MathWorks AUTOMOTIVE CONFERENCE 2023 Europe

Agile Behavior-Driven and Test-Driven Development with Model-Based Design

Marc Segelken, MathWorks





Agenda

- Definition Behavior-Driven and Test-Driven Development
- How can I write good requirements?
- How do BDD and TDD work with Model-Based Design?
- Example of BDD with MBD

Key Takeaway: How to apply BDD and TDD in Model-Based Design

Introduction Problem statement

- Textual requirements are often incomplete, inconsistent, incorrect and ambiguous.
- Testing often happens very late during a development cycle and it is rushed due to deadlines.

How can you prevent this?

The Scaled Agile Framework (SAFe) suggests the "test-first" approaches of Behavior-Driven Development and Test-Driven Development.

How can BDD and TDD be practiced in the context of Model-Based Systems Engineering and Model-Based Design?

Introduction Model-Based Design

- Simulink and Stateflow referenced in standards for safetycritical system development such as, e.g., ISO 26262
- Higher level of abstraction
- Enables early validation of requirements through simulation
- Generate documentation and production code automatically, saving effort and eliminating manual error







Benefit Hypothesis:

I believe that Model-Based Design with system simulation and production code generation will result in increased productivity and efficiency in large-scale software projects, as measured by correctness of requirements and quality of generated production code.

Introduction Behavior-Driven Development (BDD)



"BDD is a collaborative process that creates a shared understanding of requirements between the business and the agile teams." (SAFe)

- Requirements elicitation is team effort, leads to understanding of
 - what needs to be done
 - why it should be done
 - when it will be good enough
- Easier to see what behavior is needed with examples in action
- BDD with system simulation is an effective way to get a baseline for requirements with acceptance criteria.



Benefit Hypothesis:

I believe that Behavior-Driven Development (BDD) in the combination with system simulation capabilities of Model-**Based Design** will result in built-in quality as measured by the efficient elicitation of correct system requirements and early validation of correct behavior of the system.

Introduction Test-Driven Development (TDD)

"[..] building and executing tests before implementing the code or a system component" (SAFe)

- Simulink simplifies creating test-frames including assessments, and mocking environments
- re-use the tests for generated code.





Benefit Hypothesis:

I believe that Test-Driven Development (TDD) in the combination with the executable specification (simulation) capabilities, model-level verification, and code generation capabilities of Model-**Based** Design will result in built-in quality, as measured by verified correct behavior of models and generated code on a component level.

The Role of Requirements The Importance of Good Requirements

Starting point for BDD and TDD: Testable requirements with acceptance criteria





The Role of Requirements in an Agile Framework How can I write good requirements?

Benefit Hypothesis	I believe that <capability>, Will produce <outcome>, As measured by <metric>.</metric></outcome></capability>	To communicate business and customer relevance	WHY?
User Stories	As a <role>, I want to <need>, So that <goal>.</goal></need></role>	To give context	WHAT?
Acceptance Criteria	Given <precondition>, When <trigger action="">, Then <expected result="">.</expected></trigger></precondition>	To communicate quality assurance criteria	WHEN WILL IT BE GOOD ENOUGH?

The Role of Requirements How Can I Write Good Requirements? Example

Requirement	Issue
The vehicle shall support DC charging of the battery.	This requirement is vague and incomplete with no clear acceptance criteria.
The vehicle shall support DC fast charging at 50kW300kW.	The requirement now is clearer, but it is still incomplete.
As an electric vehicle driver, I want to have the option to charge my electric vehicle fast, So that I can get additional driving range in a short amount of time.	The end-user is put into focus and context is given by the user-story.
Given the battery's state of charge is below 50%, And the vehicle is plugged into a DC fast charging station capable of up to 320 kW, When the battery is charged for 5 minutes, Then the vehicle shall be able to travel for at least an additional 100 km.	The acceptance criteria is formulated in such a way that a test could be derived from it easily without too much interpretation.

How do BDD and TDD work with Model-Based Design?

- BDD and TDD follow "test-first" approach:
 - 1. Have clear requirements and clear acceptance criteria
 - 2. A minimum architecture or design



3. Only then start **building test harnesses and test cases** with assessments.



BDD and TDD lifecycles incorporating the benefits of MBD



A: Analyze System Requirements Capture and review system requirements – Use Case Diagram



A: Analyze System Requirements

Capture and review system requirements - "WHY?"

- Benefit Hypothesis: ("WHY?")
 - I believe, that DC fast charging in the range of 50 kW to 320 kW,
 - will result in greater adoption of electric vehicles due to more flexibility in charging options,
 - as measured by #users indicating DC fast charging as a significant reason for choosing the electric vehicle.

A: Analyze System Requirements

Capture and review system requirements - "WHAT? WHEN?"

- User Story: ("WHAT?")
 - As an electric vehicle manufacturer,
 - I want to integrate DC charging stations with my EV charging system,
 - So that the vehicles can charge fast, safe and efficiently.
- Acceptance Criteria: ("WHEN IS IT GOOD ENOUGH?")
 - Given the vehicle is plugged into a compatible DC fast charging station capable of up to 320kW,
 - And the battery temperature is between 10°C and 40°C,
 - And the State of Charge (SOC) is between 5 and 50%,
 - When the battery has been charging for at least 100ms,
 - Then, the charging power shall be at least 300 kW.



A: Analyze System Requirements Capture and review system requirements

A: Analyze System Requirements Capture and review system requirements

- Work in a structured way:
 - Create a MATLAB Project
 - Use Version Control,
 e.g., GIT that is integrated with MATLAB Project
 - Create a Requirement Set
 - Create a Requirements Model
 - Create Virtual Prototype Model
 - Do systematic analysis



A: Analyze System Requirements Capture and review system requirements

- Create a Requirement Set
- Requirements Editor (Create / Import / Update Requirements)
- Keep track of changes and do change impact analysis

뉡 Requ	irements Editor																	
REQUIF	REMENTS																	
N Require	ew Open La Import ment Set FILE	Load Drofile	ige e Editor	Add Fequirement Requirement Requirement Requirement Requirement Requirement REQUIREMENTS	Add Clear Issue	Show Requirements	Show C Refresh Links VIEW	iew 🔻 🖪	Colum	nns 👻 nation 👻	C Search EDIT	Traceability Matrix	Traceability Diagram ANALYSIS	Model Testing Dashboard	Export SHARE			
									Rec	quirement	#5							
	Index	Туре	ID	Summary	Implemented		Verified		s ▼ P	Propertie	s							
>)	Туре	e:	Functional	~						
🕶 🔓 R	equirements_DC_Charge)	Inde	ex: 1	.5							
~ ≣	1	Container	#1	Scenario: DC Charger Station Integration				1	Cust	tom ID:	#5							
	≣ 1.1	Informational	#2	User Story: EV Driver				2	Sum	nmarv:	Acceptance	Criteria: charo	e power					
	≣ 1.2	Functional	#8	Acceptance Criteria: charge speed				8										
	≣ 1.3	Informational	#3	User Story: OEM				3	De	escription	Rationa	ile						
	≣ 1.4	Functional	#4	Acceptance Criteria: start time				4	<u>w</u>	Arial							~ 10	✓ B
	I.5	Functional	#5	Acceptance Criteria: charge power				5	Gi	iven the nd the ba	vehicle is	plugged into	a compatible	e DC fast char and 40°C	ging station	capable of	up to 350k	.W,
									Ar W Th	nd the St /hen the I hen, the	ate of Chattery ha	arge (SOC) is s been charg ower shall b	s between 5 ging for at least 30 e at least 30	and 50%, ast 100ms, 0 kW.				

BDD and TDD lifecycles incorporating the benefits of MBD

A: Analyze System Requirements Capture and review system requirements

B: Define System Architecture Create architecture and allocate requirements

BDD for system

C: Write System Test Cases Create test cases for system behavior

F: Simulate and Test System Run system test cases

G: Rapid Prototyping with Real Hardware Test system behavior on hardware

I, J: Run System SIL Tests, then HIL Tests Test system behavior on hardware TDD for each component

D: Write Component Test Cases Create test case(s) for each requirement

E: Detailed Component Design Model just enough to pass test case(s)

H: Component Implementation Refactor and prepare for code generation. Perform full set of quality checks and tests.

B: Define System Architecture

Concept phase: use virtual prototypes to gain better understanding

- Use a virtual prototypes
- Get started quickly making use of reference examples, e.g.,
- DC Fast Charger for Electric
 Vehicle





BDD and TDD lifecycles incorporating the benefits of MBD



C: Write System Test Cases

Create test cases for system behavior: Formalize Requirements

- Test-Case: System Under Test + Input Stimuli (Scenario) + <u>Assessments</u>
- Formalize requirements (acceptance criteria) == Assessments
- Use, for example, a "Requirements Table" in Simulink



A: Analyze System Requirements Analyze Requirements for Complete and Consistency

 Requirements Tables assist you to analyze for completeness and consistency using advanced formal methods

4	
Analyze Table 🔻	
่≜ ′,	n alyze Table nalyze table for issues

Requirements Assumptions

			\odot
Date: 02-May-2023 1	1:41:25		
Block: Requirements_ charge power	_Model_DC_Charge/Variant Subsystem/Usin	g Requirements Tables/Acceptance Criteria:	
Inconsistency l	lssues		
No inconsistency issu	es found.		
Incompletenes	e Iccuec		
meompretenes	5 135uc5		
Incompleteness 1: 'po	wCharge' is not specified for the following	inputs:	
Incompleteness 1: 'po Time	owCharge' is not specified for the following	inputs: 0.1	
Incompleteness 1: 'po Time Step	owCharge' is not specified for the following 0-0.09 1-10	inputs: 0.1 11	
Incompleteness 1: 'po Time Step ChargeStatus	owCharge' is not specified for the following 0-0.09 1-10 ChargeStatus_Type.CST_Charging	inputs: 0.1 11 ChargeStatus_Type.CST_Charging	
Incompleteness 1: 'pc Time Step ChargeStatus ChargerCompatibility	owCharge' is not specified for the following 0-0.09 1-10 ChargeStatus_Type.CST_Charging 'ChargerCompatibility_Type.CCT_Compatible	inputs: 0.1 11 ChargeStatus_Type.CST_Charging eChargerCompatibility_Type.CCT_Compatibil	le
Incompleteness 1: 'po Time Step ChargeStatus ChargerCompatibility tempBat	owCharge' is not specified for the following 0-0.09 1-10 ChargeStatus_Type.CST_Charging ChargerCompatibility_Type.CCT_Compatible 41	inputs: 0.1 11 ChargeStatus_Type.CST_Charging e ChargerCompatibility_Type.CCT_Compatibl -	le
Incompleteness 1: 'po Time Step ChargeStatus ChargerCompatibility tempBat socBat	owCharge' is not specified for the following 0-0.09 1-10 ChargeStatus_Type.CST_Charging ChargerCompatibility_Type.CCT_Compatible 41	inputs: 0.1 11 ChargeStatus_Type.CST_Charging e ChargerCompatibility_Type.CCT_Compatibl -	le

· ·								
🔒 Index	Summary	Precondition			Descritions	Postcondition		
		ChargerCompatibility	socBat	tempBat	ChargeStatus	Duration	powCharge	
1	Acceptance Criteria: charge power	CCT_Compatible	[5,50]	[10,40]	CST_Charging	0.1	> 300	

A: Analyze System Requirements Analyze Requirements for Complete and Consistency

 Requirements Table needs to be complete to be used in simulation as Assessments



. .

Analysis Results (🖲 🗙
Date: 02-May-2023 11:36:43 Block: Requirements_Model_DC_Charge/Variant Subsystem/Using Requirements Tables/Acceptance Criteria: charge power
Inconsistency Issues
No inconsistency issues found.
Incompleteness Issues
No incompleteness issues found.

Requiremen	its Assumptions						
Index	Summary	Precondition					Postcondition
		ChargerCompatibility	socBat	tempBat	ChargeStatus	Duration	powCharge
1	Acceptance Criteria: charge power	CCT_Compatible	[5,50]	[10,40]	CST_Charging	0.1	> 300
2 D	Don't care	Else					[-2000,2000]

C: Write System Test Cases Check that Assessments FAIL as expected



BDD and TDD lifecycles incorporating the benefits of MBD



F: Simulate and Test System

Run system test cases, Check that Assessments pass as expected

- Add mocking logic and stubs to virtual prototype
- Run Simulation including Assessments, view results

Concept_e_dc_fast_charger - Simulink SIMULATION DEBUG MODELING FORMAT APPS		Data
Image: Construction of the state of the	R:1 (Acceptance Criteria: charge power)	
Bugger Puggersor: StateMachine Fail Pail Concept.ee.d.fail_charger Fail Pail Concept.ee.d.fail_fail_charger Fail Pail Concept.ee.d.fail_fail_charger Fail Concept.ee.d.fail_charger Concept.ee.d.fail_fail_charger Fail Pail Concept.ee.d.fail_fail_fail_fail_fail_charger Fail Pail Concept.ee.d.fail_fail_fail_fail_fail_fail_fail_fail_	Measurement Panel DC bus Voltage(V) 1197 Charging Voltage(V) 897.8044 Charging Current(A) 375.8853	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
		26

Conclusions

- Model-Based Design fits very well with BDD and TDD
- Refine requirements make them complete, consistent, correct, testable
- Follow test first approach –
 Use test-cases, catching missing or wrong functionality
- Use virtual prototypes and test-benches to validate requirements
- Avoid unnecessary work Gain confidence in requirements, architecture, and designs before implementation





Thank you! For more information on this topic, please visit this page:

