

BUILDING GOOD-QUALITY FUNCTIONAL SPECIFICATION MODEL



Research and Consultancy on Systems Engineering

- Requirement engineering
- Model-Based Systems Engineering
- Co-simulation



Mainly involved in avionics domain (but not limited to)







Rockwell_

Accelerate Systems Design

Strong links with high schools and research institutes







Challenges of « Good functional specification »

MBSE approach to reach those challenges

Demonstration



What is a « Good-Quality » specification ?



Correctness is the degree to which an individual requirement is <u>unambiguous</u>, <u>verifiable</u>, <u>consistent</u> with other requirements and <u>necessary</u> for the requirement set.

Accelerate Systems Design

Completeness is the degree to which a set of correct requirements, when met by a system, <u>satisfy the</u> <u>interests</u> of customers, users, maintainers, certification authorities as well as aircraft, system and item developers under <u>all modes of operation</u> and <u>lifecycle phases</u> for the <u>defined operating environment</u>

<image/> <section-header><section-header><section-header><text></text></section-header></section-header></section-header>	The second secon	2	CHARA 2.1 (2.2 (2.3 (2.4 (2.5 (2.6 (2.7 (2.8 (2.9 (2.1 (2.9 (2.1 (2.1 (2.2 (2.1 (2.2 (2.3 (ACTERISTICS OF REQUIREMENT STATEMENTS C1 - NECESSARY C2 - APPROPRIATE C3 - UNAMBIGUOUS C4 - COMPLETE C5 - SINGULAR C6 - FEASIBLE C7 - VERIFIABLE C7 - VERIFIABLE C8 - CORRECT C9 - CONFORMING ACTERISTICS OF SETS OF REQUIREMENTS C10 - COMPLETE
			3.1 (
7			3.3 (C12 - FFASIBI F
FOURTH EDITION			3.4 (C13 - COMPREHENSIBLE
For INCOSE member, Corporate Advisory Board, and Academic Council use only.			3.5	C14 - ABLE TO BE VALIDATED

Consistency is the strongest challenge

Can you ensure there is no conflict in 200 pages spec?

Accelerate Systems Design

- Can you afford to review the total combinatory of req?
 - Example: for 200 requirements it would mean 40 000 checks...
- With rising number of requirements we need help !
 - Models can help => A diagram provides "local consistency"
 - But it is not enough... we need to address also <u>global consistency</u>
- Ultimate goal: reach "consistent-by-construction"

Goal: define a MBSE methodology able to formalize requirements as <u>connected model elements</u> that make a <u>consistent specification</u>

- 1. Use of standard <u>Sys</u>tem <u>Modeling Language</u>
- 2. Function based definition (prescribed by ARP4754A) with support of recursive refinement (different levels)

Accelerate Systems Design

- 3. Support of simulation to ease early validation
 - Ability to run specification model in its operating context
- 4. Support of bidirectional traceability
 - Prescribed to reach avionics system certification
 - Useful for change management (key for long term sustainability)
- 5. All elements connected (consistent-by-construction)
- 6. Agile approach (late new or changed requirements)
- 7. And obviously a Modeling Tool conformant to SysML

Complete and correct requirements (1) SAMARES

• Suggestion #1: structure inputs into UC and actors

 Motivation 1: a UC provides goal for users (actors) and then ensures <u>completeness of functionality</u>. UC is the right container to organize future validation scenarios

Accelerate Systems Design

• Motivation 2: actors provide functional boundaries and then help ensuring that <u>functional scope for system is correctly understood</u>



Complete and correct requirements (2)

Suggestion #2: use black box scenario-based approach

- Motivation 1: capture all intended normal and non-normal behaviors (<u>completeness</u>) with special focus on interactions → can check ICD (<u>necessary and sufficient functional interfaces</u>)
- Motivation 2: Capture abnormal behaviors (address <u>safety</u>)
- Motivation 3: Scenarios are <u>easy to review and validate</u>
- Motivation 4: straightforward derived into validation scenarios

UC is scenario umbrella and provides guidance

- Same scope, start, end, actors
- Can detect inconsistencies early

🔄 💛 Execute ground test

P→ Relations
P→ Sunny day S1 with data - Select target member system and test1 «Reviewed Operational Scenarional Scenariona



ENGIN

Accelerate Systems Design

Which SysML concept for function?



Common SysML concepts to formalize a function

- Block or Activity for function definition
- Respectively Port or Parameter for functional interface
- Respectively Part or Call Behavior Action for function call (usage)
- Respectively Connector or Object Flow for functional flow



Function behavior (for simulation)



Function behavior = Activity or State machine (SM)

 If function is represented by *Block* and behavior is represented by Activity → have to map ports and parameters at each level



• If function as *Block* and behavior as *State* machine, cannot see subfunctions in the behavior (less intuitive)

• We choose to represent function as Activity as it can be completed with behavior consistently



Pattern to <u>identify</u> top-level functions and system functional interfaces from intended behaviors (usages)



Scenario is focused on capture of top level functions and interactions, not on function logics Idea is to ensure it remains simple enough to be validated and to leave logics for further stage

Then we use tool (plugin) to combine scenarios and define function required logics iteratively

Complete top-level functions and interfaces

Accelerate Systems Design

Once identified, top-level functions are completed

- Definition of their inputs and outputs
- Functional flows between elements

Data continuity guidance principle

- Each function input either comes from parent function input or from another function output
- For top-level function, inputs may come from input signal that triggered function => lead to update interface



Specify function validity conditions



- Functions are not valid at any time
 - Have to specify conditions in which their activation is valid
- We use state machine to represent mission states
 - Input signals are put on transitions
 - Top-level functions set in states (doActivity) or on transition effect



Refine functions down to allocation



- Each function is refined (realized with lower-level functions) down to the level where :
 - Either it can be allocated to existing products
 - Or it is fully understood from its specification (no realization risk)



At any time can simulate a scenario (1)





At any time can simulate a scenario (2)





Traceability matrix



	1-	-				-		_	_															_
Legend			1.Stakeholde	rs Requi	rements		Legend			B1.1	Requ	ired	func	tion	s ar	nd pe	rform	ance						
Refines			customer			Ε				🖿 E>	ecut	te gro	ound	tes	t								E	Ð
		E	Aircraft	requirem	ner ⊞					± E		Func	tion	s									ō	r s
Between messages and source functional			quirements	ements = 1	irements ■ Requirements ing]		Between messages and functions -			test(context	equest 🗉	nhibit condi ⊞ shibit condi ⊞	ue test requ ⊞	ed test displ ⊞	ed test inhik ⊞	ed test retrik ⊞	rrget 🚦	on informat 8	on message s	in-test con pr	selection re 🗄		m datapase wa	target membe
			Rec	Ŭ E G	em					ŭ	Ē	11	<u>-</u>	ate	ate	ate	ta	ti a	ij	s i	et	8	an	5
requirements - Manual			ctional I	ety Req ditional	rived Re ics syst [Engine	ards	automated			ite grot	ss aboi	ss clea	ss cont	ss initi	ss initi	ss initi	ve test	t condi	t condi	rogres:	ss targ	ss test	target i or and	d data
			a 3 a 4 Fun- a 7 Cra	a 57 Cra a 67 Sat a 85 Ad	3 97 De Avion Safety	stand				þ Execu	proce	proce	proce	broce	proce	proce) :retrie	inhibi	inhibi	test p	proce	broce	Boad Monit] Uploa
- C - Operational [A2 Lifecure]	+							_	_	لي.	6	00		6			UP		M					4
C - Operational [A2.Lifecycle Concepts] E 1 - Operational Use Cases and Scenarios							1 - Operational Use Cases and Scenarios										3	2	2 :	5 3				
							Execute ground test										3	2	2 :	5 3				
Execute ground test E Suppy day S1 with data – Select target member system and test1							🖃 🛤 Sunny day S1 with data – Select target member										1			1 1				
→ asynchSignal Created Test Information Message	3	3 3	÷ 2		1		→ asynchSignal Created Test Information Messa	1	1	1						1					_		_	
→ asynchSignal In-test message	2	2 2	2				→ asynchSignal In-test message	1	1	1						1	Ľ	<u> </u>						
→ asynchSignal Initiated Test Information	2	2 2	2				→ asynchSignal Initiated Test Information	1	1	1						1								
→ asynchSignal Initiated Test Retrieve Request	3	3 3	3				→ asynchSignal Initiated Test Retrieve Request	2	2	2														
→ asynchSignal Initiated Test Run-Test Command	1	1 1	. 1				→ asynchSignal Initiated Test Run-Test Comma	1	1	1						1			E.	/				
→ asynchSignal Start Test Request	1	1 1	. 1				→ asynchSignal Start Test Request	2	2	2														
→ asynchSignal Target Selection Request	2	2 2	2				→ asynchSignal Target Selection Request	2	2	2														
→ asynchSignal Test Creation Request	1	1 1	. 1				→ asynchSignal Test Creation Request	1	1	1						1								
→ asynchSignal Test Progress Information	3	3 3	2		1		→ asynchSignal Test Progress Information	1	1	1						1				1				
synchCall Create new test	1	1 1	. 1				SynchCall Create new test	1	1	1						1								
synchCall Perform inhibit condition check	4	4 4	4				SynchCall Perform inhibit condition check																	
😐 synchCall Perform target inhibit test	4	4 4	4				synchCall Perform target inhibit test																	
SynchCall Retrieve Initiated Tests	2	2 2	2				SynchCall Retrieve Initiated Tests	1	1	1						1								
→ synchCall Set Test Target	1	1 1	. 1				O synchCall Set Test Target	1	1	1						-					1			
E Sunny day S2 with data - Non-interactive, execute target test with in	1		20		3		E E Suppy day S2 with data - Non-interactive ever	-	-	6		3 3						dina i			- 1			
E Sunny day S3 - Non-interactive, uninhibited initiated test with autom	A		14				E Sunny day S2 - Non-interactive, uninhibited init			6		2 2	1						1 3)		
E 🖼 Sunny day S4 – Non-interactive, execute target test with inhibit condi	t 👘		17				Sunny day 53 - Non-interactive, driministed mit			6			<u> </u>					1	1 .	L 2 1	1	-		
🗄 🖼 Sunny day S5 – Interactive, uninhibited initiated test with automatic re	s		11				Sunny day 54 - Non-Interactive, execute target			4								1	1 4	2 I 1 1		L		
🗄 🖼 Sunny day S6 – Interactive, uninhibited initiated test that later becom	e		21				Gunny day SS - Interactive, uninnibited initiated i			4			-							1 1	4			
E Sunny day S7 – Abort execution of ground test			2		10		🗄 🖼 Sunny day S6 – Interactive, uninnibited initiated			12	-		2	1	1						2			
Monitor and analyze aircraft conditions					10		E 🖾 Sunny day S7 – Abort execution of ground test			2	3													
E O Monitor member system laut			1		10		Monitor and analyze aircraft conditions																7	
			1		2		⊞ ○ Monitor member system fault																	
View and print failure report					4		Upload data to target member system																	36
View member system configurations report					3		🗉 🔿 View aircraft parametric data																	
View Onboard Maintenance Documentation					1		🗉 💭 View and print failure report																	
					2		Uiew or reset members system fault history																	

End to end traceability





No Magic European Conference 2018 - Samares Engineering

System functional requirements?



- Where is system functional specification ? Where are system functional requirements?
- They live in the model ! They can be derived from all functions and their definitions, starting from top-level functions down to leaf functions

Functional requirements shall take into account

- Activity parameters
- States that restrict function validity
- Activation condition (end of other functions, signal event)
- Specification of actions to perform if any (design excluded)

Big picture (work presented in orange)

Collect stakeholder Requirements Architecture needs and constraints capture and definition definition of top (functional and level functions doc physical) Structure functional doc needs into scenarios doc Define top level system **Define functional** functions and flows architecture Formalize physical interfaces **Define physical** architecture I had target and system design Full set of System requirements requirements

ENGINEERING Accelerate Systems Design

Demonstration (if remaining time)



UAV for agriculture

• Treat field against pathogenic agents





Iteration 1

- Structure input requirements into UC and scenarios
- Identification of top-level functions
- Refinement of functions and identification of missing behavior...

Iteration 2

- Go back to scenarios and updates
- Identification/update of top-level functions
- Refinement of functions
- Simulation

Future work



- Complete formalization of requirements
 - Design and Physical constraints
 - Complete function definition with performance and safety concerns
- Connection of operational context with 3D animation framework
- Generate / update textual requirements from pattern
- Automated impact analysis (through traceability)
- Functional design and architecture trade-offs
 - Ability to represent different function realizations
 - Ability to select a given set of realizations for simulation



• Any question ?



Appendix - Import/synchronization of requirements



🖧 Cameo DataHub Explorer 🗉 Cameo DataHub Properties OMS-inputs:(IBM Rational DOORS-DOORS Database)
Cameo DataHub Explorer 🛛 🖓 🕂 🗙
Uperation: Copy Data with Sync 🔻
Iype text to search 📀 🕤
🖃 DH Cameo DataHub
🗇 📑 DOORS Database
E sample
□ [*] Avionics System Requirements Definitions
Errig 2 Onboard Maintenance
□ 1.2 Crew interface Requirements
□ 1.2.0-2 AVIONUES shall provide the capability to allow the maintainers to print(
1.2.0-4 AVIONICS shall provide the capability to allow maintainers to access and
1.2.0-6 AVIONICS shall provide the capacity to alphay maintenance asta in su
Tr 2 1 Functional Requirements
trian 2.3 Safety Requirements
The 2.4 Additional Certification Requirements
⊕ ≣ FMS Requirements
⊡ ∰ Member System Requirement for OMS
Trime 1 SCOPE
Trial 2 APPLICABLE DOCUMENTS
HIT 3 OVERVIEW
the definition of the second s
4.2 Central Maintenance Function.
4.2.1 Juring normal operation, the UMS continuously monitors the nealth state
+ 4.2.2 Ferlit Detection and Reporting Function
4.2.3 Thitisted Test Function
🖶 🗋 4.2.4.1 Description and philosophy.
4.2.4.2.0-3 [COM] The OMS will encode "Fault History Get First Block Comm
4.2.4.2.0-4 The member system shall store fault history data in persistent 🗸
4