MathWorks AUTOMOTIVE CONFERENCE 2023 Korea

Development of Fault Detection and Reaction Software Using Model-Based Development

Song Eun Jae, Hyundai Mobis





Table of Contents

- Hyundai Mobis Business Area and ADAS/AD Division
- Key Features of the Fail-Safe Software
- How Fail-Safe software is developed in Mobis
- Case in developing the Fail-Safe software: from the skeleton model to embedded code
- Case in improving work efficiency
- Concluding : Why the model-based development is applied for the Fail-Safe software
- Further Works

Hyundai Mobis Business Area







In-Vehicle-Infotainment

- Multimedia (Headunit)
- Head-Up Display
- Sound system
- Center Stack Display
- Switch Controller
- Telematics Unit

Auto Parking System (ECU, Sensor)
Surround View Monitoring System

Core Automotive Technologies

MOBIS

HYUNDAI

<u>Safety</u>

- Airbag Module : Front/Side/Roof Airbag
- Airbag Control Unit (incl. Remote impact sensor)
- Safety Control Module (using video data)

Module

- Chassis Module(Axle Assembly)
- Cockpit Module
- Front End Module

E-Powertrain

- Traction Motor / EDU / Starter & Generator
- OBC / DC-DC Converter / Inverter / ICCU
- BMS & Battery System
- 48V System
- In-Wheel Motor

Lighting

Radar (Front / Corner)

Driver Monitoring SystemOccupant Monitoring System

In-Cabin Sensing

ADAS

- ADB Headlamp
- HD Headlamp
- Homogeneous Rear Lamp
- Grille Lamp

Chassis

- Conventional Brake : Caliper/Drum Brake/EPB
- Slip-control, e-Boost : One box, Two box
- Steering EPS : C/DP/R type with full redundancy
- Air Suspension

How the Fail-Safe Software ensures the system safety (1)

- It detects faults that could cause system malfunctions,
- It executes safety mechanisms to maintain the system within safe states despite the faults occurred,
 - Faults may come up anywhere: Supply Power, Bus-Off, Message Timeout, Invalid Signal, etc.
- It stores diagnostic trouble codes (DTC) in Non-Volatile Memory to facilitate self-diagnosis for fault identification and cause analysis.



How the Fail-Safe Software ensures the system safety (2)

- The Fail-Safe Software transitions to Safety state and notifies other systems or driver of the status of fault status,
- It ensures that the system operates with minimal performance to prevent accidents.



What the key features of the Fail-Safe Software are

- To detect system faults (error) and react about the faults (error), uses <u>standardized AUTOSAR</u> interfaces,
 - BswM, Dem, E2EXf, ComXf, Complex Device driver (CDD) ...
- The platform software related to the Fail-Safe notifies fault status to the application software periodically.
 - CRC error, Alive Counter error, Timeout error, Bus Off error, Out of range …
- The Fail-Safe application software qualifies the system faults and determines whether to transition the system operating mode to safe state or another state.



Software Architecture Scenario #n

How Mobis develops the Fail-Safe software (1)

Why not Bottom-up nor Round-Trip

- The Fail-Safe Software has high dependency on the embedded system,
 - It cannot be dependent from the hardware architecture
 - It employs AUTOSAR services in its design
- Rapid-prototyped Simulink models are not necessary,
- Simulink including AUTOSAR Toolbox is not yet 100% AUTOSAR compatible.

Top-Down Workflow

Top-down

How Mobis develops the Fail-Safe software (2)

Model-Based Development

- The Fail-Safe application software is developed with Simulink, Stateflow and Embedded Coder
- The Simulink models are auto-code generated and integrated in the CI environment

OS, RTE, NM, EcuM, BswM, Com ...

Creating Skeleton model from architecture design

- Design software architectural attributes
 - runnable and trigger type, port interfaces, data types, mapping information ...
- Generate ARXML file from the architecture design
 - In-house converter fills out XML metadata based on AUTOSAR shema
- Refine ARXML file with the AUTOSAR authoring tool (DaVinci, Mobilgene ...)
 - VFB is realized by connecting with BSW
- Create a Simulink Skeleton model by importing the refined ARXML file with using Matlab commands.

Top Level

1+3

3+3 +Cord_CONNELLOFF_N (++3) Proceed Weller and

2+3

 data
 2121_C10000_[0_1]/00_2^{-1021}_{10

Subsystem Level

8

Designing the Fail-Safe algorithm within the Skeleton (MIL)

- Constructing a simulation model for verifying the goals of the software
 - Testing of increment/decrement of fault count, expectation when changing preconditions or inputs ...
- Designing the internal behaviors of the function (runnable) for fault detection and reaction
 - Out of voltage range, Timeout, E2E violations, Invalidity, Bus-Off ...
- Verifying the functionalities via MIL and improving the function behaviors.

Developing the Fail-Safe Software code using the embedded coder and HIL

- Auto C code generation from the verified module using the embedded coder and generating executable image (ELF, Hex ...) after integrating with AUTOSAR codes,
- Simulating the faults situation using a hardware simulator and monitoring the Fail-Safe Software via the debugger and network monitoring equipment,
 - Out of voltage range, Timeout, E2E violations, Invalidity, Bus-Off, DTC status ...
- Verifying the functionalities and generated C code via HIL and improving the function behaviors.

Requirement management using the Requirement Tool Box

- To define and descript the unit of Fail-Safe functionalities, use the Requirement Manager and Requirement Editor Apps,
- Using Requirement Manager App and Requirement Editor App, adding the new requirement set on the top-level and creating new requirement on the subsystem (meaning unit, runnable ...) to define and descript the functionalities,
- Creating the traceability between sub-system (meaning unit, runnable ...) and requirements.

Re_CtApFaultIIngr_Init Re_CtApFaultIIngr_MainFunction10ms_sys Re_CtApFaultIIngr_PreConChk_10ms_sys Re_CtApFaultIIngr_CMR_Ctri01_RxChk_sy

(2) Adding New Requirement Set and Creating Requirements

						Property inspector (0.3
	Requirements - CtApFault	Mngr				Requirement: #1
	View Requirements of			t 🔗 🚳 Salact View		* Properties
	new. [nedniements]			Geneer view	· · ·	Type: Functional ~
	Index		ID			Custom ID: 61
	CtApFaultMngr					Summary: [Re_CtApFault/IngrUnit
	B 1	#1			Re_CtApFaultM	Description Rationale
	E 2	#2			Re_CtApFaultM	
	E 3	#3			Re_CtApFaultM	Design intention This unit is responsible to initialize global variables and fault condition.
	E 4	#4			Re_CtApFaultM	Fault condition is set as "Passed" Set OperationCycleState as "DEM_CYCLE_STATE_START"
	E 5	#5			Re_CtApFaultM	Initiate IVV synchronization for DEM
	iii 6	#6			Re_CtApFaultM	0
	iii 7	#7			Re_CtApFaultM	Verification Criteria Input : None
	iii 8	#8			Re_CtApFaultM	Initialze FPIusVariant as '0'
	16 9	#9			Re_CtApFaultM	HWER, Temperature, CAVBusOR) as "OEM_EVEND_BTATUS_PASSED" HWER, Temperature, CAVBusOR) as "OEM_EVEND_STATUS_PASSED" Set Document (CAVBUSOR) STATE START
	E 10	#10			Re_CtApFaultM	Read ERR by calling ReCtApFaultiling_IV/Synchronization_SynchronizeIV/DataCl
	A DAS DRV 01 10ms	uirem	nents			Revision Information View Bac Outer Institute Line Bac Outer Institute Line
RpClApFaultMngr_BowM_MSI_ACA	N_BUSOFF_BswM_MDGI ationControl_ComMConf_ ngr_PreConChk_10ms	P_ACAN_BUSOFI ComMChannel_CP _sys	F_reak			Connects
	Select View	v				

Improving work efficiency – Automated Model Creating and Updating (1)

 Following the metrics for sub-system (meaning units, runnables...) count,

S.No	Failure Mode	Runnable Count (in Model)/DTC Count
1	CAN Bus Off	9
2	Message Time-out	218
3	Signal Invalidity	218
4	Battery Voltage Failures	5
5	Others	20

- As part of this automation, the following Fail-Safe modes are considered (100% automated)
 - CAN Bus Off, Message Time Out, Signal Invalidity

APPS			\frown				
	*** C	S		\mathbf{X}	\mathbf{X}	5	2
Optimizatio	n MATLAB Coder	Application Compiler	Failsafe Logic Automation Tool	Fuzzy Logic Designer	Neuro-Fuzzy Designer	Driving Scenario Designer	Ground Truth Labeler
						APPS	
Failsafe_Logic	_Automation_V2p0						
	🕤 Comman	d Winge					
🙀 Fail:	safe Logic Auto	mation To	ol		- C	ı x	
Ske	leton Mode	l Genero	ation:		Generat	е	
Reti	ned lemplo	ite Gen	eration:		Generat	e	
E.ul		т.					
Fau	t Manager	Logic Ir	npiementa	rion:	Generat	e	
AATI	Test Envir		Cuentinu				
/V\IL	. Test Envir	onmen	creation		Genera	e	
PTI	Test Envin	onment	Creation		Generat		
			er ourronn		Seneral		
					Manual		
					wanua		

Improving work efficiency – Automated Model Creating and Updating (2)

Improving work efficiency – Automated Model Re-Architecting

- All runnables are created on the Top-Level in the skeleton model,
- Necessary to group similar runnables together to improve readability,
 - Timeout, Invalidity ...
- Creating sub-system and move the runnables to corresponding sub-system automatically.

		R @ /~ /												
				arch Documentatio	on									
					D		1							
	on Tool Re-Architectur	⊂ 🔺 Failsafe Re-	Arch Automation To	ol	- 0	I X								
	APPS	_				_								
	h App Failsate Model g <u>Started</u> .	Model R	e-Architectur	e 📃	Generate									
		- Determ	ine Subsystem Iandlina		Generate									
					Manual									
				Qui										
				Sui	o-syste									
				~										
Model	Re-Archite	ecture						Relation	LMgGr_CCAN_EECI_FG.	read				
	<u> </u>			-		RpEEC	-T-	form2	ApNon_Relivent_EBS					
							.T. >	Ravocs	2_EBS_MsgGr_CCAN_VDC2	EBS_mad				
			0	-		2		Ra_CpA	ApNon_Relivent_SJB					
								Ra Col	JMagGr_CCAN_OEL_read					
		•	0			Ē		RpCLU	mag06_MagGr_CCAN_CLUm	isg06_read				
							.T.	Ra_CpA	QNon_Relivent_HRW					
	-				—		\times	RaHRM	(MagGr_CCAN_HRW_mad					
		0			Internation			Ra_CpA	ApMon_Relivent_EBC					
		Se	etting the po	ositions of	blocks			RpEBC1	1_MsgGr_CCAN_EBC1_read		invalidity	check		
						Car	ICOL							
						Oai								
	-					-T-	>	Re_CpApNon_R	trTOutEvent_EMS					
				~		-T-	5	Re_CpApMon_R	trOutEvent_EBS					
						·Ţ.	>	Re_CpApMon_R	aTOuEvent_SJS		- V			1
			_	-		·T-	>	Re_CpApMon_R	trTOutEvent_CLU					1
						·T-	>	Re_CpApMon_R	tcTOutEvent_HRW					1
			_	-		-T-	\geq	Ra_CpApNon_R	DIE_trevEuOTs					
											Timeout che	ic.		
				-										
							•							
				Sub	o-syste	m								

Concluding Remarks : 'Why Mobis applies Model-Based Development to the Fail-Safe Software?'

Reusability

Different Sensors/ECUs but similar fault detection algorithms.

How can we make it easy to reuse?

Quality Maintenance

 Developers' different coding capabilities. ASPICE consistency hard to achieve. Coding rule compliance.

How can we make it easy to maintain?

Collaboration

Ambiguity in natural language (especially in international collaboration)

How can we make it easy to share design intention precisely?

Future Works

System Composer and Requirement Management

 Designing the Fail-Safe Software Architecture in Simulink and managing software requirement & software architecture requirement ...

Advanced usage of Autosar Tool Set

Improving the simulation environment using Dem, Nvram Tool box

Enhanced Fail-Safe automation

 Automatically adding the Fail-Safe mechanism to the software component ports, interfaces ... and validating whether it has been applied as intended.

MathWorks AUTOMOTIVE CONFERENCE 2023 Korea

Thank you

© 2023 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See *mathworks.com/trademarks* for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.