### Model-Based Testing: Automated Generation of Test Cases, Test Data, and Test Procedures from SysML Models

erified

Jörg Brauer and Jan Peleska

Verified Systems International GmbH

support@verified.de

Space Tech Expo Europe – 2015-11-19

#### Motivation

- Model-driven system and software development has become an established best practice – at least in certain application fields
- Model-based testing, however, is an active research area, but many enterprises are hesitant to adopt it as an integrated part of their V&V processes

#### Motivation

- In this presentation, it is explained why
  - Model-based testing is fit for industrial application
  - The return of investment into test model development is significant, leading to
    - reduced V&V costs
    - increased test strength
    - simpler accumulation of certification evidence

- Workflow overview
- Test model development
- Model-based requirements tracing
- Test procedure generation and execution
- Conclusions

- Workflow overview
- Test model development
- Model-based requirements tracing
- Test procedure generation and execution
- Conclusions

#### **MBT Workflow**







- Workflow overview
- Test model development
- Model-based requirements tracing
- Test procedure generation and execution
- Conclusions

#### Test Model Development

- The test model specifies
  - Interfaces between SUT and environment are represented as readable or writable in the test campaign
  - Structural aspects functional decomposition
  - Behavioral aspects transformation of inputs into outputs, sequencing, synchronisation ...
- Two alternatives for test model creation and utilisation (see next slides)

#### Validated Test Models – Variant 1



Validated Test Models – Variant 2



- Workflow overview
- Test model development
- Model-based requirements tracing
- Test procedure generation and execution
- Conclusions

#### Model-Based Requirements Tracing

- Objective
  - Link model elements to associated requirements that are represented by these elements
  - This allows us to identify test cases suitable for verifying a given requirement in an automated way
- SysML is a system modeling language providing (graphical and textual) syntax for linking requirements to behavioral and structural model elements

#### Model-Based Requirements Tracing



- Workflow overview
- Test model development
- Model-based requirements tracing
- Test procedure generation and execution
- Conclusions

# Test Procedure Generation and Execution

- Test cases can be identified by evaluating the links between model elements and requirements
- The behavior expressed by the model can be internally encoded by logical formulas
- The test case is internally represented by a logical formula as well
- → Concrete test data can be automatically calculated using mathematical constraint solvers

# Test Procedure Generation and Execution

- For test procedure generation, users just configure which test cases should be covered by the procedure to be created
  - Requirements-driven generation is performed by identifying the requirement (or a subset of related test cases) to be tested by the procedure
  - Model-driven generation is performed by identifying the model portions to be covered by the test procedure

#### Requirements-Driven Test Procedure Generation

Model	<u>R</u> equirements	Test Cases		<u>M</u> odel	<u>R</u> equirements	<u>T</u> est Cases	
Model + R R + R R + R R + R R + R R + R R - R R - R R + R R	Requirements REQ-001 Indication lights are REQ-002 If any lights are flas REQ-003 An input change fro REQ-004 An input change fro REQ-005 An input change fro REQ-006 Activation of the tu REQ-007 If turn indication lef REQ-008 If emergency flashin C Referenced Te TC-turn_in TC-turn_in TC-turn_in TC-turn_in TC-turn_in REQ-009 If turn indication lef	Test Cases e only active if the shing, this is dor om TurnIndLvr = ( om TurnIndLvr = ( om EmerFlash = ( um indication left ft or right is switt ng is turned off a est Cases dication-HITR dication-HITR dication-TR-O dication-UD-O ft or right is switt	e electrical voltage is > 10.3 and 14.0 V. e synchronously with a 340ms ON - 32( 0 or 2 to:TurnIndLvr = 1 switches indicati 0 or 1 to:TurnIndLvr = 2 switches indicati 0 to EmerFlash = 1 switches indication lig or right overrides emergency flashing, if 1 cched off and emergency flashing is still ac ind turn indication left or right is still activ -0001 -0002 005 0001 cched off before three flashing periods har	Model Basi Basi Hiel MC, Trai	Requirements ic Control State TC-turn_indica	Test Cases e Coverage ation-BCS-00 ation-	01 02 03 04 05 06 07 08 09 10 4 rage ge
<			2	<	TC-turn_indica	tion-TR-000	2

#### Requirements-P



Requirements are displayed with their associated test cases, to be selected for test procedure generation

- TC-turn\_indication-BCS-0005
- TC-turn\_indication-BCS-0006
- TC-turn\_indication-BCS-0007
- TC-turn\_indication-BCS-0008
- TC-turn\_indication-BCS-0009
- TC-turn\_indication-BCS-0010

Basic Control State Pairs Coverage

Hierarchical Transition Coverage

MC/DC Transition Coverage

#### Transition Coverage

- TC-turn\_indication-TR-0001 Cover transition of component IMR.SystemUnderTest.FLASH\_CTRL FLASH\_CTRL.EMER\_OFF -- [ IMR.EmerSwitch ] -->
  - FLASH\_CTRL.EMER\_ON
- TC-turn\_indication-TR-0002

#### Model-Driven Test Procedure Generation

<u>M</u> odel	<u>R</u> equirements	Test Cases		Goals	Solver																
SystemUnderTest			Ordered Goals Add LTL																		
□ ▲ FLASH_CTRL				TC-turn_indication-BCS-0001																	
+ Initial			Unordered Goals						Add LTL												
EMER_OFF			-	→ EmerS	witch		BCS	BCSPAIRS	HITR	MCDC	TR										
→ EmerSwitch				D EMER.	_ON		BCS	BCSPAIRS	HITR	MCDC	TR										
Referenced Test Cases			i		UT_CTRL		BCS	BCSPAIRS	HITR	MCDC	TR										
TC-turn_indication-BCS-0002				OUTPI	UT_CTRL		BCS	BCSPAIRS	HITR	MCDC	TR										
+ • EMER_ON			REQ-001																		
E Referenced Test Cases			Indication lights are only active if the electrical voltage is > 10.3 and 14.0 V.																		
			REQ-005																		
				An input change from EmerFlash = 0 to EmerFlash = 1 switches indication ligi TC-turn_indication-TR-0001 Cover transition of component IMR.SystemUnderTest.FLASH_CTRL FLASH_CTRL.EMER_OFF																	
															[ IMR.EmerSwitch ]>						
																FLAS	H_CTRLEMER_	ON			
				♥ User Defined LTL Edit																	
					Finally	y ([(((IMR.System	UnderTest.pr_[	Decisio	on != 6) && (	IMR.Sy	stemUnd	lerTes									
<			<								э										



# Test Procedure Generation and Execution

- 6 Different testing strategies are supported
  - Simple: Basic control state coverage, Transition coverage, Hierarchic transition coverage.
  - Complex:
    - MC/DC coverage. Complex guard conditions are exercised with different valuations of their atomic condition parts
    - **Basic control state pairs coverage**. Interacting state machine pairs are tested in every possible state combination
    - Equivalence class testing strategy. Inputs with large data types are automatically partitioned into equivalence classes

# Test Procedure Generation and Execution

- Applicable strategies can be automatically adapted to the criticality of SUT component under consideration
  - For DAL-C components, transition coverage typically suffices
  - For DAL-A components, the more complex strategies have to be applied as well

- Workflow overview
- Test model development
- Model-based requirements tracing
- Test procedure generation and execution
- Conclusions

### Conclusions

- Model-based testing is fit for large-scale application in industry
- Efficiency measurements performed by Verified Systems show that the following effort reductions in comparison to conventional testing approaches can be expected
  - 30% for a new project, where a new test model has to be created
  - 90% in the best case, where an optimised re-usable workflow has been set up for regular regression testing of a long-lived product