2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Accellera Functional Safety Working Group Update and Next Steps

Alessandra Nardi, Accellera FS WG Chair

Ghani Kanawati, Technical Director of System Reliability, ARM

accellera SYSTEMS INITIATIVE

Agenda

- The Accellera Functional Safety Working Group (FS WG)
  - Mission and the FS Standardization Landscape
  - Scope and Key Objectives
- The Accellera Functional Safety Standard
  - Data Model Development
  - Data Model White Paper
  - An example
- What’s Next?

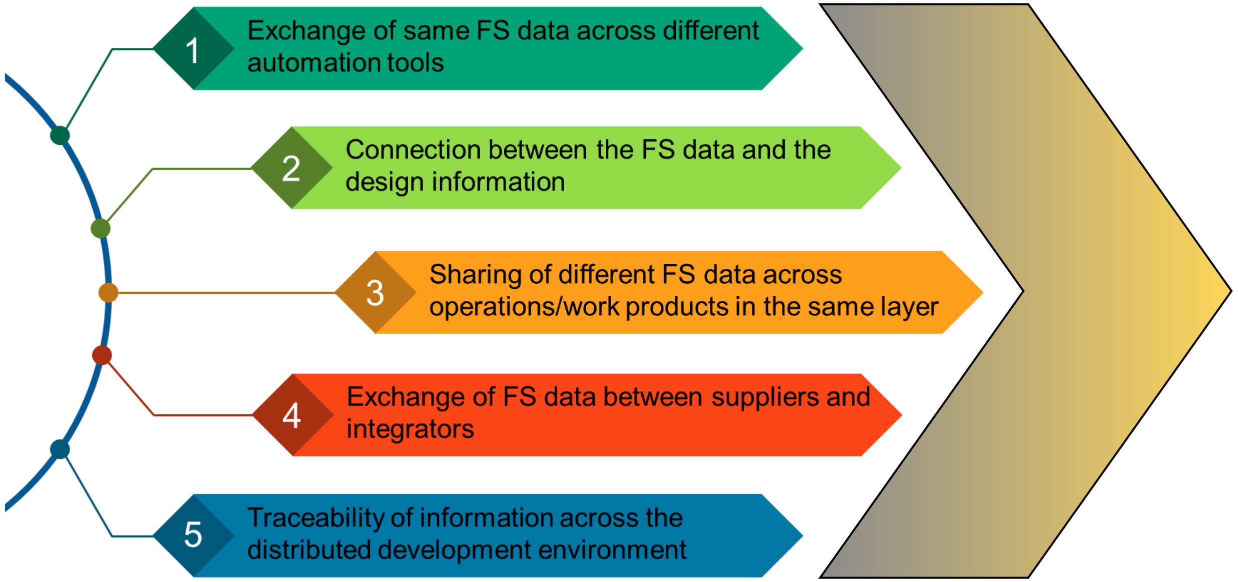
Work in Progress

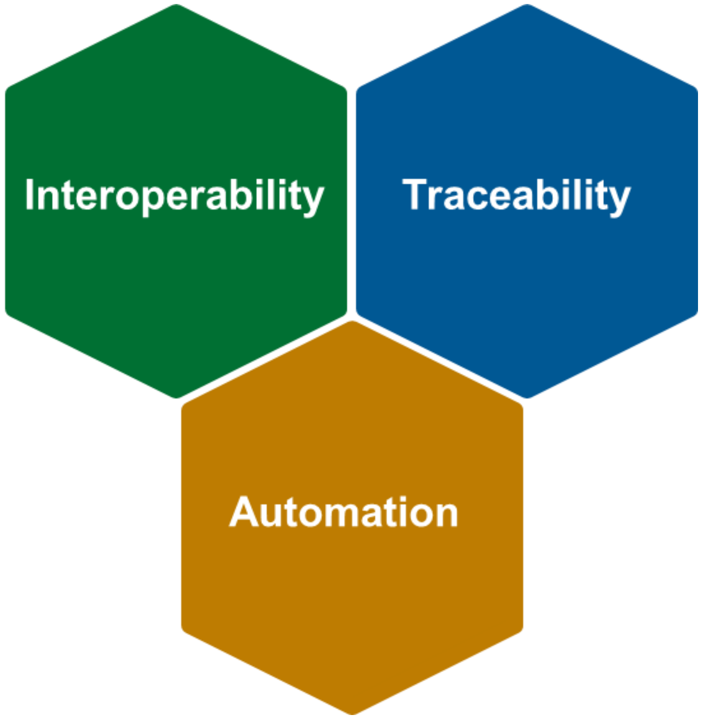
2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Mission and Key Objectives

accellera SYSTEMS INITIATIVE

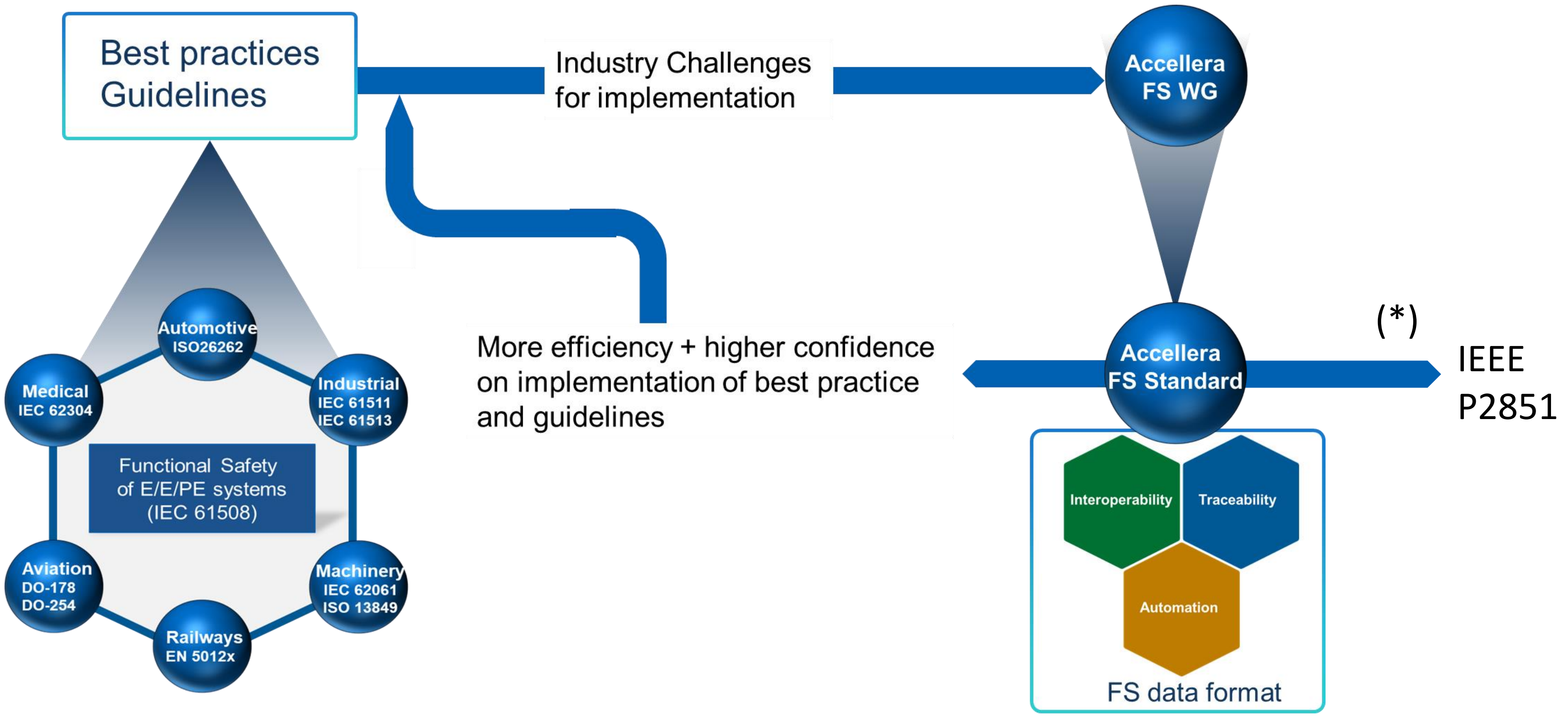
Mission of the FS WG





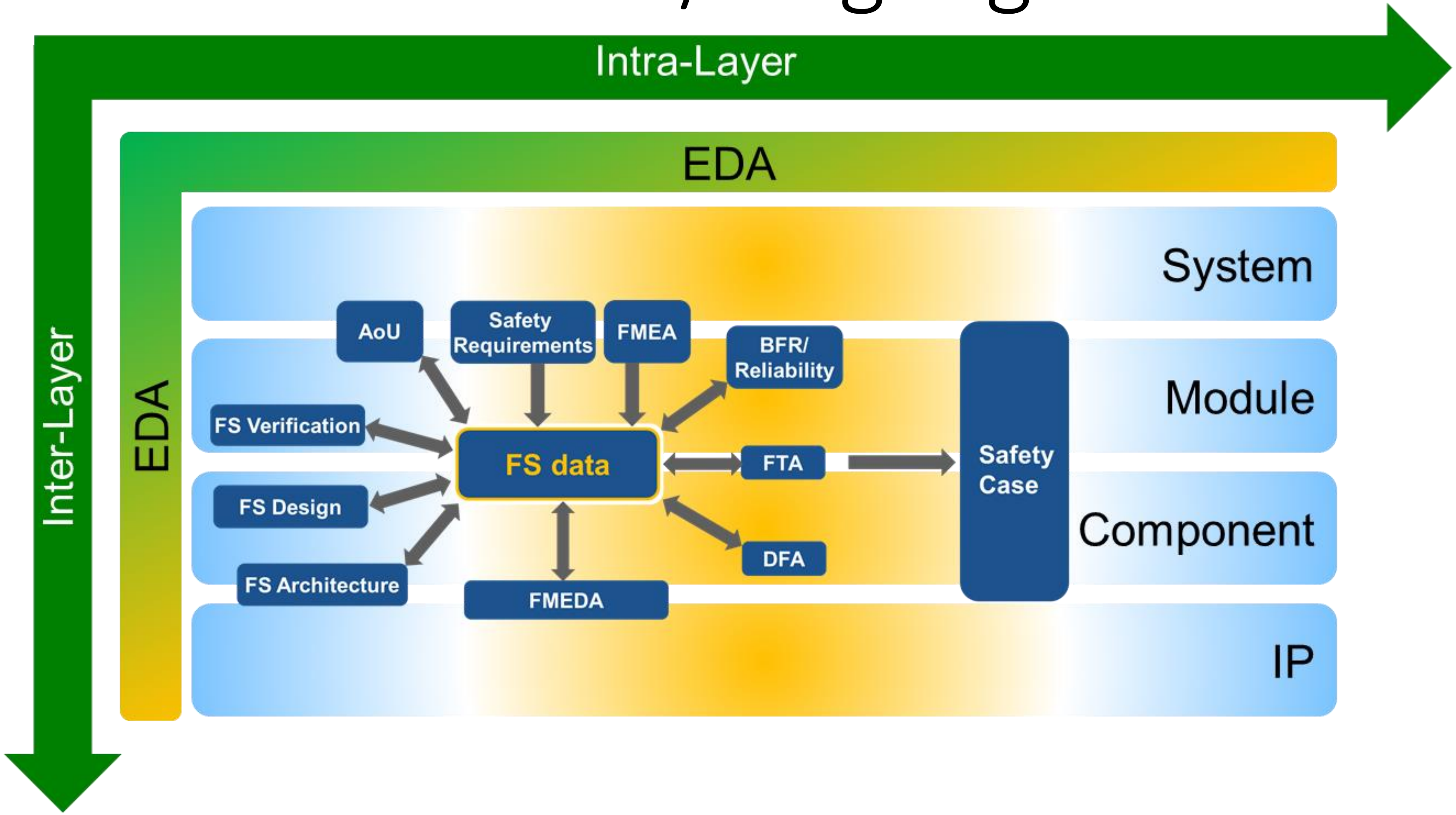
- Define a FS language to capture and propagate the functional safety data through the flow/supply chain
- Enable interoperability, traceability and automation

Mission and the FS standardization Landscape



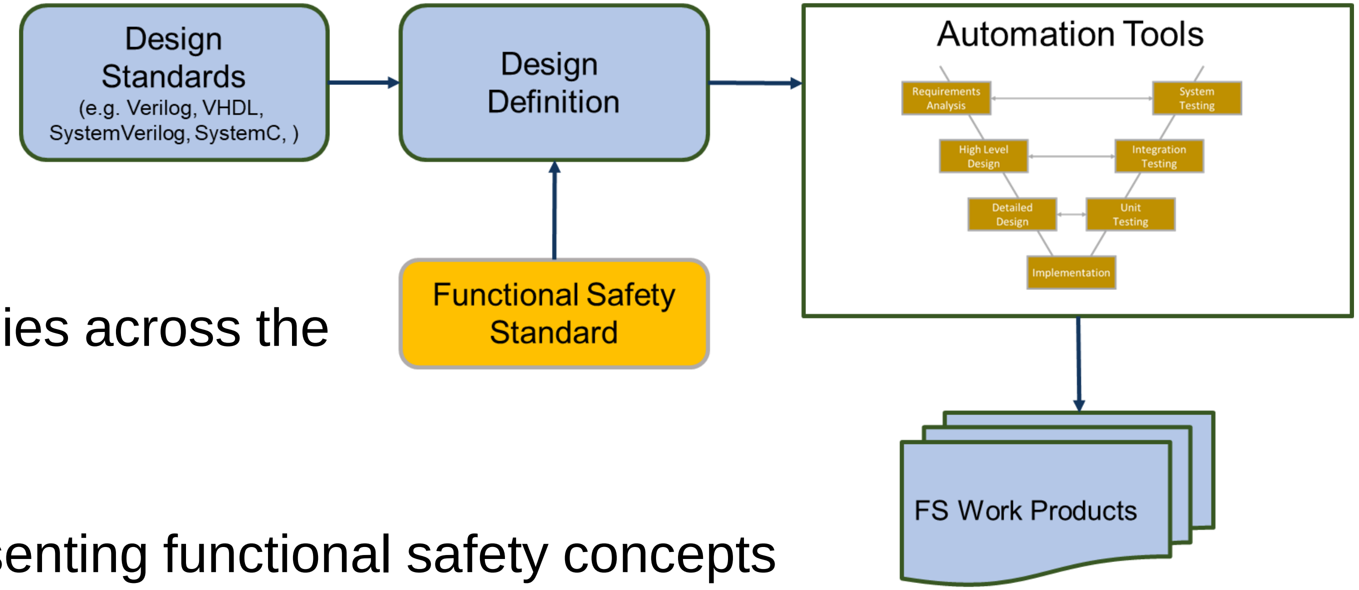
(*) Once completed and published, the Accellera FS standard is planned to be contributed to IEEE as per traditional collaboration between Accellera and IEEE

Accellera FS data format/language



FS data = set of data needed to perform safety activities and to generate work products

Key Objectives



- Harmonize best practices and methodologies across the industry via common language

- Enable efficient interchange of data representing functional safety concepts
  - across the diverse lifecycle development tool chain and
  - among organizations engaged in distributed development

- Be comprehensive, flexible, and scalable to minimize future perceived needs for local or proprietary customization

The data model is in addition to the existing design standards

Data Model Development

Approach to Data Model Development

1. Functional Safety Analysis Process Formalization
2. Conceptual Data Model
3. Functional Safety Language

Conceptual Data Model → LRM (*)
Functional Safety Language → LRM (*)

The actual exchange of information will happen through the FS Language

(*) Language Reference Manual

The conceptual data model approach

Goals:

- Define FS data
- Not to provide a reference implementation
- Systematic approach to define a language/format

Conceptual Data Model:

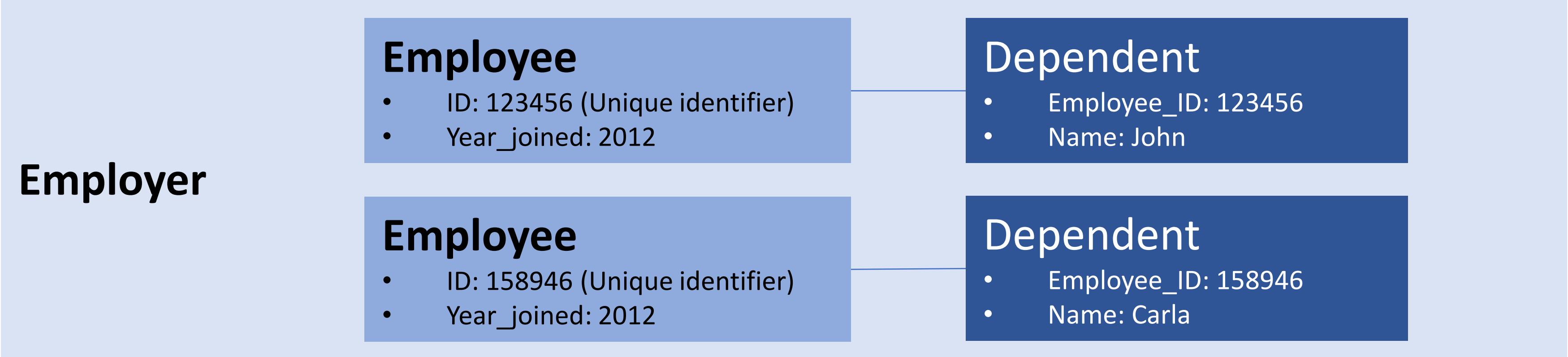
- Defines WHAT the system contains
- Does NOT define HOW the system should be implemented

Using the Entity Relationship model

The 3 basic tenants:

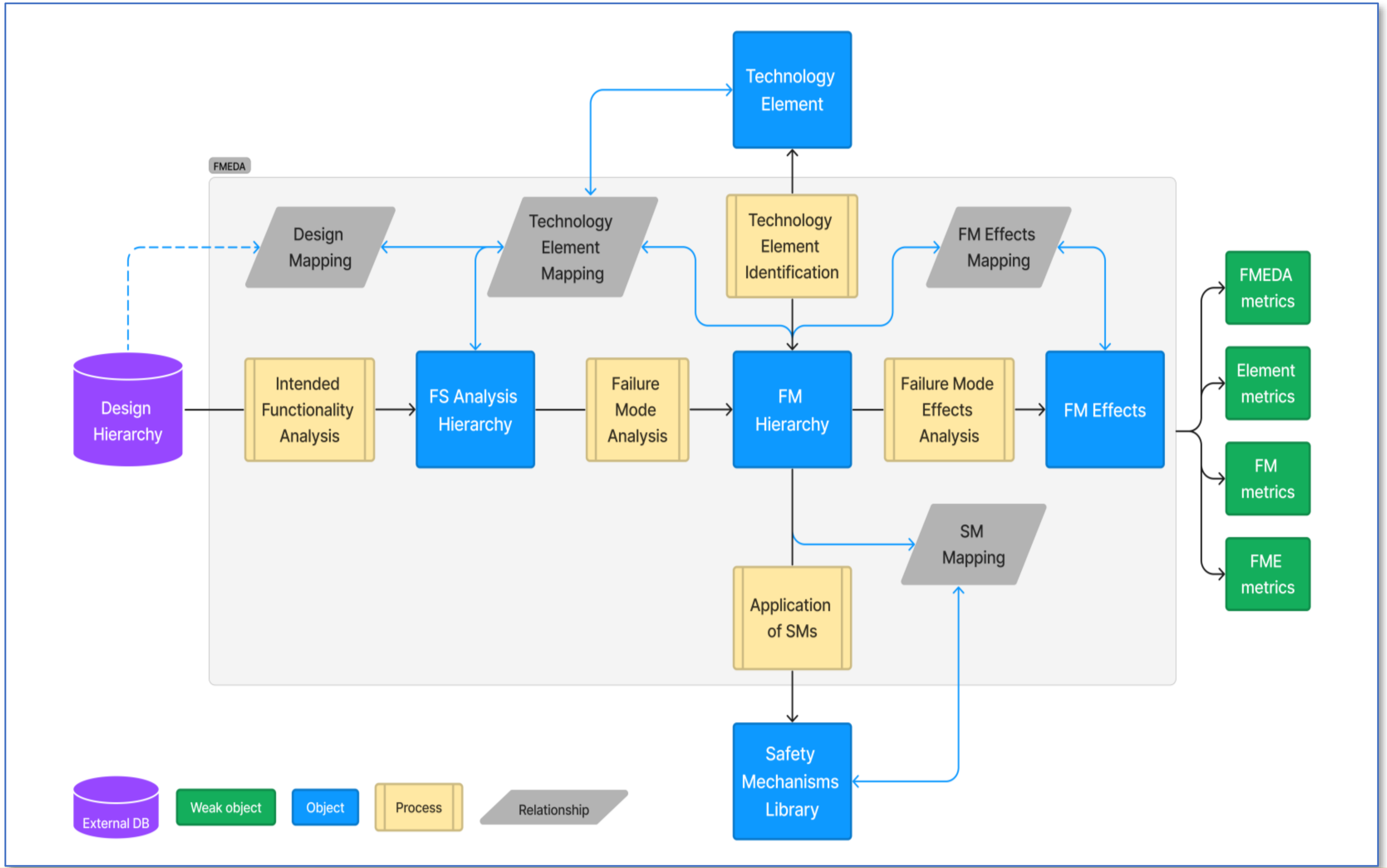
- **Entity**: The object/data describing the system to be modeled
- **Attribute**: Characteristics or properties of an entity
- **Relationship**: Dependency or association between two entities

In addition, we rely on the concept **Weak entity**, which cannot be identified by its attributes alone, but only exists in the context of another entity

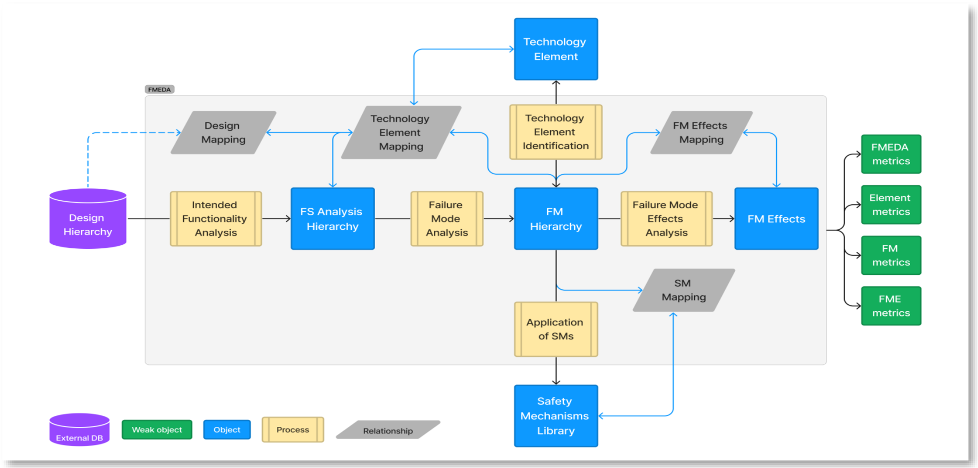


Source: https://www.guru99.com/data-modelling-conceptual-logical.html

FMEDA Process



Conceptual Data Model derived from the FMEDA process



Functional Safety Data Model			
FMEDA process data	**Entity Type**	**Information Type**	
FMEDA	FMEDA	Object	Object
FS Analysis Hierarchy	Element	Object	Object
FM Hierarchy	Failure_Mode	Object	Object
Technology Element	Technology_Element	Object	Object
Safety Mechanism Library	Safety_Mechanism	Object	Object
FM Effects	Failure_Mode_Effect	Object	Object
SM Mapping	SM-FM	Relationship	Relationship
FM Effects Mapping	FM-FME	Relationship	Relationship
Technology Element Mapping	TE-FM	Relationship	Relationship
Technology Element Mapping	TE-Element	Relationship	Relationship
Design Mapping	Inside the TE-FM since there is no Design Hierarchy in the data model	Relationship	Relationship
Design Mapping	Inside the TE-Element since there is no Design Hierarchy in the data model	Relationship	Relationship
Calculated FR	FR_ISO26262	Weak object (*)	Weak object
Calculated metrics	Metrics_ISO26262	Weak object (*)	Weak object
Calculated FR	FR_IEC61508	Weak object (*)	Weak object
Calculated metrics	Metrics_IEC61508	Weak object (*)	Weak object

Direct traceability from the data + mapping of FMEDA process to data model

Sample Language

Functional Safety Data Model			
FMEDA process data	**Entity Type**	**Information Type**	
FMEDA	FMEDA	Object	Object
FS Analysis Hierarchy	Element	Object	Object
FM Hierarchy	Failure_Mode	Object	Object
Technology Element	Technology_Element	Object	Object
Safety Mechanism Library	Safety_Mechanism	Object	Object
FM Effects	Failure_Mode_Effect	Object	Object
SM Mapping	SM-FM	Relationship	Relationship
FM Effects Mapping	FM-FME	Relationship	Relationship
Technology Element Mapping	TE-FM	Relationship	Relationship
Technology Element Mapping	TE-Element	Relationship	Relationship
Design Mapping	Inside the TE-FM since there is no Design Hierarchy in the data model	Relationship	Relationship
Design Mapping	Inside the TE-Element since there is no Design Hierarchy in the data model	Relationship	Relationship
Calculated FR	FR_ISO26262	Weak object (*)	Weak object
Calculated metrics	Metrics_ISO26262	Weak object (*)	Weak object
Calculated FR	FR_IEC61508	Weak object (*)	Weak object
Calculated metrics	Metrics_IEC61508	Weak object (*)	Weak object

Annex B: Language

Introduction

In this paper we defined a sample language for the only purpose of showing some concrete examples of usage of the Functional Safety Standard. The final LRM defined in the standard might differ from the sample used in this paper.

Following the principle of traceability, the sample language is derived directly from the conceptual data model with remarkably simple rules:

- Objects are created with “create” commands and updated with the “-update” option.
- Relationships are created with the "assign" commands.
- Weak objects are assigned a value with the "define" command.

In other words, the sample language is the implementation of the requirements defined in the conceptual data model.

A special rule stands for the Design mapping since it connects objects in the data model to objects in the design hierarchy, which are not part of the data model. The design mapping connection is described through the “-mapping” and “-exclude_mapping” options inside the design mapping relationship commands.

Conceptual Data Model + sample commands

FMEDA process data	Entity Type	Information Type	Commands
FMEDA	FMEDA	Object	create_fmeda
FS Analysis Hierarchy	Element	Object	create_element
FM Hierarchy	Failure_Mode	Object	create_failure_mode
Technology Element	Technology_Element	Object	create_failure_mode
Safety Mechanism Library	Safety_Mechanism	Object	create_failure_mode
FM Effects	Failure_Mode_Effect	Object	create_failure_mode
SM Mapping	SM-FM	Relationship	assign_SM_FM
FM Effects Mapping	FM-FME	Relationship	assign_FM_FME
Technology Element Mapping	TE-FM	Relationship	assign_TE_FM
Technology Element Mapping	TE-Element	Relationship	Assign_TE_Element
Design Mapping	Inside the TE-FM since there is no Design Hierarchy in the datamodel	Relationship	assign_TE_FM –mapping {…} –exclude_mapping
Design Mapping	Inside the TE-Element since there is no Design Hierarchy in the datamodel	Relationship	assign_TE_Element –mapping {…} –exclude_mapping
Calculated FR	FR_ISO26262	Weak object (*)	define_FR_ISO26262
Calculated metrics	Metrics_ISO26262	Weak object (*)	define_metric_ISO26262
Calculated FR	FR_IEC61508	Weak object (*)	define_FR_IEC61508
Calculated metrics	Metrics_IEC61508	Weak object (*)	define_metric_IEC61508

Functional Safety Data Model			
FMEDA process data	**Entity Type**	**Information Type**	
FMEDA	FMEDA	Object	Object
FS Analysis Hierarchy	Element	Object	Object
FM Hierarchy	Failure_Mode	Object	Object
Technology Element	Technology_Element	Object	Object
Safety Mechanism Library	Safety_Mechanism	Object	Object
FM Effects	Failure_Mode_Effect	Object	Object
SM Mapping	SM-FM	Relationship	Relationship
FM Effects Mapping	FM-FME	Relationship	Relationship
Technology Element Mapping	TE-FM	Relationship	Relationship
Technology Element Mapping	TE-Element	Relationship	Relationship
Design Mapping	Inside the TE-FM since there is no Design Hierarchy in the data model	Relationship	Relationship
Design Mapping	Inside the TE-Element since there is no Design Hierarchy in the data model	Relationship	Relationship
Calculated FR	FR_ISO26262	Weak object (*)	Weak object
Calculated metrics	Metrics_ISO26262	Weak object (*)	Weak object
Calculated FR	FR_IEC61508	Weak object (*)	Weak object
Calculated metrics	Metrics_IEC61508	Weak object (*)	Weak object

Use Cases

The data model implementation supports two main use cases:

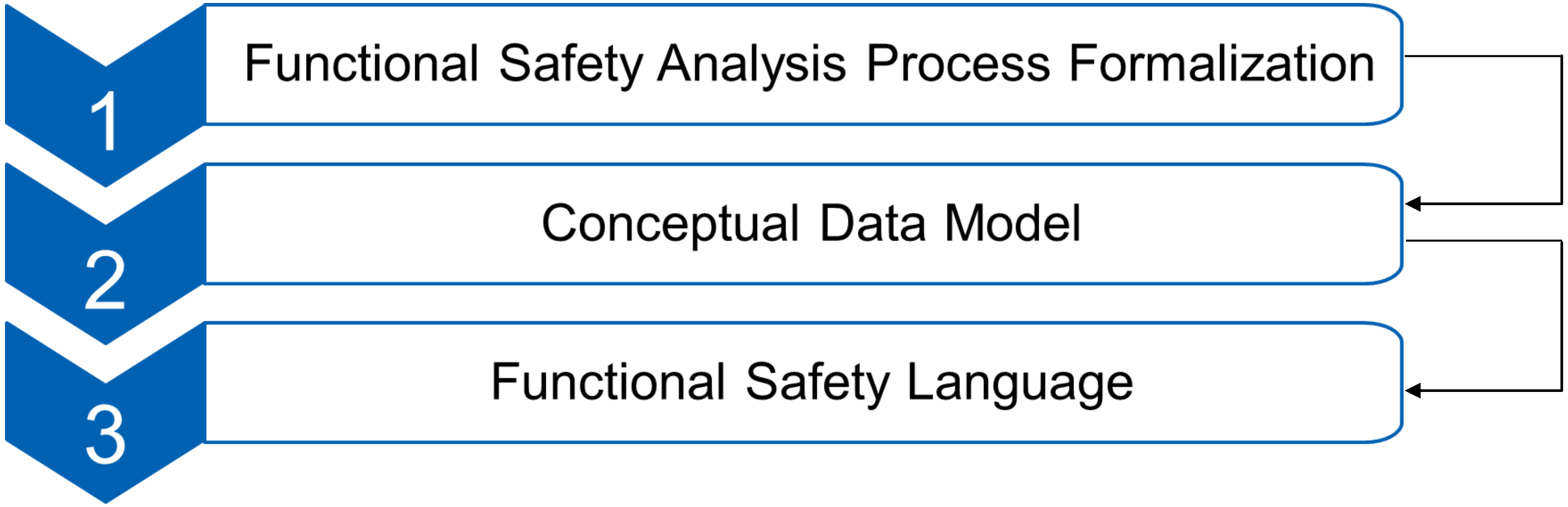
1) FMEDA evaluation: A safety analysis is performed and described, for example, by using a command-based formalism describing the atomic actions (e.g., create the safety analysis, create a failure mode, etc.). When the user decides to generate final reports, all of the outputs are also stored in the data model. In this use case the provided authoring information is evaluated with the intent to populate the data model and to be able to generate final reports.

2) “As is”: A safety analysis is shared “as is,” as for example an FMEDA table or summary. In this use case there is no authoring information but only failure rates and metrics to be exchanged as outputs (for example, following a numerical evaluation of the data model) or imported as inputs.

As stated in the Accellera FS WG white paper [1], the goal for the Accellera FS standard is to work in alignment with well-established safety standards (e.g., ISO26262 [2] and IEC61508 [3]) and to facilitate their implementation. Hence, calculations and definitions are meant to be consistent with such standards (unless stated otherwise).

Point of Discussion: Input and Output in the same format/file?

Traceability of Data Model Development



Direct consistent mapping from process to data model objects

Direct translation from data model to language with defined rules

Traceability from:
- Requirements (FMEDA process objects and mapping) to
- Implementation of requirements (FS data model and then language commands)

Data Model White Paper

Data Model White Paper

- Publish shortly after DVCon Europe 2023
- Status: Final Editorial Review
- Main Body:
  - FMEDA Process
  - Data Model
  - Associated methodology discussions
- Annexes:
  - Detailed Data Model
  - Prototype Language
  - What's after this version
  - Repository Example
- Annexes to evolve into LRM/User Guide

Contents

I. Introduction ... 5
II. FMEDA Process ... 8
III. Design Representation and Mapping of Data ... 9
Design Representation ... 9
Mapping ... 12
Design Mapping ... 12
Failure Modes Mapping ... 13
Safety Mechanism Mapping ... 14
Technology Element Mapping ... 15
Failure Mode Effects Mapping ... 16
Complex Use Cases ... 17
IV. FMEDA type ... 19
Assumption-based ... 19
Calculation-based ... 19
Mixing FMEDA Types ... 19
V. Conceptual Data Model ... 20
Introduction to the Entity-Relationship model ... 20
General considerations ... 21
VI. Detailed Annotations on the Data Model ... 23
FMEDA type (assumption-based, calculation-base) ... 23
FS Hierarchy and FM Hierarchy ... 24
Technology element ... 25
Digital ... 25
RAM/ROM/Flash ... 26
Analog ... 26
FS Hierarchy modeling ... 26
Operations on design mapping ... 27
DC aggregation methods ... 27
Failure Mode effect ... 28
VII. Concluding Remarks ... 32
Accellera FS WG Supporting Entities ... 33

Annexes of the Data Model White Paper

Acknowledgements 33
Annex A: Data model 34
FMEDA 36
Element 38
Failure Mode 39
Technology element 42
Safety mechanism 43
Failure mode effect 44
Mapping safety mechanism - failure mode 45
Mapping failure mode - failure mode effect 46
Mapping technology element - failure mode 47
Mapping technology element – element 49
Define ISO26262 failure rate 51
Define ISO26262 metric 52
Define IEC61508 failure rate 53
Define IEC61508 metric 54
Annex B: Language 55
Introduction 55
Conventions 56
Safety analysis commands v0.1 57
create_fmeda 58
create_element 60
create_fm 60
create_te 63
create_sm 65
create_fme 67
add_attribute 68
add_collection 70
assign_sm_fm 72
assign_fm_fme 74
assign_te_fm 75
assign_te_element 77
define_fr_iso26262 79
define_metric_iso26262 80
define_fr_iec61508 82
define_metric_iec61508 83
Annex C: Add-on to v0.1 84
load_slf 85
save_slf 86
set_scope 87
add_parameter 88
attr_expr 90
assign_fmeda_fmeda 91
assign_fmeda_element 92
Annex D: Repository 94
Example 1 94
Example 2 95
Example 3 95
Example 4 98
Introduction 98
Step 0. Understand the difference between a language and a data model 99
Step 1. Create a library of Collections of attributes 100
Step 2. Create a library of Safety mechanisms 102
Step 3. Create the Safety hierarchy 104
Step 4. Create Failure modes and assisting collections 105
Step 5. Assign Safety mechanisms to Failure modes 108
Step 6. Create technology elements 109
Step 7. Assign Technology elements to Failure modes, mapping 110
Step 8. Create Failure mode effects and connect them to Failure modes 112
Step 9. Update objects according to verification strategy 113
Step 10. Create FMEDA-scoped metrics 114
Step 11. Create FME-scoped metrics 115
Data tracing 116
Equivalent tables 117
Bibliography 120

2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
10 YEAR ANNIVERSARY

Detailed Data Model

accellera
SYSTEMS INITIATIVE

FMEDA

Entity name FMEDA

Key identifier FMEDA_Name

Attribute Name	Attribute Type	Description	Required
FMEDA_Name	String	Name (identifier) of the FMEDA of the project.	Yes
Type	Enumerate {assumption-based, calculation-based}	Defines the source of the failure mode distribution in case a choice needs to be made. The failure mode distributions can be calculated based on: • Estimations provided with the options fm_size or element_size • Design metrics extracted from the design mapping as specified in the fm_mapping and element_mapping When both options (*_size and *_mapping) are specified for an FM, the FMEDA type will select as follows: • assumption-based: The *_size takes precedence over *_mapping • calculation-based: The *_mapping takes precedence over *_size	No
ASIL	Enumerate {None, A, B, C, D}	Defines the ASIL target for the FMEDA (for a given Safety Goal) according to ISO26262. Used also to specify that the FMEDA is for ISO26262.	No
SIL	Enumerate {None, 1, 2, 3, 4}	Defines the SIL target for the FMEDA according to IEC61508. Used also to specify that the FMEDA is for IEC61508.	No
Analysis_Type	List of Enumerate {Permanent, Transient, All}	Defines the failure types to be considered and which metrics to be calculated within the safety analysis. More than one value can be specified, e.g., Analysis_Type = {Permanent} or Analysis_Type = {Permanent, Transient} The value "All" implies all Failure Types are activated. Defined as "All" instead of "Both" allows for plans for more than just Transient and Permanent.	Yes

Creator	String
Date	Date
Version	Float
Data_Model_Version	Float
Comment	String
Hierarchical	Enumerate {Yes, No}
User_Defined_Attribute	List of tuples

Element

Entity name: Element

Key identifier: Element_Name + Parent_Element + FMEDA_Name

Attribute Name	Attribute Type	Description	Required
Element_Name	String	Name (identifier) of the Element.	Yes
Element_Description	String	Description of the intended functionality of the Element.	No
Element_Type	Enum {System, Element, SubElement, Component, SubComponent, Part, SubPart}	Specifies the type of the Element. Element_Type = Component or SubComponent can only be defined if the analysis is for IEC61508, inferred from the FMEDA entity, whether it has ASIL or SIL defined.	Yes
Parent_Element	String	Connects the Element to its Parent in the FS hierarchy.	No
FMEDA_Name	String	Connects the FS hierarchy to the FMEDA project.	Yes
User_Defined_Attribute	List of tuples	List of previously created user-defined attributes and their values.	No

Point of Discussion: Required or Defaults?

Safety mechanism

Entity name Safety mechanism

Key identifier Safety_Mechanism_Name

Attribute Name	Attribute Type	Description	Required
SM_Name	String	Name (identifier) of the Safety Mechanism.	Yes
SM_Description	String	Description of the SM.	No
FMEDA_Name	String	Connects the FS hierarchy to the FMEDA project.	No
Class	Enumerate {HW, SW, AoU, AoU-SW, AoU-HW, user-defined}	Method by which the safety mechanism is to be realized. Notes: 1) AoU is to capture when the SM is not part of the product (potentially raise a flag during FMEDA integration) 2) HW allows for further specification for downstream tools	No
Class_description	String	Description of the class. This is specially meant in the case in which the class is user-defined, but available for all classes.	No
Configurable	Boolean {yes, no}	Captures whether the SM can be turned on or off by the user/integrator. If configurable=yes, then the "SM-FM active" attribute can be used.	Yes
DC_Perm	Float [0, 100]	Diagnostic coverage of the SM in isolation for permanent faults.	Yes
DC_Trans	Float [0, 100]	Diagnostic coverage of the SM in isolation for transient faults.	Yes
DC_Lat	Float [0, 100]	Diagnostic coverage of the SM in isolation for latent faults. This attribute is only available when the ASIL target level is defined. Not available if only the SIL target is defined.	Yes
User_Defined_Attribute	List of tuples	List of previously created user-defined attributes and their values.	No

To apply a diagnostic coverage specific to an SM-FM pair, use the DC_ *type* attribute in the SM-FM category. When SM:DC_ *type* and SM-FM:DC_ *type* are specified, the SM-FM:DC_ *type* attribute takes precedence. See Mapping safety mechanism - failure mode for details.

Mapping safety mechanism - failure mode

Entity name SM_FM

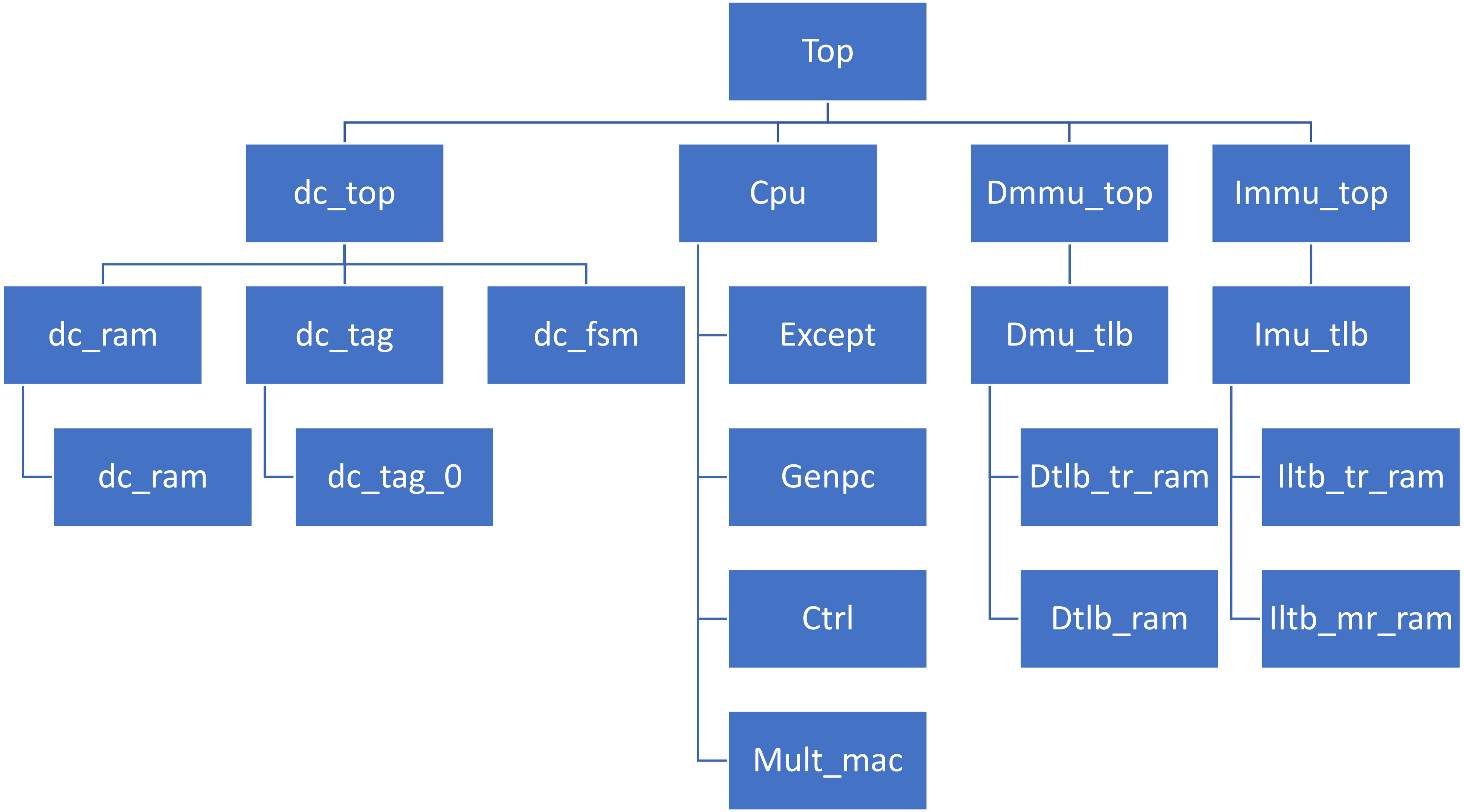
Key identifier Assignment_Name + FMEDA_Name

Attribute Name	Attribute Type	Description	Required
SM_Name	String	Name (identifier) of the SM applied to the FM.	Yes
FM_Name	String	Name (identifier) of the FM covered by the SM.	Yes
Parent_Element	String	Connects the Failure Mode to its Parent in the FS hierarchy.	Yes
FMEDA_Name	String	Connects to the FMEDA project.	Yes
DC_Perm_Estimated	Float [0, 100]	Diagnostic coverage of the SM applied to the FM for permanent faults.	No
DC_Trans_Estimated	Float [0, 100]	Diagnostic coverage of the SM applied to the FM for transient faults.	No
DC_Lat_Estimated	Float [0, 100]	Diagnostic coverage of the SM applied to the FM for latent faults.	No
DC_Perm_Measured	Float [0, 100]	Diagnostic coverage of the SM applied to the FM for permanent faults as a result of Fault Injection Activities.	No
DC_Trans_Measured	Float [0, 100]	Diagnostic coverage of the SM applied to the FM for transient faults as a result of Fault Injection Activities.	No
DC_Lat_Measured	Float [0, 100]	Diagnostic coverage of the SM applied to the FM for latent faults as a result of Fault Injection Activities.	No
Active	Boolean {yes, no}	Specifies whether the SM is enabled for this FM. Only accessible if the SM_Configurable attribute=yes.	Yes
User_Defined_Attribute	List of tuples	List of previously created user-defined attributes and their values.	No

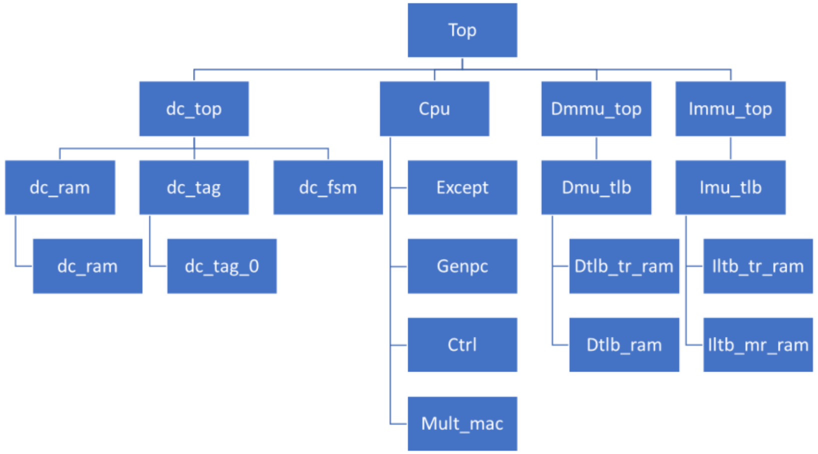
DC_ *type* value is specific to the SM-FM pair and takes precedence over the DC_ *type* of the SM category. If such value is not specified, then the value is taken from the DC_ *type* attribute of the SM category.

FS Standard Example

Example: Design Under Analysis

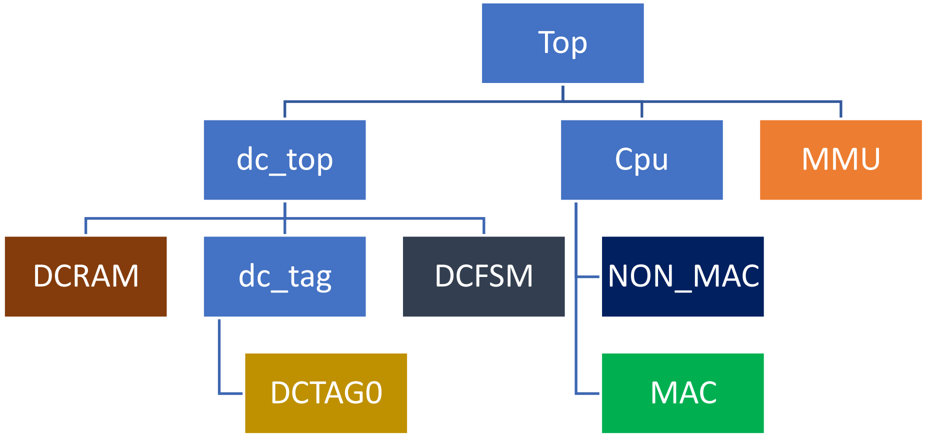


Example: FS Analysis Hierarchy



Design Under Analysis

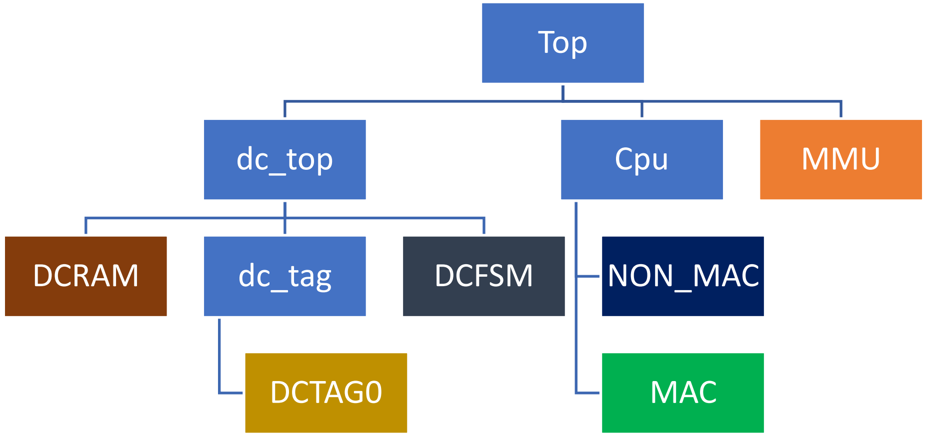
Analysis of the Intended Functionality



FS Analysis Hierarchy

Part	Subpart
TOP	MAC
	NON_MAC
	MMU
	DCFSM
	DCTAG0
	DCRAM

Example: FS Analysis Hierarchy

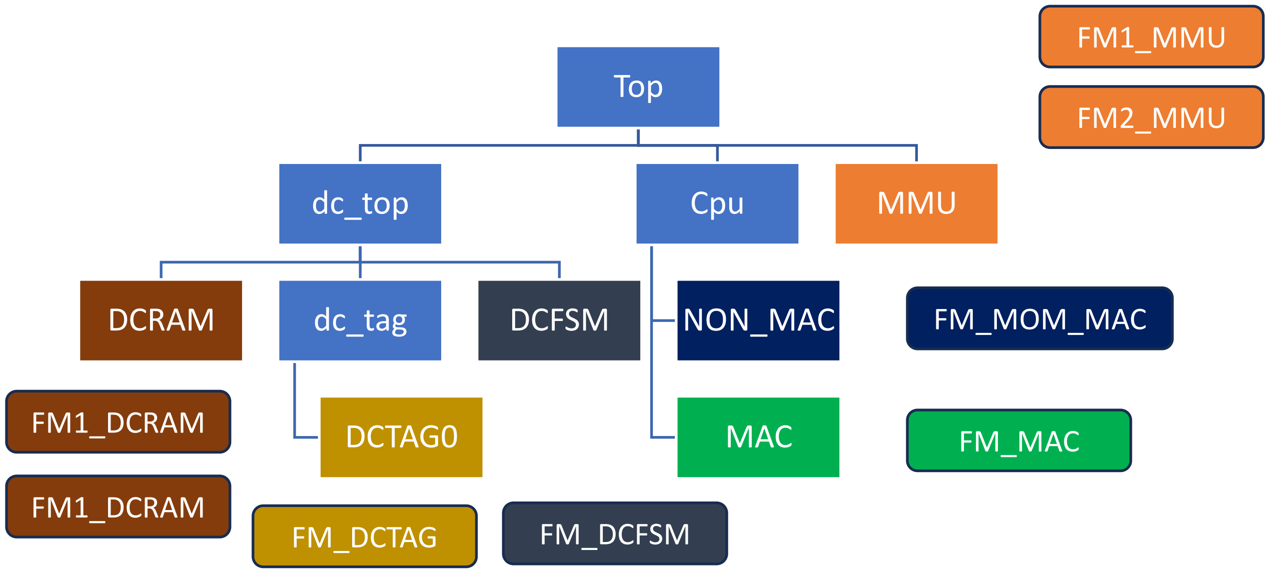


Part	Subpart
TOP	MAC
	NON_MAC
	MMU
	DCFSM
	DCTAG0
	DCRAM

Create_fmeda MY_FMEDA
Create_element –type part TOP
Create_element –type subpart MAC –parent TOP -fmeda MY_FMEDA
Create_element –type subpart NON_MAC –parent TOP -fmeda MY_FMEDA
Create_element –type subpart MMU –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCFSM –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCTAG0 –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCRAM –parent TOP -fmeda MY_FMEDA

Example: FM Hierarchy

Create_fmeda MY_FMEDA
Create_element –type part TOP
Create_element –type subpart MAC –parent TOP -fmeda MY_FMEDA
Create_element –type subpart NON_MAC –parent TOP -fmeda MY_FMEDA
Create_element –type subpart MMU –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCFSM –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCTAG0 –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCRAM –parent TOP -fmeda MY_FMEDA

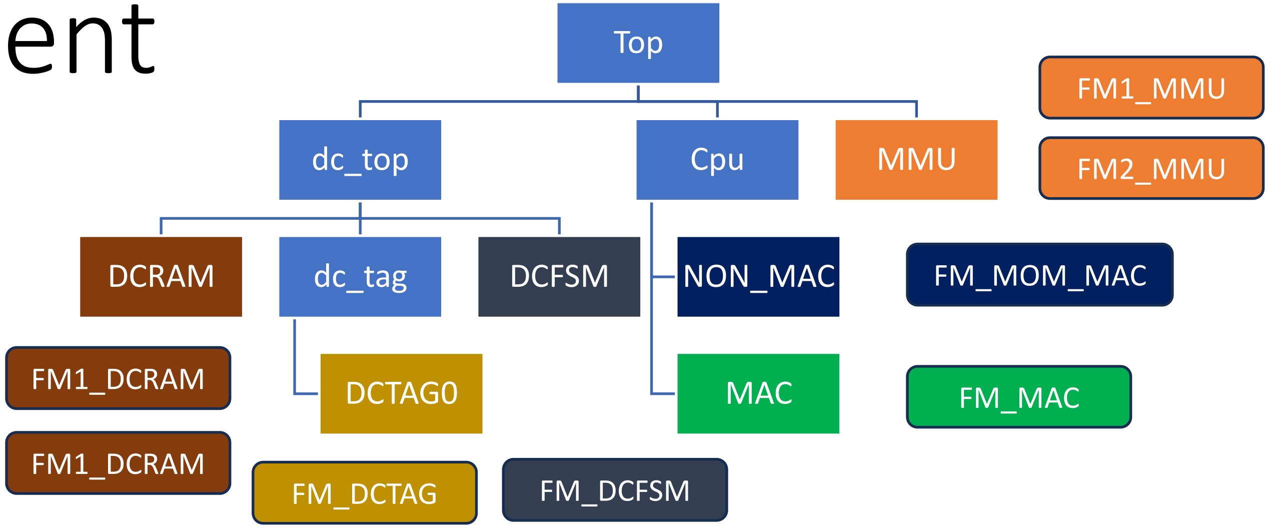


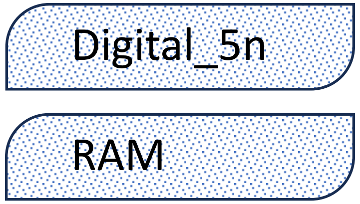
Create_fm FM1_DCRAM –type Mission –parent TOP.DCRAM –fmeda MY_FMEDA
Create_fm FM2_DCRAM –type Mission –parent TOP.DCRAM –fmeda MY_FMEDA
Create_fm FM_DCTAG –type Mission –parent TOP.DCTAG0 –fmeda MY_FMEDA
Create_fm FM_DCFSM –type Mission –parent TOP.DCFSM –fmeda MY_FMEDA
Create_fm FM1_MMU –type Mission –parent TOP.MMU –fmeda MY_FMEDA
Create_fm FM2_MMU –type Mission –parent TOP.MMU –fmeda MY_FMEDA
Create_fm FM_NON_MAC –type Mission –parent TOP.NON_MAC –fmeda MY_FMEDA
Create_fm FM_MAC –type Mission –parent TOP.MAC –fmeda MY_FMEDA

FS Analysis Hierarchy + FM Hierarchy

Part	Subpart	Failure Mode
TOP	MAC	FM_MAC
	NON_MAC	FM_NON_MAC
	MMU	FM1_MMU
		FM2_MMU
	DCFSM	FM_DCFSM
	DCTAG0	FM_DCTAG
	DCRAM	FM1_DCRAM
		FM2_DCRAM

Example: TE and assignment





Create_TE Digital_5n –type Digital –fr 1e-6
Create_TE RAM –type RAM –fr 1e-5

Part	Subpart	Failure Mode	Technology	FM_size
TOP	MAC	FM_MAC	Digital_5n	10
	NON_ MAC	FM_NON_MAC	Digital_5n	15
	MMU	FM1_MMU	RAM	35
		FM2_MMU	Digital_5n, RAM	5, 25
	DCFSM	FM_DCFSM	Digital_5n	…
	DCTAG0	FM_DCTAG	Digital_5n	…
	DCRAM	FM1_DCRAM	RAM	…
		FM2_DCRAM	RAM	…

Assign_TE_fm -te_name Digital_5n -fm_name FM_MAC -parent TOP.*MAC* -fmeda *MY_FMEDA* -fm_size 10

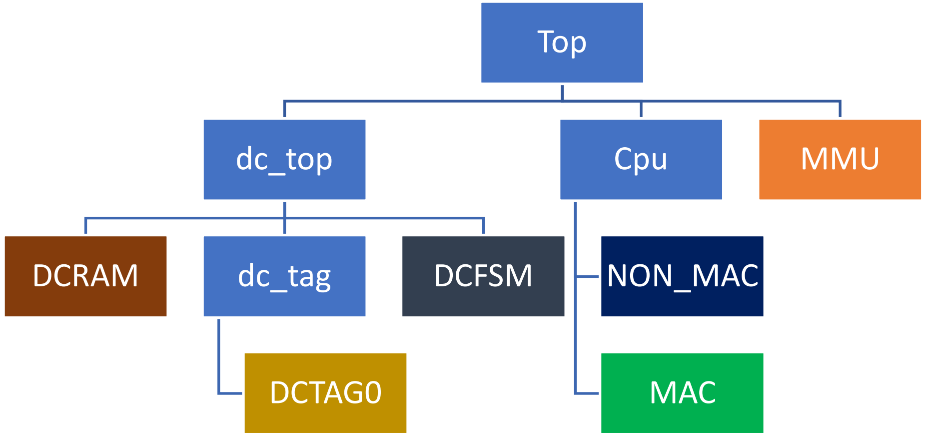
Assign_TE_fm -te_name Digital_5n -fm_name FM_NON_MAC -parent TOP.NON_*MAC* -fmeda *MY_FMEDA* -fm_size 15

Assign_TE_fm -te_name RAM -fm_name FM1_MMU -parent TOP.MMU -fmeda *MY_FMEDA* -fm_size 35

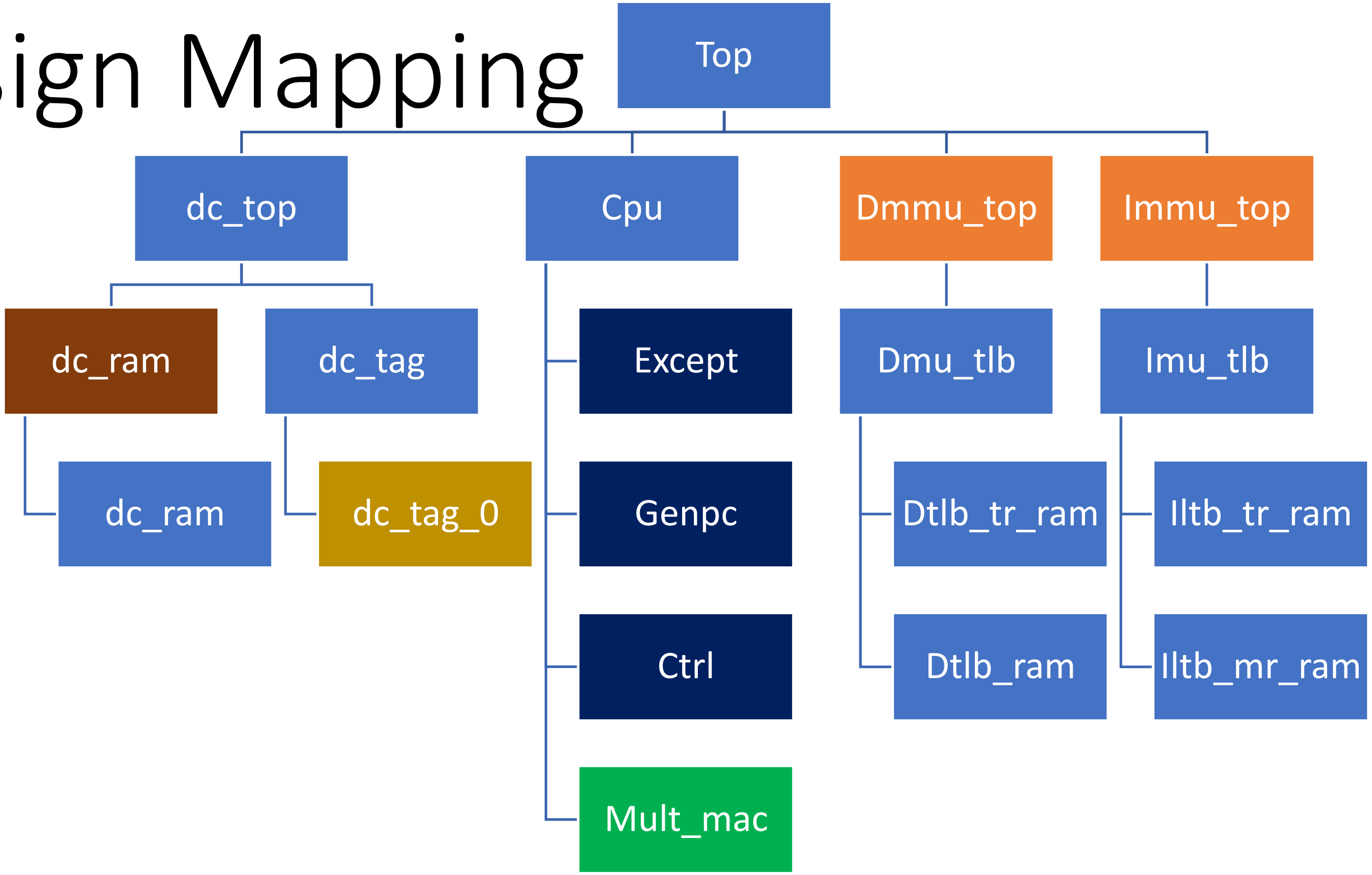
Assign_TE_fm -te_name Digital_5n -fm_name FM2_MMU -parent TOP.MMU -fmeda *MY_FMEDA* -fm_size 5
Assign_TE_fm -te_name RAM -fm_name FM2_MMU -parent TOP.MMU -fmeda *MY_FMEDA* -fm_size 25

….

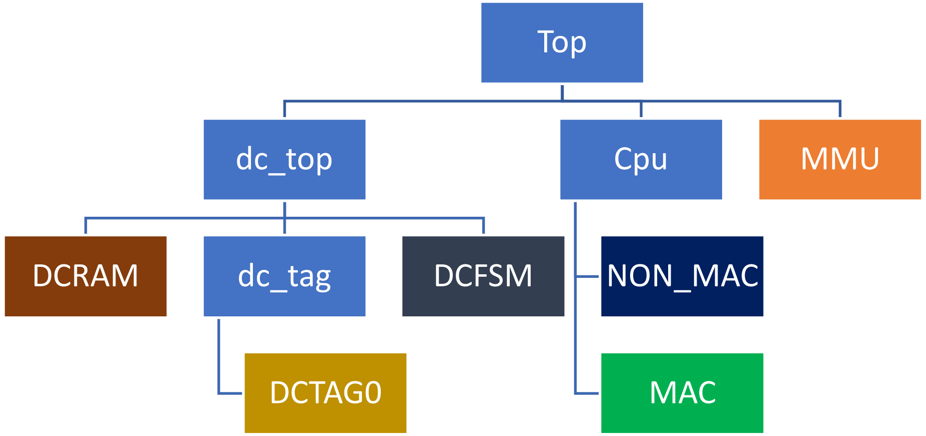
Example: Subpart Design Mapping

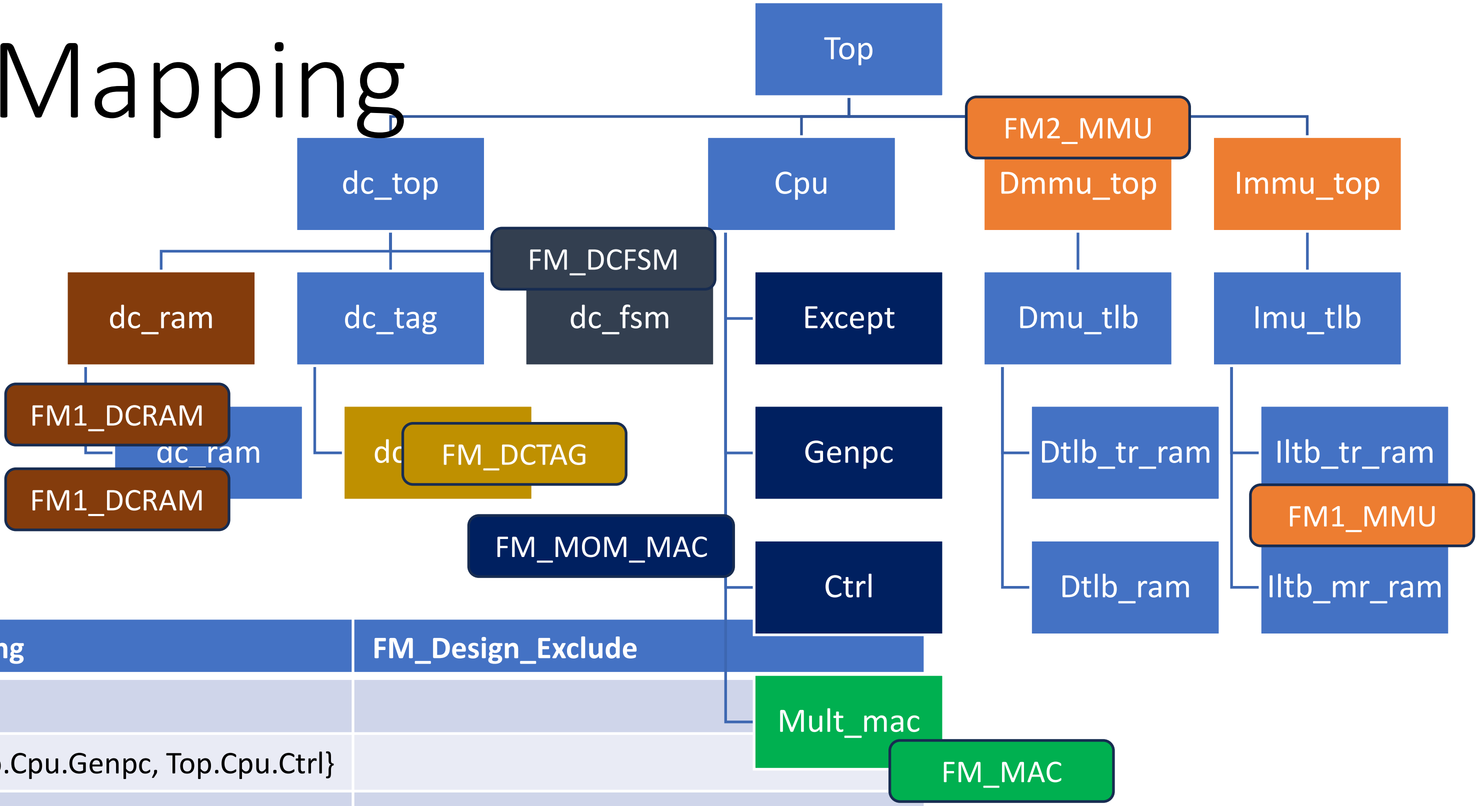


Part	Subpart	Failure Mode	Technology
TOP	MAC	FM_MAC	Digital_5n
	NON_MAC	FM_NON_MAC	Digital_5n
	MMU	FM1_MMU	RAM
		FM2_MMU	Digital_5n, RAM
	DCFSM	FM_DCFSM	Digital_5n
	DCTAG0	FM_DCTAG	Digital_5n
	DCRAM	FM1_DCRAM	RAM
		FM2_DCRAM	RAM



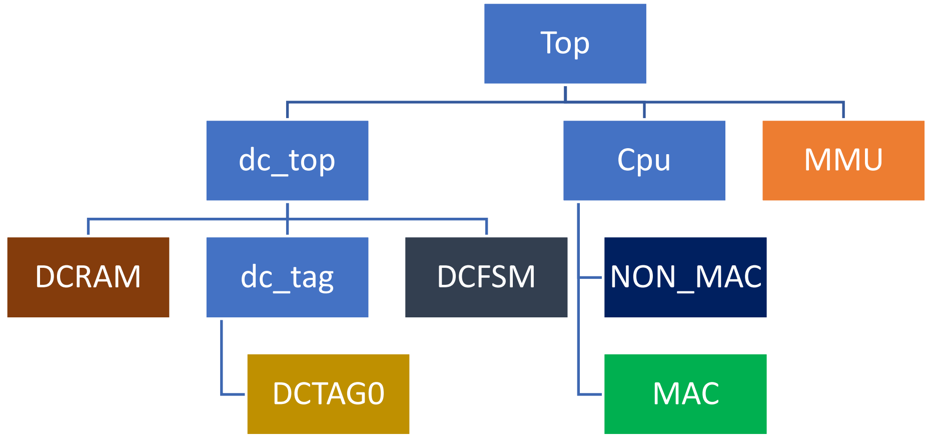
Example: FM Design Mapping

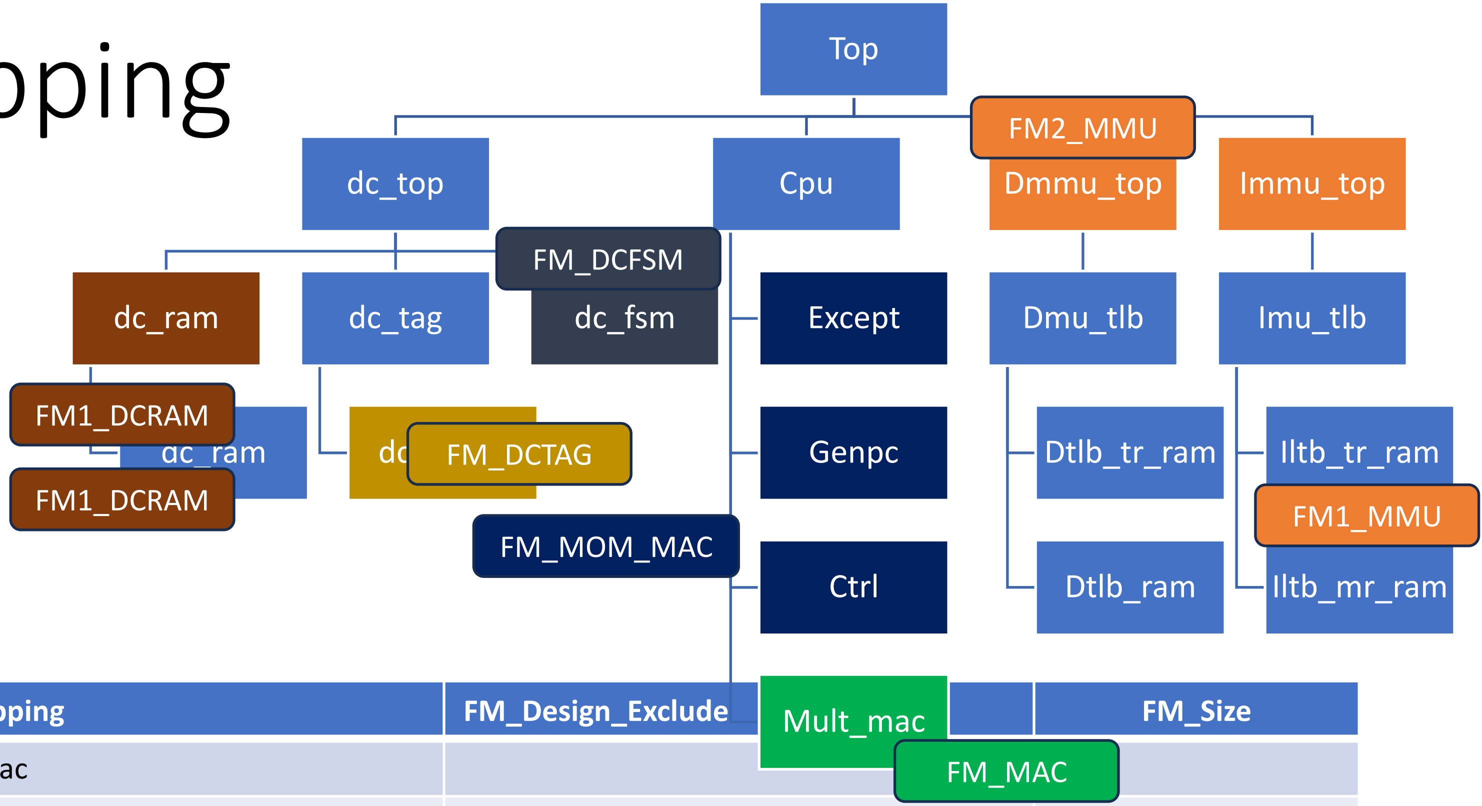




Part	Subpart	Failure Mode	Technology	FM_Design_Mapping	FM_Design_Exclude
TOP	MAC	FM_MAC	Digital_5n	Top.Cpu.Mult_mac	
	NON_MAC	FM_NON_MAC	Digital_5n	Top.Cpu.Except, Top.Cpu.Genpc, Top.Cpu.Ctrl}	
	MMU	FM1_MMU	RAM	{Top.Immu_top.Imu_tlb.Iltb_mr_ram, Top.Immu_top.Imu_tlb.Iltb_tr_ram}	
		FM2_MMU	Digital_5n, RAM	{Top.Dmmu_top, Top.Immu_top}	{Top.Immu_top.Imu_tlb.Iltb_mr_ram, Top.Immu_top.Imu_tlb.Iltb_tr_ram}
	DCFSM	FM_DCFSM	Digital_5n	Top.dc_top.dc_fsm	
	DCTAG0	FM_DCTAG	Digital_5n	Top.dc_top.dc_tag.DCTAG0	
	DCRAM	FM1_DCRAM	RAM	Top.dc_top.dc_ram.dc_ram	
		FM2_DCRAM	RAM	Top.dc_top.dc_ram.dc_ram	

Example: Design Mapping





Part	Subpart	Failure Mode	Technology	FM_Design_Mapping	FM_Design_Exclude	FM_Size
TOP	MAC	FM_MAC	Digital_5n	Top.Cpu.Mult_mac		
	NON_MAC	FM_NON_MAC	Digital_5n	Top.Cpu.Except, Top.Cpu.Genpc, Top.Cpu.Ctrl}		15
	MMU	FM1_MMU	RAM	{Top.Immu_top.Imu_tlb.Iltb_mr_ram, Top.Immu_top.Imu_tlb.Iltb_tr_ram}		35
		FM2_MMU	Digital_5n, RAM	{Top.Dmmu_top, Top.Immu_top}	{Top.Immu_top.Imu_tlb.Iltb_mr_ram, Top.Immu_top.Imu_tlb.Iltb_tr_ram}	5, 25
	DCFSM	FM_DCFSM	Digital_5n	Top.dc_top.dc_fsm		…
	DCTAG0	FM_DCTAG	Digital_5n	Top.dc_top.dc_tag.DCTAG0		…
	DCRAM	FM1_DCRAM	RAM	Top.dc_top.dc_ram.dc_ram		…
		FM2_DCRAM	RAM	Top.dc_top.dc_ram.dc_ram		…

```
Create_fmeda MY_FMEDA –type assumption_based

Create_element –type part TOP
Create_element –type subpart MAC –parent TOP -fmeda MY_FMEDA
Create_element –type subpart NON_MAC –parent TOP -fmeda MY_FMEDA
Create_element –type subpart MMU –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCFSM –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCTAG0 –parent TOP -fmeda MY_FMEDA
Create_element –type subpart DCRAM –parent TOP -fmeda MY_FMEDA

Create_fm FM1_DCRAM –type Mission –parent TOP.DCRAM –fmeda MY_FMEDA
Create_fm FM2_DCRAM –type Mission –parent TOP.DCRAM –fmeda MY_FMEDA
Create_fm FM_DCTAG –type Mission –parent TOP.DCTAG0 –fmeda MY_FMEDA
Create_fm FM_DCFSM –type Mission –parent TOP.DCFSM –fmeda MY_FMEDA
Create_fm FM1_MMU –type Mission –parent TOP.MMU –fmeda MY_FMEDA
Create_fm FM2_MMU –type Mission –parent TOP.MMU –fmeda MY_FMEDA
Create_fm FM_NON_MAC –type Mission –parent TOP.NON_MAC –fmeda MY_FMEDA
Create_fm FM_MAC –type Mission –parent TOP.MAC –fmeda MY_FMEDA

Create_TE Digital_5n –type Digital –fr 1e-6
Create_TE RAM –type RAM –fr 1e-5

Assign_TE_fm -te_name Digital_5n -fm_name FM_MAC -parent TOP.MAC -fmeda MY_FMEDA -fm_size 10 –FM_mapping {Top.Cpu.Mult_mac}
Assign_TE_fm -te_name Digital_5n -fm_name FM_NON_MAC -parent TOP.NON_MAC -fmeda MY_FMEDA -fm_size 15 –FM_mapping {Top.Cpu.Except, Top.Cpu.Genpc, Top.Cpu.Ctrl}
Assign_TE_fm -te_name RAM -fm_name FM1_MMU -parent TOP.MMU -fmeda MY_FMEDA -fm_size 35 –FM_mapping {Top.Immu_top.Imu_tlb.Iltb_mr_ram, Top.Immu_top.Imu_tlb.Iltb_tr_ram}
Assign_TE_fm -te_name Digital_5n -fm_name FM2_MMU -parent TOP.MMU -fmeda MY_FMEDA -fm_size 5 –FM_mapping {Top.Dmmu_top}
Assign_TE_fm -te_name RAM -fm_name FM2_MMU -parent TOP.MMU -fmeda MY_FMEDA -fm_size 25 –FM_mapping {Top.Immu_top} –FM_mapping_exclude
{Top.Immu_top.Imu_tlb.Iltb_mr_ram, Top.Immu_top.Imu_tlb.Iltb_tr_ram}

….
```

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

What’s Next

accellera SYSTEMS INITIATIVE

What’s next

- Data Model White Paper: coming out shortly. Stay tuned!
- LRM/User Guide + Validation
  - Content
    - Intent: data model content // how is info stored/exchange
    - API: interaction with data model // how to interact with info
  - Language
    - Formal or pseudo (even specific implementations e.g. Tcl)
    - Single or different for use cases (authoring and exchange)
    - Usage of default value
- Baseline and extensions:
  - Version 0.1: FMEDA, semiconductors
  - Post version 0.1: language feature, hierarchical support
  - Version 0.2: Safety Goals, Extension for verification support, FMEA?, system-level?

Annex C: Add-on to v0.1

This chapter describes commands that were considered by the working group, but no decision was agreed on whether accept or decline them. This chapter is for informative purposes only.

The full list of commands defined according to this extension is as follows:

- load_slf
- save_slf
- set_scope
- add_parameter
- attr_expr
- assign_fmeda_fmeda
- assign_fmeda_element

Language extensions

Hierarchical/SoC

Point of discussion: data model vs integration/compression of FMEDA

Thank you

More information on the Functional Safety WG:
https://www.accellera.org/activities/working-groups/functional-safety

White paper: https://www.accellera.org/images/downloads/standards/functional-safety/Functional_Safety_White_Paper_051020.pdf

Data model white paper: Coming soon!!!