Continental
The Future in Motion

Entwicklungsprozesse mit der UML für eingebettete Systeme

10 Jahre Modellierung bei Continental, eine ehrliche Bilanz

Continental Corporation
Five Strong Divisions

Chassis & Safety	Powertrain	Interior	Tires	ContiTech
Vehicle Dynamics	Engine Systems	Instrumentation & Driver HMI	PLT, Original Equipment	Air Spring Systems
Hydraulic Brake Systems	Transmission	Infotainment & Connectivity	PLT, Repl. Business, EMEA	Benecke-Kaliko Group
Passive Safety & Sensorics	Hybrid Electric Vehicle	Intelligent Transportation Systems	PLT, Repl. Business, The Americas	Compounding Technology
Advanced Driver Assistance Systems (ADAS)	Sensors & Actuators	Body & Security	PLT, Repl. Business, Asia Pacific	Conveyor Belt Group
	Fuel & Exhaust Management	Commercial Vehicles & Aftermarket	Commercial Vehicle Tires	Elastomer Coatings
			Two Wheel Tires	Industrial Fluid Systems
				Mobile Fluid Systems
				Power Transmission Group
				Vibration Control

PLT – Passenger and Light Truck Tires

Business Unit Commercial Vehicles & Aftermarket

Segments and their Portfolio

Tachographs, Telematics & Services

- Digital tachograph solutions
- Analogue tachograph solutions
- Accessories & services for workshops
- Fleet management
- Telematic products

Vehicle Electronics*

- Instrument clusters OE & platform solutions
- Single gauges
- Body controller for trucks
- Driver working place
- Body builder module
- Off & On-Highway transmission ECU
- Chassis master control unit
- Light control module
- Platform multiplex solutions

Independent Aftermarket

- ATE wear and tear parts for brakes
- VDO replacement parts (fuel systems, actuators for central locking systems, engine actuators, HVAC blower & fan systems, screen & headlight washer systems, sensors, tire pressure monitoring systems)
- Multi-brand diagnostics
- OE diagnostics & services
- Diesel repair service
- Brake service equipment & tools

Original Equipment Services

- All original parts of Automotive Group
- Portfolio for service & replacement to OEMs (commercial & passenger vehicles)

*Selective products from product categories

Agenda

1. Motivation
2. Introduction Tachograph
3. Project Structure Tachograph
4. Legacy Development
5. Future Development
6. Support and Barriers

Motivation

Target Achievement

- Abstract and understand complex Systems
- Formal Development Approach
  - How do we come to a solution (break down)
  - Why have we chosen this solution (rationale)
  - Where are requirements implemented (traceability)
- Effective and reliable Verification and Validation
- Consistency between Architecture, Documentation, Code, Test, …

Agenda

1. Motivation
2. Introduction Tachograph
3. Project Structure Tachograph
4. Legacy Development
5. Future Development
6. Support and Barriers

The Project [redacted]

- Tachograph for European market
  - Common Criteria Security Evaluation (highest Attack level)
  - Major Project Goal: Cost reduction (using new Security-Architecture)

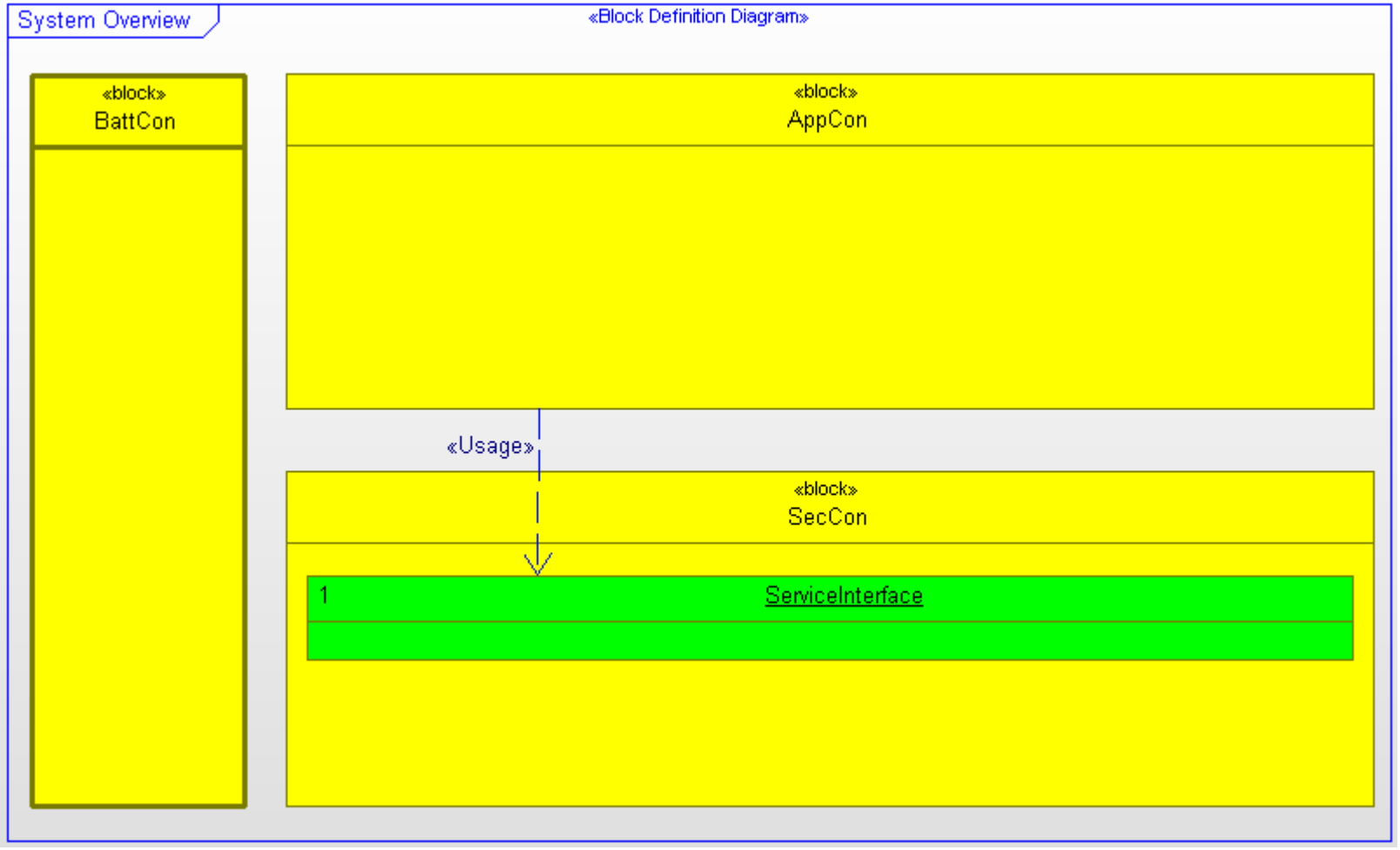
- Project Domain
  - Team: Multi-site / Multi-team
  - Approach: Migration based on existing software, by isolating security related features into additional microcontroller (called „SecCon“)
  - Restricted HW:
    - CPU: [redacted]
    - Code/Data: [redacted]
    - RAM:[redacted]

Architectural Principles

- Definition of a Static Structure
  - Central Entry Point (Architectural Views)
  - Clear Hierarchy for SW Modules (Layers)
  - Clear Structure within the Modules (Interface, Implementation)

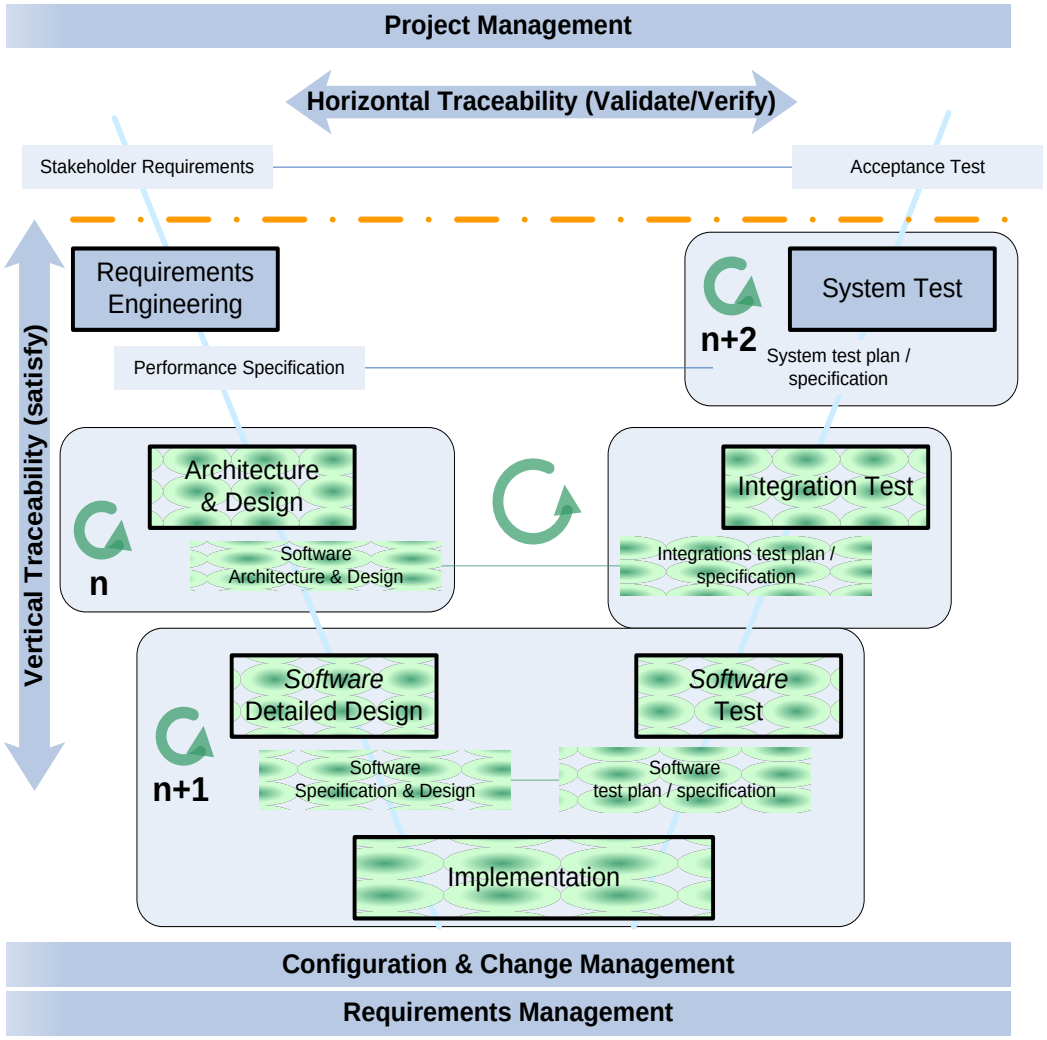
- Definition of a Workflow
  - Abstraction Methodology
  - Definition of Abstraction Level 0 – 3
    - Level 0: Cross System Level (Collaboration)
    - Level 1: System Level (Device)
    - Level 2: Sub-system Level (Controller – SW Architecture)
    - Level 3: Component Level (SW Module – SW Design)

System Overview (level 1)



Process – V-Model

Feature Driven Development (FDD)

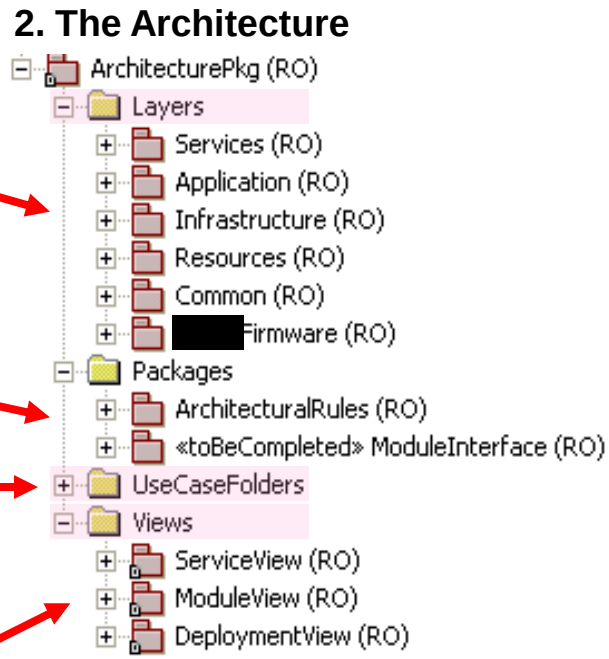


Agenda

1. Motivation
2. Introduction Tachograph
3. **Project Structure Tachograph**
4. Legacy Development
5. Future Development
6. Support and Barriers

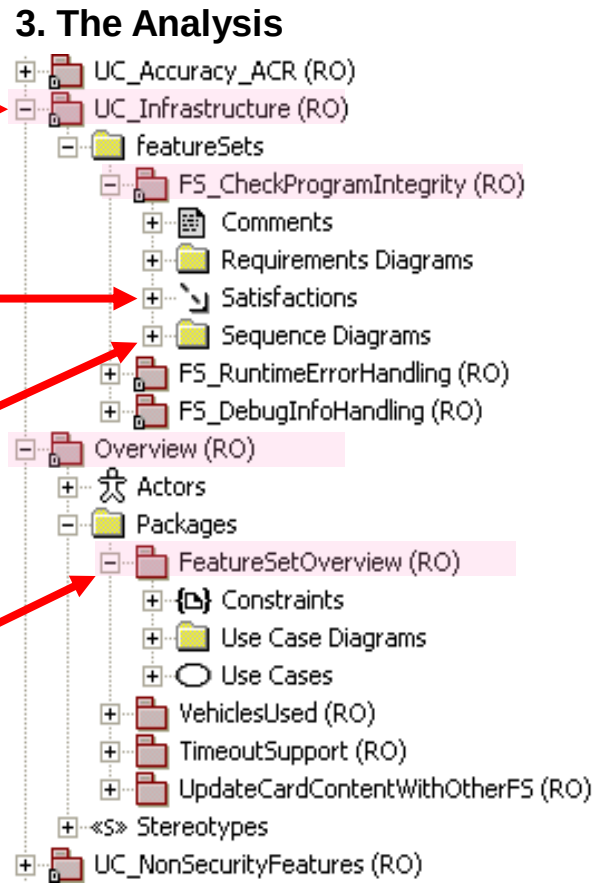
Structure – Overview (level 1 & level 2)

- The Architecture consists of several Layers
- Additional Packages provides Architectural Rules
- Use Case Folders provide high level Abstraction
- Different Views on the Model provides a good Overview



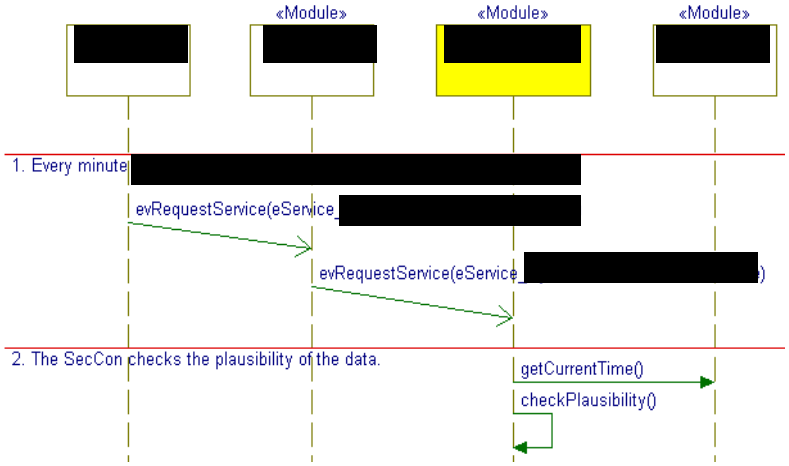
Structure – Overview (level 1 & level 2)

- The Use Case folders are container for corresponding Feature Sets
- The Feature Sets are satisfying Requirements
- Sequence Diagrams are showing Scenarios of the Feature Sets on different levels and impact
- The Overview consists of all identified Feature Set represented as UML Use Cases

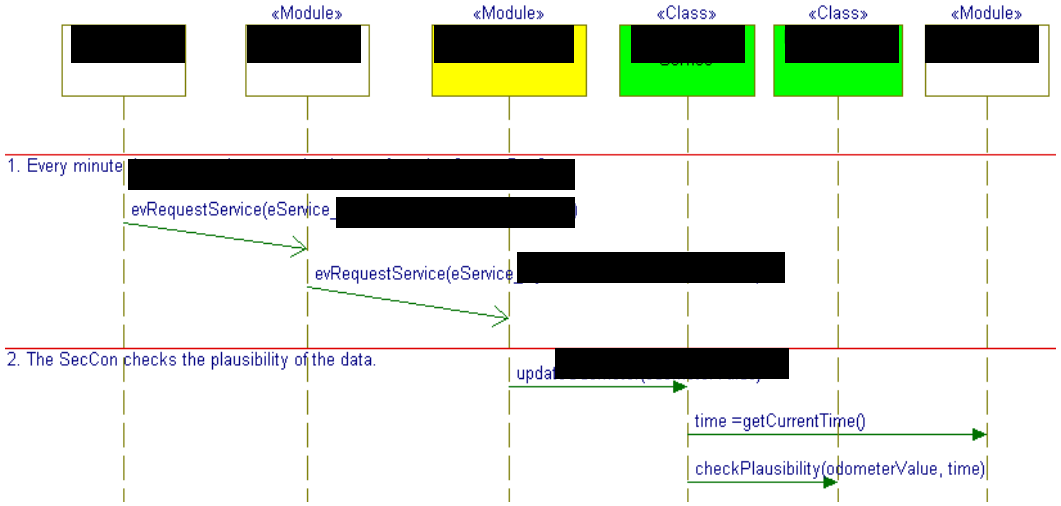


Structure – Architecture & Design

SW Architecture
(Level 2 Diagram)



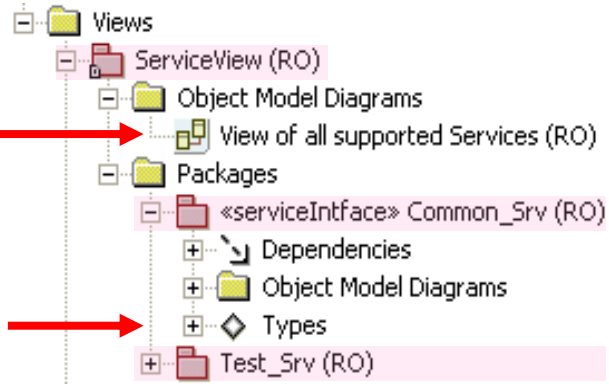
SW Design
(Level 3 Diagram)



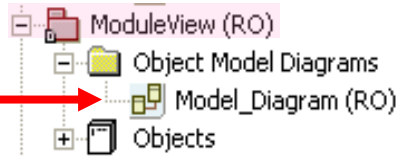
Structure – Views (level 2)

- The Service View shows an Overview of all existing Services
- It defines Types which represents the Message structure and the Service IDs
- The Module View shows all existing Modules and how they collaborate

4. The Service View

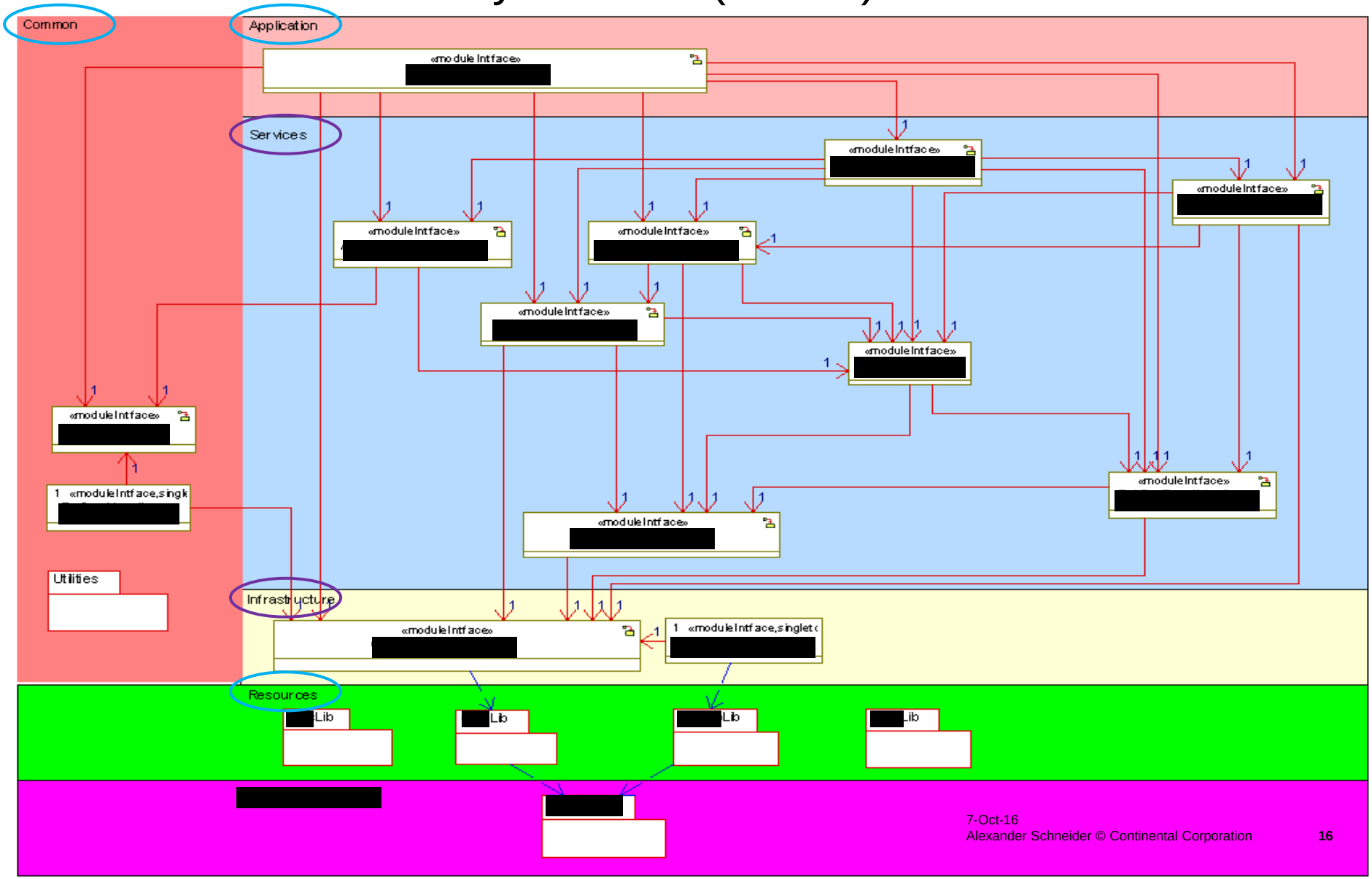


5. The Module View



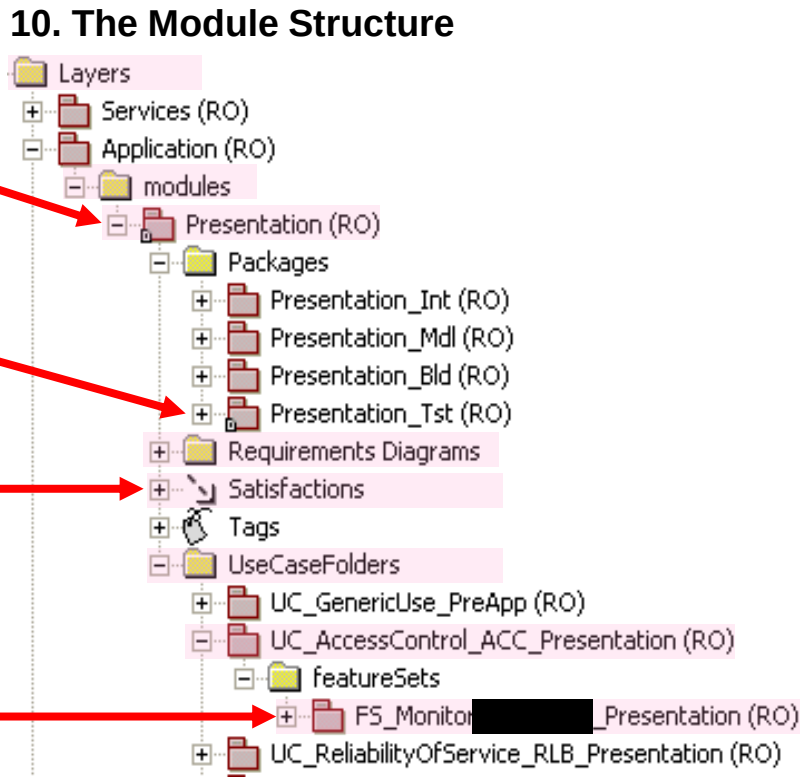
Structure – Module Interface Overview

SW Architecture – Layer model (level 2)



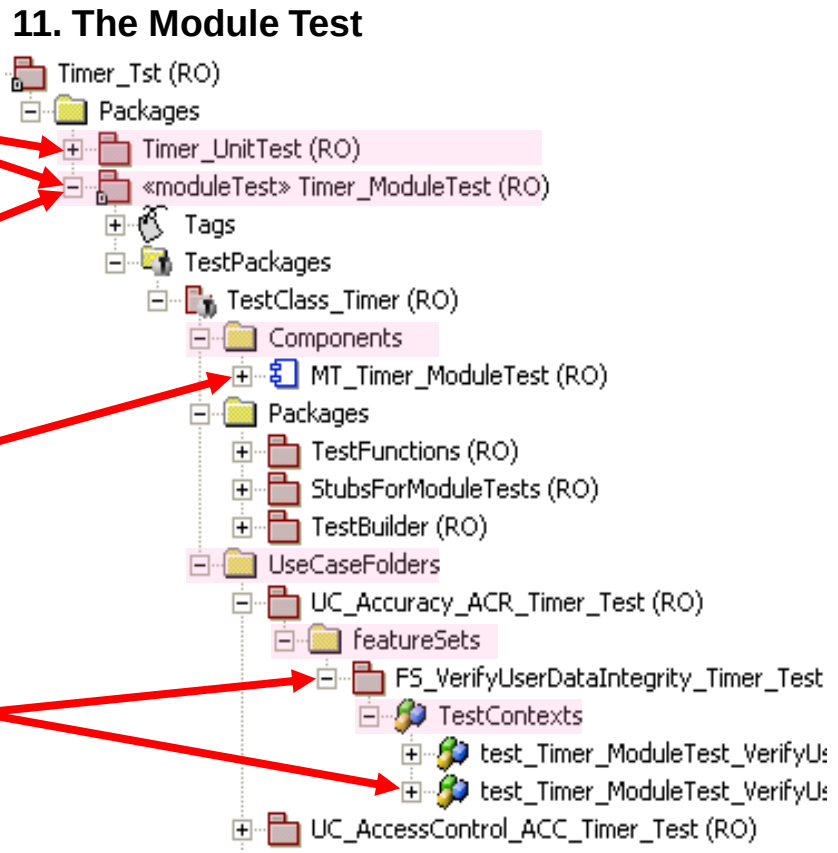
Structure – Implementation / Test (level 3)

- Each Module is stored in its own CI
- The Test Package is stored in a separate CI
- Requirements are satisfied on Module level
- Each Module can have more detailed Level3 Sequence Diagrams



Structure – Implementation / Test (level 3)

- The Test Package contains Unit and Module Tests
- The Module Tests are again stored in a separate CI
- The Component consists of several Configurations
- The Tests are created according the defined Feature Sets

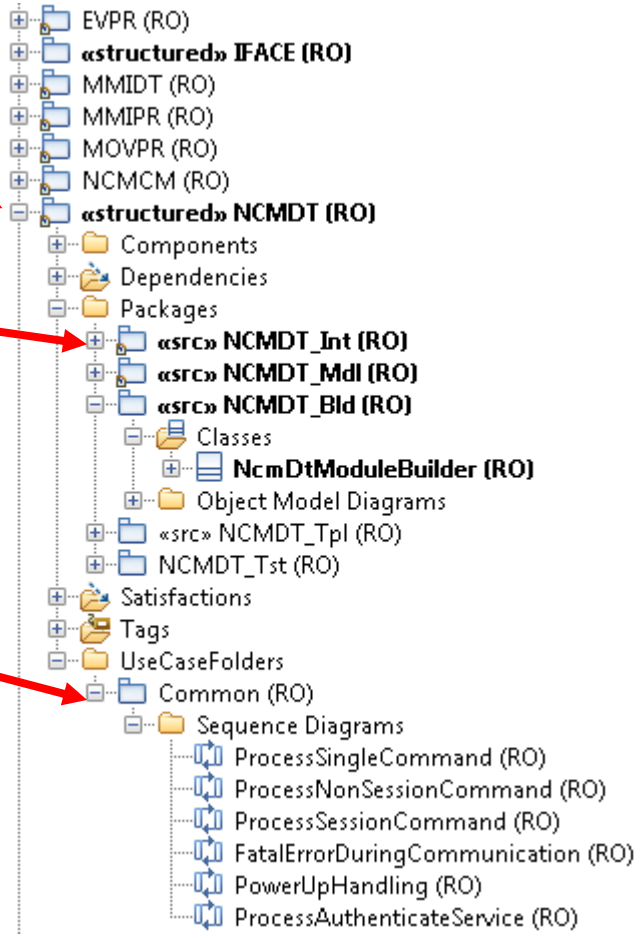


Agenda

1. Motivation
2. Introduction Tachograph
3. Project Structure Tachograph
4. Legacy Development
5. Future Development
6. Support and Barriers

The Project [redacted]
New Module Development

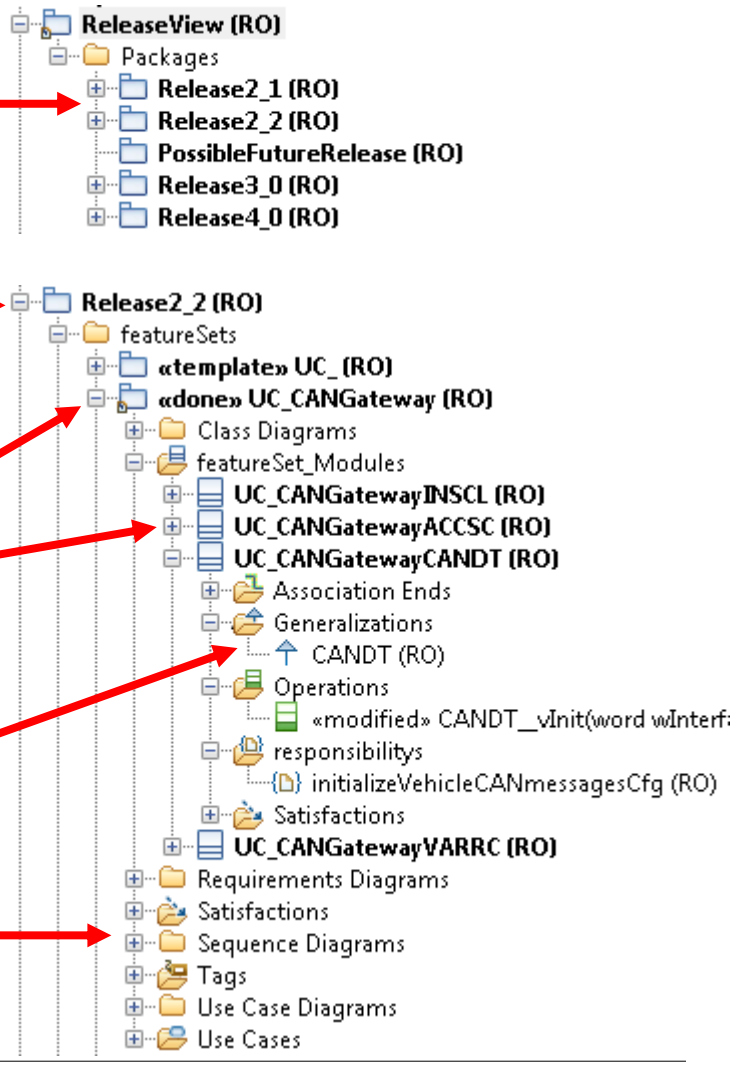
- NCMDT is a new Module
  - Uses object oriented C
  - Provides Singleton Objects for accessing the Services
  - Provides callback Interfaces to be implemented by the clients
  - Documents its behavior with several SDs
  - Uses Code Generation for Statecharts



A Complex Statechart

The Project [redacted]

- The Release View provides information for several releases
- Within a Release is Information about all the changes
  - Each Feature has its own Package
  - FeatureSet Modules show the related Module changes
  - They inherit from the real Modules and show the needed changes and responsibilities
  - The Features have also SDs and Requirement traceability



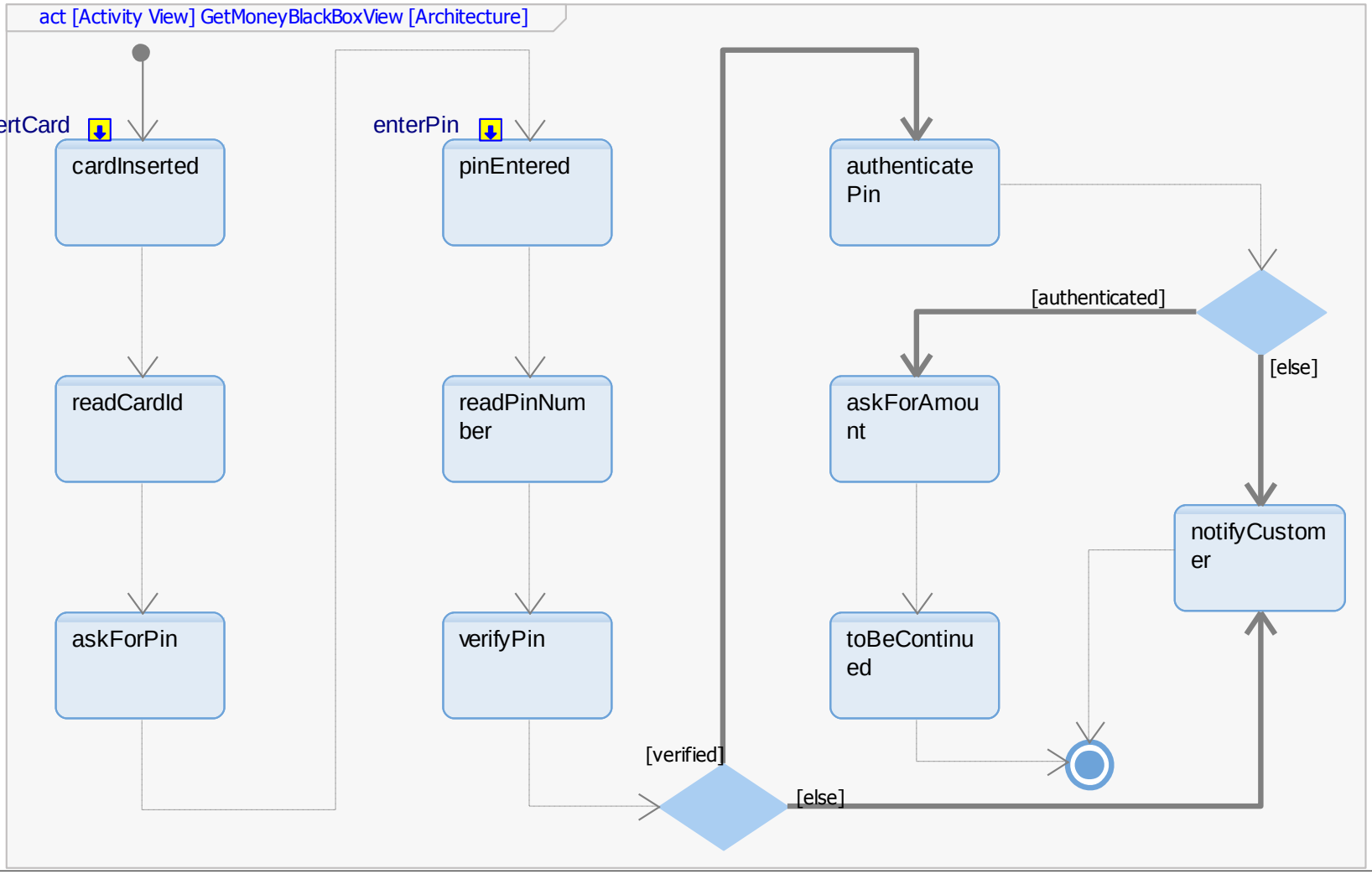
Agenda

1. Motivation
2. Introduction Tachograph
3. Project Structure Tachograph
4. Legacy Development
5. Future Development
6. Support and Barriers

System: Cash Machine

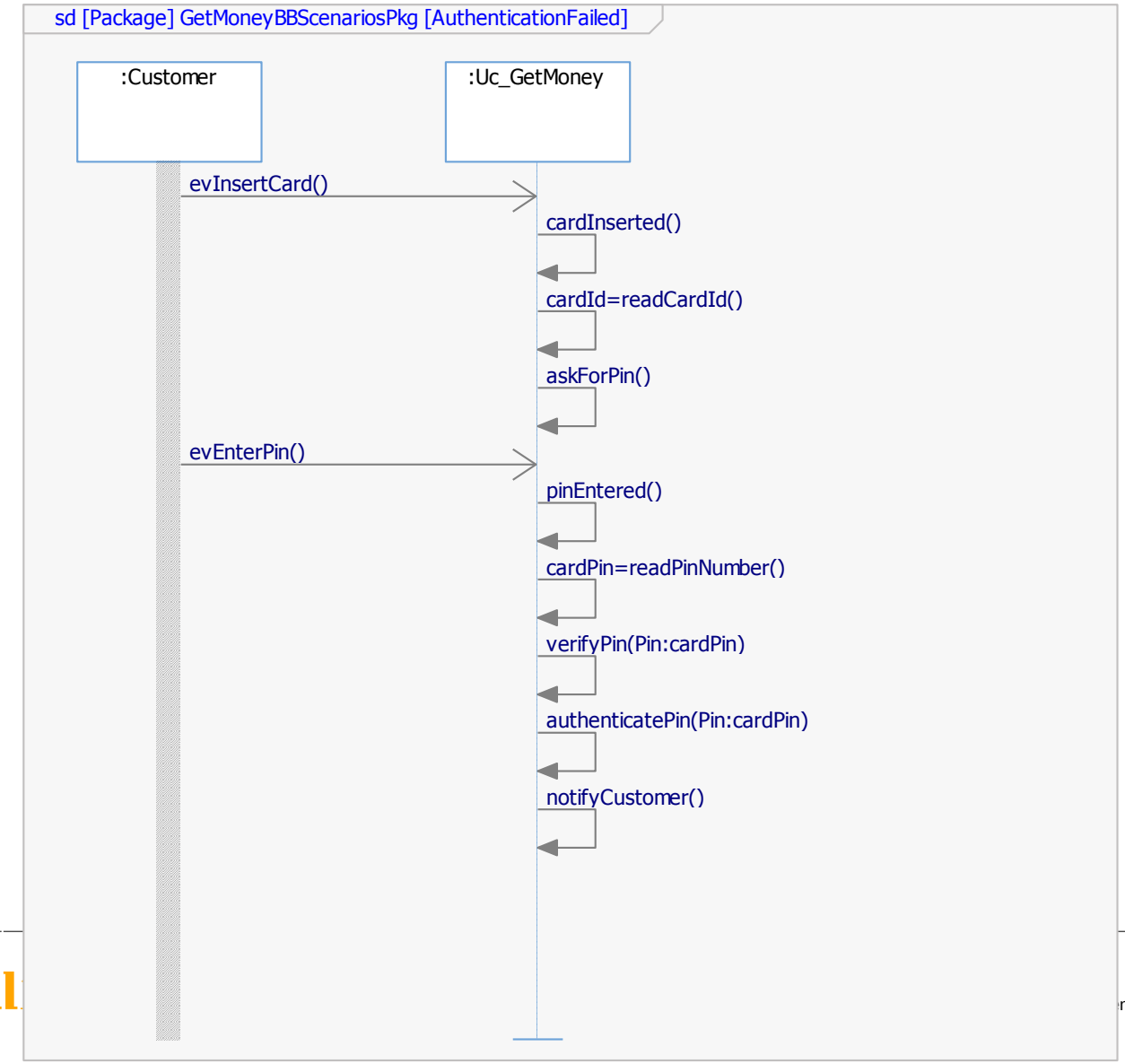
System: Cash Machine

Use Cases Analysis (level 0)



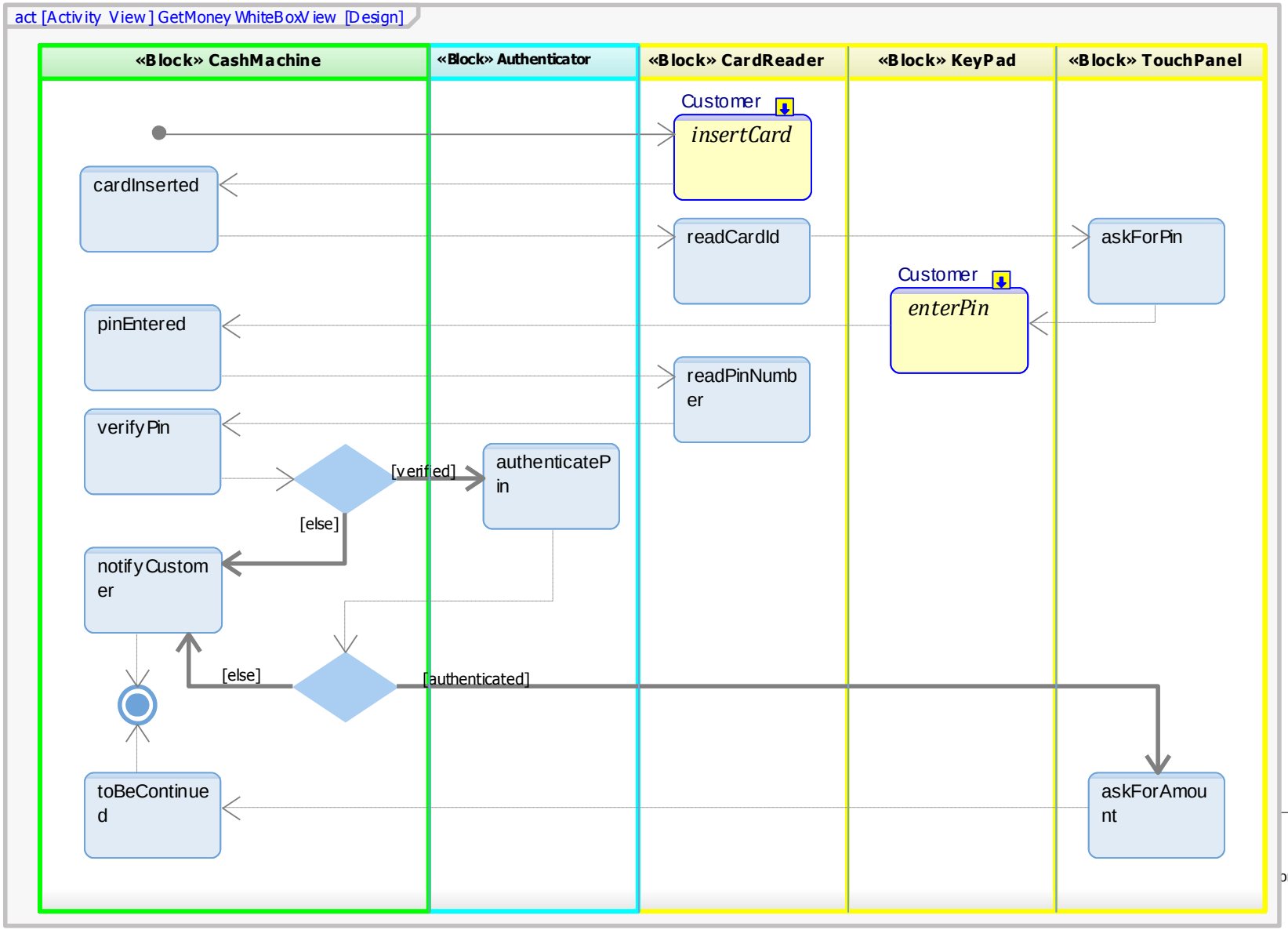
System: Cash Machine

Scenario: Use Case Execution I (level 0)



System: Cash Machine

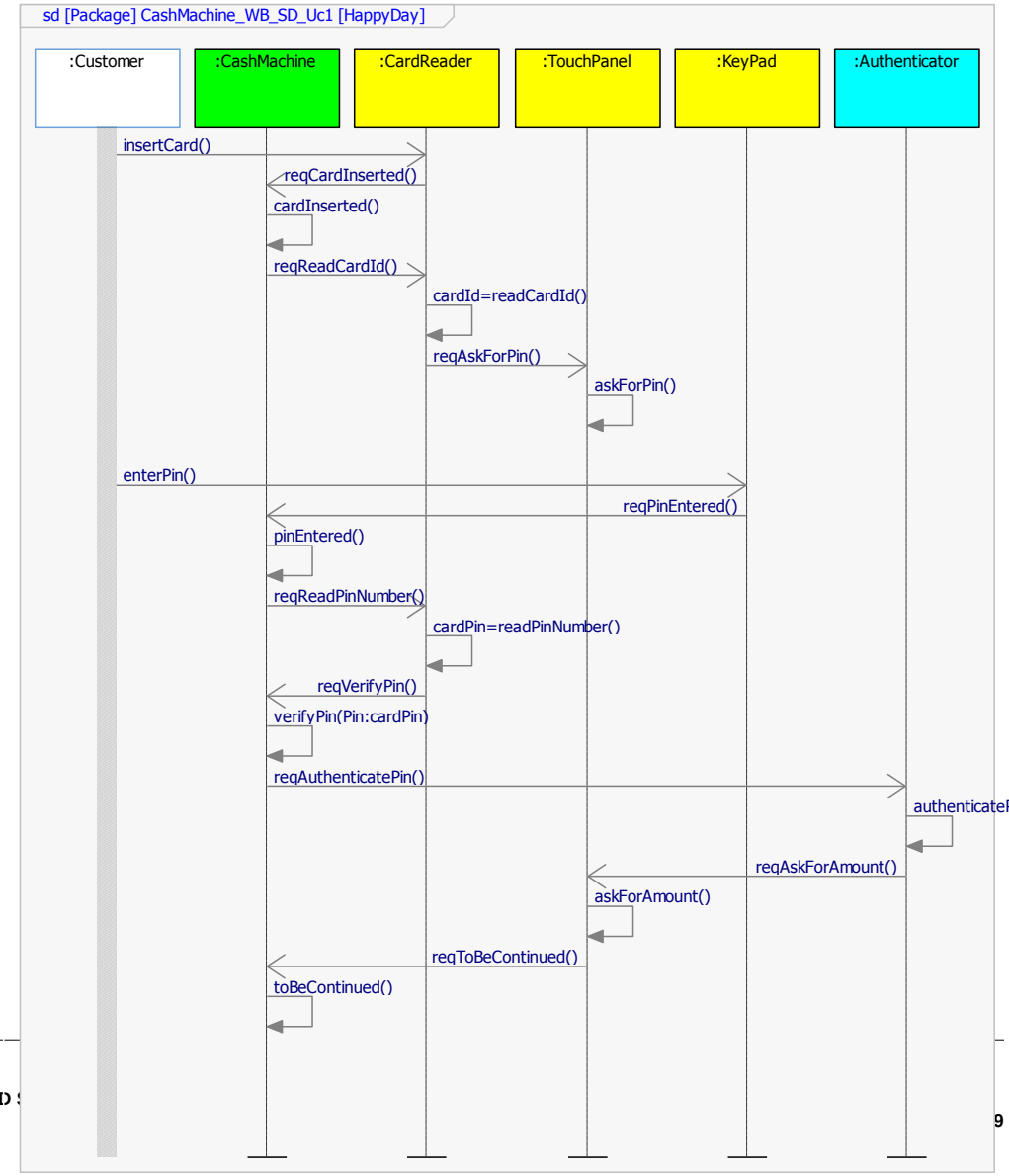
Use Cases Partitioning (level 0)



System: Cash Machine

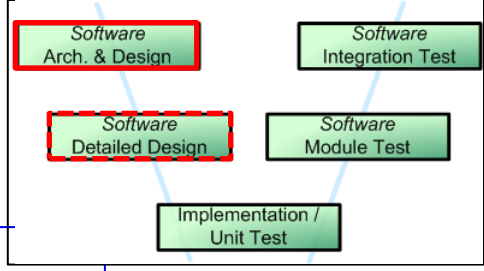
Scenario: Use Case Execution II (Happy Day – level 0)

- The Lead Architect defines the Specification
- All parties are involved
- Result of an Iterative Analysis and Design Process



System: Cash Machine

Static software architecture (level 2)



ENG.5.BP1
Architecture

ENG.5.BP1
Component

ENG.5.BP2
Allocate Req.

ENG.5.BP9
Traceability
Architecture

ENG.5.BP10
Traceability
Design

ModuleOverview

«Interface»
CardListener

«Interface»
KeyPadListene

«Interface»
TouchPanelLis

«Requirement»
Requirements.i
ID = 10

«satisfy»

CashMachine

1

Authenticator

«Requirement»
Requirements.i
ID = 7

«satisfy»

1 1 1

CardReader

KeyPad

TouchPanel

«Requirement»
Requirements.i
ID = 8

«satisfy»

«satisfy» «satisfy» «satisfy» «satisfy»

«Requirement»
Requirements.i
= 4

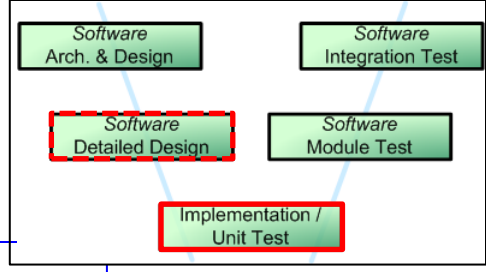
«Requirement»
Requirements.i
ID = 6

«Requirement»
Requirements.i
ID = 5

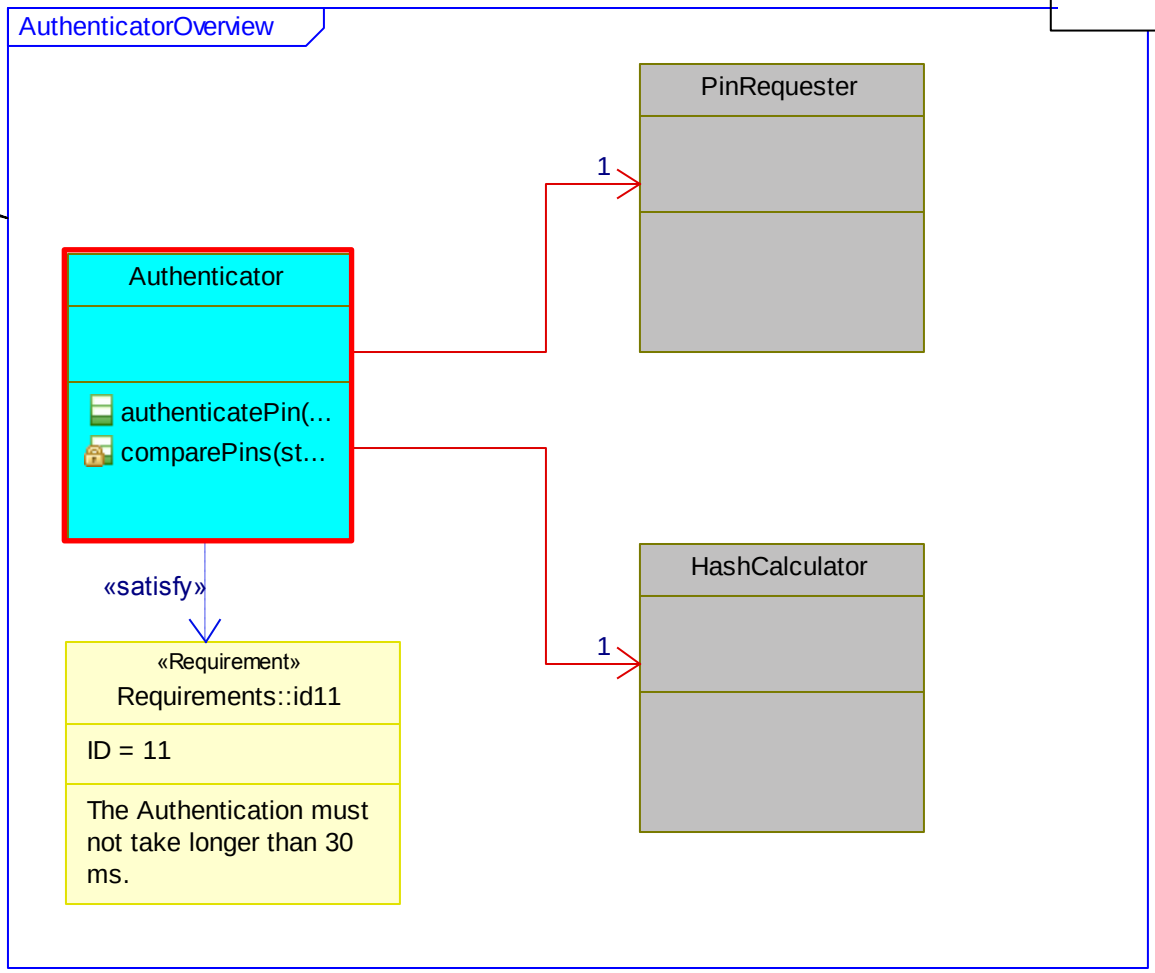
«Requirement»
Requirements.i
ID = 9

Open Main Diagram
Show Relations in New Diagram
Show Relations Between Elements
Locate in Browser Strg+L
Search...
Search Inside... Strg+Alt+F
References... Strg+R
Browse from Here
Add to Favorites Strg+D

System: Cash Machine (Not shown in talk)
Software construction (level 3)

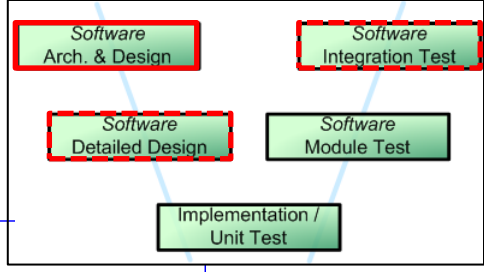


ENG.5.BP6
Detailed
Design



System: Cash Machine

Dynamic software architecture (level 2)



ENG.5.BP3 Interfaces

ENG.5.BP4 Behavior

ENG.5.BP7 Verification Criteria

ENG.5.BP8 Verify Design UC Step

ENG.7.BP3 Test Spec

ModuleBehavior_HappyDay_mod

:Customer | :CashMachine | :Authenticator | :CardReader | :KeyPad | :TouchPanel

1. Insert the card

insertCard()

cardInserted()

readCardId()

cardId()

askForPin()

If a card is inserted, it is read to identify the customer.

To authenticate the customer, a pin is requested.

2. Enter the pin

enterPin()

pinEntered()

readPinNumber()

pinNumber()

verifyPin()

authenticatePin(cardId, pinNumber)

authenticated()

askForAmount()

If a pin is entered, it is read to authenticate the customer.

The stored and hashed pin on the card is authenticated against the entered pin.

If the customer is authenticated, it is asked for the amount of money.

«Requirement» Requirements::id11

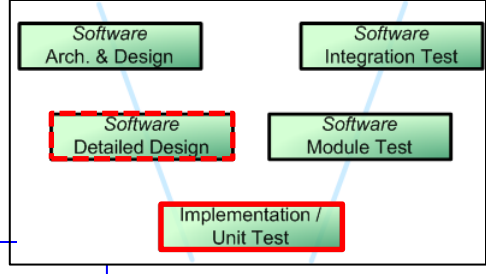
ID = 11

The Authentication must not take longer than 30 ms.

{D} {4 digit}

3. Request the money --> to be continued ...

System: Cash Machine (Not shown in talk)
Software construction (level 3)



ENG.5.BP6
Detailed
Design

ENG.6.BP9
Traceability
SW Unit

ENG.6.BP5
Verification
Criteria

ENG.6.BP10
Traceability
Test Spec

ENG.6.BP4
Develop

AuthenticatorOverview

PinRequester

1

Authenticator

authenticatePin(...
comparePins(st...

«satisfy»

«Requirement»
Requirements::id11

ID = 11

The Authentication must not take longer than 30 ms.

HashCalculator

1

Relations of Class : Authen ator in Authenticator_Int

View References

Name

in Architecture::ModuleView::ModuleBehavior_HappyDay.

Operation : authenticatePin in A thenticator

General | Description | Implementation | Arguments | Relations | T

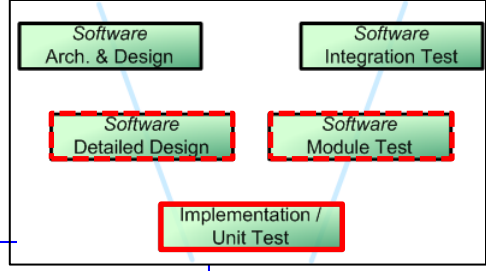
bool Authenticator_authenticatePin(const uint32 cardId,const uint8

```
uint32 storedPinHash = 0;
uint32 enteredPinHash = 0;
bool compareResult = false;

storedPinHash = PinRequester_getPin(
enteredPinHash = HashCalculator_calc

compareResult = comparePins(me, stor
```

System: Cash Machine (Not shown in talk)
Software construction (level 3)



ENG.6.BP2 Analyze SW Units

ENG.6.BP3 Prioritize

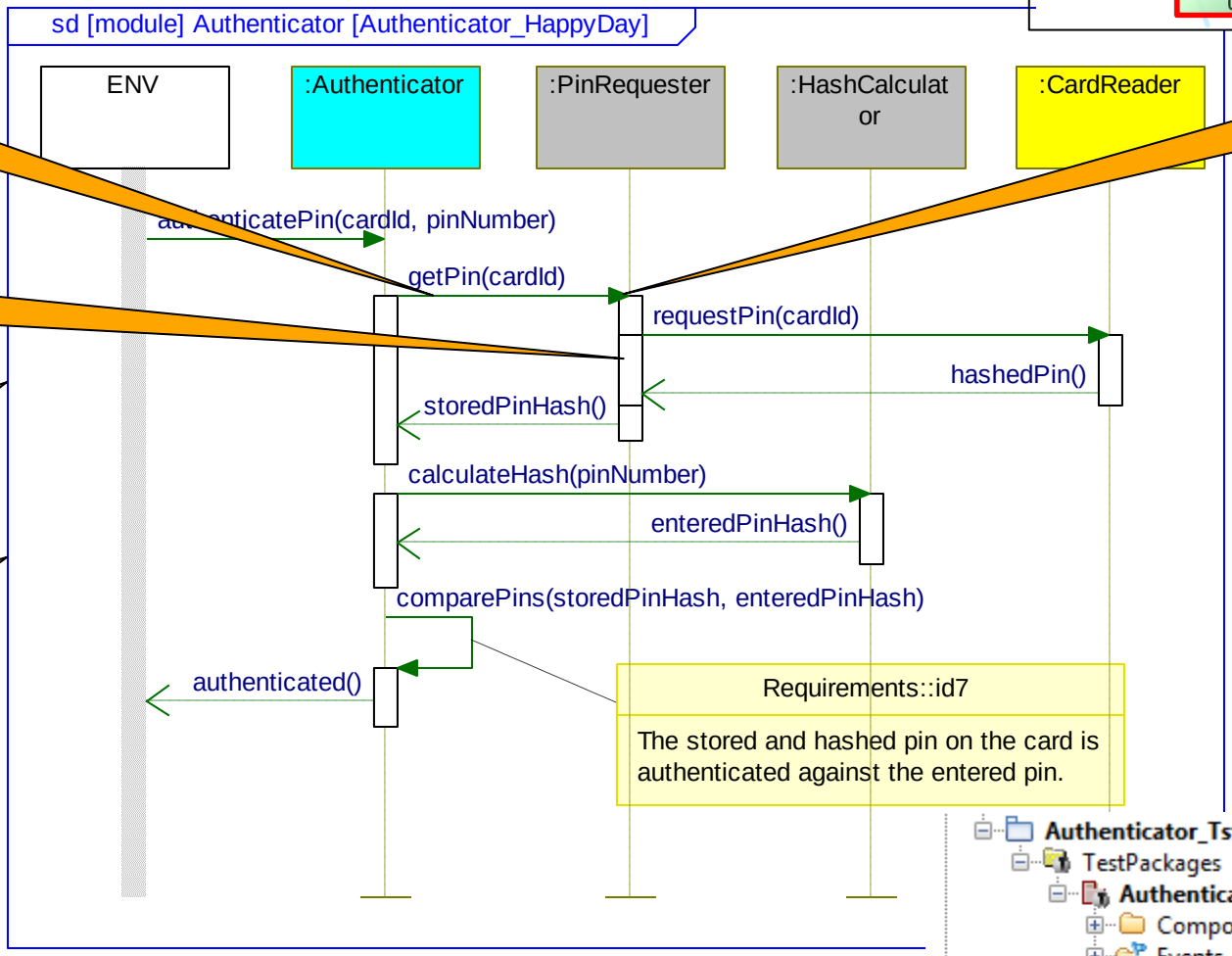
ENG.6.BP5 Verification Criteria

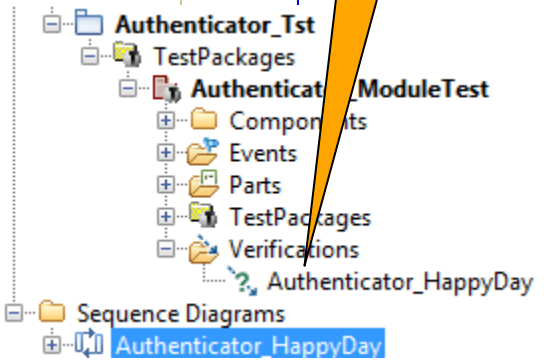
ENG.6.BP6 Verify

ENG.7.BP4 Integrate SW

ENG.6.BP8 Traceability SW Unit

ENG.7.BP7 Traceability Arch./Design





Agenda

1. Motivation
2. Introduction Tachograph
3. Project Structure Tachograph
4. Legacy Development
5. Future Development
6. Support and Barriers

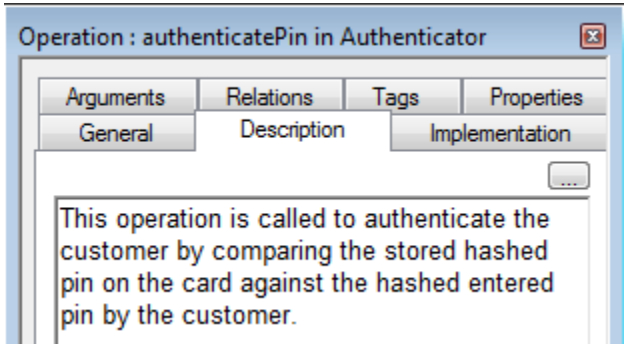
Summary

- Full Bilateral Traceability
- Full Consistency
- Improves Communication
- Validity of Documentation
- Reuse of Documentation

```
/**
 * @brief This operation is called to authenticate the customer by comparing the stored hashed pin on the
 *
 * @param cardId  [In] The cardId to get the hashed pin for.
 * @param pinNumber  [In] The entered pin number to be verified.
 */
/*#]*/
/*## operation authenticatePin(uint32,uint8) */
bool Authenticator_authenticatePin(Authenticator* const me, const uint32 cardId, const uint8 pinNumber) {
    /*#[ operation authenticatePin(uint32,uint8) */
    uint32 storedPinHash = 0;
    uint32 enteredPinHash = 0;
    bool compareResult = false;

    storedPinHash = PinRequester_getPin(me->itsPinRequester, cardId);
    enteredPinHash = HashCalculator_calculateHash(me->itsHashCalculator, pinNumber);

    compareResult = comparePins(me, storedPinHash, enteredPinHash);
    /*#]*/
}
```



5.2.1.5 Public Methods (export) for "Authenticator"

bool authenticatePin(const uint32 cardId, const uint8 pinNumber)

Description:
This operation is called to authenticate the customer by comparing the stored hashed pin on the card against the hashed entered pin by the customer.
Return type: (Type: bool)
No return description available.

Direction/Type/Name		Description/Range
In	const uint32 cardId	The cardId to get the hashed pin for.
In	const uint8 pinNumber	The entered pin number to be verified.

- Lots of Powerful Diagrams
- Performed in real Projects with Assessments

Summary

- Guided and Formal development process
  - Reducing Errors
  - Simpler Tracing
  - Growing Design
- Single Source
  - Common data storage
  - Less redundancies
  - Reliable, Consistent and Traceable
  - More re-use
- Automation
  - Executable Models

Support and Barriers

- Support for lots of Standards
  - Process Models (CMMI, (Automotive)SPICE, …)
  - Quality Charachteristics (ISO9126, ISO25010, …)
  - Safety Standards (ISO26262, ISO61508, …)

- Barriers
  - Management needs to understand the need of this technology
  - Management needs to support it (Time, Budget, Commitment)
  - Employees need to be qualified
  - Employees need to get the chance to collect experience
  - Projects need to have support from Experts (Coaching)

Thank you
for your attention!