Testing Solutions with UML/SysML

Matthew Hause, Jon Holt

Agenda

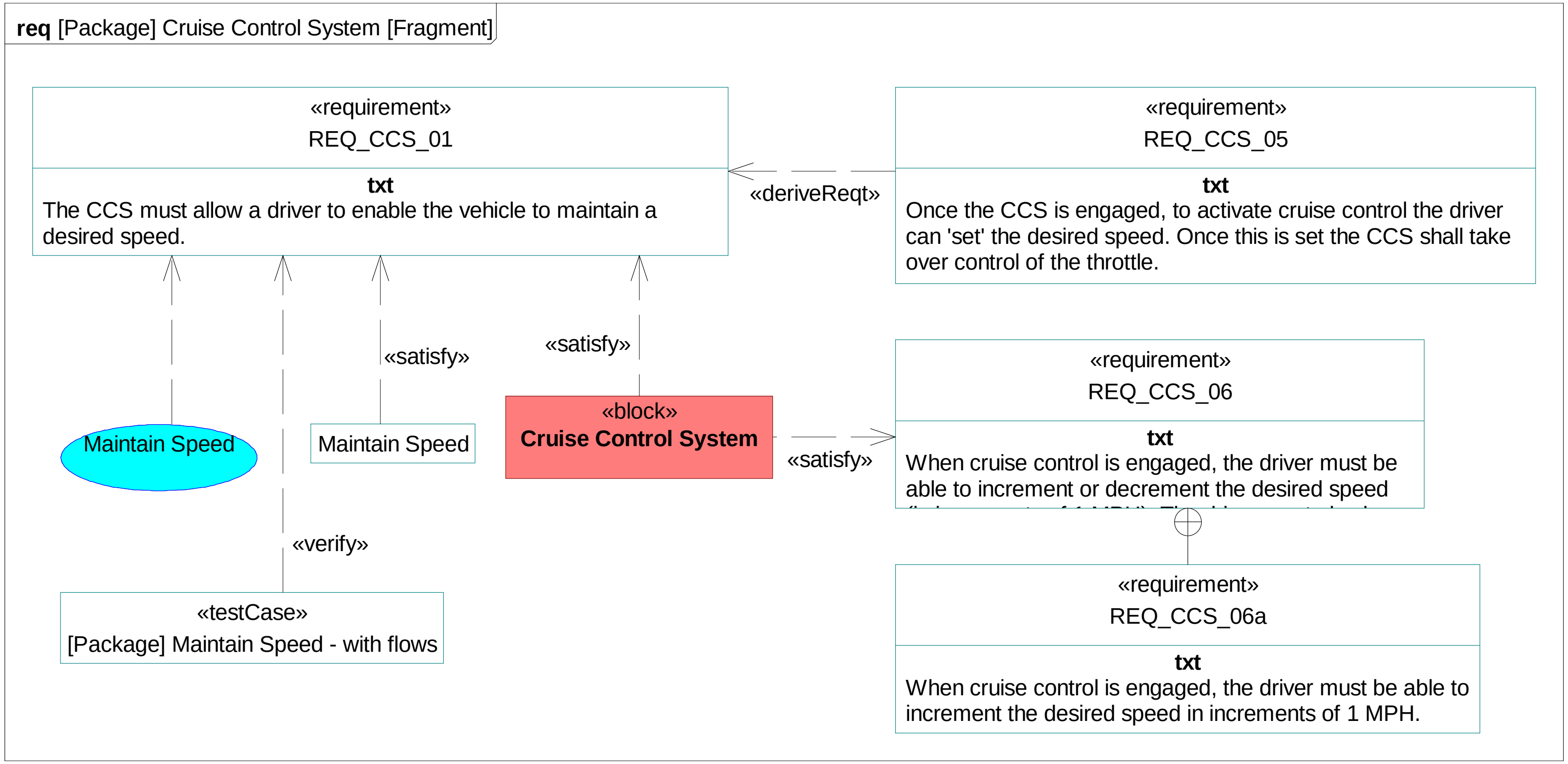
- Safety Critical Systems and Testing
- SysML and Testing
- Rail Case Study
- Automotive Case Study
- Conclusion

Safety Critical Systems and Testing

- The Problem
  - We are running out of time.
    - Systems are becoming more complex.
    - Time to market is decreasing.
    - Most important, safety critical systems must be properly tested.
- The Solution
  - SysML and Model Based Systems Engineering (MBSE)
    - SysML supports modelling of the requirements, behaviour, structure, etc.
    - Can aid the verification, validation, and simulation of software, firmware and mechanical systems.
    - SysML can provide a practical solution of integrating testing into an MBSE approach.

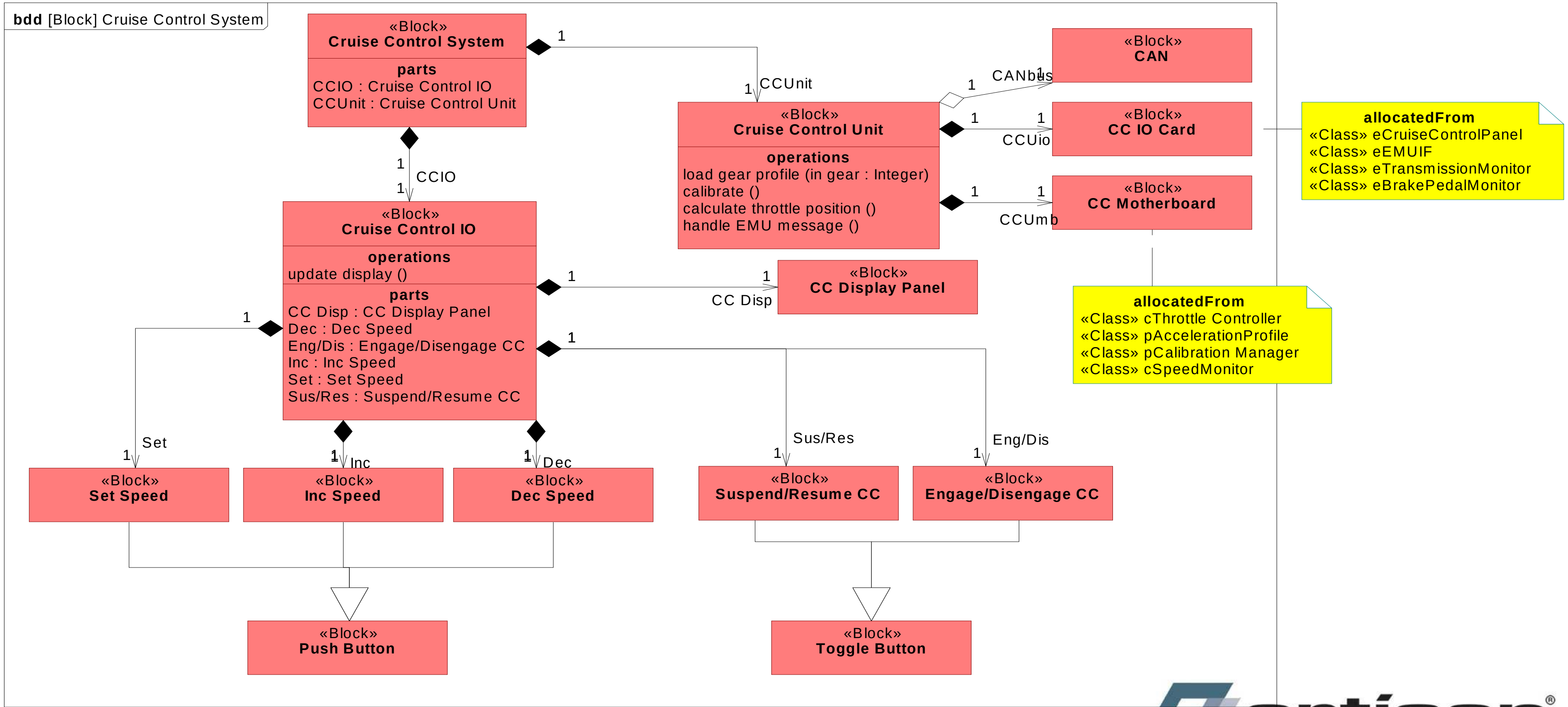
SysML Requirements Flow Down

- Requirements are integrated into the model
- Supports requirements hierarchies
- Verify, Satisfy, Derive, Trace and Refine relationships



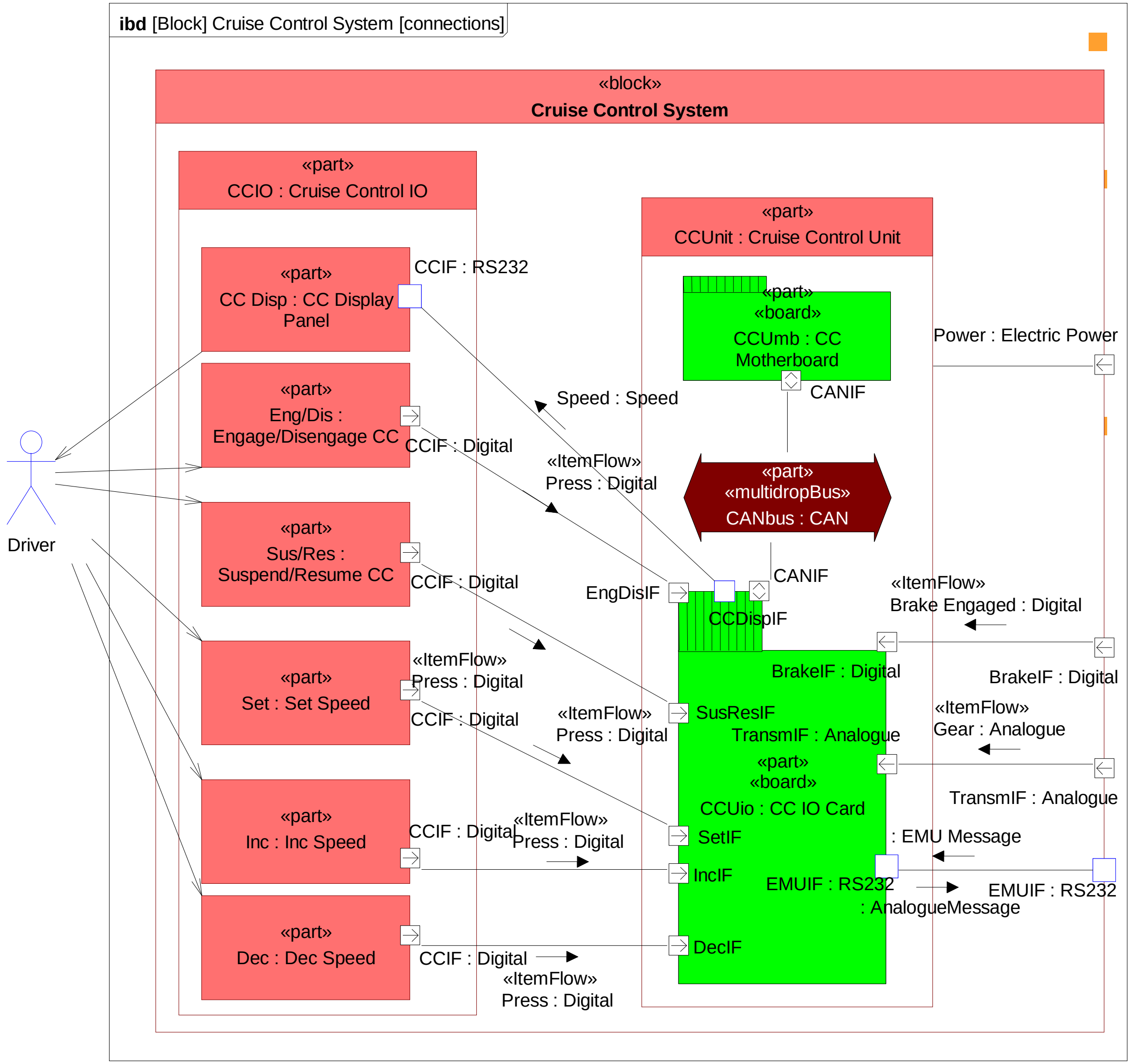
Physical and Logical Structural Modelling

- Parts shown by diamond notation, or by Parts Compartment
- Compartments show properties of the block
- Attributes are inherited from the parent



your model for success

Internal Block Diagram



Shows parts (structural children) …

… and ports (interaction points on blocks and parts)

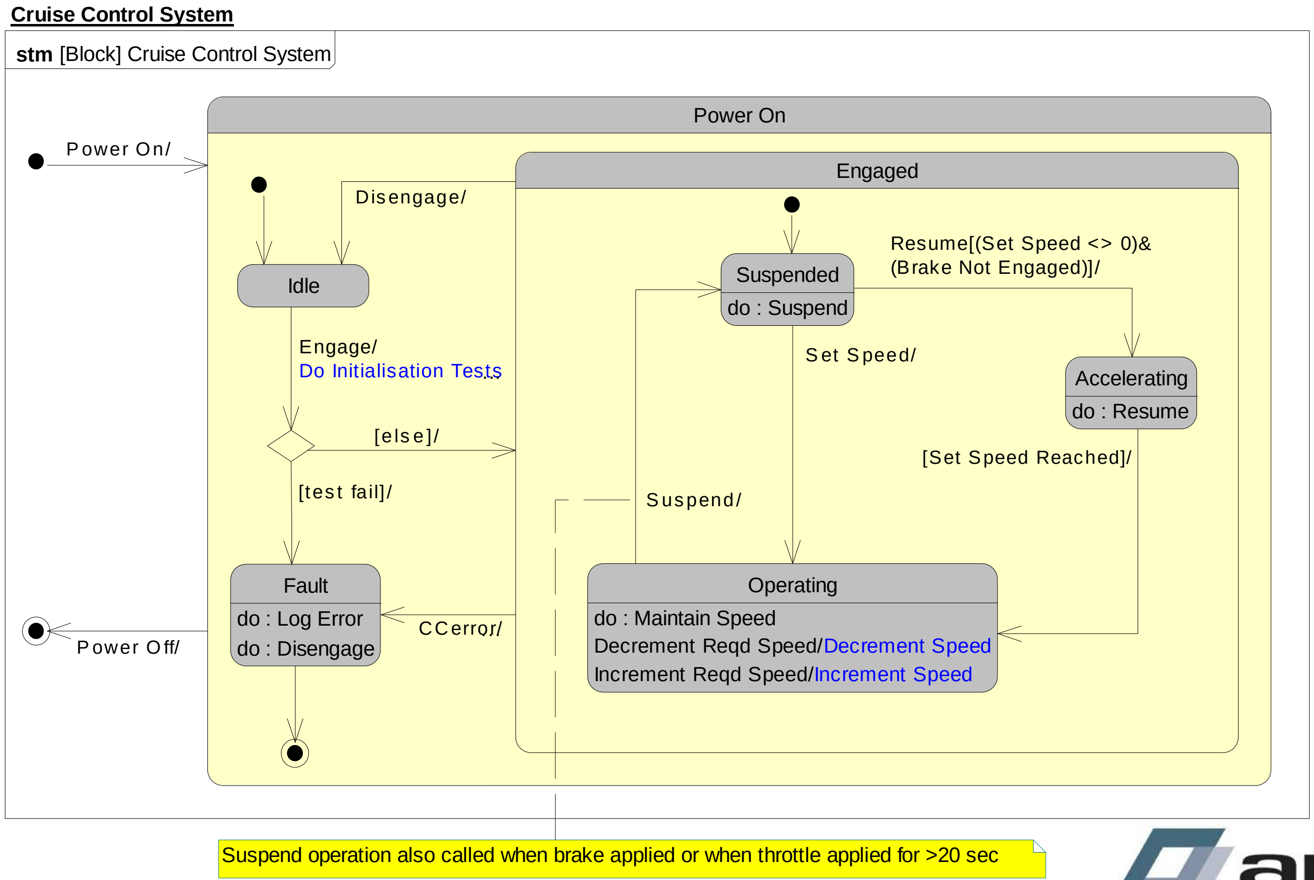
- Supports integration of behavior and structure

Port types

- Standard Ports
  - Specify a set of operations and/or signals
  - Typed by a UML interface
- Flow Ports
  - Specify what can flow in or out of block/part
  - Typed by a flow specification

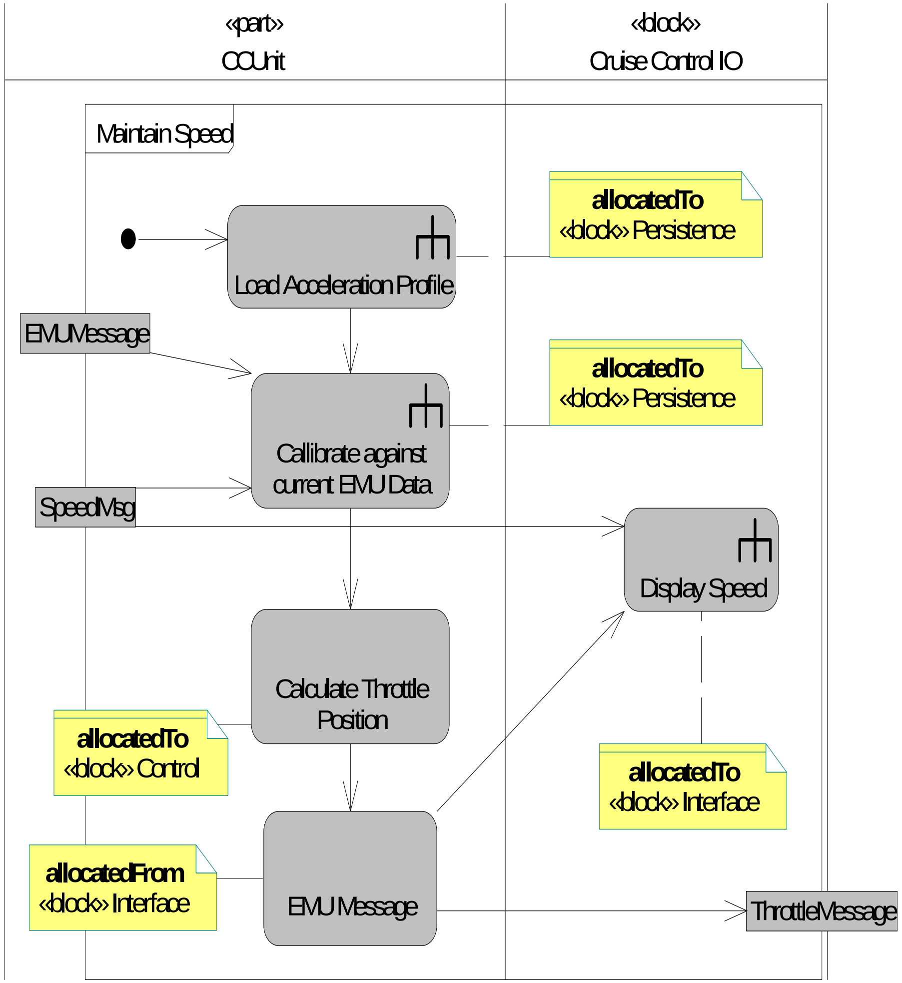
State Modelling

- Models Complex state-based behaviour of system elements
- Atomic, composite and concurrent states
- Conditional behaviour



Activity Modelling

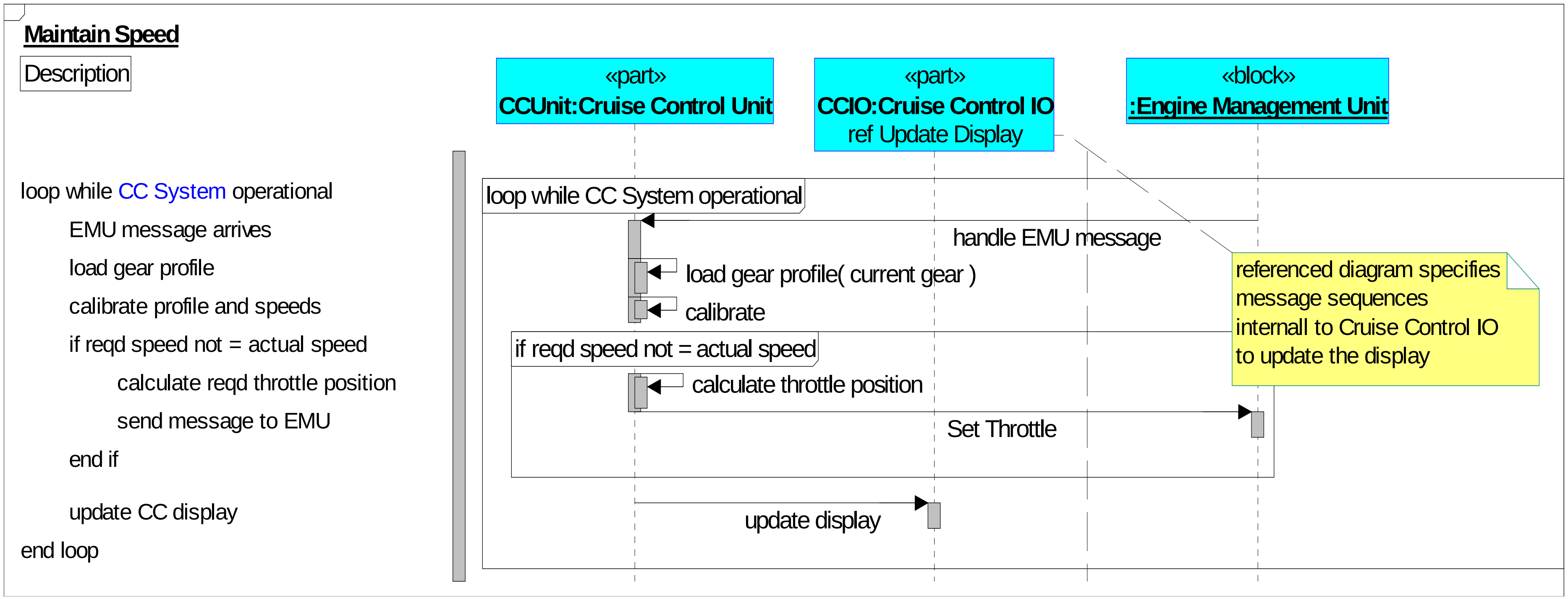
- Allows low level behavioural modelling of the system.
- SysML Additions
  - «streaming» activities consume inputs after initialization
  - «continuous» flows
  - Probabilities
  - Interruptible regions



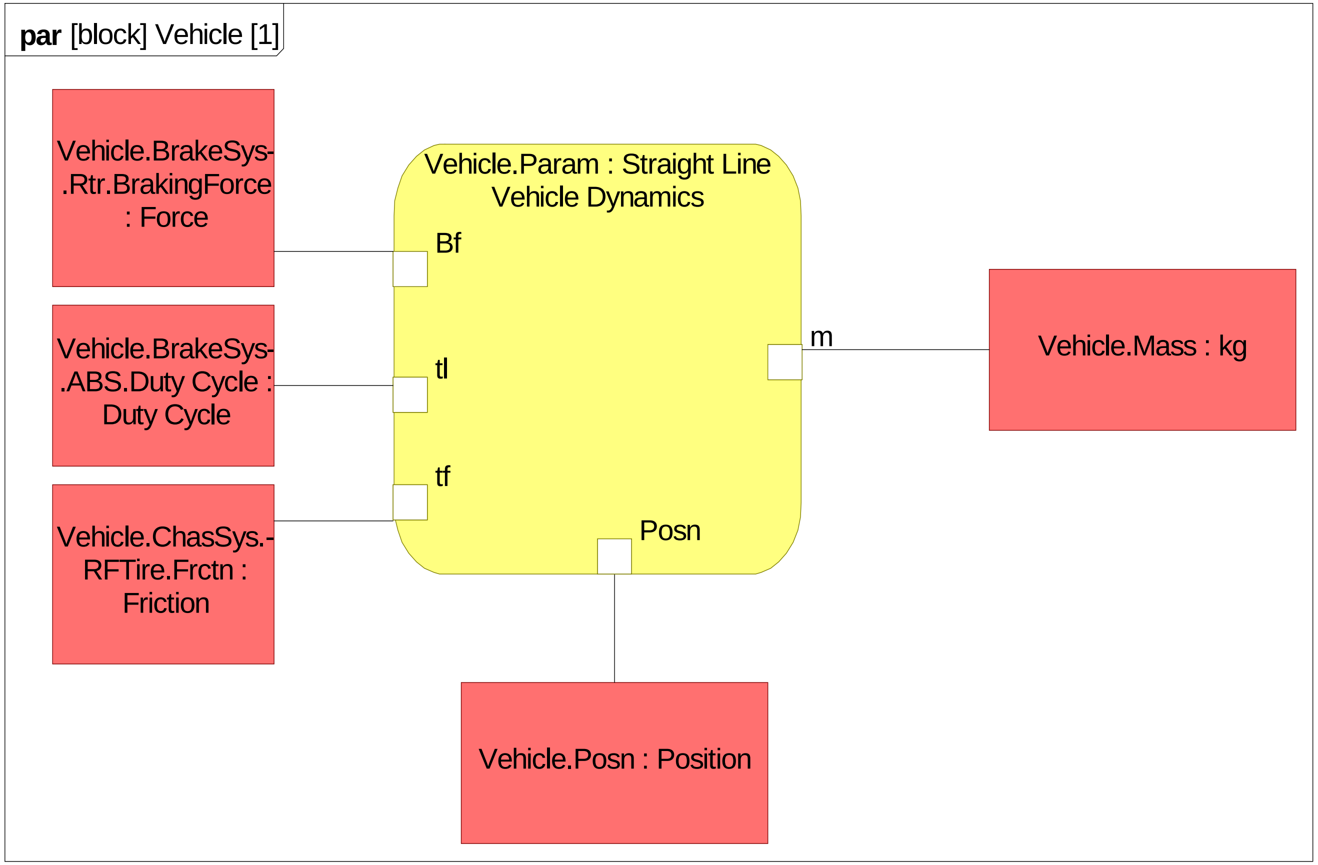
artisan®
your model for success

Sequence Diagrams

- Represents the interaction between collaborating parts of a system
- Supports conditional behaviour, loops, parallel, etc.
- Interactions can be messages, operations, etc.
- Can be used to define test cases

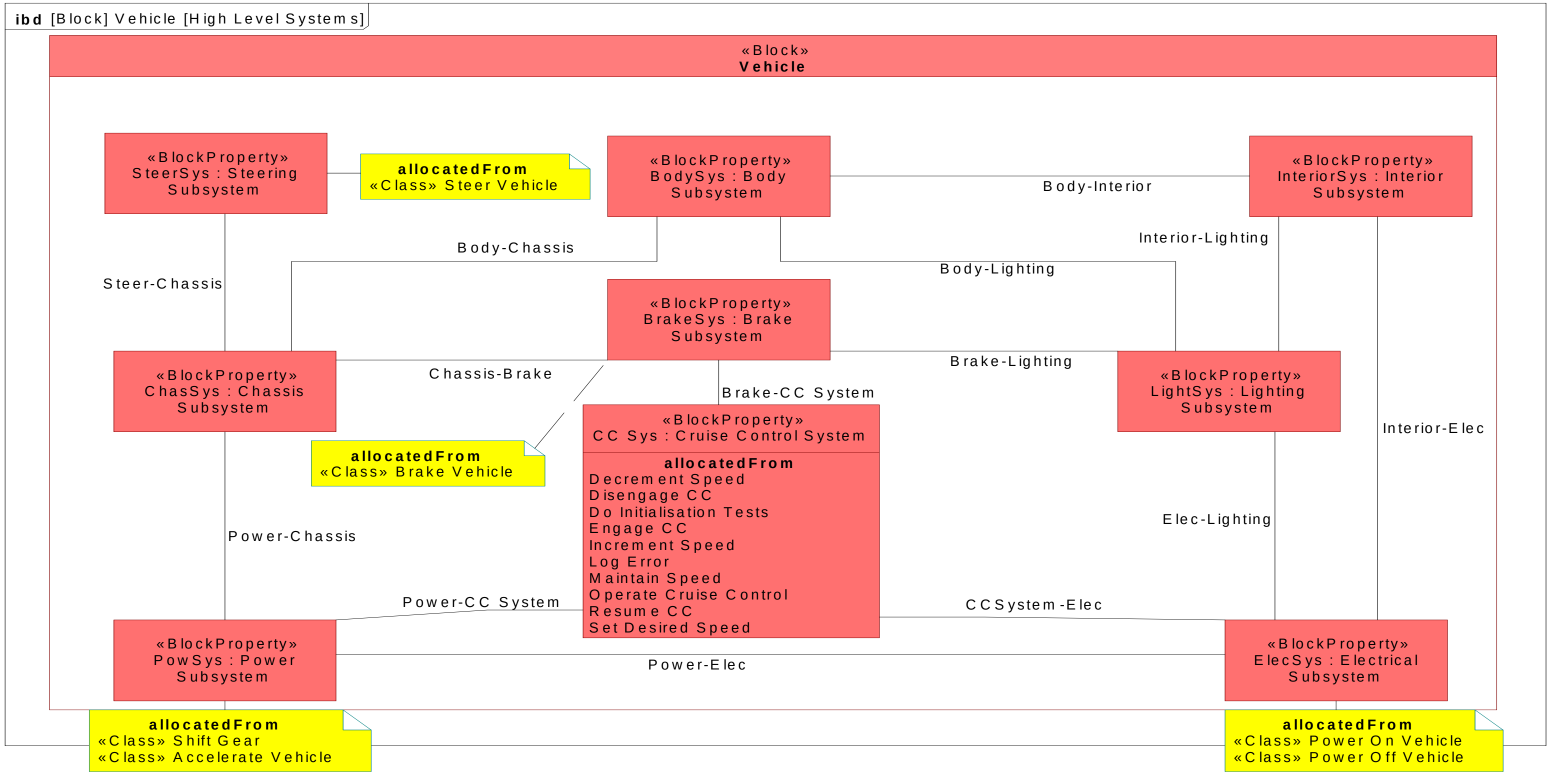


Parametric Diagram



- Defines parametric relationships between properties
- Constraint can be shown in compartment or in attached note

Allocation Shown on IBD

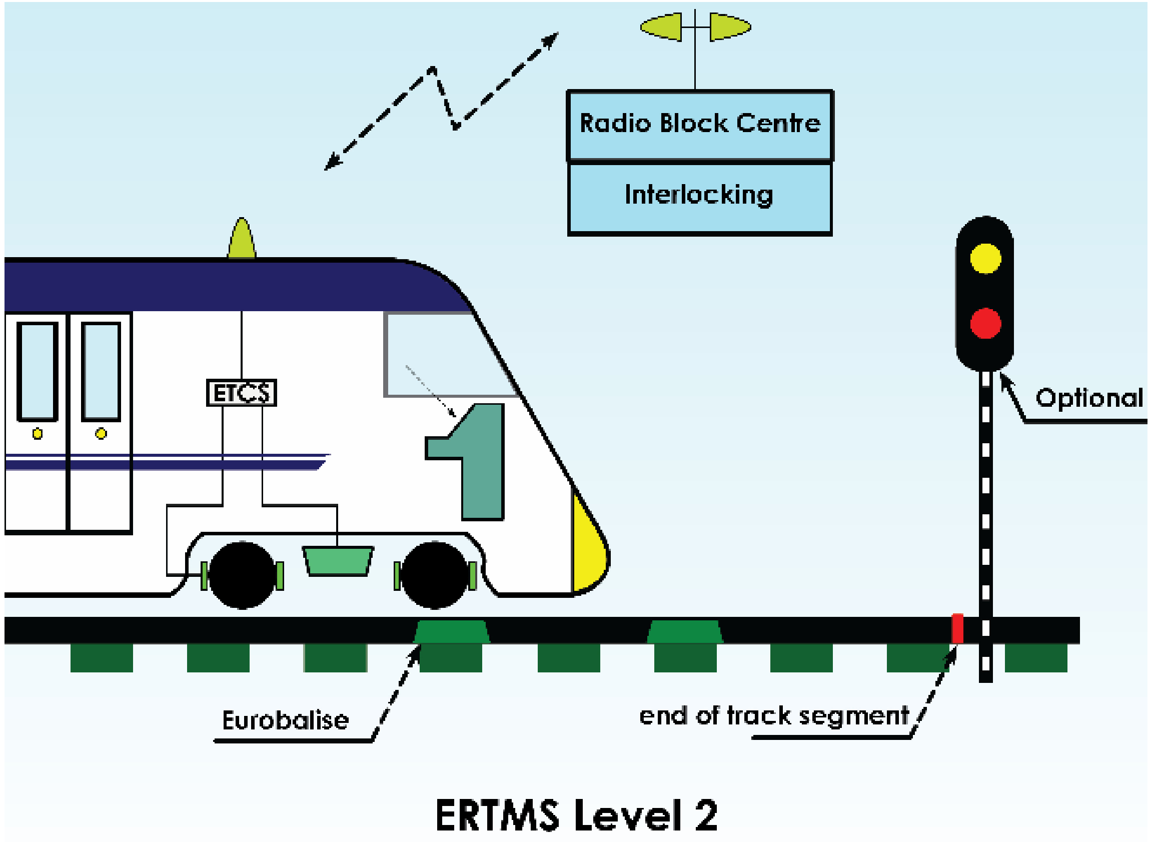


- Allocations
  - Allocation of activity invocation to parts shown in allocatedFrom compartment.
  - Allocation of activity object nodes to Item flows shown via call-outs.

Rail Case Study

Radio Block Control (RBC) Validation Scenario

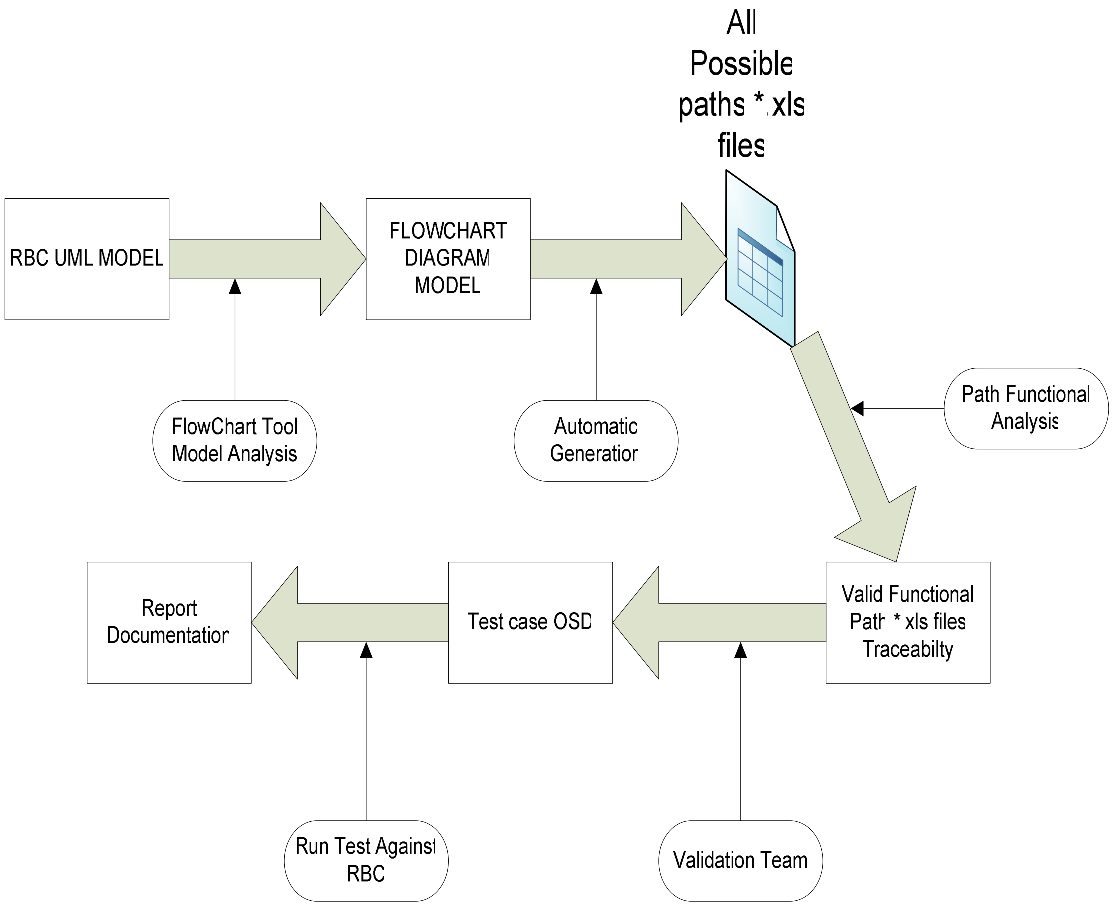
- The RBC is responsible for continuous speed supervision and movement authority of the train. Safety Integrity Level 4 Product.
- Eurobalise determines the Train Location and sends it to the RBC



artisan® your model for success

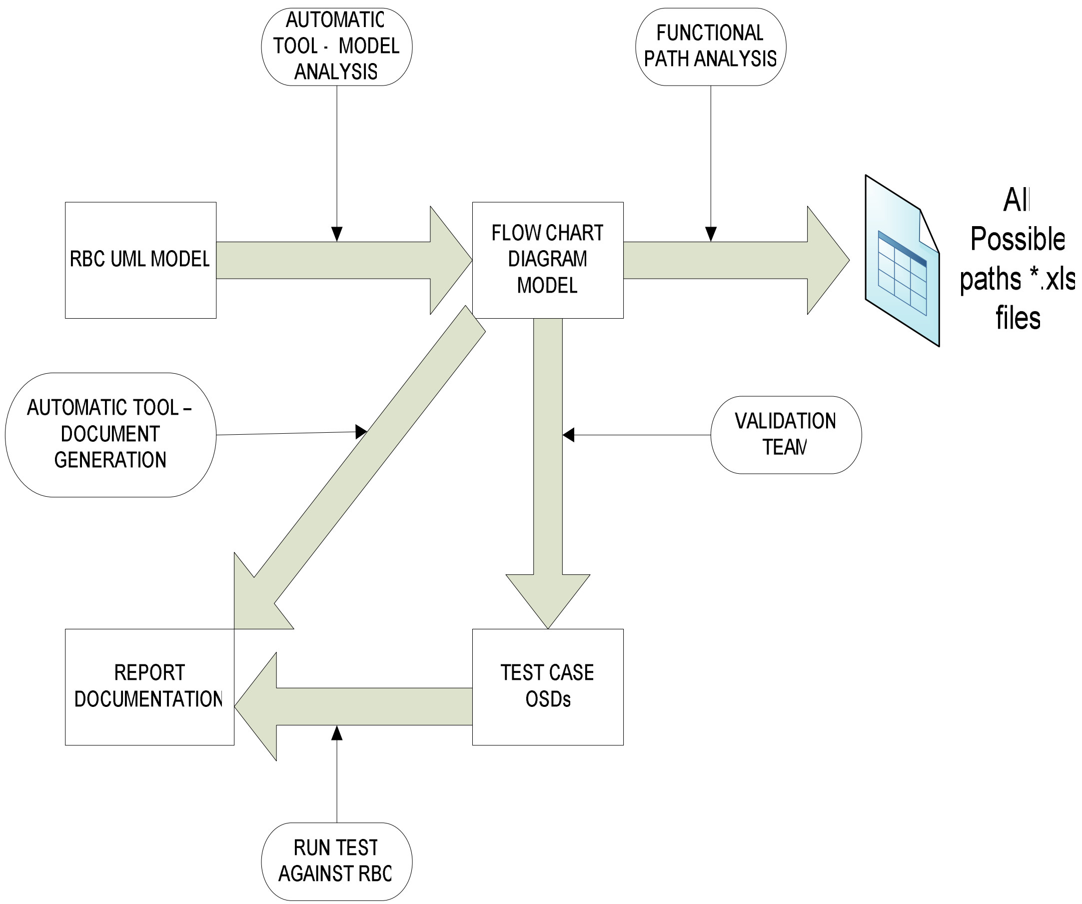
Traditional RBC Validation

- Use case requirements model defined in Artisan
  - Analysis and design models used sequence and state diagrams
- Flowcharts constructed from the model that capture all possible execution paths
  - Sequence diagrams (SD) generated for all valid paths.
  - Python scripts generated from the SD
  - Scripts exercised SUT
  - Not flexible when model changes



Automated RBC Validation

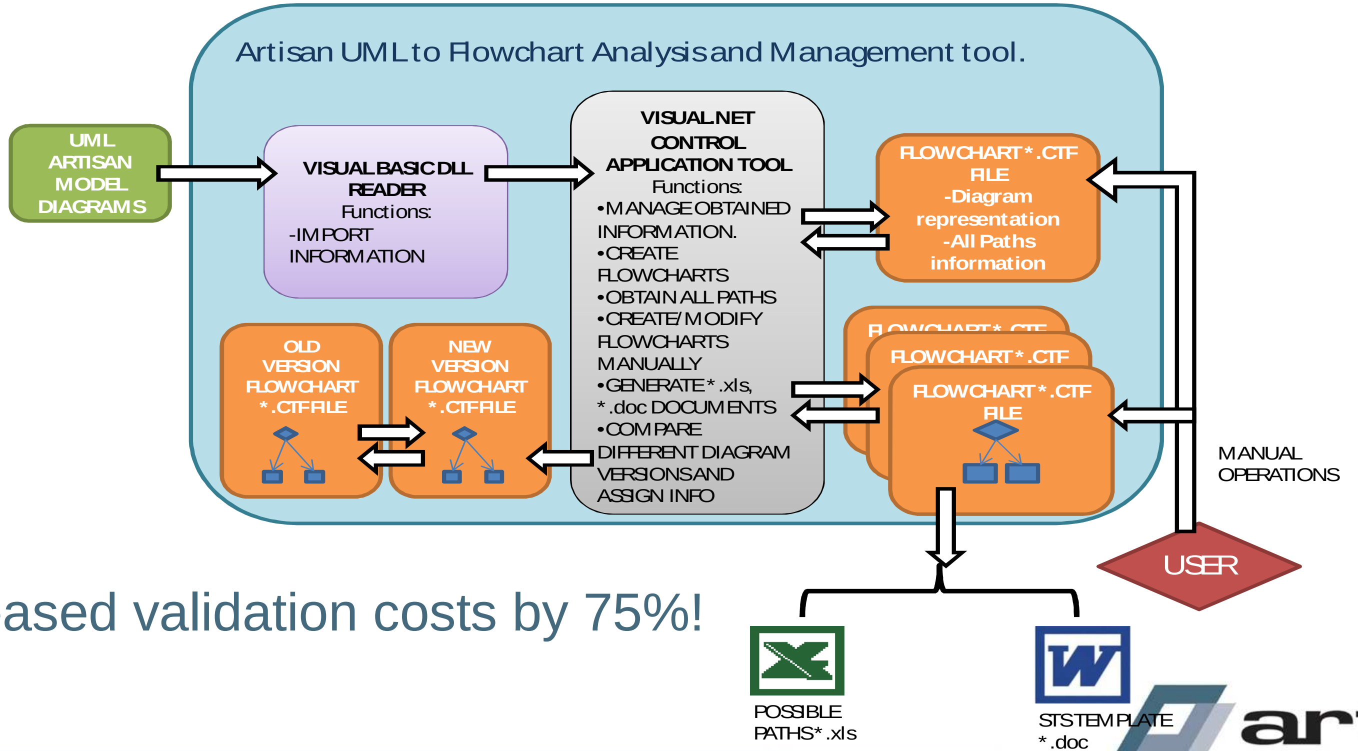
- Eliminates manual work
  - Excel files created automatically. Used as evidence.
- Reduces human errors
  - Originally the files were hand-coded
- Decreases the number of files used.
- Enforces design standards.



artisan® your model for success

RBC Tool Integration

- VB DLL reader imports from the model.
- .NET application manages imported information
- Provides execution paths
- Automatically produces documentation



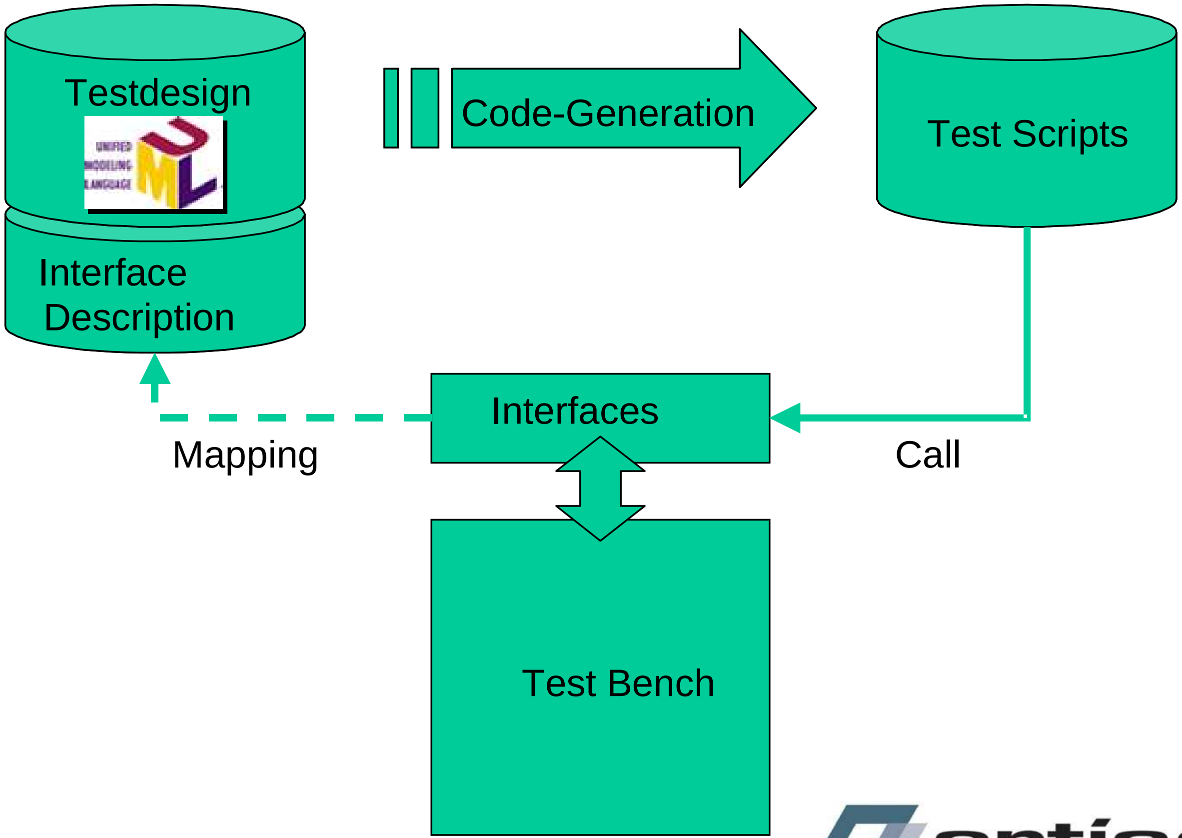
- Decreased validation costs by 75%!

artisan®
your model for success

Automotive Case Study

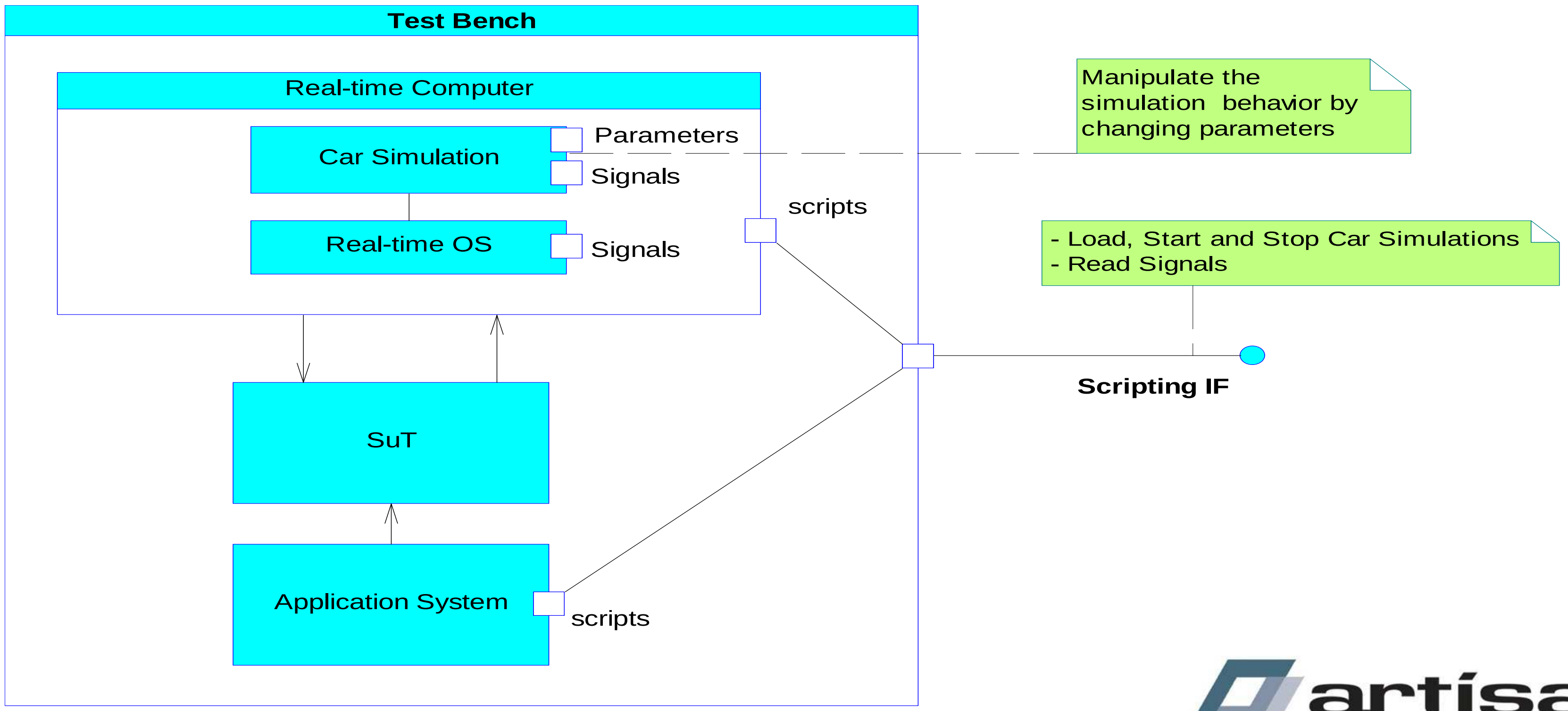
Automotive Platform Independent System Test

- Test cases derived from use cases and modelled in sequence diagrams.
- White box test cases can be derived from activity diagrams
- Test cases automatically generated through a code generator and based on the model.
- Executed on the test rig.
  - Results captured.
  - Documentation generated



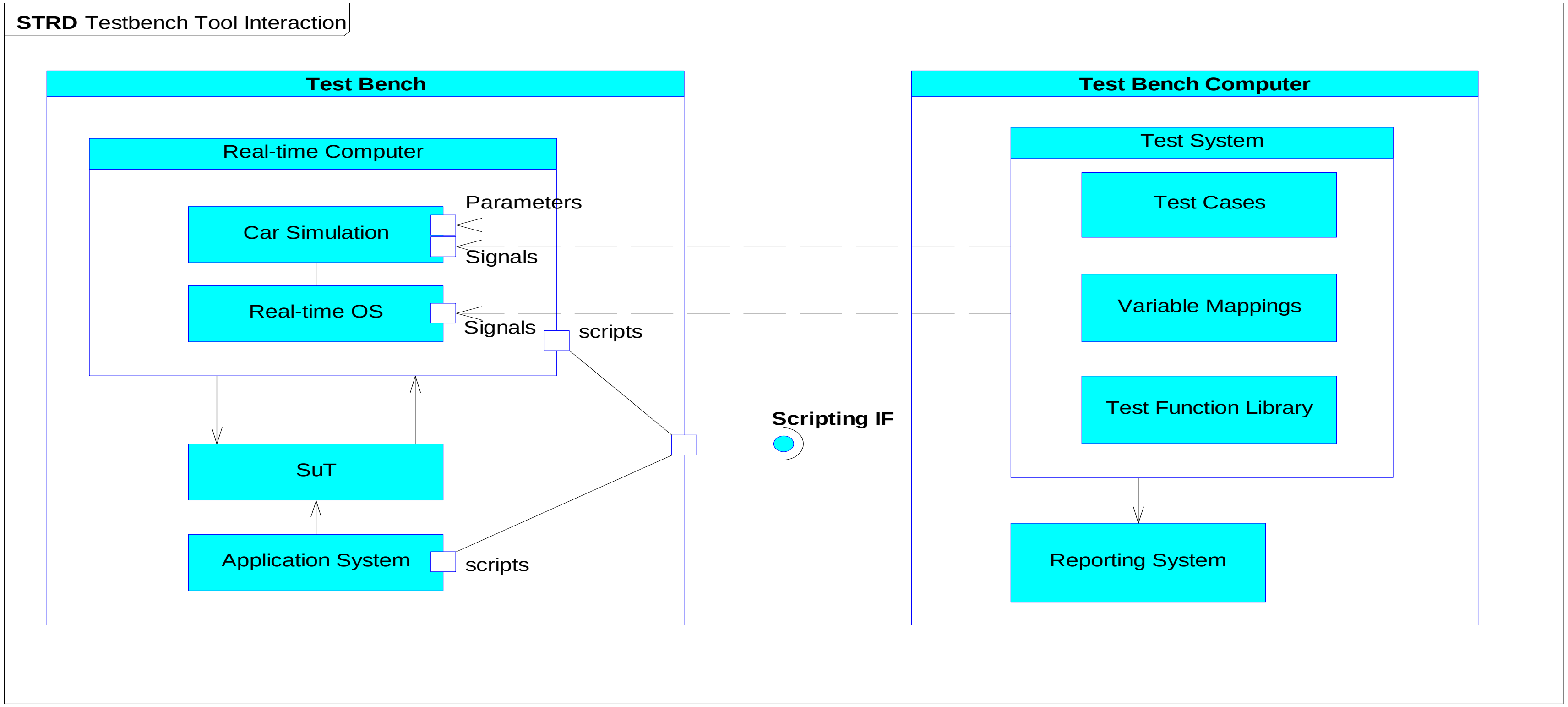
Testbench Overview

- Provides a configurable remote test of the car thru the I/F
- Interface to RTOS and AppUT through pre-defined test interface
- Allows mapping of the test to the equipment to be used separately



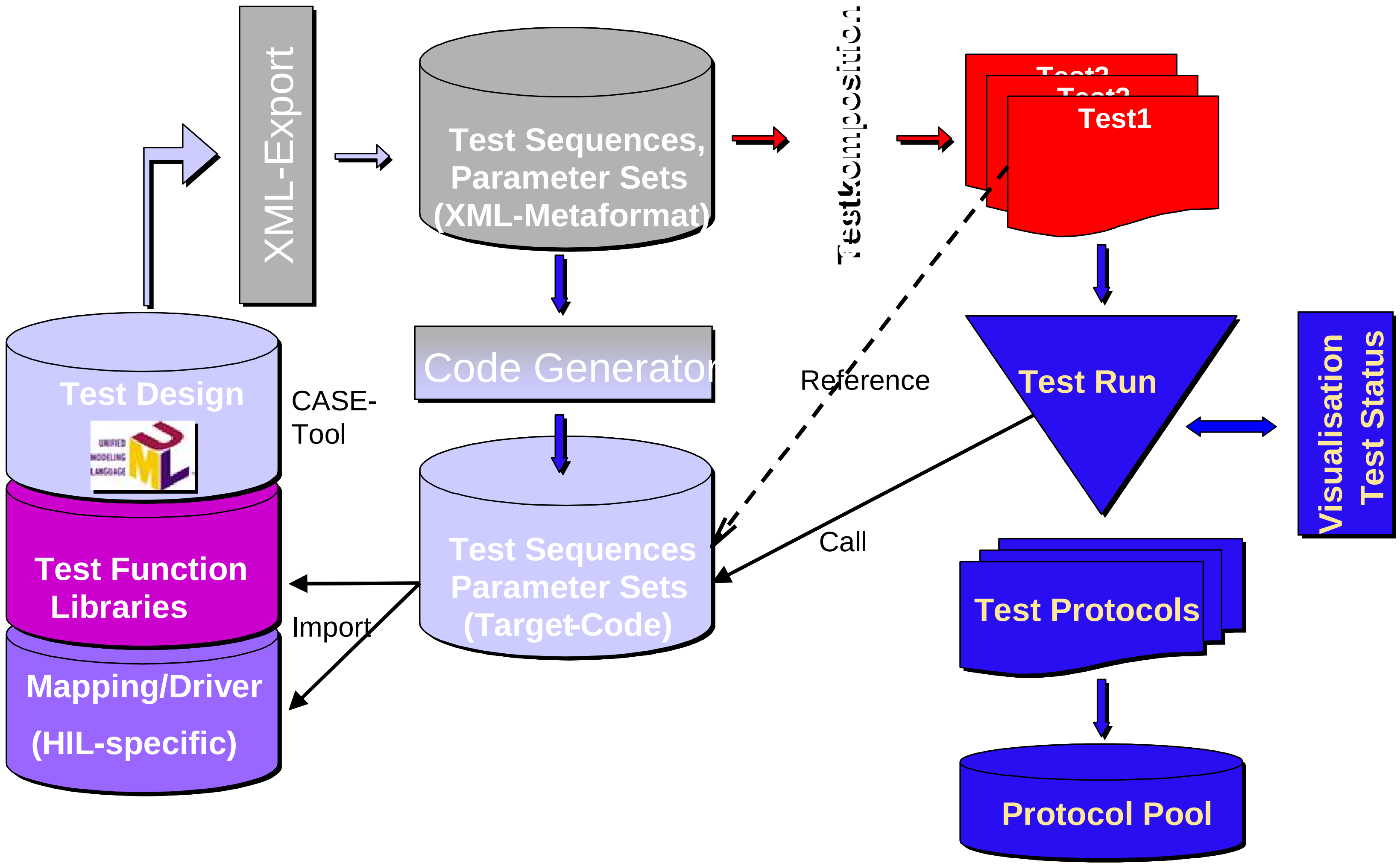
Testbench Tool Interaction

- Dependencies show mapping between variables in test descriptions and parameters and signals performed using name matching
- Library holds drivers for each platform.



artisan® your model for success

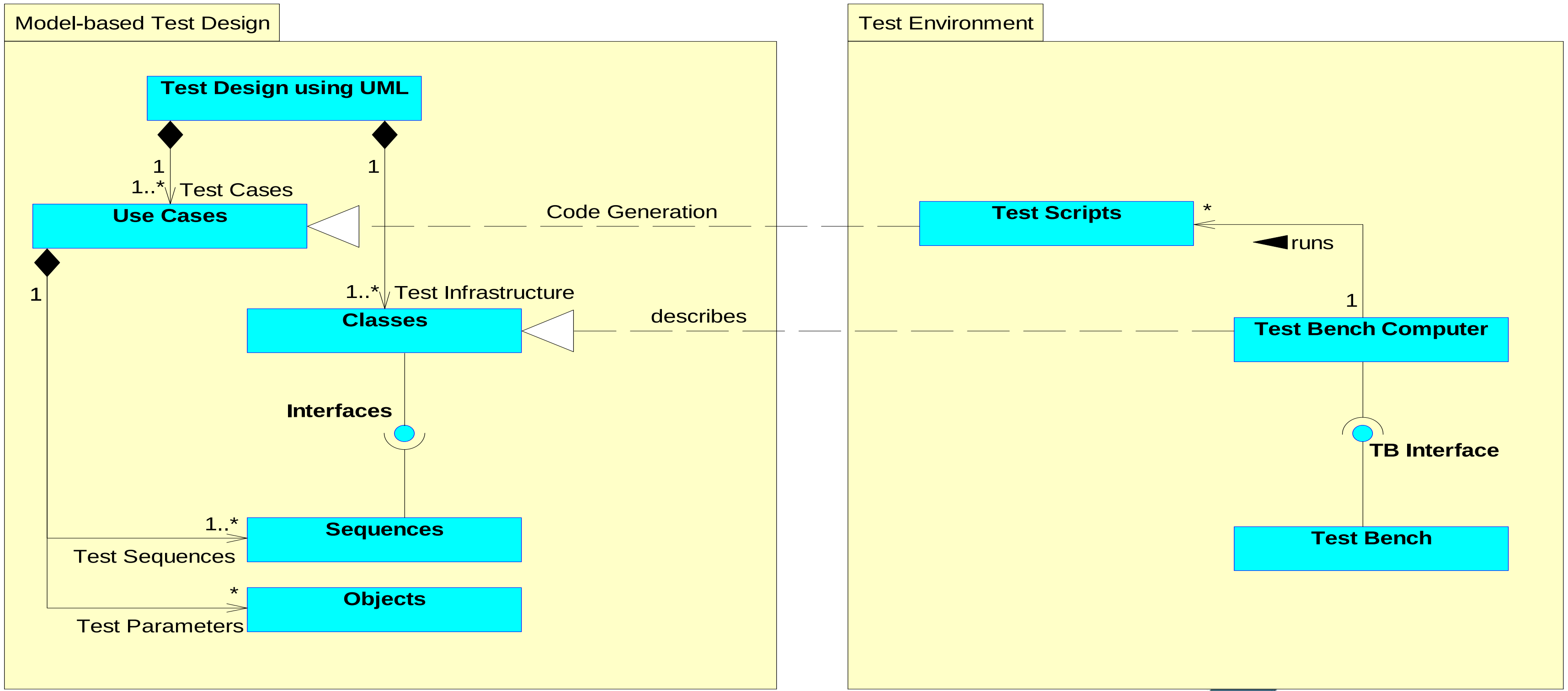
Automotive Tool Integration and Workflow



artisan your model for success

Description of Automotive Model-based Testing

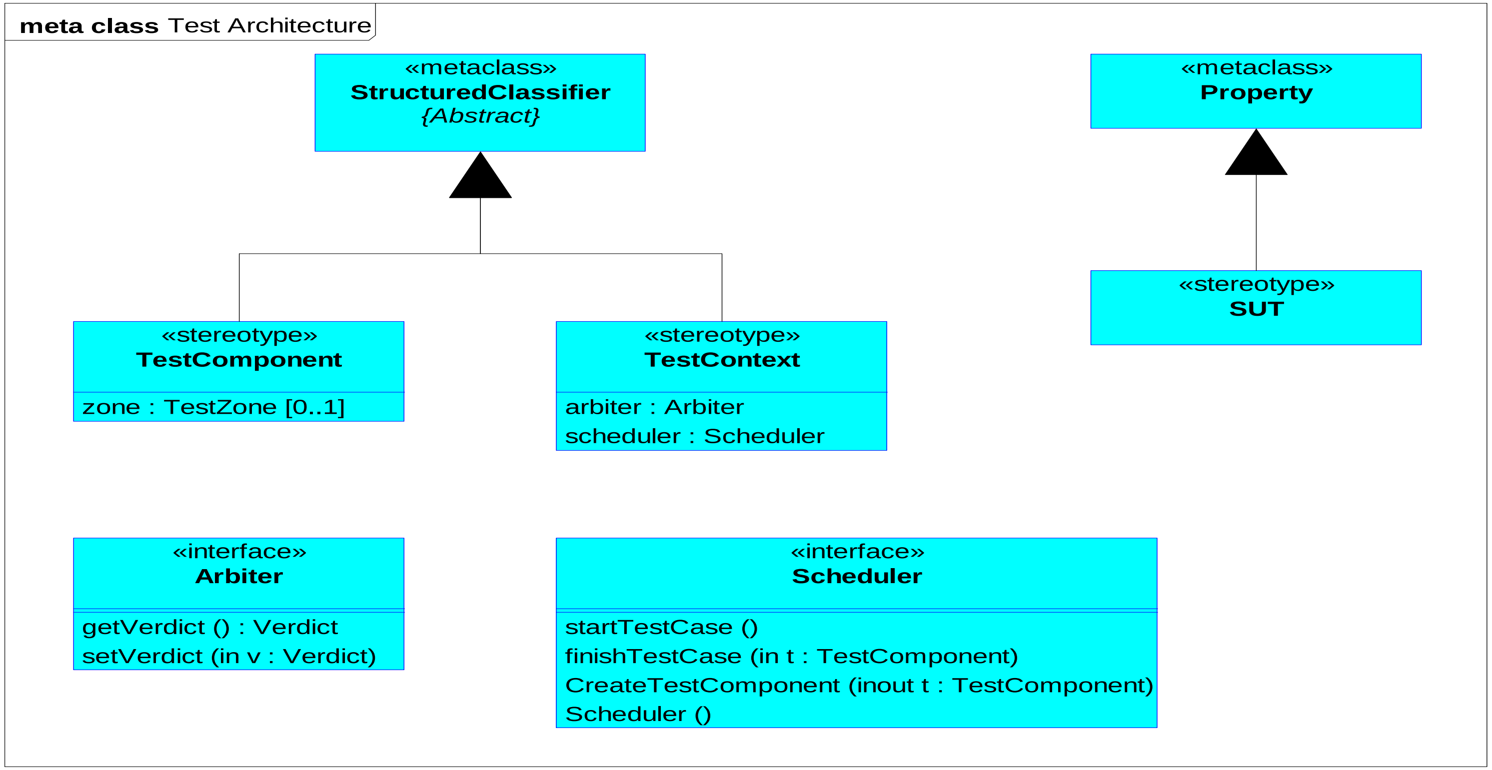
- Shows how test scripts are abstracted in MBD
- Scripts cross all levels; the models contain these perspectives.



Standards for Model-based Testing

UML 2.0 Testing Profile (U2TP) Test Architecture

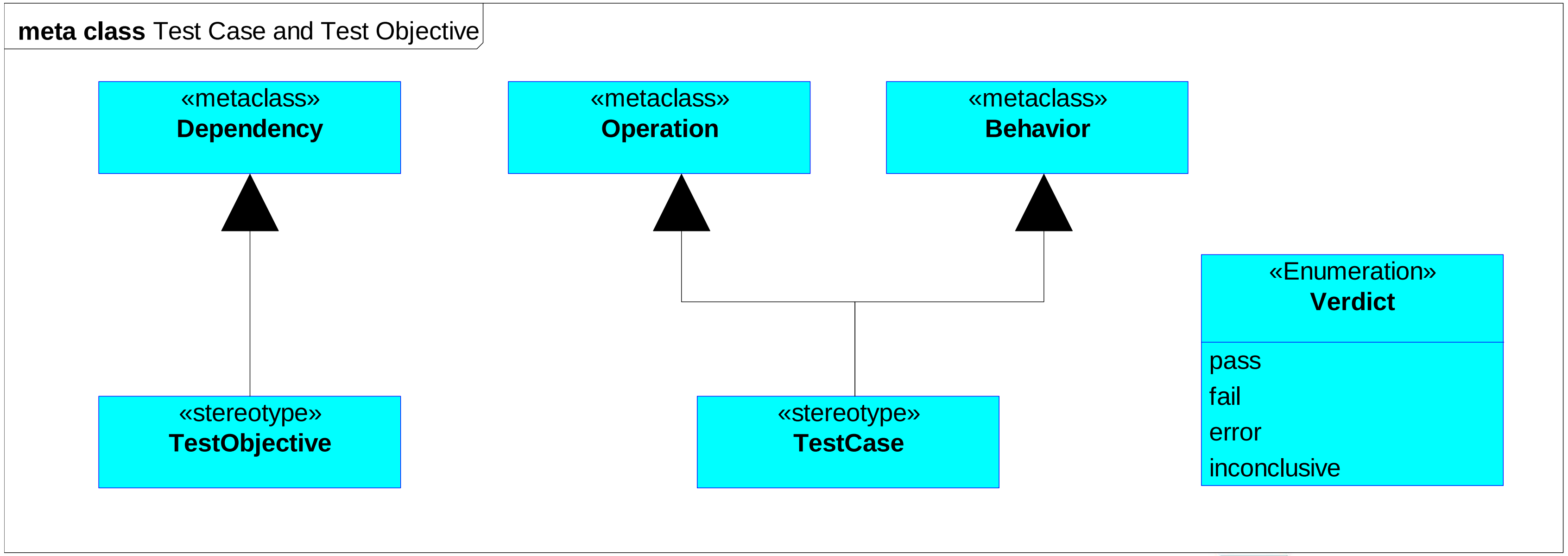
- Created in 2005 for modelling tests
- Both case studies use this approach
- Can make use of the SysML testcase stereotype.



artisan® your model for success

Test Case and Test Objective

- Test context contains an arbiter and a scheduler (Previous slide)
- Test case can be an operation or a behaviour
- Test results can be pass, fail, error or inconclusive
- Tests generally run using a sequence diagram
- Automating test runs not included in U2TP, as it is tool dependent



Conclusion

- Test plans can and should be modelled
- A model based approach provides the logical progression from:
  - Test Plan (and models) to,
  - Test Suite(s) (and models) to,
  - Test Case(s) (and models) to,
  - Test Scripts (and models)
- Building models to test models and systems is more natural
- ROI has been proven – in this instance saving 75% on testing

artisan®
your model for success

Discussion and Comments

artisan®
your model for success