



Presented by:
Sean Wyatt

Content

- | | |
|---|--|
| 1 | Need and Challenges for Function Developers |
| 2 | What is RT2? |
| 3 | Test Case Modeling
Test Case Execution
Test Case Assessment and Analysis
Test Report Generation |
| 4 | Benefits of RT2 |

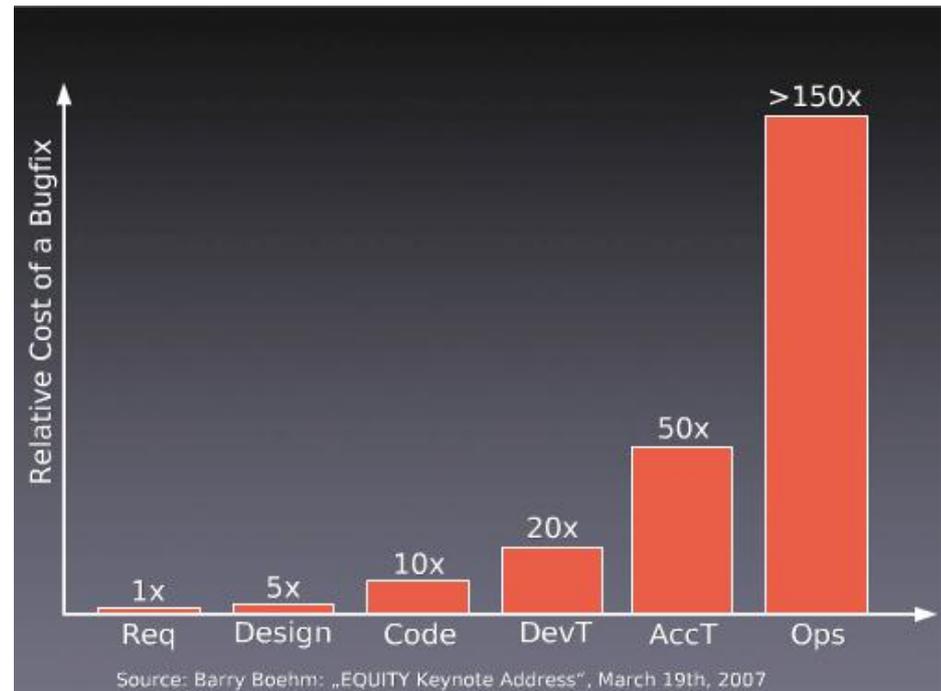
Content

1	Need and Challenges for Function Developers
2	What is RT2?
3	Test Case Modeling Test Case Execution Test Case Assessment and Analysis Test Report Generation
4	Benefits of RT2

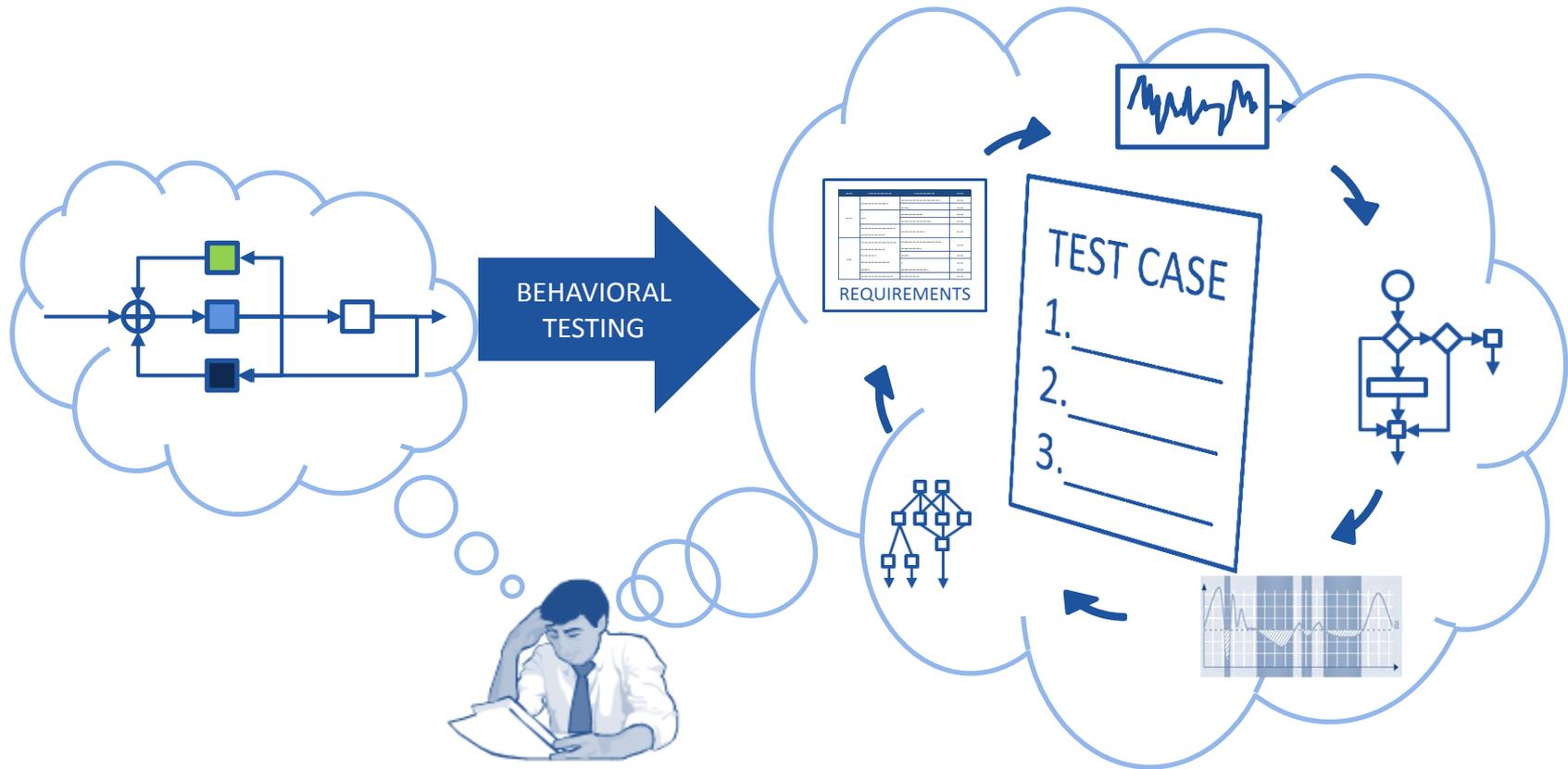
The Need

- Software defects originate from all stages of the development process
 - (Model-based) function development
 - Software code generation
 - Software build and integration
 - Hardware development
 - ...
- Errors are often found far too late
 - High pressure and risk at the end of the engineering cycle

..... resulting in high cost of fixing errors



Barry Boehm, Software Engineering Economics



Challenges Summarized

- Tracking requirements coverage and changes to their dispositions
- Deciding how to stimulate the function
- Creating test variants of the stimulations in a systematic and repeatable manner
- Assessing the results
- Creating reports with rich details that support ISO 26262:6 and 8

ETAS' solution is RT2

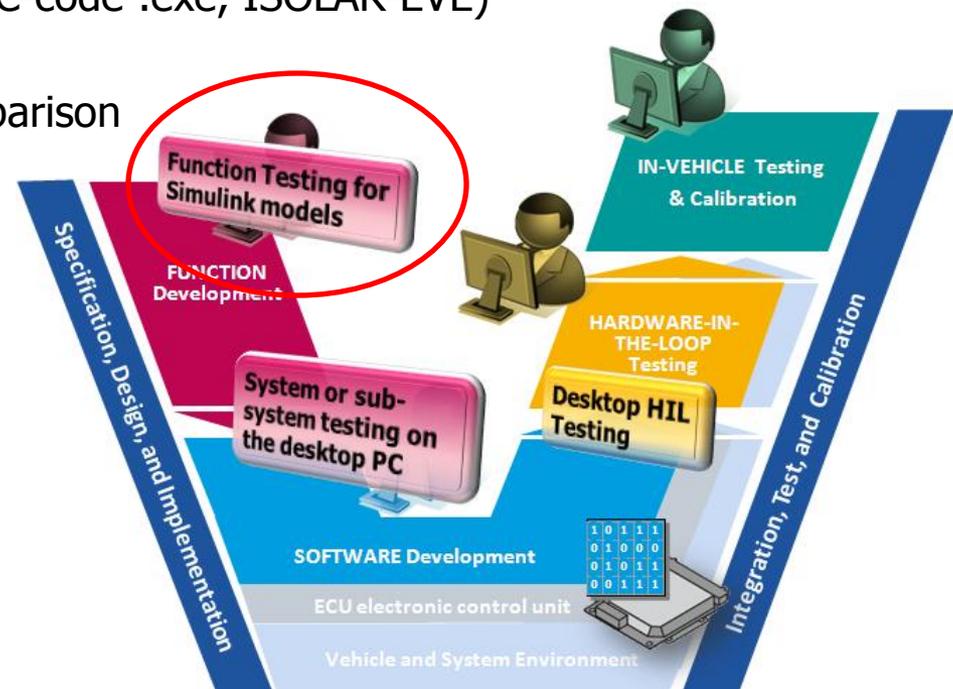
Content

1	Need and Challenges for Function Developers
2	What is RT2?
3	Test Case Modeling Test Case Execution Test Case Assessment and Analysis Test Report Generation
4	Benefits of RT2



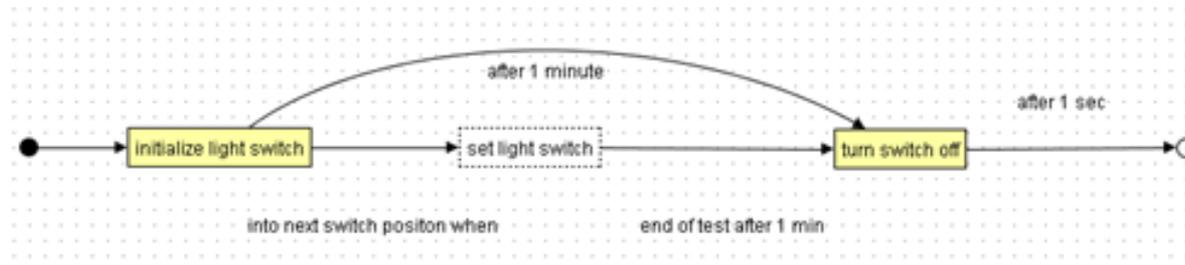
ETAS RT2 is a tool for **designing, executing and assessing systematic tests** for:

- **Functional models** (Simulink or ASCET)
- **Software-in-the-loop** platforms (e.g. C-code .exe, ISOLAR-EVE)
- **Back-to-back** testing: automated comparison of test results between model and software test



RT2 Testing Approach: Model-based Testing

- Normally, test case design is done using scripts
- What's the challenge of scripting?
 - Manage huge amounts of variants
 - Hard to overview the testing strategy and coverage
 - Programming skills required
 - High maintenance effort
- RT2 takes a different approach: test cases are described using **models**
 - Intuitive representation of complexity
 - Efficient management of variants
 - A "language" function developers understand

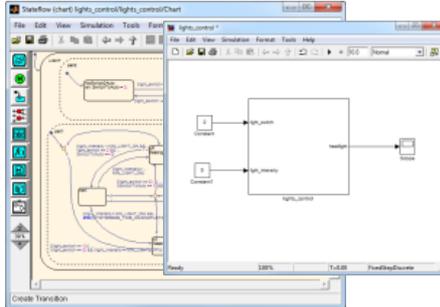


Content

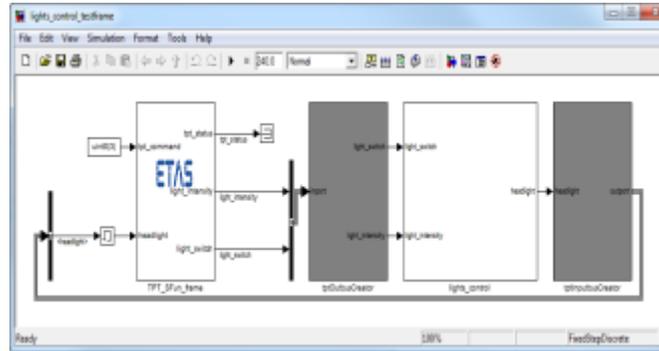
1	Need and Challenges for Function Developers
2	What is RT2?
3	Test Case Modeling Test Case Execution Test Case Assessment and Analysis Test Report Generation
4	Benefits of RT2

RT2 Testing Process

Simulink® Model



Instrumented Simulink® Model

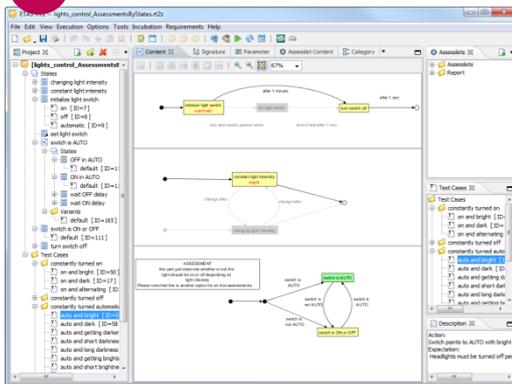


Interface

Parameters



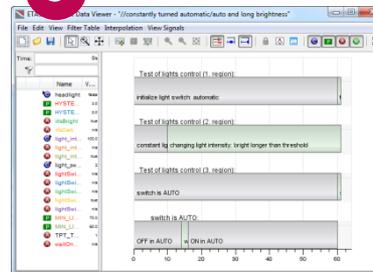
1 Test Case Design Model



2 Execution

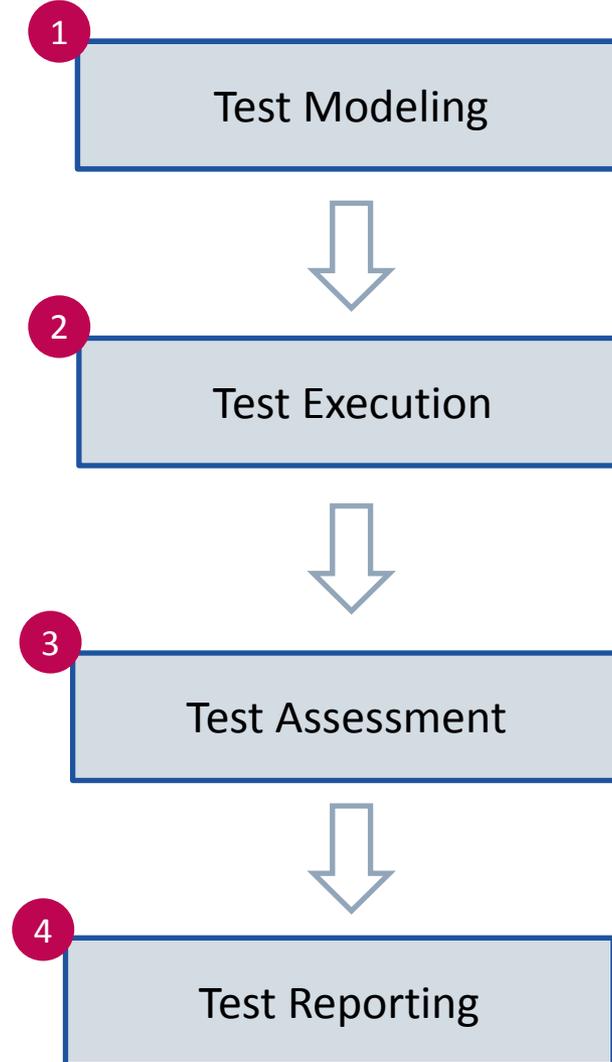


3 Automated Test Assessment



4 Test Report



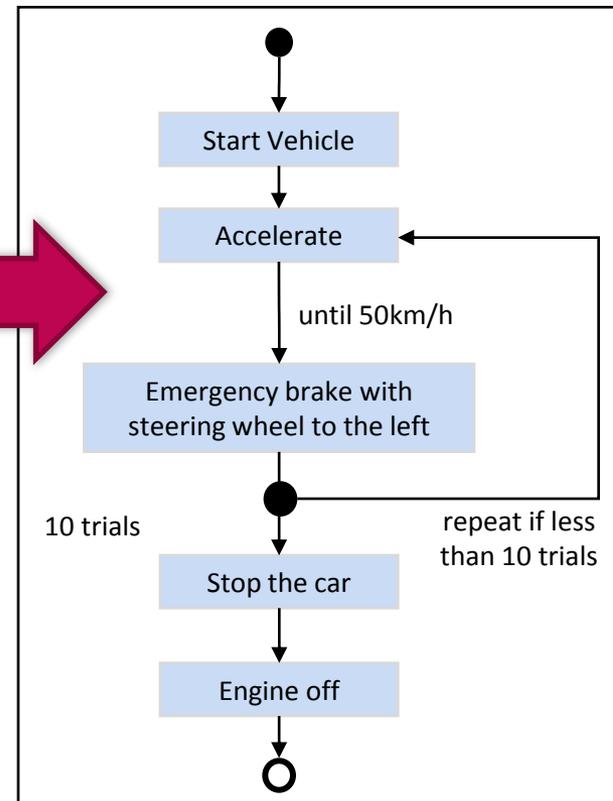


1 Test Modeling: Test Case Sequence

Test case scenarios often consist of sequences of logical phases.

1. Start Vehicle
2. Accelerate until 50 km/h
3. Emergency brake with steering wheel to the left
4. Repeat steps 2 and 3 a total of 10 times
5. Stop the car
6. Engine off

State machine with textual annotations are descriptive and easy to understand

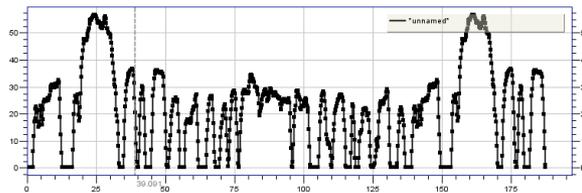


1 Test Modeling: Test State Definitions

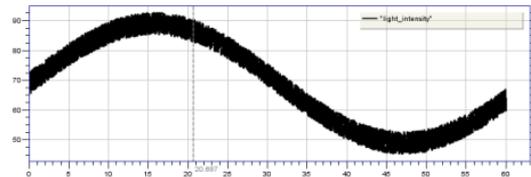
– Direct definition



Imported Signals



Signal Editor



Formulas

```
MIN_LIGHT_OFF
+ 20 * sin(t/10)
+ 7.5 * random ()
```

– Test step list: definition of a step sequence supported by a graphical editor

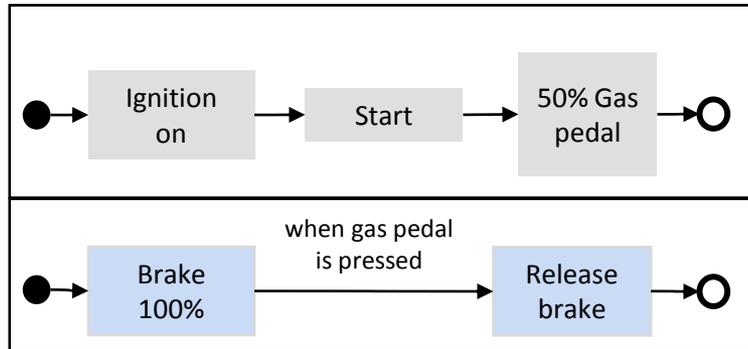
1		Set channel	Brake	:= 80
2		Set channel	Klemme_15	:= 1
3		Wait	10ms	
4		Set channel	Klemme_50	:= 1
5		Wait for value	Engine_RPM	>= 700
6		Set channel	Klemme_50	:= 0
7		Set channel	PRNDL	:= DRIVE
8		Set channel	Brake	:= 0
9		Set channel	Acceleration_Pedal	:= 50
10		Wait for value	Speed	>= 50

– Time partitioned tests: definition through a hierarchical state machine

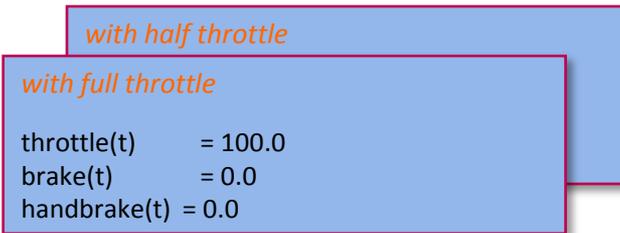
1 Test Modeling: State Arrangements and Variations



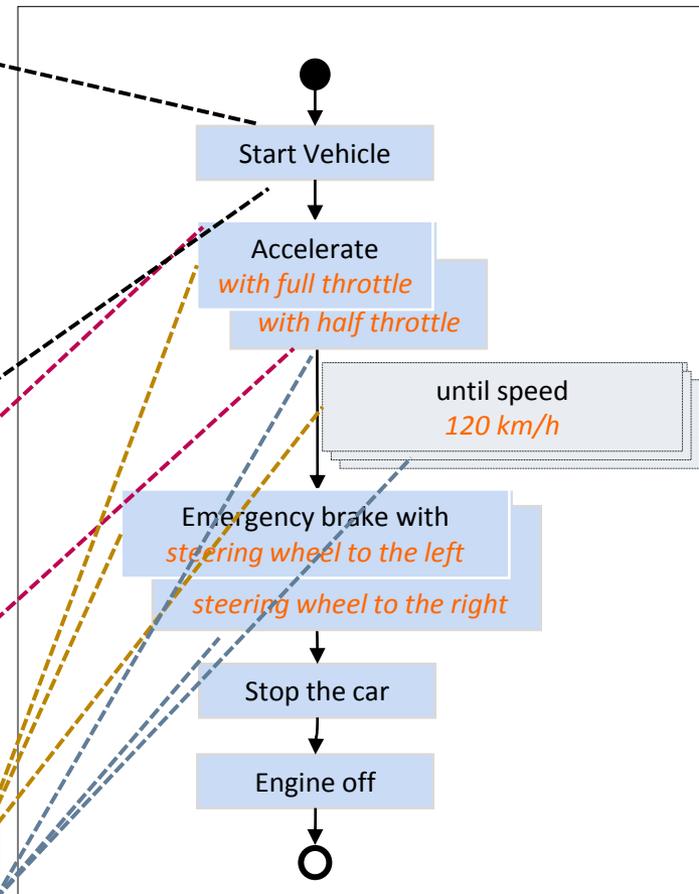
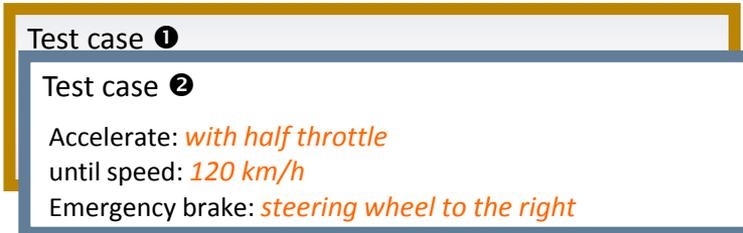
Hierarchical and parallel states



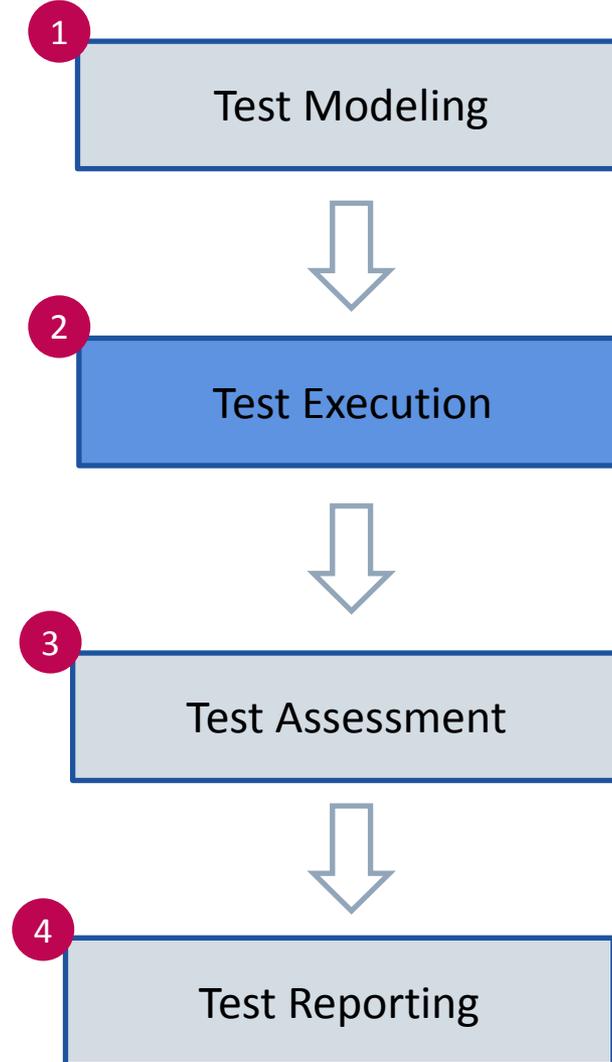
Variants of a state



Variants of combinations of states



RT2 Testing Process

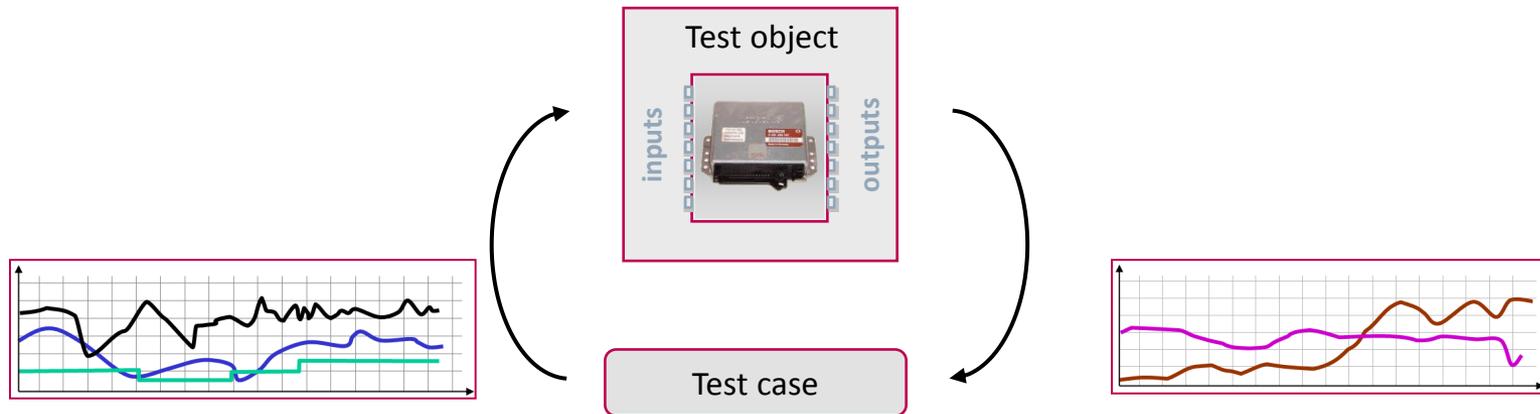


2 Test Execution (1/2)

- Test cases stimulate the test object by continuously affecting system quantities (inputs)
- Test cases can react to system behavior by observing system quantities (outputs)

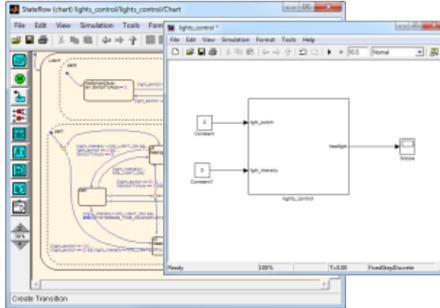


Test Object = Simulink, ASCET, C-code etc.

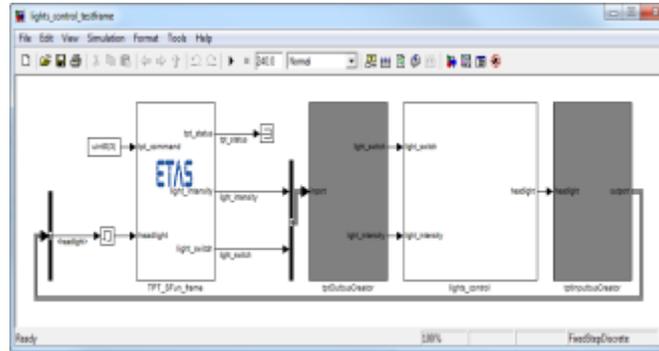


2 Test Execution (2/2)

Simulink® Model



Instrumented Simulink® Model

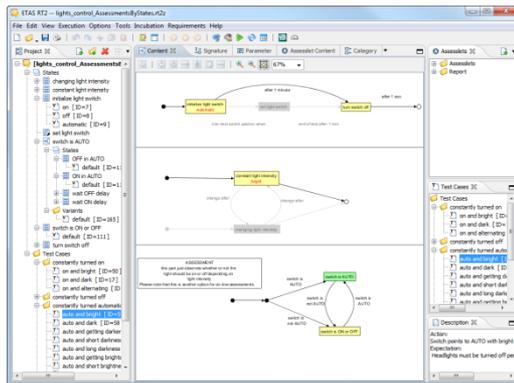


Interface

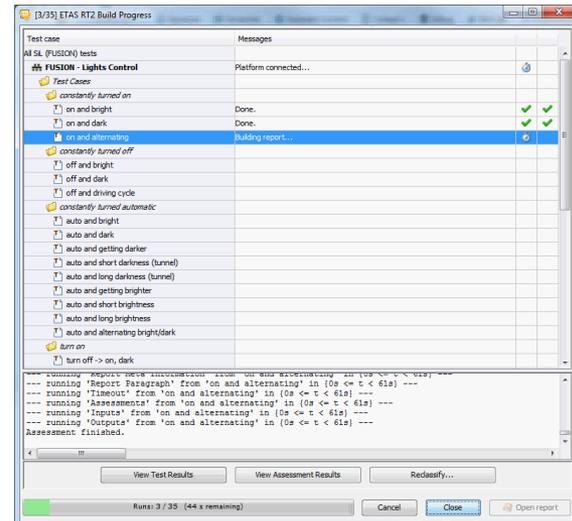
Parameters



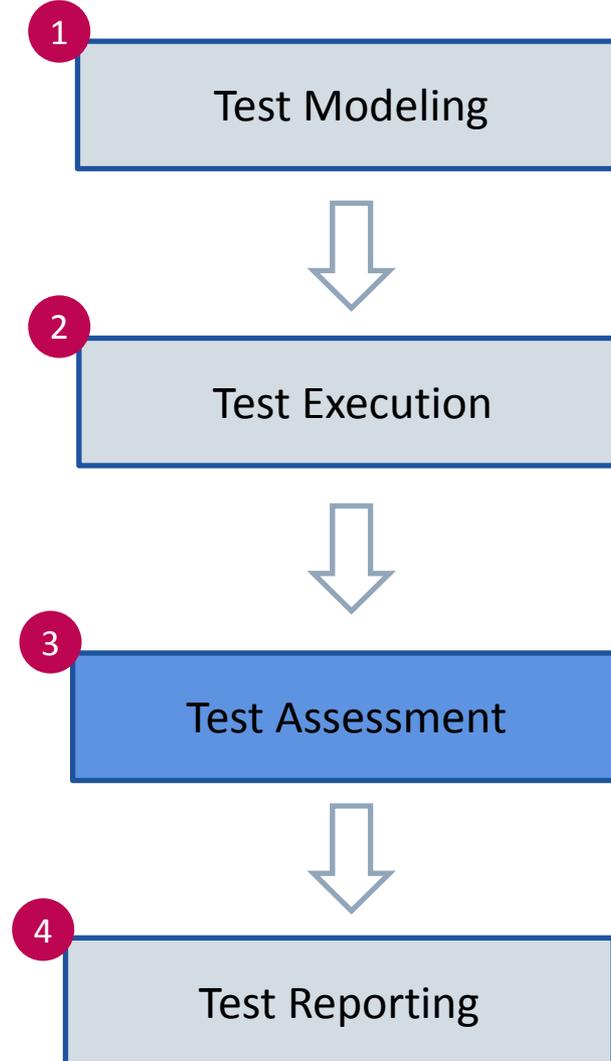
Test Case Design Model



Execution

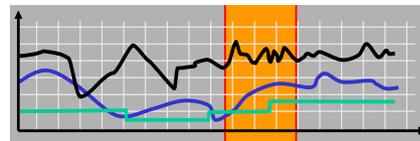


RT2 Testing Process



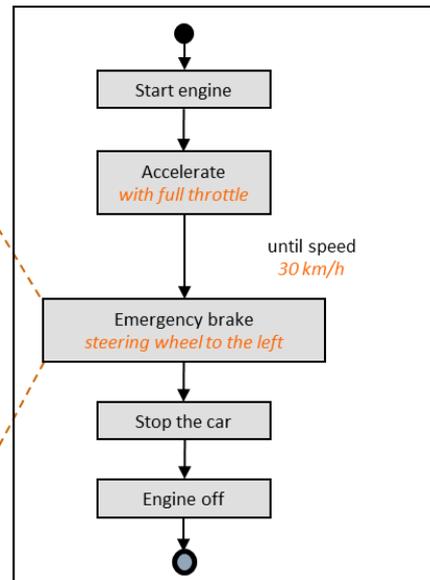
3 Test Assessment – Time Interval Analysis (1/2)

- Assessments conducted after test data is collected
- **When specific state is active:** Analyses focused on a time interval when a specific state is active



Steering wheel to the left

- Duration of the emergency brake may not exceed 10sec.
- Deceleration must be less than 1.5 m/s².
- Revs per minute must be less than 6000 min⁻¹.

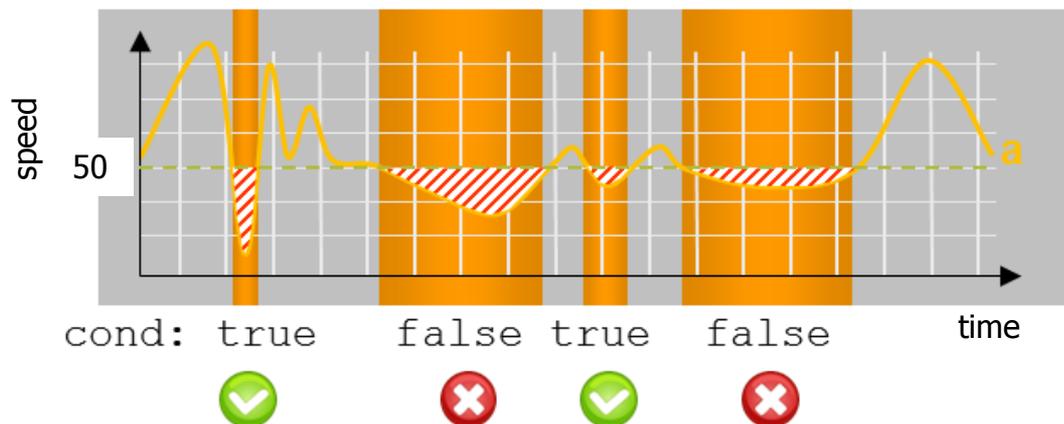


3 Test Assessment – Time Interval Analysis (1/2)

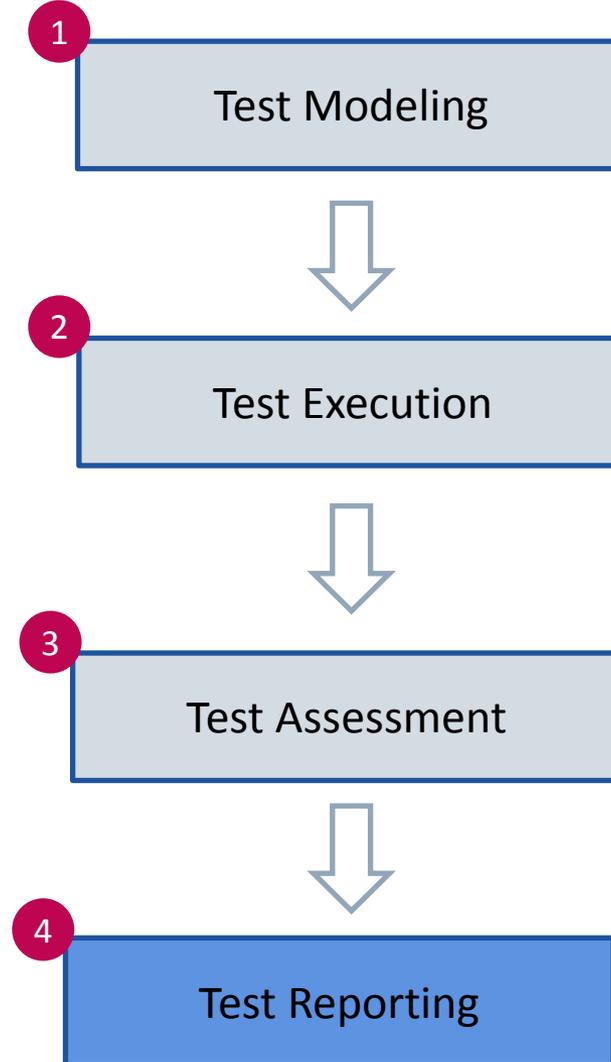
- **Absolute time:** A time interval specified by absolute time
 - `[t >= 50s]`

- **Time patterns:** Explicit time intervals specified
 - `[v_vehicle(t) >= 100 kph]` *time intervals with speed ≥ 100*
 - `[foo(t) == 1]` `[foo(t) == 2]` *time intervals with foo=1 followed by foo=2*

- **Temporal regular expressions:** Time patterns as special cases
 - Assessments can be analyzed in multiple time intervals:
 - e.g. *time intervals where vehicle speed drops below 50kph for less than 10s*



RT2 Testing Process





Detail-rich reports

Requirements Management

- Import/Synchronization of requirements
- Import/Export/Synchronization of test cases
- Import/Export/Synchronization of links
- Impact analysis when requirement changes
- DOORS® Integration

Overview
Variables

EXE - Lights Control

- Test Cases
 - on and bright
 - on and dark
 - on and starting
 - on and alternating
 - constantly turned off
 - off and bright
 - off and dark
 - off and printing cycle
 - constantly turned automatic
 - auto and bright
 - auto and dark
 - auto and getting darker
 - auto and short darkness (turns)
 - auto and long darkness (turns)
 - auto and getting brighter
 - auto and short brightness
 - auto and long brightness
 - auto and alternating bright/dark
- turn on
 - turn on -> on, dark
 - turn off -> on, bright
 - turn auto -> on, getting brighter
 - turn auto -> on, dark
- turn off
 - turn on -> off, just below threshold
 - turn on -> off, bright
 - turn auto -> off, getting darker
 - turn auto -> off, bright
- turn auto
 - turn on -> auto, dark
 - turn on -> auto, bright
 - turn on -> auto, getting darker afterwards
 - turn on -> auto, getting brighter afterwards
 - turn on -> auto, between threshold
 - turn off -> auto, just below threshold
 - turn off -> auto, just above threshold
 - turn off -> auto, getting darker afterwards
 - turn off -> auto, getting brighter afterwards
 - turn off -> auto, between threshold
- signal comparison
 - correct reference signal
 - wrong reference signal

ETAS RT2 Report

ETAS

1 Test Information

1.1 Report Meta Information

Test Case Name	turn auto -> off, getting darker
Test Result	Success
Test Case ID	73
ETAS RT2-File	C:\Program Files (x86)\ETAS\RT2 V5.0\examples\lights_control_AssessmentsByStates.rtz
Directory	C:\Program Files (x86)\ETAS\RT2 V5.0\examples\testdata\EXE_Lights_Control\004_turn_off\002_turn_auto_off_getting_da
Execution Config	All SIL (EXE) tests
Platform Config	EXE - Lights Control
Execution started	09:50:15 02.07.2013

Actions: - Switch first points to AUTO and will be turned to OFF, the light intensity is getting lower. Expectations: 1- Switch starts on AUTO; the headlights should be turned off as the light intensity is above the threshold value MIN_LIGHT_OFF (70%). --> headlight() = 0 2- Switch on Auto; the headlights should be turned on as the light intensity is below the threshold MIN_LIGHT_ON for at least 2s, headlight() = 1 3- Switch OFF; the headlight should be turned automatically off. --> headlight() = 0

2 Signals

2.1 Assessments

Name	Value	From	To	Comment
lightSwitch_Off	true	5.35s	65.35s	light switch OFF is okay
lightSwitchAuto_hold	true	3.34s	5.34s	Headlights have hold last value (0) in AUTO mode during hysteresis
lightSwitchAuto_off	true	0s	3.34s	Headlights have been turned off correctly in AUTO mode
lightSwitchAuto_on	true	5.34s	5.35s	Headlights have been turned on correctly in AUTO mode
light_intensity_range_check	true	0s	20s	Signal light_intensity between min (0%) and max (100%) limits.
light_intensity_range_check	true	20s	66.35s	Signal light_intensity between min (0%) and max (100%) limits.
ItIsBright	true	0s	3.34s	It is bright longer than threshold time
ItIsDark	true	5.34s	5.35s	It is dark longer than threshold time
waitOnDelayTime		2.0	3.34s	Wait for time

2.2 Inputs

Name	Value	From	To	Comment
headlight	1	0s	66.35s	

2.3 Outputs

Name	Value	From	To	Comment
light_switch	1	0s	66.35s	
light_intensity	100	0s	66.35s	

2.1 Assessments

Name	Value	From	To	Comment
lightSwitch_Off	true	5.35s	65.35s	light switch OFF is okay
lightSwitchAuto_hold	true	3.34s	5.34s	Headlights have hold last value (0) in AUTO mode during hysteresis
lightSwitchAuto_off	true	0s	3.34s	Headlights have been turned off correctly in AUTO mode
lightSwitchAuto_on	true	5.34s	5.35s	Headlights have been turned on correctly in AUTO mode
light_intensity_range_check	true	0s	20s	Signal light_intensity between min (0%) and max (100%) limits.
light_intensity_range_check	true	20s	66.35s	Signal light_intensity between min (0%) and max (100%) limits.
ItIsBright	true	0s	3.34s	It is bright longer than threshold time
ItIsDark	true	5.34s	5.35s	It is dark longer than threshold time
waitOnDelayTime		2.0	3.34s	Wait for time hysteresis ON

Content

1	Need and Challenges for Function Developers
2	What is RT2?
3	Test Case Modeling Test Case Execution Test Case Assessment and Analysis Test Report Generation
4	Benefits of RT2

Comprehensive tools suite for test design, automation, assessment and documentation for Simulink® models



- Intuitive test case **creation** - Enables users to quickly develop test cases with a short learning curve
- Test **variant** management - Enables users to explore functional behavior in multi-faceted test cases, in large test campaigns
- Powerful **assessment** capabilities - Allows users to make decisions from test results without additional tools
- **Traceability** between requirements and models - Improves quality and ensures that deliverables meet the needs and expectations of stakeholders
- **Test reuse** across development phases (MiL/SiL)
- Supports **ISO26262** compliant development processes

The ETAS logo is displayed in a large, blue, stylized font. The letters are bold and blocky, with a slight shadow effect. The background of the slide is a blurred image of a car's side mirror and a road stretching into the distance under a cloudy sky.

有難うございました

谢谢

감사합니다.

Thank you

Merci

Vielen Dank