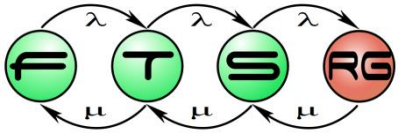
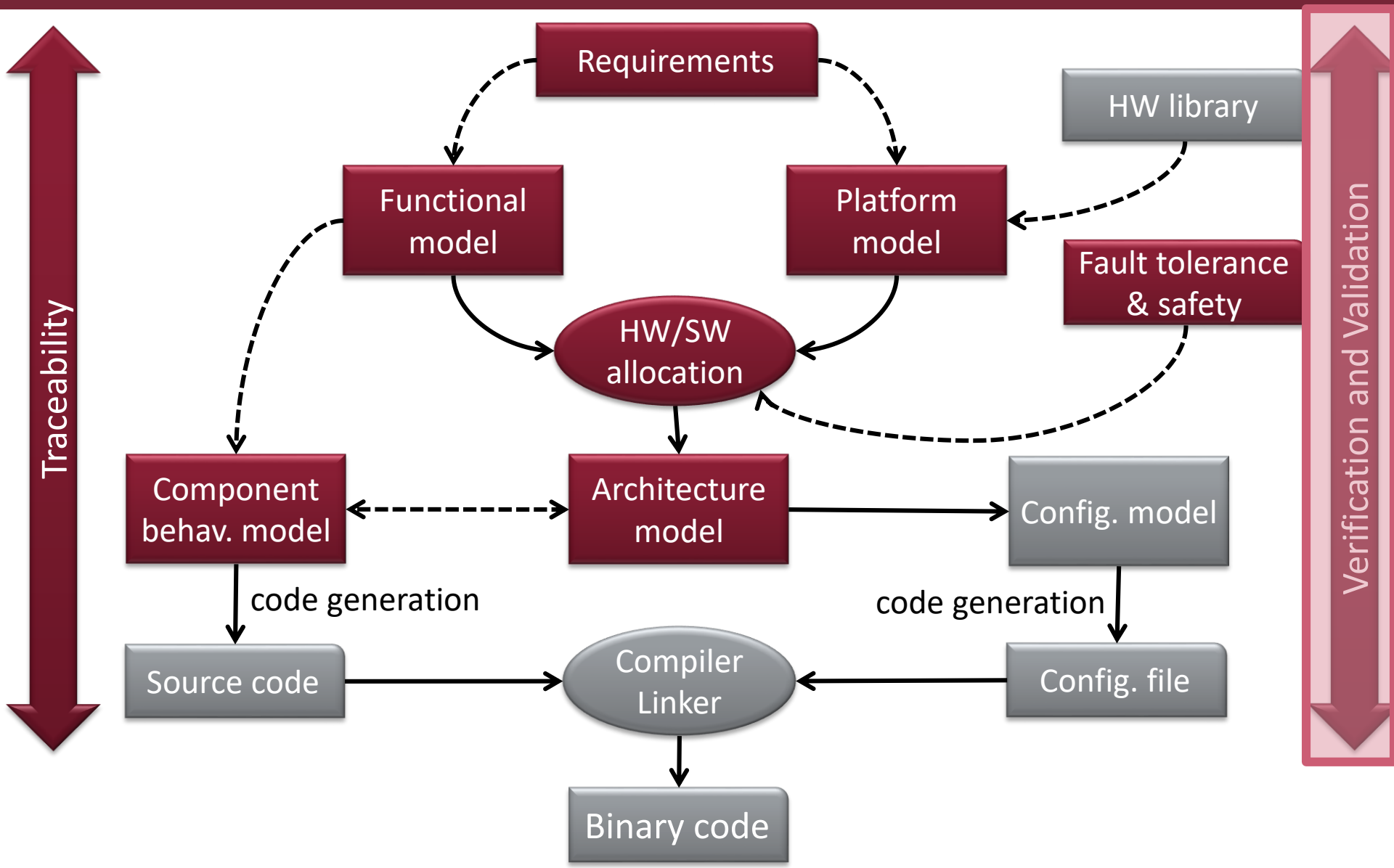
Verification & Validation: Overview, Requirement-based testing

Systems Engineering BSc Course



Budapest University of Technology and Economics
Department of Measurement and Information Systems

Platform-based systems design



Learning Objectives

V&V overview

- List typical V&V activities
- Classify verification techniques according to their place in the lifecycle

Requirement-based testing

- Recall basic testing concepts
- Describe the goal of specification-based test design techniques
- Use basic test design techniques

Overview of V&V techniques

Typical steps in development lifecycle

- **Requirement analysis**
- **System specification**

→ System engineer

- **Architecture design**

→ Architect

- **Module design**
- **Module implementation**

→ Developer, coder

- **System integration**
- **System delivery**
- **Operation, maintenance**

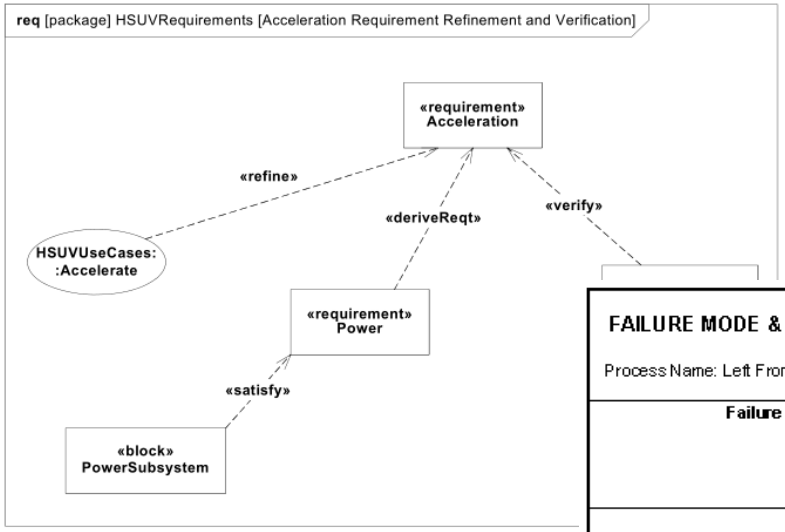
→ Test engineer

Schedule, sequencing depends on lifecycle model!

Requirement analysis

- **Requirement analysis**
- **System specification**
- **Architecture design**
- **Module design**
- **Module implementation**
- **System integration**
- **System delivery**
- **Operation, maintenance**

Task	V&V criteria	V&V technique
Defining functions, actors, use cases	- Risks - Criticality	- Checklists - Failure mode and effects analysis



FAILURE MODE & EFFECTS ANALYSIS (FMEA)

Date: 1/1/2000
Revision: 1.3

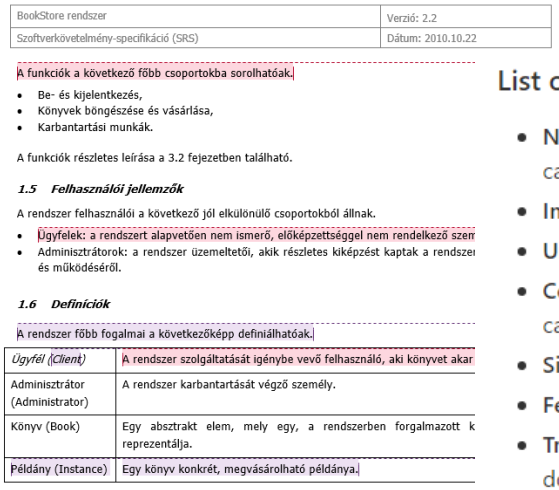
Process Name: Left Front Seat Belt Install Process Number: SBT 445

Failure Mode	A) Severity Rate 1-10 10 = Most Severe	B) Probability of Occurance Rate 1-10 10 = Highest Probability	C) Probability of Detection Rate 1 - 10 10 = Lowest Probability	Risk Preference Number (RPN) AxBxC
1) Select Wrong Color Seat Belt	5	4	3	**60**
2) Seat Belt Bolt Not Fully Tightened	9	2	8	**144**
3) Trim Cover Clip Misaligned	2	3	4	**24**

System specification

- Requirement analysis
- **System specification**
- Architecture design
- Module design
- Module implementation
- System integration
- System delivery
- Operation, maintenance

Task	V&V criteria	V&V technique
Defining functional and non-functional requirements	- Completeness - Unambiguity - Verifiability - Feasibility	- Reviews - Static analysis - Simulation



BookStore rendszer	Verzió: 2.2
Szoftverkövetelmény-specifikáció (SRS)	Dátum: 2010.10.22

A funkciók a következő főbb csoportokba sorolhatóak.

- Be- és kijelentkezés,
- Könyvek böngészése és vásárlása,
- Karbantartási munkák.

A funkciók részletes leírása a 3.2 fejezetben található.

1.5 Felhasználói jellemzők

A rendszer felhasználói a következő jól elkülönülő csoportokból állnak.

- Ügyfelek: a rendszert alapvetően nem ismerő, előképzettséggel nem rendelkező szem
- Adminisztrátorok: a rendszer üzemeltetői, akik részletes kiképzést kaptak a rendszer és működéséről.

1.6 Definíciók

A rendszer főbb fogalmai a következőképp definiálhatóak.

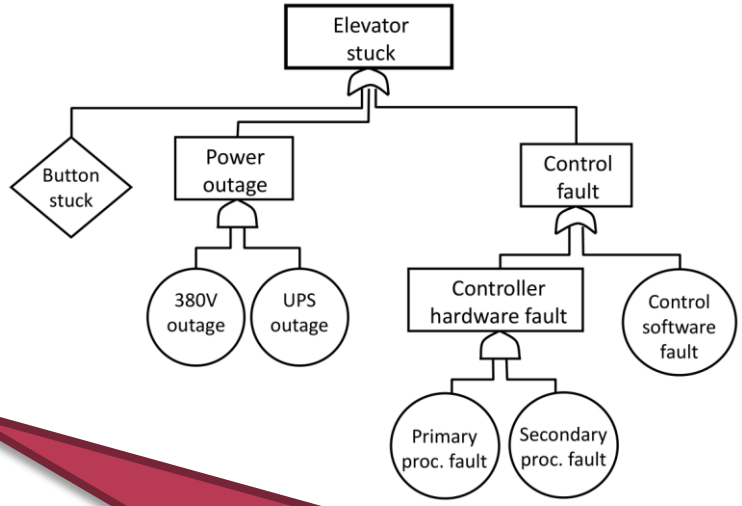
Ügyfél (Client)	A rendszer szolgáltatását igénybe vevő felhasználó, aki könyvet akar
Adminisztrátor (Administrator)	A rendszer karbantartását végző személy.
Könyv (Book)	Egy absztrakt elem, mely egy, a rendszerben forgalmazott k reprezentálja.
Példány (Instance)	Egy könyv konkrét, megvásárolható példánya.

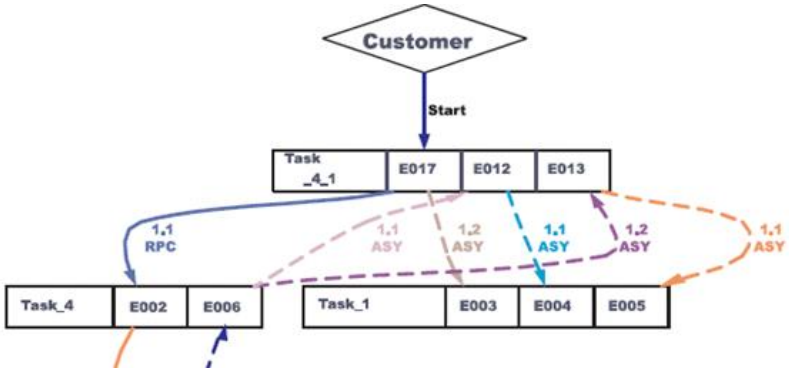
List of desired requirement characteristics

- **Necessary**: If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities
- **Implementation Free**: Avoids placing unnecessary constraints on the design
- **Unambiguous**: It can be interpreted in only one way; is simple and easy to understand
- **Complete**: Needs no further amplification (measurable and sufficiently describes the capability)
- **Singular**: Includes only one requirement with no use of conjunctions
- **Feasible**: Technically achievable, fits within system constraints (cost, schedule, regulatory...)
- **Traceable**: Upwards traceable to the stakeholder statements; downwards traceable to other documents
- **Verifiable**: Has the means to prove that the system satisfies the specified requirement

Architecture design

- Requirement analysis
- System specification
- **Architecture design**
- Module design
- Module implementation
- System integration
- System delivery
- Operation, maintenance

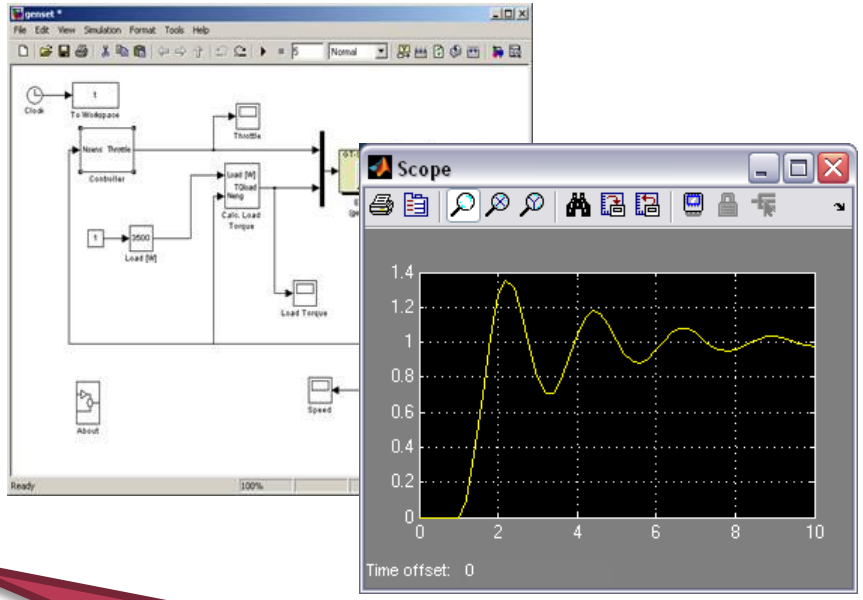


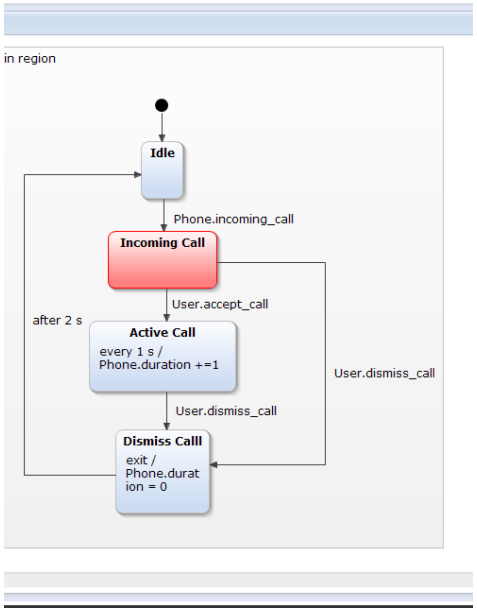


Task	V&V criteria	V&V technique
- Decomposing modules - HW-SW co-design - Designing communication	- Function coverage - Conformance of interfaces - Non-functional properties	- Static analysis - Simulation - Performance, dependability, security analysis

Module design (detailed design)

- Requirement analysis
- System specification
- Architecture design
- **Module design**
- Module implementation
- System integration
- System delivery
- Operation, maintenance

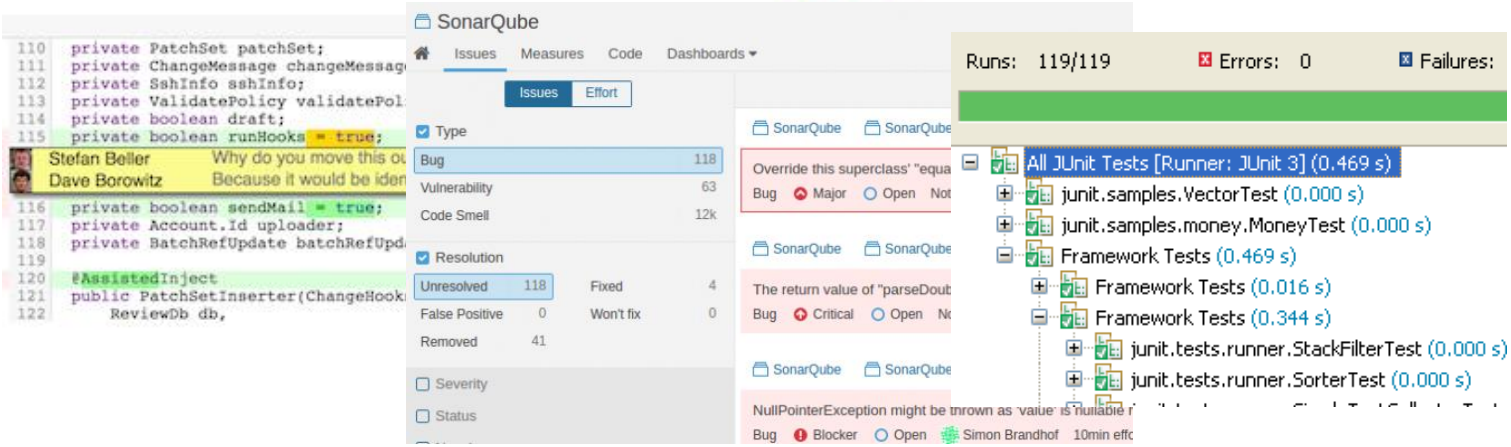




Task	V&V criteria	V&V technique
- Designing detailed behavior (data structures, algorithms)	- Correctness of critical internal algorithms and protocols	- Static analysis - Simulation - Formal verification - Rapid prototyping

Module implementation

- Requirement analysis
- System specification
- Architecture design
- Module design
- **Module implementation**
- System integration
- System delivery
- Operation, maintenance

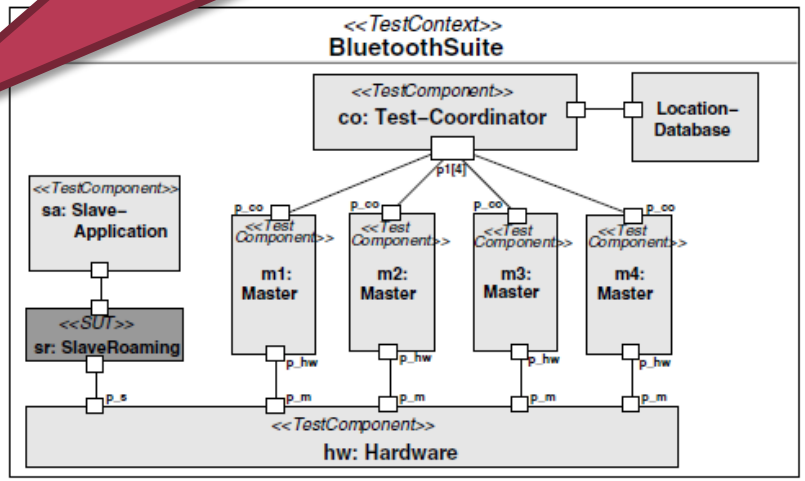


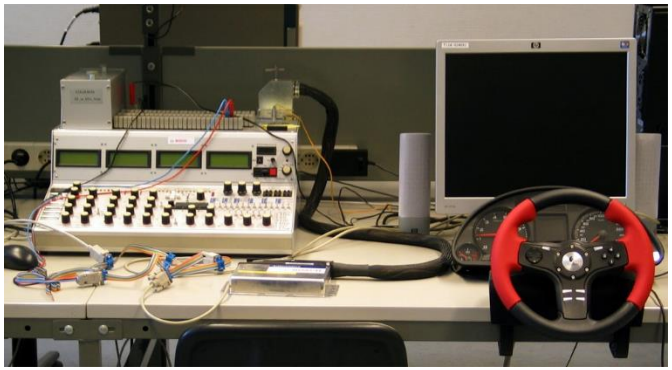
Task	V&V criteria	V&V technique
- Software implementation	Code is - Safe - Verifiable - Maintainable	- Coding conventions - Code reviews - Static code analysis
- Verifying module implementation	- Conformance to module designs	- Unit testing - Regression testing

System integration

- Requirement analysis
- System specification
- Architecture design
- Module design
- Module implementation
- **System integration**
- System delivery
- Operation, maintenance

Task	V&V criteria	V&V technique
- Integrating modules - Integrating SW with HW	- Conformance of integrated behavior - Verifying communication	- Integration testing (incremental)





System delivery and deployment

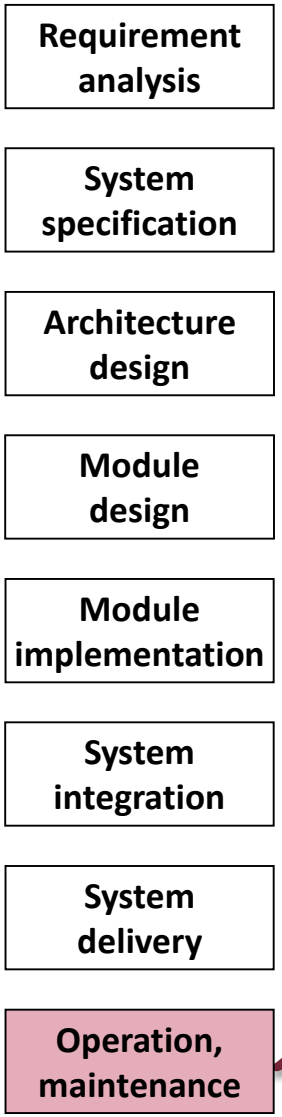
- Requirement analysis
- System specification
- Architecture design
- Module design
- Module implementation
- System integration
- **System delivery**
- Operation, maintenance

Source: Video and radar test (Bosch)

Source: Consumer Reports

Task	V&V criteria	V&V technique
- Assembling complete system	- Conformance to system specification	- System testing - Measurements, monitoring
- Fulfilling user expectations	- Conformance to requirements and expectations	- Validation testing - Acceptance testing - Alfa/beta testing

Operation and maintenance



Tasks during operation and maintenance:

- Failure logging and analysis (for failure prediction)
- V&V of modifications

Mini-lifecycle for each modification

Basic V&V Concepts

Recap from *Software Engineering* course

V&V techniques

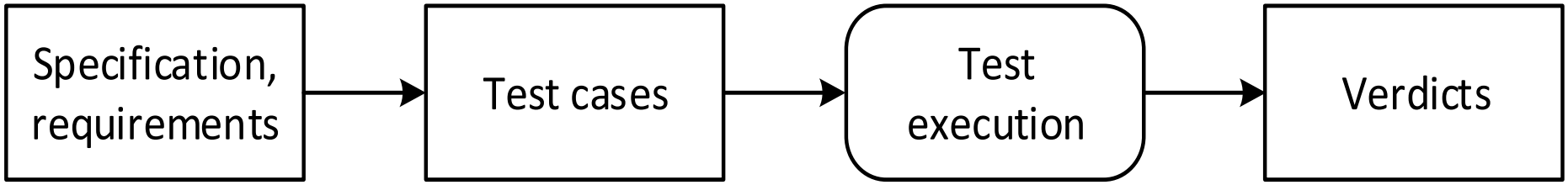
Static

- What: any artefact (documentation, model, code)
- How: without execution
- E.g.: review, static analysis

Dynamic

- What: executable artefacts (model, code…)
- How: with execution
- E.g.: simulation, testing

Basic concepts

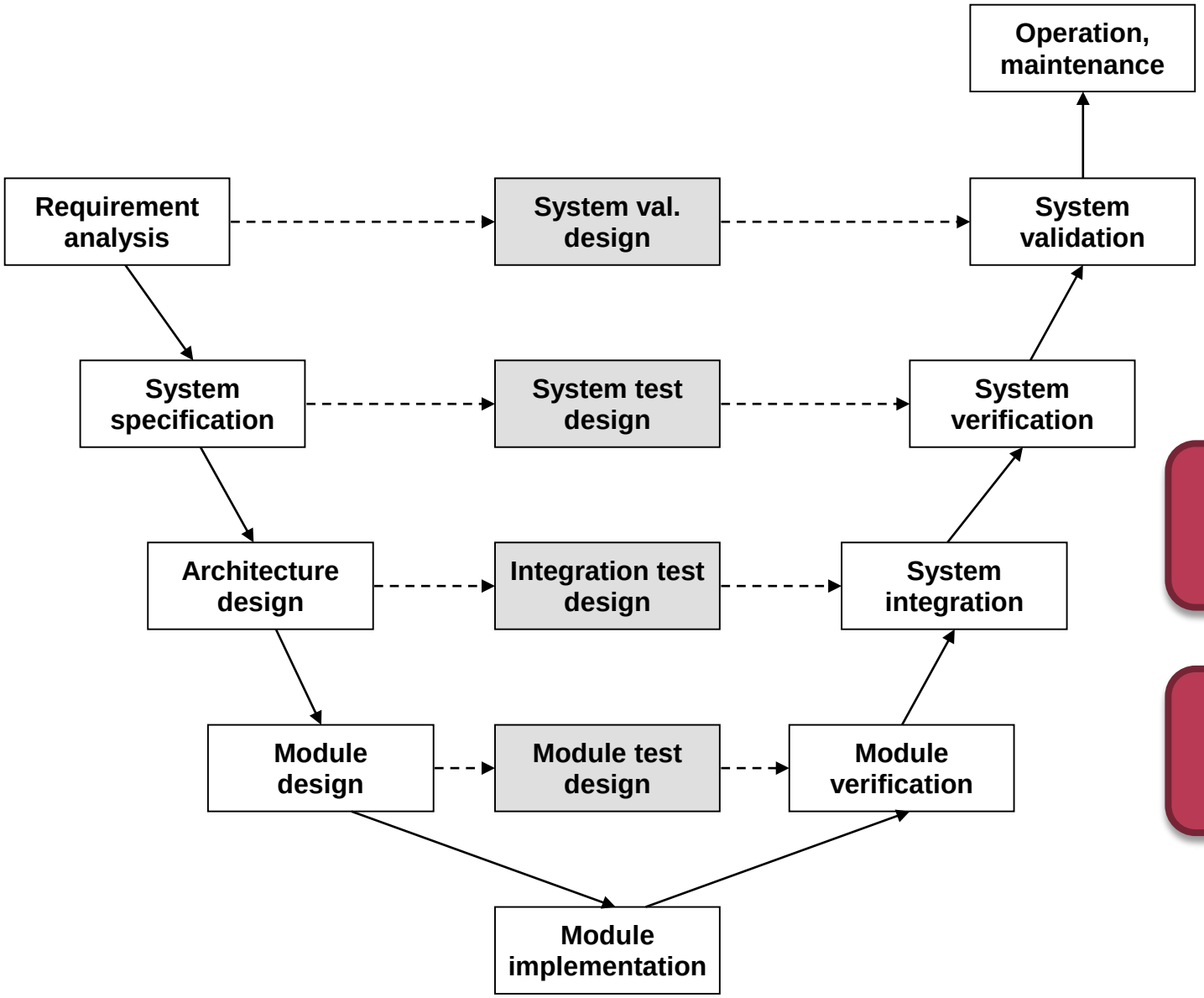


- SUT: system under test
- Test case
  - a set of test inputs, execution conditions, and expected results developed for a particular objective
- Test suite
- Test oracle
  - A principle or mechanism that helps you decide whether the program passed the test
- Verdict: result (pass / fail /error / inconclusive…)

Problems and tasks

- Test selection
  - What test inputs and test data to use?
- Oracle problem
  - How to get/create reliable oracle?
- Exit criteria
  - How long to test?
- Testability
  - Observability + controllability

V&V in the V-model



V&V in each step!

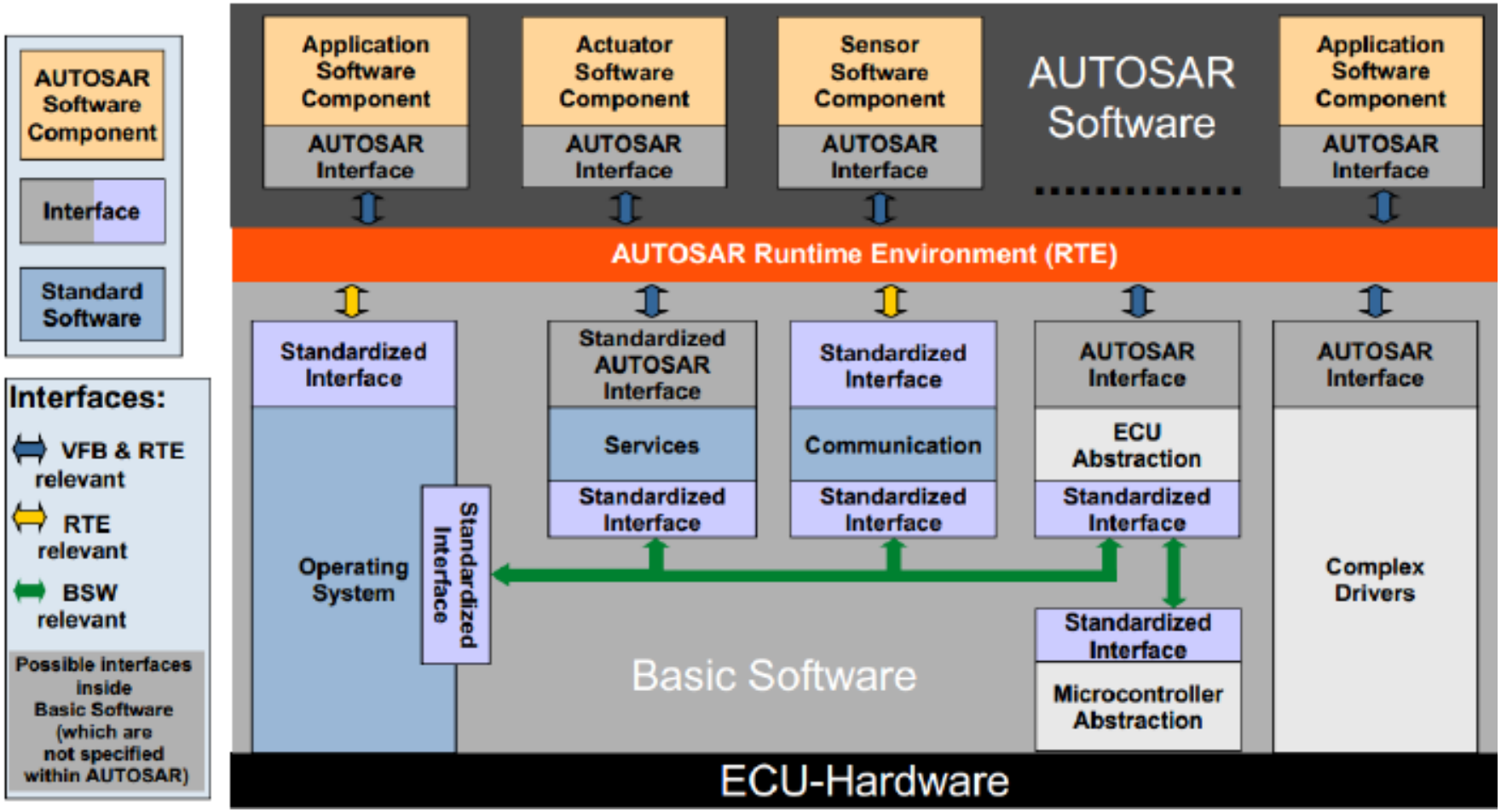
Not just after coding

Case study: AUTOSAR Acceptance Tests

Source: AUTOSAR ATS Overview, AUTOSAR ATS RTE

19

AUTOSAR concepts (recap)



AUTOSAR Acceptance Tests

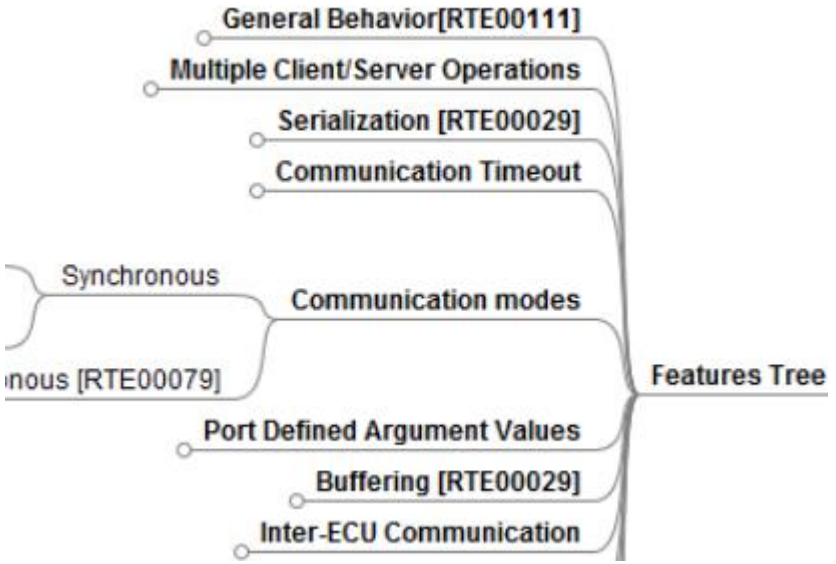
- System-level tests based on specification
- Checks visible functionalities
  - Application level and Bus level
- Acceptance Test Specifications (ATS)
- Test suites for different specifications
  - Communication (CAN, FlexRay…), Memory stack, **Runtime Environment [RTE]**…

Example: AUTOSAR ATS RTE

- Tests RTE functionality
- 5 features, 68 test cases, 251 pages (!)

Feature: RTE Client Server Communication

- *General Test Objectives*: cover the Client Server feature of the RTE [RS_BRF_01312]

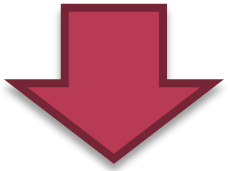


Requirements and specification to test

[RS_BRF_01312] *AUTOSAR RTE shall support calling of subroutines (client/server call, including remote procedure calls).*

Refine

[SRS_Rte_00029] *The RTE shall support multiple-client-single-server ("n:1") client-server (function invocation) communication.*



Refine

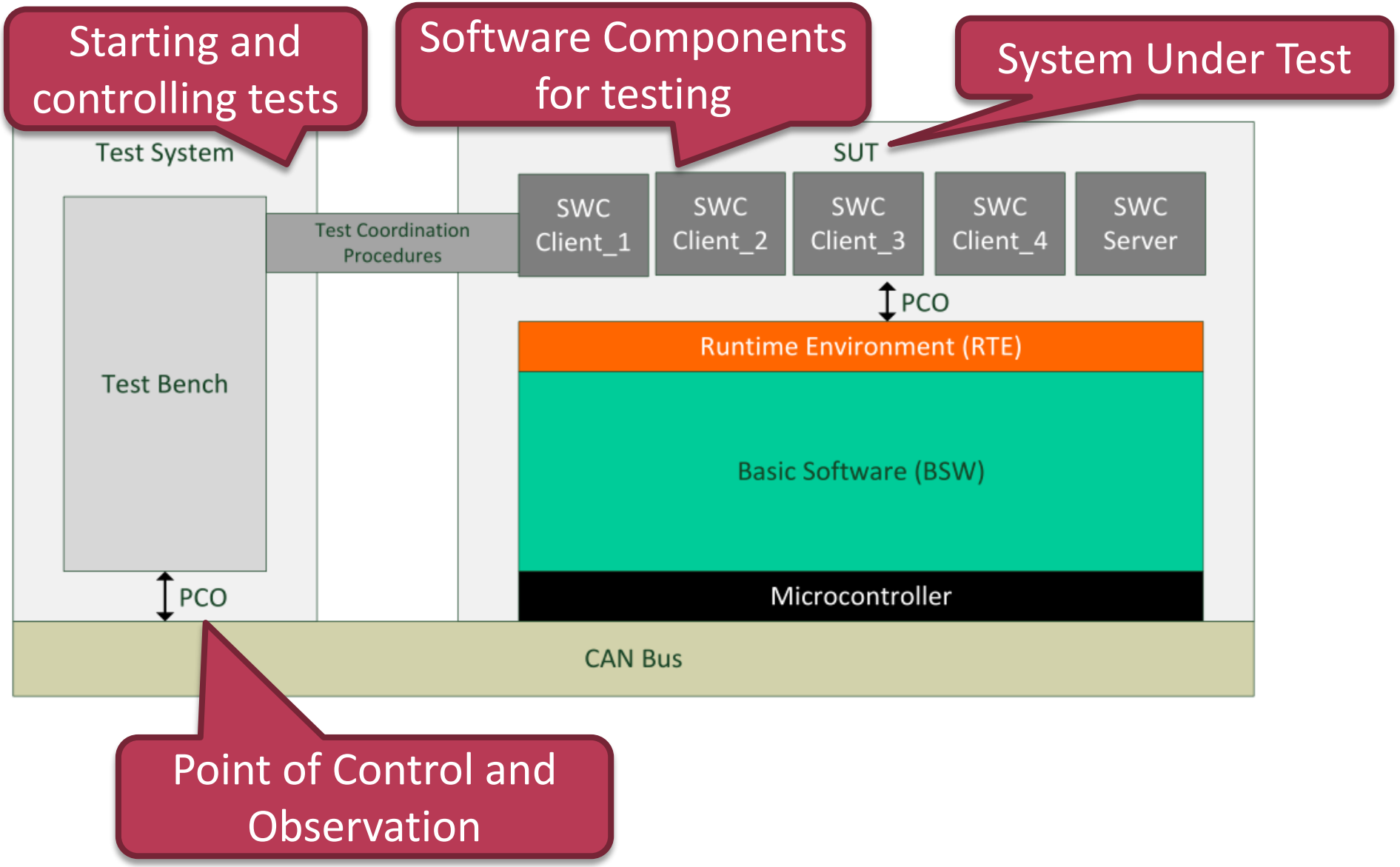
[SWS_Rte_04516] *The RTE's implementation of the client-server communication shall ensure that a service result is dispatched to the correct client if more than one client uses a service.*

How can we test this functionality?

What is needed to define a test

- Test architecture
  - SUT, simulated components, test drivers and stubs…
- Test configuration and data
  - Parameters, message data…
- Test cases
  - Test goal, pre-conditions, sequence of steps (input + expected output), post-conditions…

Test architecture



Test configuration (excerpt)

SWC Name	Tester_Server		
PORTS	**Name**	ServerA	
	Type	PPortPrototype	
	Interface	PrimitiveData_IF	
	Requirements		
RUNNABLE ENTITIES	**Name**	serverRead	
	Requirements	canBeInvokedConcurrently=false send back data from global variable written by serverWrite	
	Started by Event	**Name**	OIE_ReadA
		Type	OperationInvokedEvent

Test case

Test Objective	Test synchronous server call for n:1 intra-ECU Client-Server communication		
ID	ATS_RTE_00052	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1
Affected Modules	RTE	**State**	reviewed
Trace to SWS Item	RTE: SWS_Rte_02527 RTE: SWS_Rte_04515 RTE: SWS_Rte_04516 RTE: SWS_Rte_04519		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Read1A, sscp_Write1A) Tester_Client_3 * port Client3A * runnable RUN_Client3 (sscp_Write3A) Tester_Server * port ServerA * runnable serverRead * runnable serverWrite Client1A and Client3A connected to ServerA.		

Test case (cont'd)

Summary	The goal of this test is to check the behavior of synchronous server calls in case of n:1 Intra-ECU Client-Server communication. 2 clients connected to the same server are invoking (synchronous server call) successively the same operation of the server. The Test Manager checks that the operations are handled correctly.
Pre-conditions	None

Test case (cont'd)

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client1, RUN_Client3	
Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Write(DataValue1)	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK
Step 3	**[RUN<RUN_Client3>]** execute Rte_Call_Client3A_Write(DataValue2)	**[RUN<RUN_Client3>]** Rte_Call returns RTE_E_OK
Step 4	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Read	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK, data returned is DataValue2
Step 5	**[CP]** terminate RUN_Client1, RUN_Client3	
Post-conditions	None	

Specification-based test design

Test design techniques

Goal: Select test cases based on test objectives

Specification-based

- SUT: black box
- Only spec. is known
- Testing specified functionality

Structure-based

- SUT: white box
- Inner structure known
- Testing based on internal behavior

Specification-based techniques

Equivalence classes

Boundary values

Based on use cases

Combinatorial testing

Decision tables

...

Equivalence class partitioning

- Input and output equivalence classes:
  - Data that are expected to cover the same faults (cover the same part of the program)
  - Goal: Each equivalence class is represented by one test input (selected test data)

- Highly context-dependent
  - Needs to know the domain and the SUT!
  - Depends on the skills and experience of the tester

Selecting equivalence classes

- Selection uses heuristics
  - Initial: valid and invalid partitions
  - Next: refine partitions
- Typical heuristics:
  - Interval (e.g. 1-1000)
    - < min, min-max, >max
  - Set (e.g. RED, GREEN, BLUE)
    - Valid elements, invalid element
  - Specific format (e.g. first character is @)
    - Condition true, condition false
  - Custom (e.g. February from the months)

Deriving test cases from equiv. classes

- Combining equiv. classes of several inputs
- For valid (normal) equivalence classes:
  - test data should cover as much equivalence classes as possible
- For invalid equivalence classes:
  - first covering the each invalid equivalence class separately
  - then combining them systematically

EXERCISE Equivalence partitions

Requirement: The loan application shall be denied if the requested amount is larger than 1M Ft and the customer is a student, unless the amount is less than 3M Ft and the customer has repaid a previous loan (of any kind).

- Input parameters? Equivalence classes?
- Any questions regarding the requirement?

Specification-based techniques

Equivalence classes

Boundary values

Based on use cases

Combinatorial testing

Decision tables

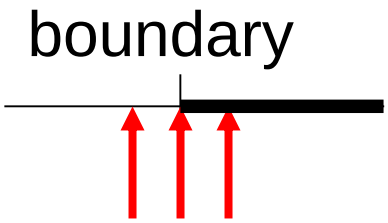
…

2. Boundary value analysis

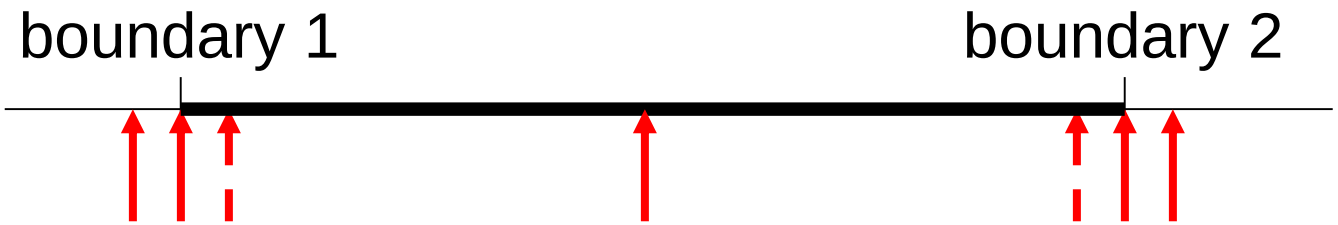
- Examining the boundaries of data partitions
  - Focusing on the boundaries of equivalence classes
  - Both input and output partitions
- Typical faults to be detected:
  - Faulty relational operators,
  - conditions in cycles,
  - size of data structures,
  - …

Typical test data for boundaries

- A boundary requires 3 tests:



- An interval requires 5-7 tests:



EXERCISE Boundary values

Requirement: If the robot detects that a human is closer than 4 meter, then it has to slow down, and if it is closer than 2 meter, then it has to stop.

- What values to use for testing?
- Any other questions regarding the requirement?

Specification-based techniques

Equivalence classes

Boundary values

Based on use cases

Combinatorial testing

Decision tables

...

Deriving tests from use cases

- Typical test cases:
  - 1 test for main path („happy path”, „mainstream”)
    - Oracle: checking post-conditions
  - Separate tests for each alternate path
  - Tests for violating pre-conditions

- Mainly higher levels (system, acceptance…)

EXERCISE Deriving tests from a use case

3.2.5 Vásárlás

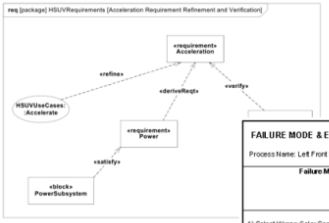
ID / Név:	UC6 / Buy
Verzió:	1.0
Leírás:	A felhasználó a megvásárolni kívánt könyvek kosárba tétele után kifizetheti azokat, ha megad ehhez egy érvényes bankkártya számot, amiről a vételár levonható.
Előfeltétel:	Van legalább egy könyv a felhasználó kosarában, megadott egy érvényes bankkártya számot a kosár megtekintésénél és ezt követően nem navigált el a kosár tartalmát listázó oldalról.
Utófeltétel:	Az ügyfél kosara kiürül, és a könyveket megvásárolja.
Trigger:	A felhasználó a fizetés funkciót választja.
Normál lefutás:	1. A kosárban lévő könyv példányok kikerülnek az adatbázisból. 2. A kosár is kiürül. 3. A fizetés ténye belekerül a tranzakció naplóba.
Alternatív lefutások:	- Ha nincs megadva vagy érvénytelen a bankkártya szám, akkor nem változik sem a készleten lévő, sem a kosárban lévő könyvek listája.

Summary

Requirement analysis

Requirement analysis
System specification
Architecture design
Module design
Module implementation
System integration
System delivery
Operation, maintenance

Task	V&V criteria	V&V technique
Defining functions, actors, use cases	- Risks - Criticality	- Checklists - Failure mode and effects analysis



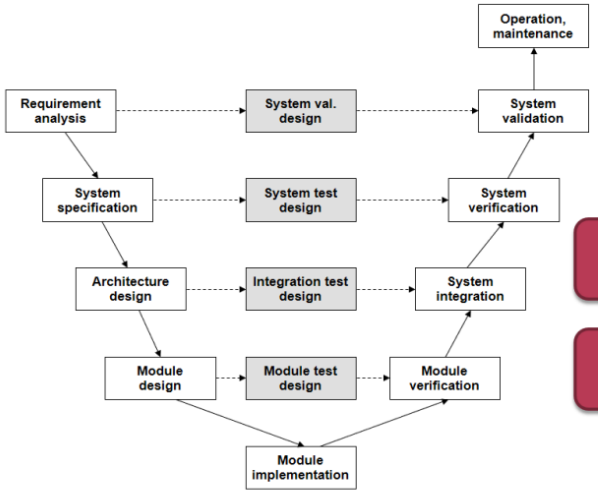
FAILURE MODE & EFFECTS ANALYSIS (FMEA)

Date: 1/1/2000
Revision: 1.3

Process Name: Left Front Seat Belt Install Process Number: SBT 445

Failure Mode	A) Severity Rate 1-10 10 = Most Severe	B) Probability of Occurance Rate 1-10 10 = Highest Probability	C) Probability of Detection Rate 1 - 10 10 = Lowest Probability	Risk Preference Number (RPN) AxBxC
1) Select Wrong Color Seat Belt	5	4	3	60
2) Seat Belt Bolt Not Fully Tightened	9	2	8	144
3) Trim Cover Clip Misaligned	2	3	4	24

V&V in the V-model



V&V in each step!

Not just after coding

Test design techniques

Goal: Select test cases based on test objectives

Specification-based

- SUT: black box
- Only spec. is known
- Testing specified functionality

Structure-based

- SUT: white box
- Inner structure known
- Testing based on internal behavior

Specification-based techniques

Equivalence classes

Boundary values

Based on use cases

Combinatorial testing

Decision tables

...