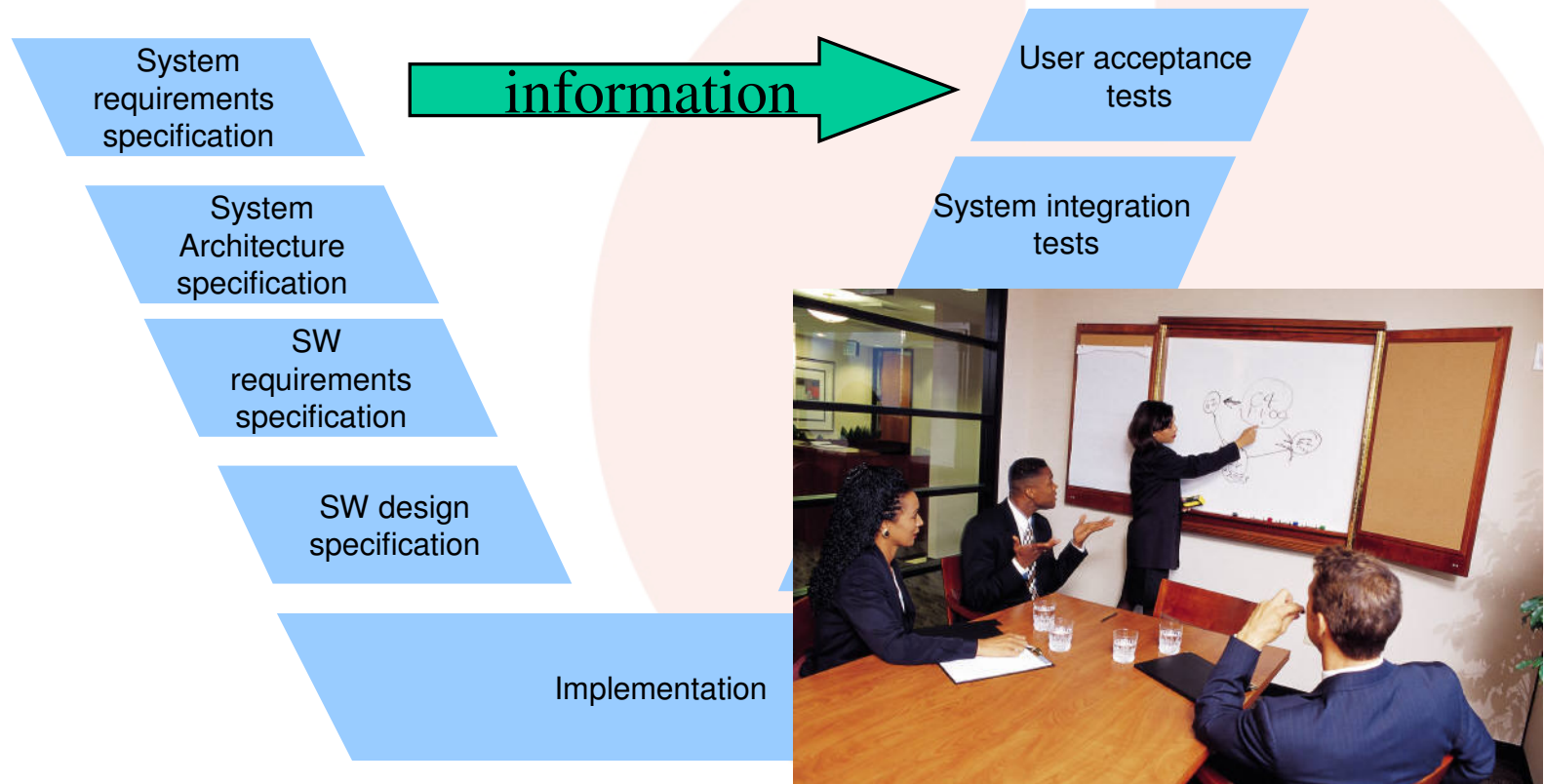


How to get test requirements?

© HOOD Group 2007

www.HOOD-Group.com

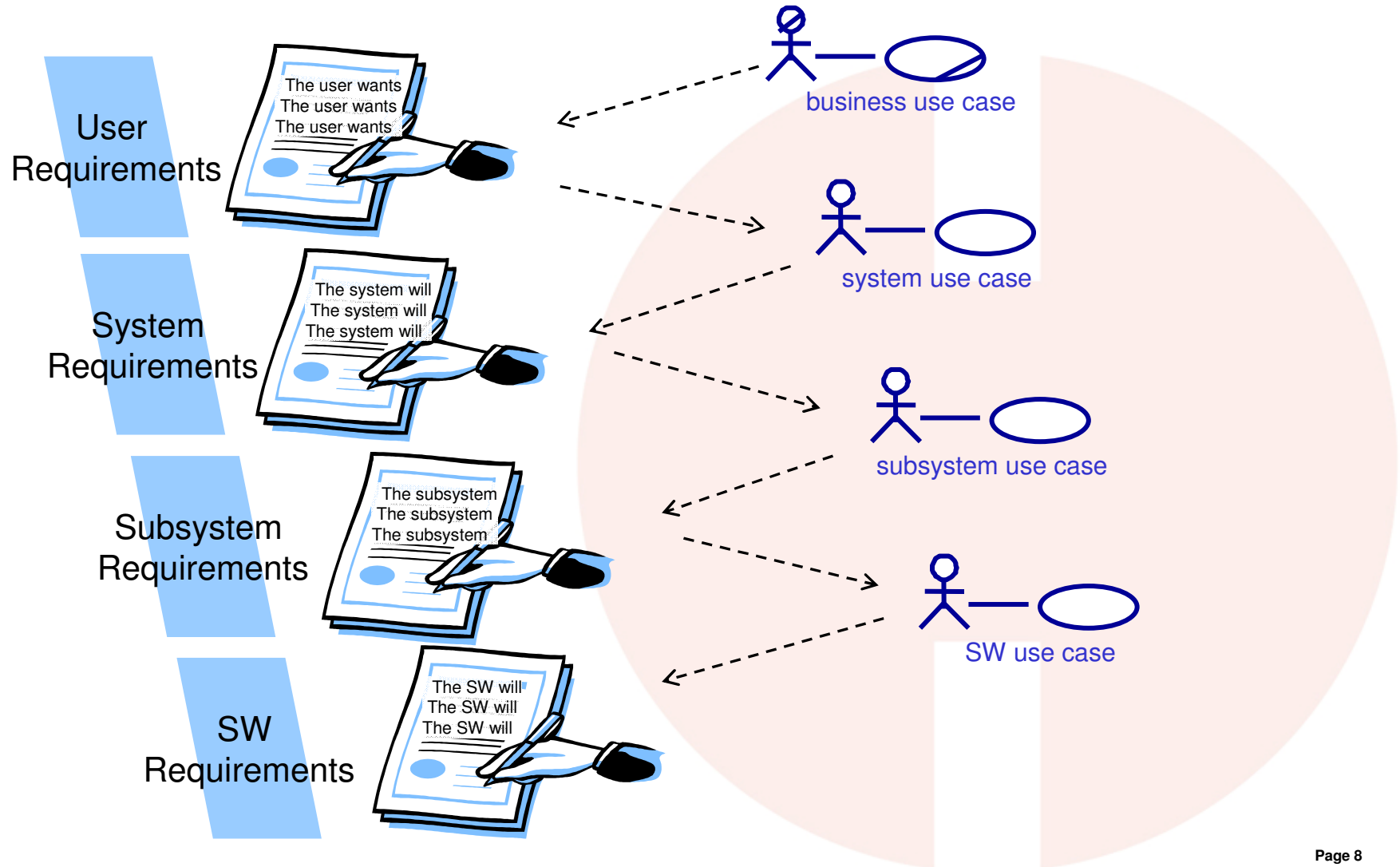
- The means we can use
 - Provide information
 - Consider all stakeholders
 - Do it in time



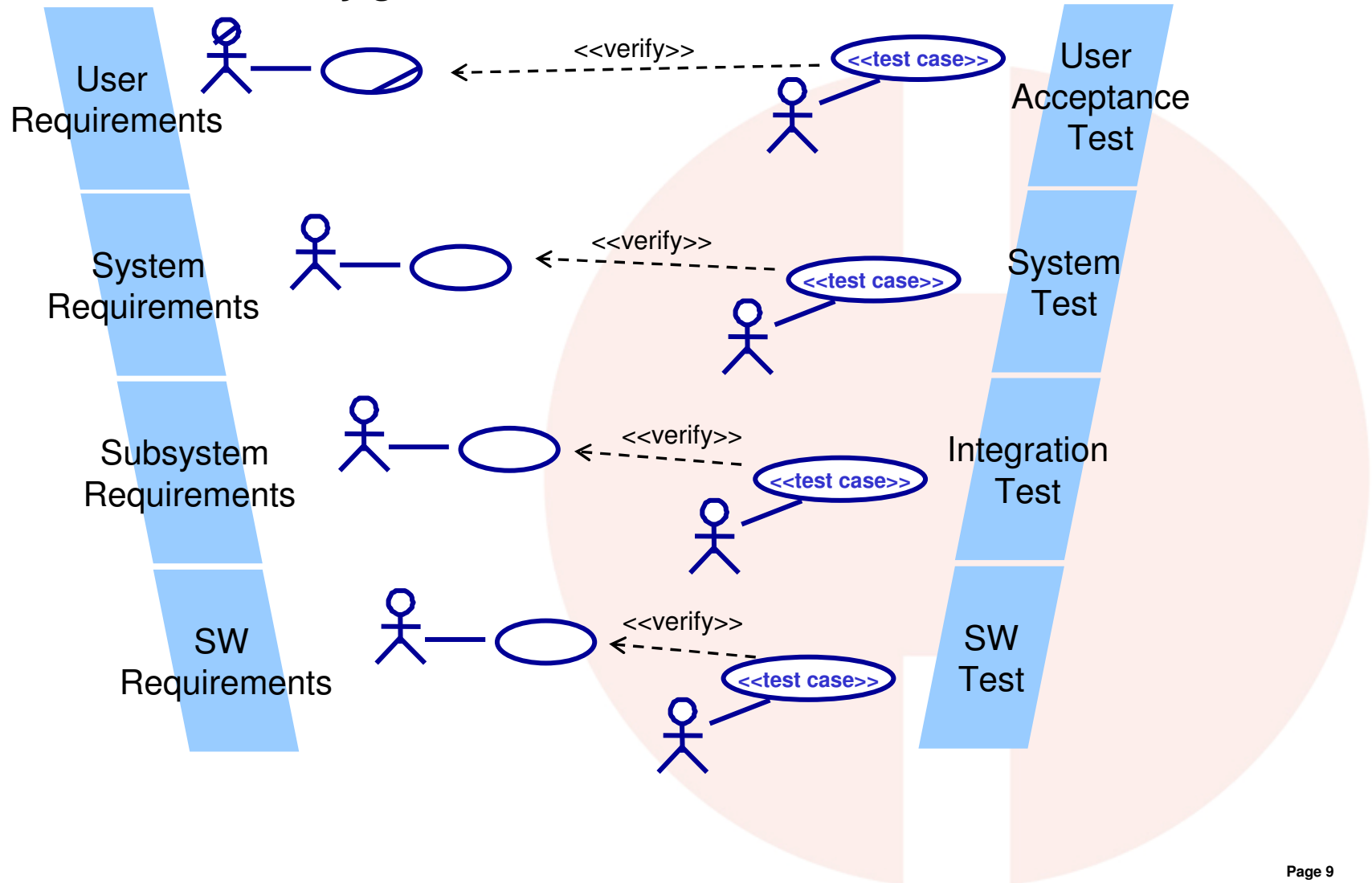
Content

- Why so much testing problems
- Use Case concept
- Simulation aims and levels
- Simulation techniques for use cases
- Testing use cases

Use Cases are a very good tool to derive requirements at all stage!



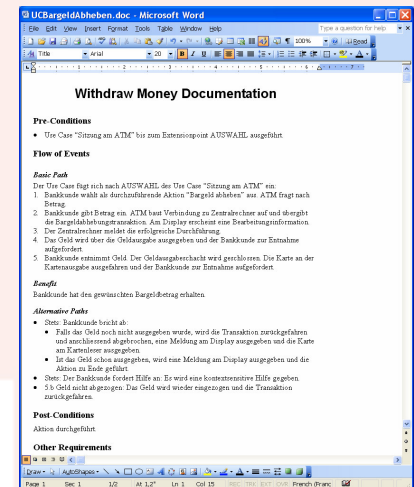
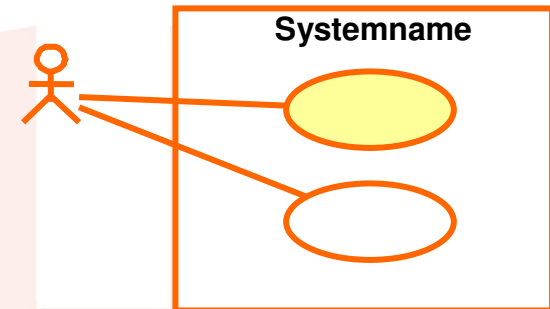
Use Cases are a very good base for verification and validation



- Use Case Diagram
 - Define scope and context
 - Requirements overview
 - Communications instrument to stakeholders

- Use Case description
 - From user perspective
 - Detailed flow of interaction
 - pre-/post-condition, constraints
 - „contract" between user and developer

- Additional information (if needed)

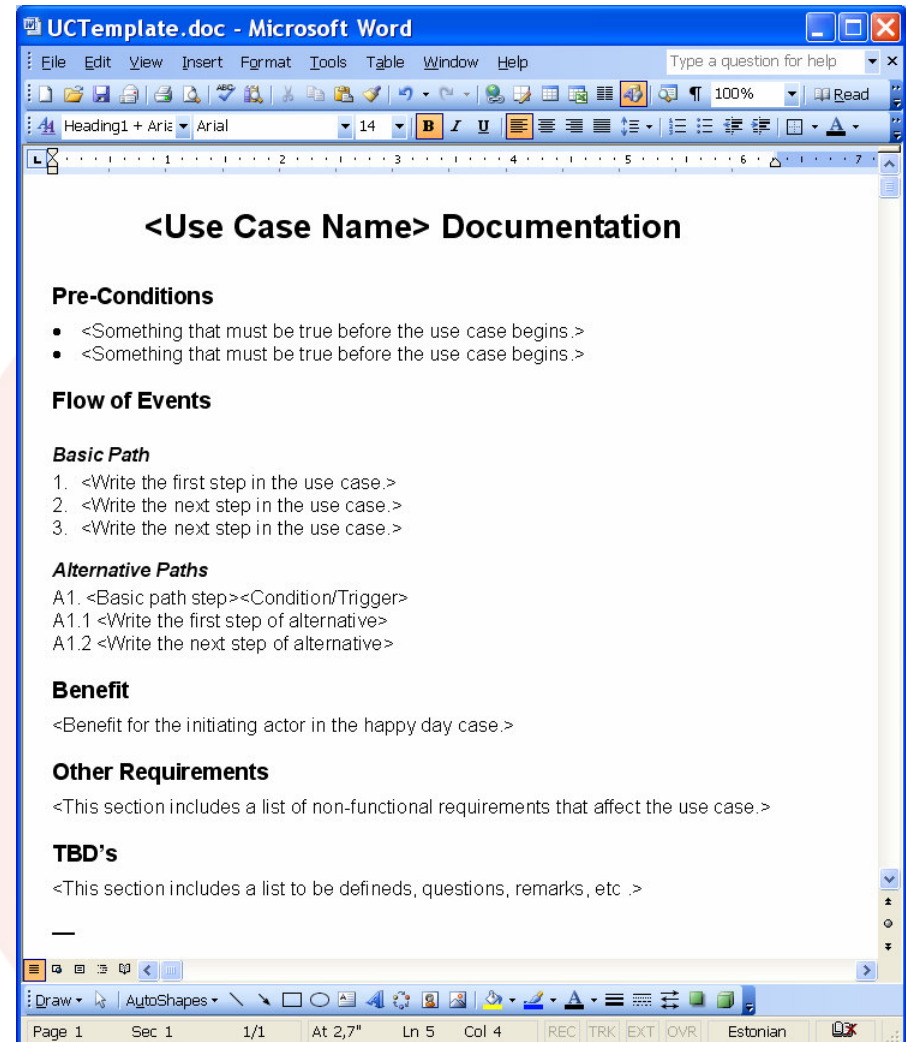


Use Case description

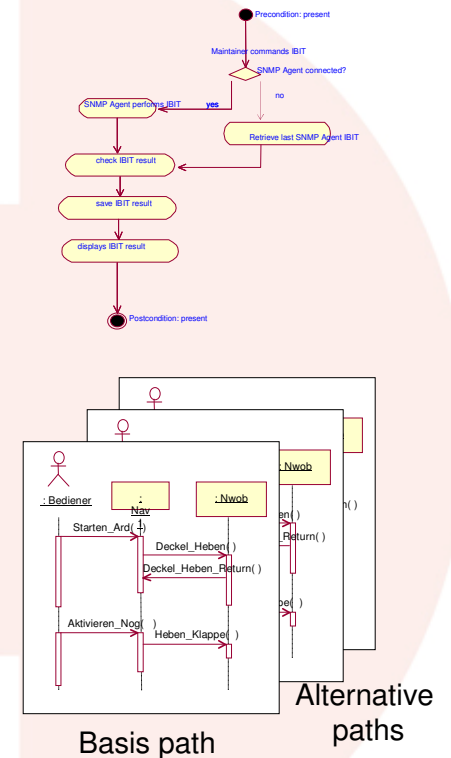
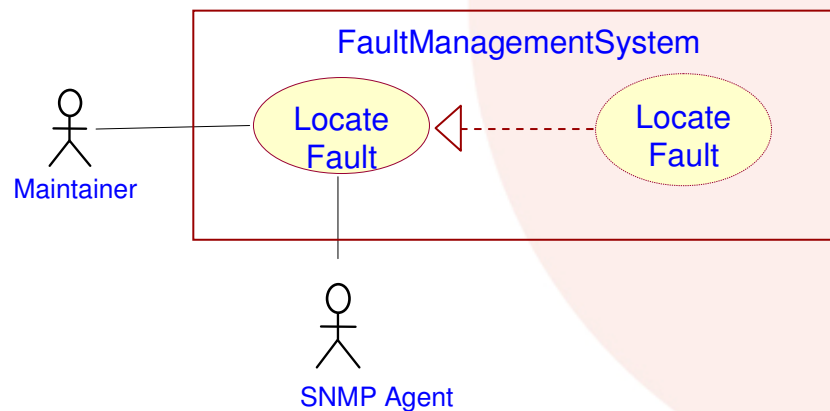
© HOOD Group 2007

www.HOOD-Group.com

- Use Case description is Use Case *Spec*
- Same rule as for *Requirement specification*
 - Clear
 - Precise, unambiguous
 - Defined terms (glossary)
 - verifiable
- Template-based
 - semi-formal
 - uniform



- For use case simulation you need modelling
- Modelling alternatives:
 - Black-Box
 - Individual flows using sequence diagrams
 - Complete use case behaviour using state machines or activities
 - White-Box
 - Use Case Realization using sequence diagrams

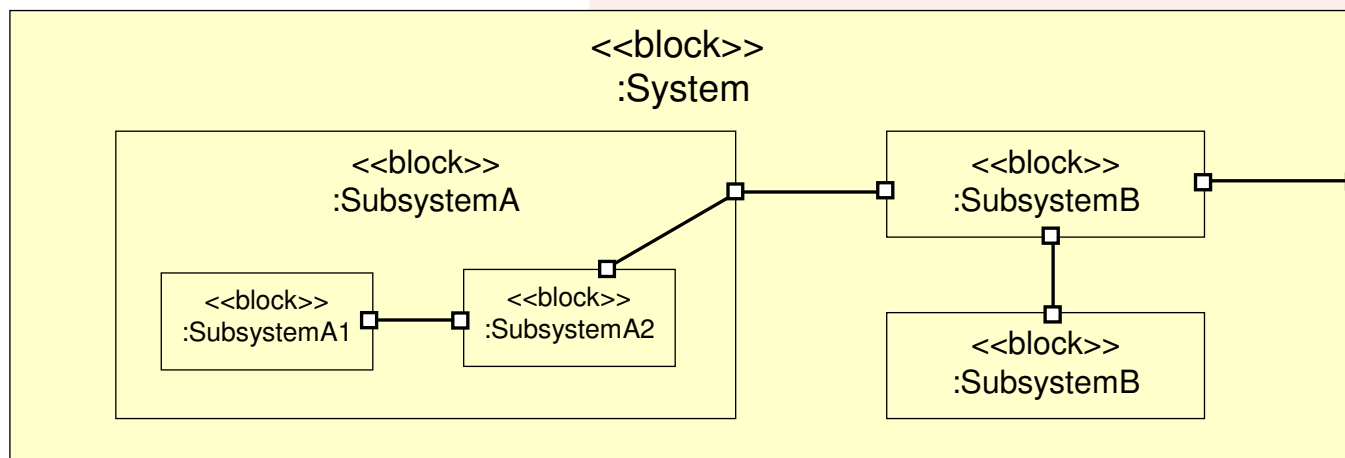


Content

- Why so much testing problems
- Use Case concept
- Simulation aims and levels
- Simulation techniques for use cases
- Testing use cases

- Aims: requirements simulation for
 - Completeness of requirements
 - Forgotten alternatives, missing preconditions, missing use case
 - Consistency of requirements
 - Ambiguous alternatives and preconditions, conflicting use cases
- Simulation is model based
 - Failures in description model transition can not be recognized!
 - Modelling must be coached and reviewed

- Simulation of functional requirements/flows on multiple levels
 - Black-Box Simulation of each use cases
 - Simulation of use case dependencies
 - White-Box simulation of each use case realisation
- Performance simulation
- For large systems: simulation of refinements for each stage recursively

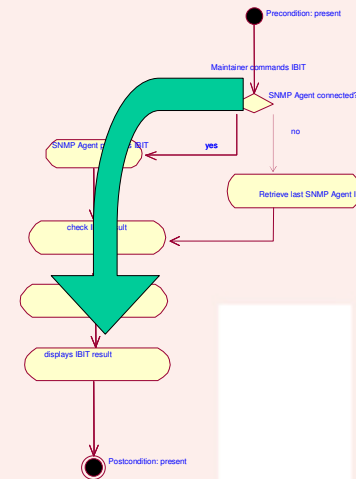
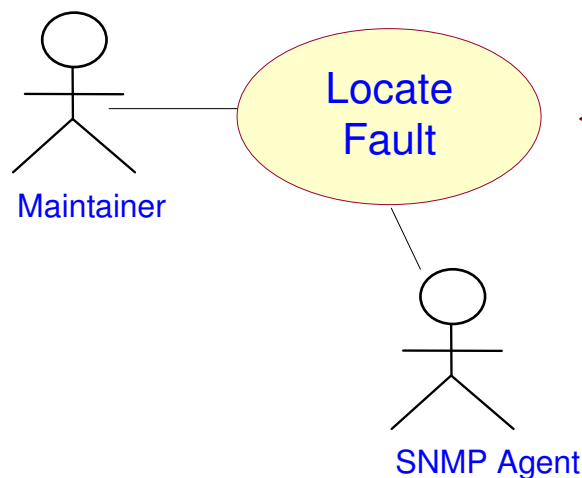


Content

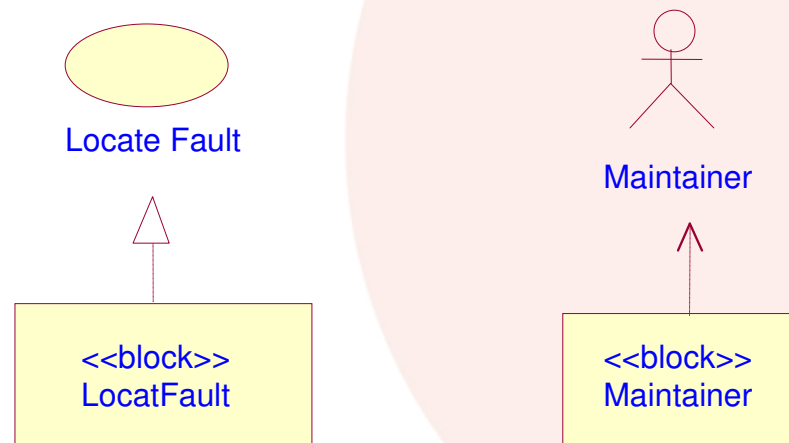
- Why so much testing problems
- Use Case concept
- Simulation aims and levels
- Simulation techniques for use cases
 - Black-Box Simulation
 - Simulation of use case dependencies
 - White-Box simulation
- Testing use cases

Use Cases Black-Box Simulation

- Aims:
 - Is the use case task correctly understood?
 - Are the use case correct?
 - Is the use case description consistent with existing interfaces?
- based on use case state machines or activities
- Simulation of concrete scenarios by injecting external event chains (e.g. from ICD)

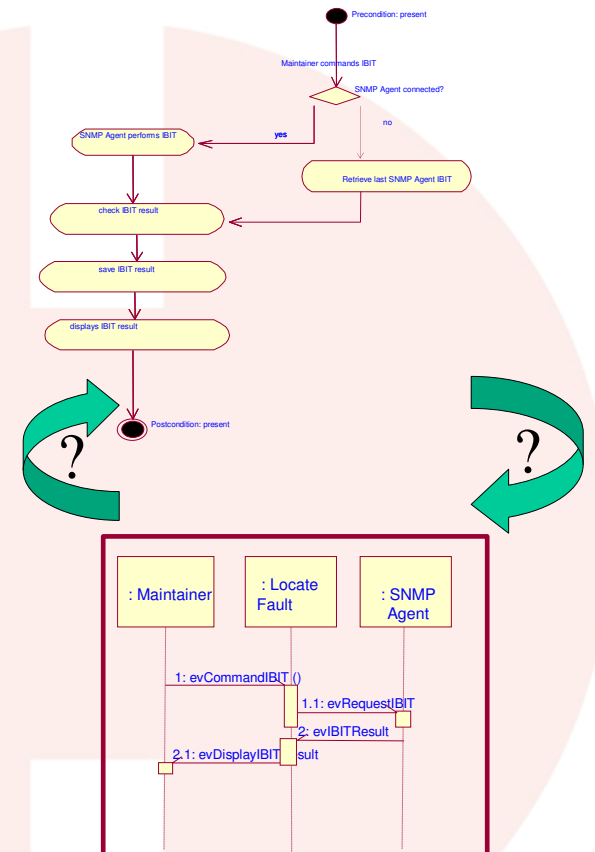


- Precondition
 - Based on UML 2 Tool (z.B. Rose RT)
 - Use Case modeled as UML 2 Structured Classes/SysML blocks
 - Use Case modeled als State Machine/activities
 - Actors modeled as UML 2 Structured Classes/SysML blocks, too

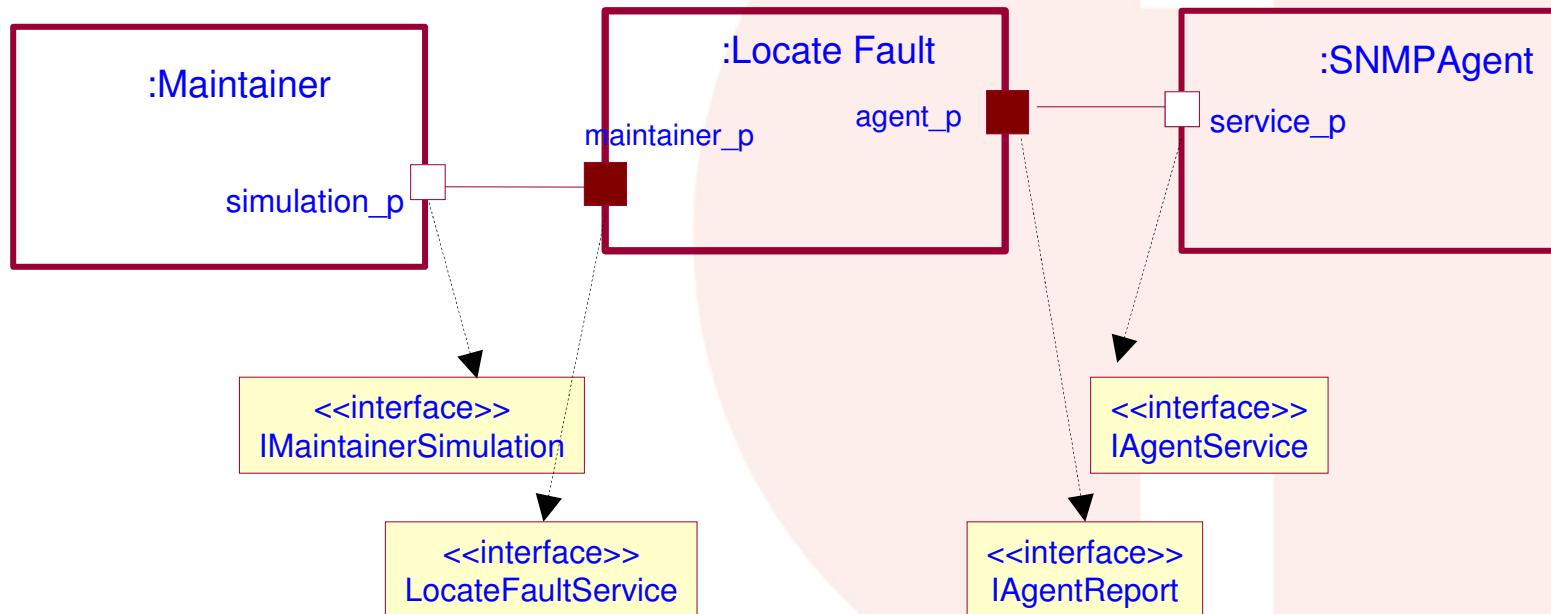


Use Cases Black-Box Simulation

- Tool based verification
 - Check use case’s state machine
 - dead ends?
 - clear decisions?
 - missing paths?
 - Check against ICD Communication
 - When to send and receive messages?
 - Which messages?
 - Conform the use case flows to the actor’s interfaces?

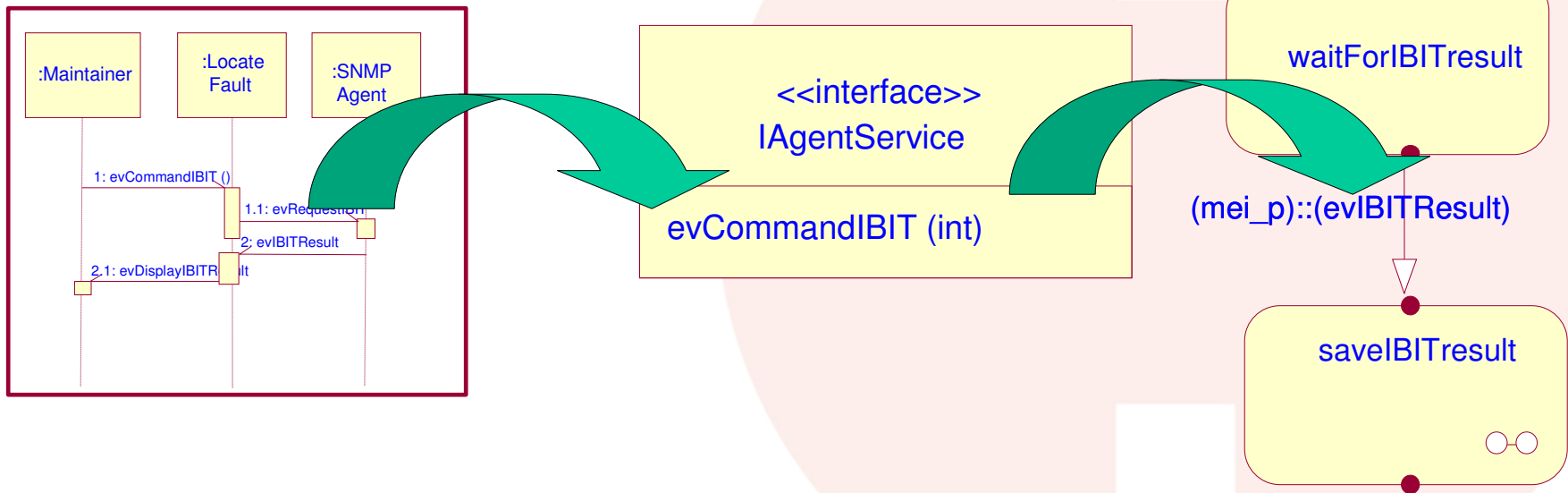


- Modeling details
 - Define a (service) port for each actor (actor interface)
 - Define ports for the use case simulation capsule
 - Define a simulation port for the actor capsule



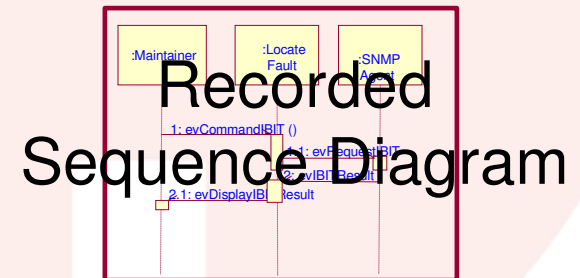
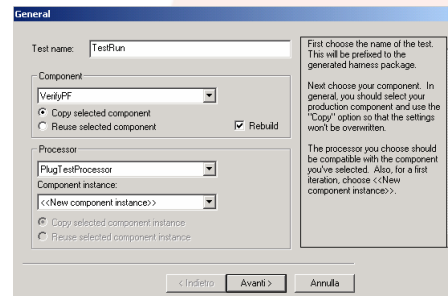
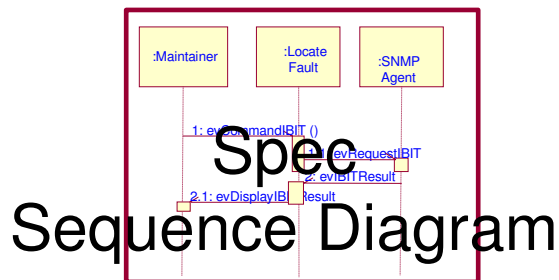
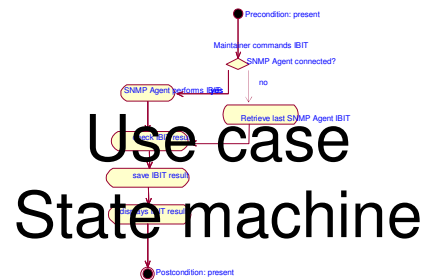
Event Approach

- Define signals for each incoming/outgoing ICD messages
- Define trigger events for event triggered transition

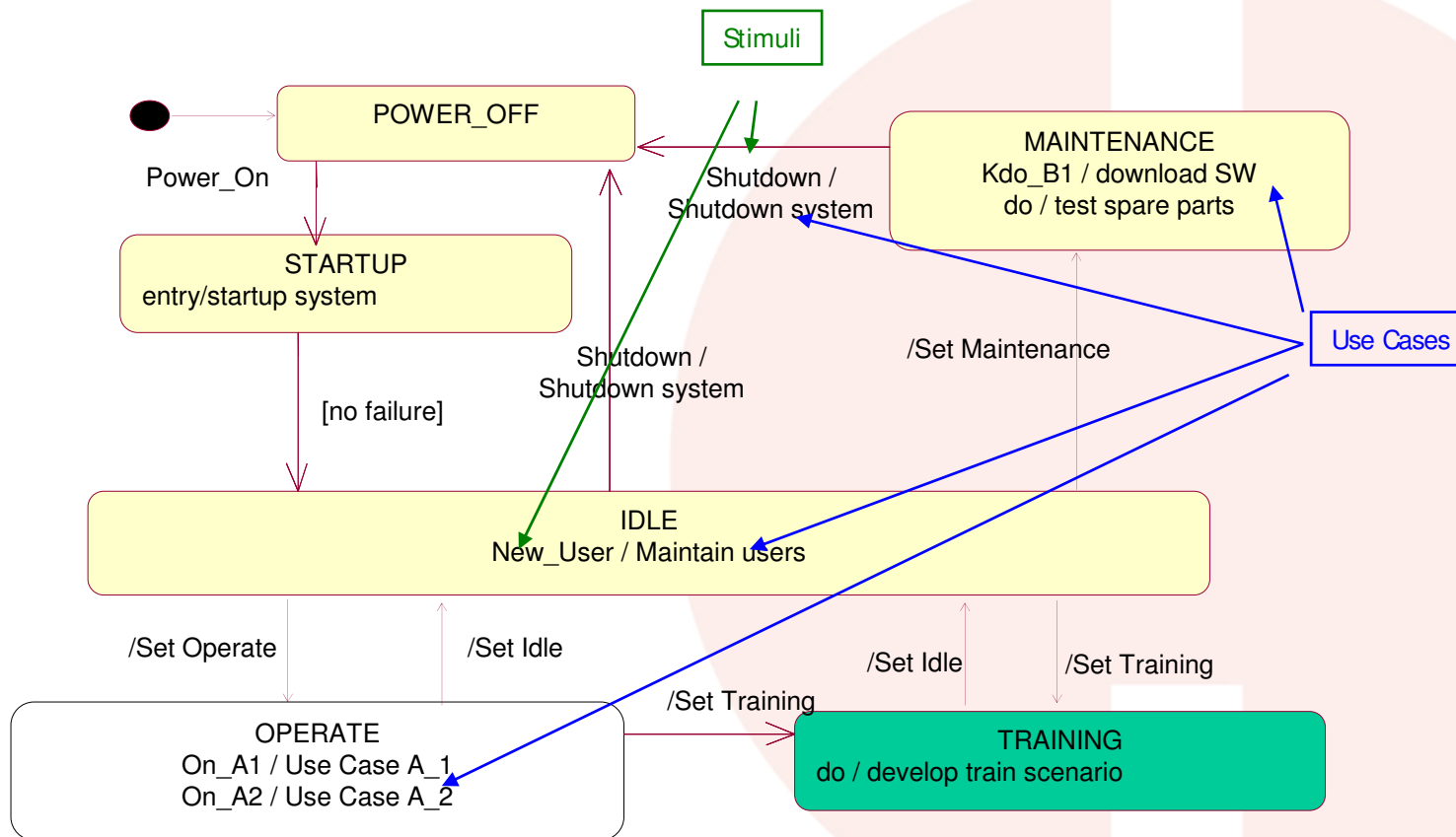


Use Cases Black-Box Simulation

- Tool based verification
 - Use Case State machine or activity is generated
 - Actors are stubed
 - Automated verification of recorded sequences against spec sequence diagrams
 - Needs a run time engine/action language



Based on state machine / activity model of use case dependencies

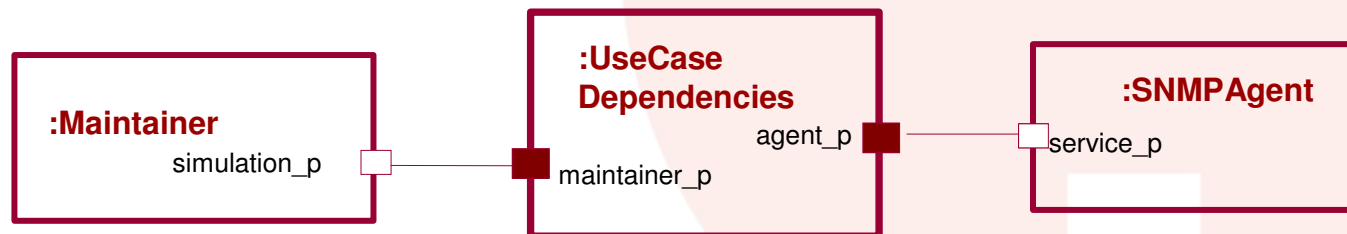


Simulation of use case dependencies

© HOOD Group 2007

www.HOOD-Group.com

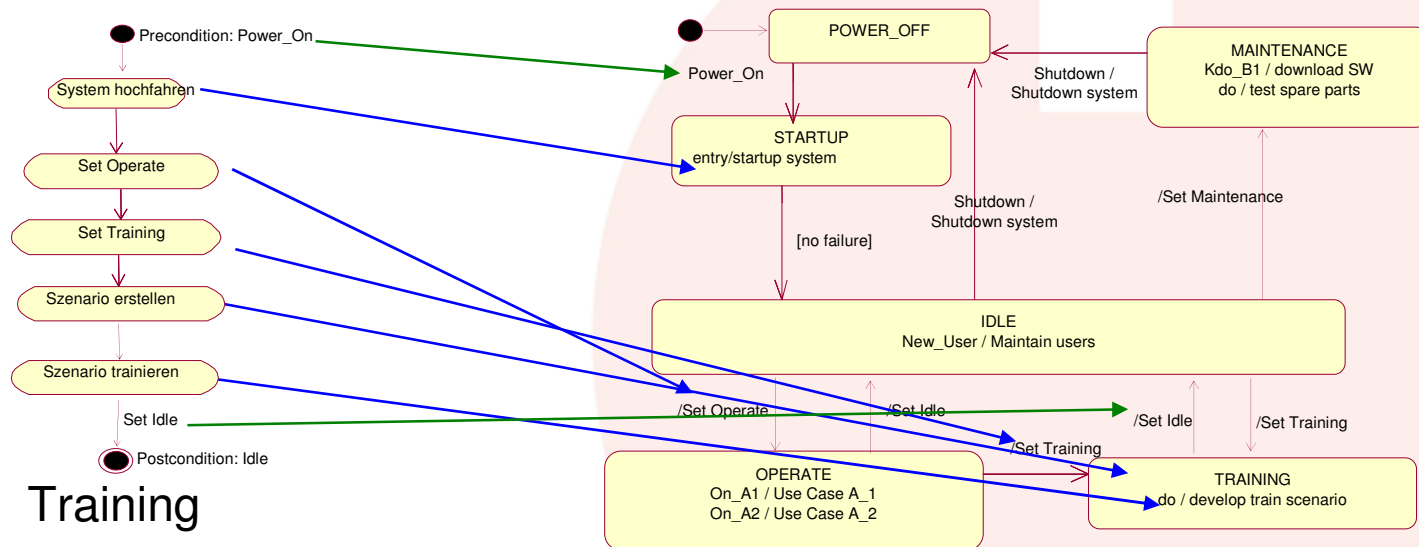
- Aims:
 - Is the problem space correctly understood?
 - Are use cases missing, exists inconsistent/ duplicated functionality?
 - Are the use cases consistent with system states & modes?
 - Fitting the use cases together?
- Simulation by concrete „Usage Profiles“ by injecting use case stimuli-chains
- Technique:
 - Same as before, but 1 structured Class/«block» for all use cases/the system



Simulation of use case dependencies

„Walk-through“ the usage profiles

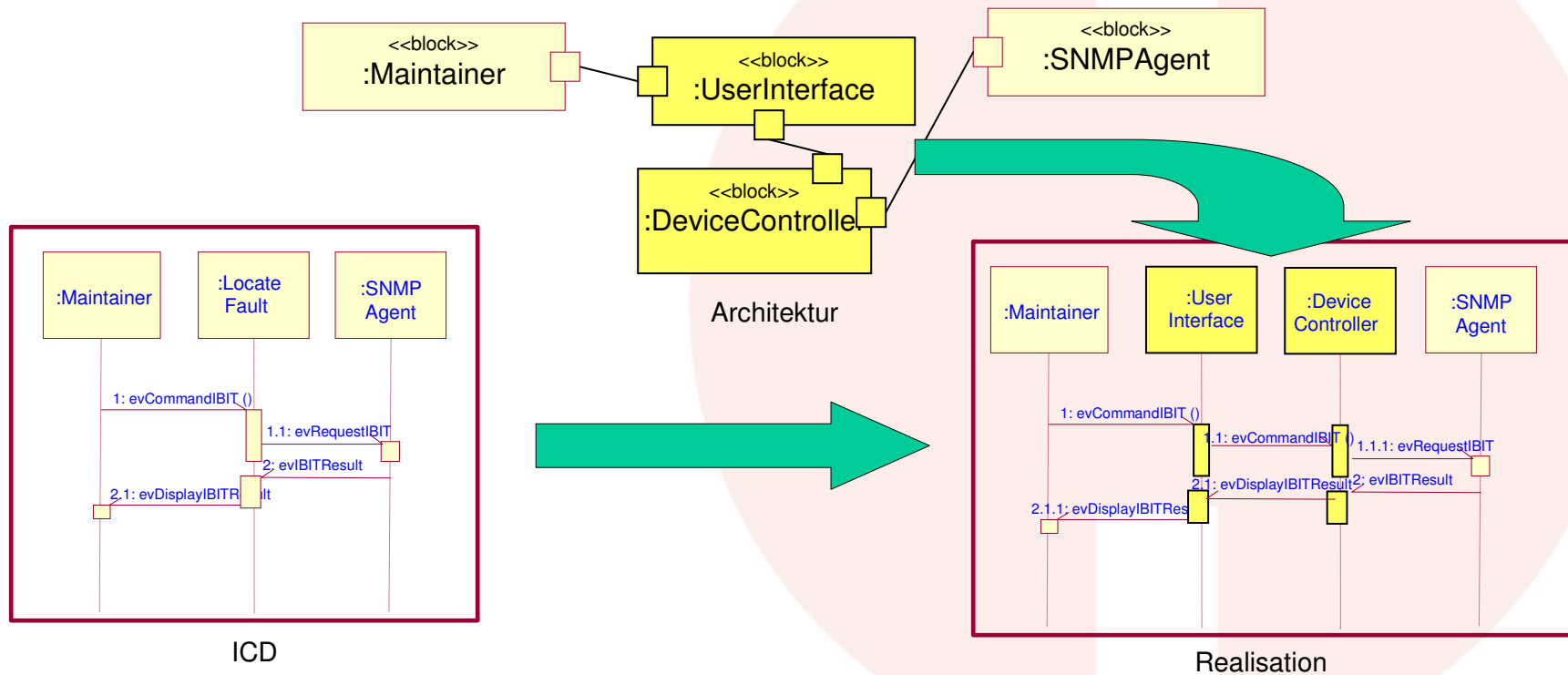
- (typically) manually
- test pre- and postconditions of the use cases



- Aim:
 - Is the use case appropriate realized?
 - Are all use case steps realized?
 - Are all subsystems appropriate designed?
(coherence and encapsulation)
 - Make the subsystem responsibilities sense?
 - Are the communication paths clearly laid out ?
- Simulation of the use case realizations
- Preconditions:
 - Architectural elements identified
 - Robustness Analysis performed

Use Case White-Box Simulation

- Simulation of Use Case Realisation against sequence diagrams
- Replace Use Case (Black-Box Simulation) block with **Realisation**

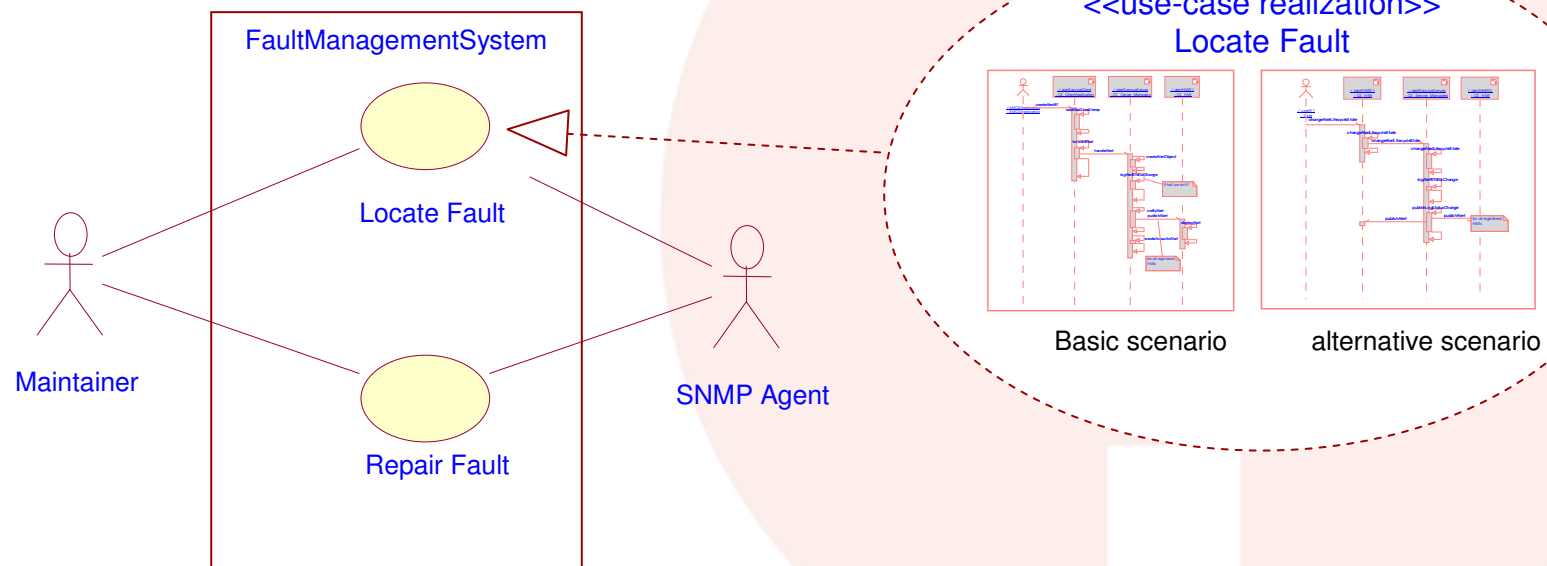


Content

- Why so much testing problems
- Use Case concept
- Simulation aims and levels
- Simulation techniques for use cases
- Testing use cases

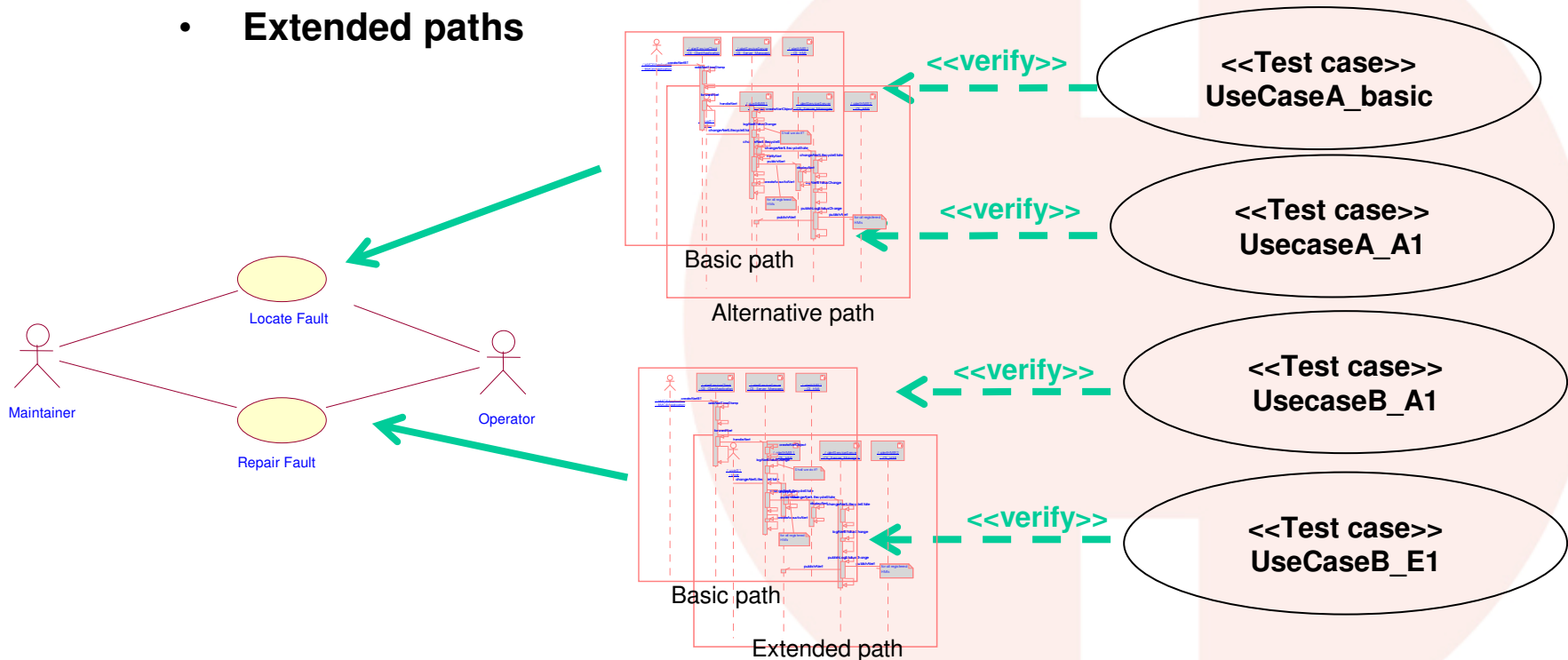
Concept

- Use cases consists of use case scenarios, which are much like „test scripts”
- Use case paths are modeled as sequence diagrams



Deriving the test specification: Stage 1 Initial test cases

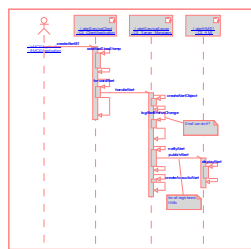
- Define a test case for each sequence diagram of each use case
 - Basic path
 - Alternative paths
 - Extended paths



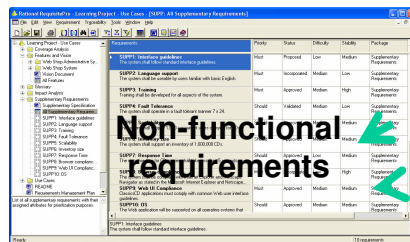
Testing use cases

Deriving the test specification: Stage 1 Initial test cases

- Check which non-functional requirements can be tested within a test case
- Define test cases for the rest of the non-functional requirements



Basic path



ID	Priority	Value	Priority	Category	Package
SRN21	High	Approved	Low	Medium	Systemwide Requirements
SRN22	High	Approved	Medium	High	Systemwide Requirements
SRN23	High	Approved	Low	Medium	Systemwide Requirements
SRN24	High	Approved	Medium	High	Systemwide Requirements
SRN25	High	Approved	Low	Medium	Systemwide Requirements
SRN26	High	Approved	Medium	High	Systemwide Requirements
SRN27	High	Approved	Low	Medium	Systemwide Requirements
SRN28	High	Approved	Medium	High	Systemwide Requirements
SRN29	High	Approved	Low	Medium	Systemwide Requirements
SRN30	High	Approved	Medium	High	Systemwide Requirements

Non-functional requirements

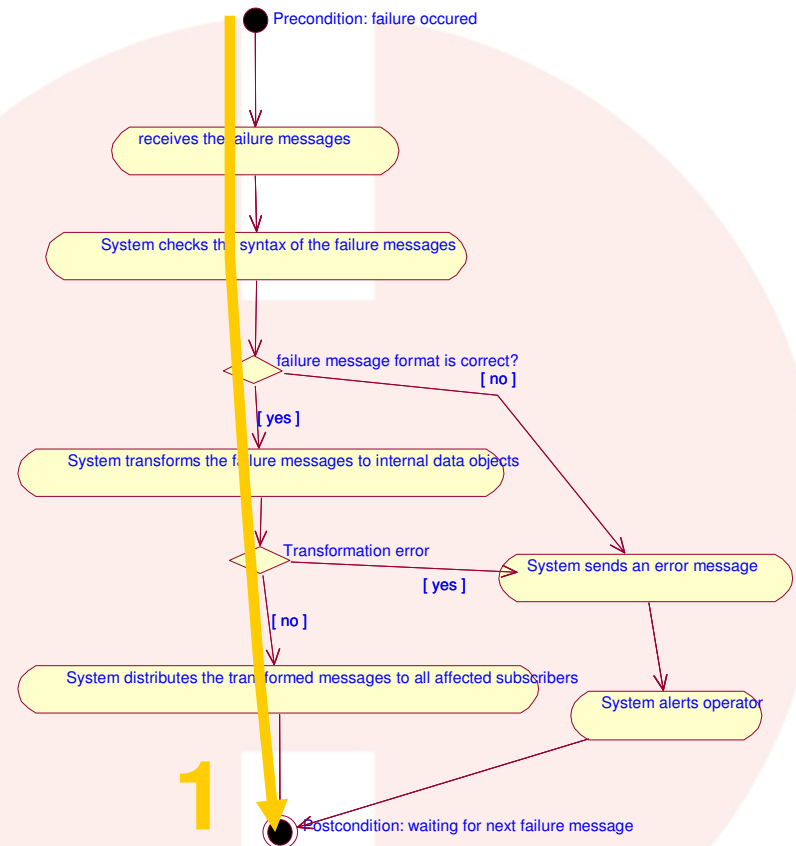
**<<Test case>>
Use case_basic**

**<<Test case>>
SRN21_test**



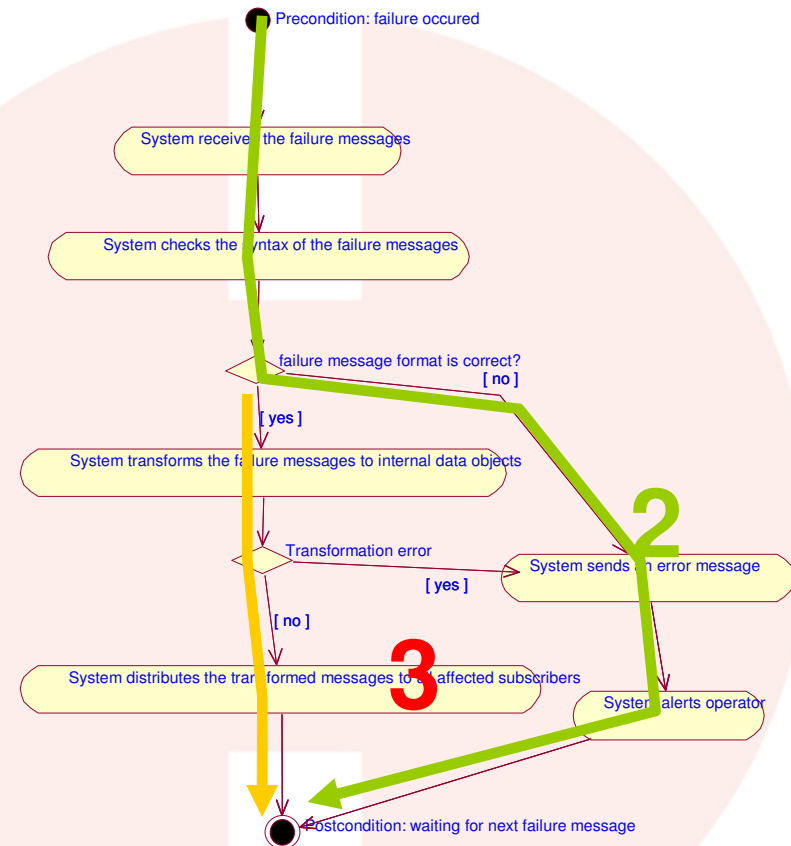
Deriving *the test specification: Stage 2 Reduced test cases*

- Check the use case state/activity model to reduce the number of test cases.
- Start with the basic path.
- Calculate the use case state/activity model coverage of the flow (states&transitions coverage)



Deriving the test specification: Stage 2 Reduced test cases

- If the path does not enhance the state/activity coverage
 - drop it
- Take the next path until states&transitions coverage
- **If important states/activities or transitions are not covered, define new path = test case that visit these states/activities.**



- *Thanks for your patience!*

Questions

