Date: January 2021

RAAML

OMG RISK ANALYSIS AND ASSESSMENT MODELING LANGUAGE

Risk Analysis and Assessment Modeling Language (RAAML) Libraries and Profiles

Version 1.0 – beta 1

OMG Document Number: ptc/2021-01-01

Standard document URL: http://www.omg.org/spec/RAAML/

Normative Machine Consumable File(s):

https://www.omg.org/spec/RAAML/20201101/CoreRAAML.xmi
https://www.omg.org/spec/RAAML/20201101/CoreRAAMLLib.xmi
https://www.omg.org/spec/RAAML/20201101/GeneralRAAML.xmi
https://www.omg.org/spec/RAAML/20201101/GeneralRAAMLLib.xmi
https://www.omg.org/spec/RAAML/20201101/GSN.xmi
https://www.omg.org/spec/RAAML/20201101/FMEA.xmi
https://www.omg.org/spec/RAAML/20201101/FMEALib.xmi
https://www.omg.org/spec/RAAML/20201101/FTA.xmi
https://www.omg.org/spec/RAAML/20201101/FTALib.xmi
https://www.omg.org/spec/RAAML/20201101/ISO26262.xmi
https://www.omg.org/spec/RAAML/20201101/ISO26262Lib.xmi
https://www.omg.org/spec/RAAML/20201101/STPA.xmi
https://www.omg.org/spec/RAAML/20201101/STPALib.xmi

This OMG document replaces the submission document (ad/2020-12-02). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org by May 01,2021.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in December 2021. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

Table of Contents

Table of Figures

TABLE OF TABLES

Preface

About the Object Management Group

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.
OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Meta-model); and industry-specific standards for dozens of vertical markets.
More information on the OMG is available at *https://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL: *https://www.omg.org/spec*
All of OMG‟s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:
OMG Headquarters
9C Medway Road, PMB 274
Milford, MA 01757
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*
Certain OMG specifications are also available as ISO/IEC standards. Please consult: http://www.iso.org

Issues

The reader is encouraged to report and technical or editing issues/problems with this specification to:
https://www.omg.org/report_issue.htm

1. Scope

1.1 Introduction

There are two parts to this specification, one being normative and another informative. The normative part is:

- The Risk Analysis and Assessment Modeling Language (RAAML) Library and Profile (this document) defines concepts and relationships for capturing safety and reliability aspects of a system in the library and profile form.

The informative part is:

- The RAAML Example Model, Annex A (see document ad/2020-11-01), which illustrates practical usages of RAAML.

1.2 RAAML Background

Model-Based Systems Engineering (MBSE) is gaining popularity in organizations creating complex systems where it is crucial to collaborate in a multi-disciplinary environment. SysML, being one of the key MBSE components, has a good foundation for capturing requirements, architecture, constraints, views and viewpoints. However, SysML does not provide the constructs to capture safety and reliability information in the system model. A group of industry experts at the OMG has been working since 2016 to define a new specification providing the necessary capabilities.

The need for a standardized UML profile/library for addressing safety and reliability aspects emerged long ago. Working group members have seen multiple commercial-grade model-based safety and reliability solution implementations being developed during the recent years and successfully used in practice. While the various safety and reliability implementations may fit the needs for a specific purpose, there are many instances where information needs to be traced and shared across multiple organizations. These inconsistent model-based solutions prohibit direct model sharing between organizations and across the various tools. One of the key goals for the working group is to reconcile these different approaches to alleviate the industry from repeatedly formulating safety and reliability constructs in their tools. The specification provides the modeling capabilities for tool vendors to build safety and reliability modeling tools that provide traditional representations (e.g. trees, tables, etc.) while using a modern model-based approach.

This RAAML 1.0 specification defines extensions to SysML needed to support safety and reliability analysis. It describes:

- the core concepts and shows how the simple concepts are powerful enough to unite all safety and reliability information across a variety of analysis methods
- the approach to automating several safety and reliability analyses, which is built on leveraging existing SysML functionalities to ensure that the profile and library is usable with existing tooling
- specific safety and reliability analysis methods and application domains that are supported
  - Failure Mode and Effect Analysis (FMEA)
  - Fault Tree Analysis (FTA)
  - Systems Theoretic Process Analysis (STPA)
  - Goal Structuring Notation (GSN)
  - ISO 26262 Road Vehicles - Functional Safety
- extension mechanisms that are typically needed by the industry to apply the specification in practice

1.3 Intended Usage

The RAAML specification provides the foundation for conducting various safety and quality engineering activities including safety and reliability analysis methods. Besides the method support, linkages to the SysML model-of-interest are provided, enabling integration with and traceability to the analyses. The specification can be used for modeling safety and reliability aspects directly in the model or as a standard language to import and export from external safety and reliability tools.

The organization of RAAML facilitates tailoring the methodologies to specific engineering domains and industries to support the various assessment and certification agencies.

1.4 Related Documents

The specification is delivered as a set of related documents. The primary normative document is this document, while a set of additional machine-readable documents is provided to specify the UML profiles and model libraries, specified by this standard.

For each safety/reliability domain, supported by this standard (FMEA, FTA, ISO-26262 and STPA) there is a pair of profile and library.

In addition to that there is a pair of profile and library for the concepts used in multiple domains – General; and a pair of profile and library for the very core concepts that might be useful for the implementers of other standards in the safety/reliability domain.

GSN stands separately, as it is an add-on, which can be used with any of the aforementioned domains for additional substantiation of the safety models. It consists of just the profile; no library is necessary.

Non-normative examples document is also provided, illustrating how to apply RAAML for capturing safety and reliability data.

Table 1.1 – Table of Related Documents

Document Number	Description	File Name	Nor-mative	Machine Readable
ad/2020-07-06	Core portion of the RAAML.	CoreRAAML.xmi	Y	Y
ad/2020-07-07	Library portion of the RAAML.	CoreRAAMLLib.xmi	Y	Y
ad/2020-07-08	General portion, shared across domains of the RAAML.	GeneralRAAML.xmi	Y	Y
ad/2020-07-09	General Library portion, shared across domains of the RAAML.	GeneralRAAMLLib.xmi	Y	Y
ad/2020-07-10	Goal Structuring Notation profile.	GSN.xmi	Y	Y
ad/2020-07-11	FMEA portion of the RAAML.	FMEA.xmi	Y	Y
ad/2020-07-12	FMEA Library portion of the RAAML.	FMEALib.xmi	Y	Y
ad/2020-07-13	FTA (Fault Tree Analysis) portion of the RAAML.	FTA.xmi	Y	Y
ad/2020-07-14	FTA (Fault Tree Analysis) Library portion of the RAAML.	FTALib.xmi	Y	Y
ad/2020-07-15	ISO26262 Functional Safety Standard portion of the RAAML	ISO26262.xmi	Y	Y
ad/2020-07-16	ISO26262 Functional Safety Standard Library portion of the RAAML	ISO26262Lib.xmi	Y	Y
ad/2020-07-17	STPA (Systems Theoretic Process Analysis) portion of the RAAML	STPA.xmi	Y	Y
ad/2020-07-18	STPA (Systems Theoretic Process Analysis) Library portion of the RAAML	STPALib.xmi	Y	Y
ad/2020-11-01	Risk Analysis and Assessment Modeling Language 1.0 Examples	OMG RAAML Examples 1.0.docx	N	N
ad/2020-12-06	Original Cameo Systems Modeler file to produce XMI files for RAAML	RAAML Libraries and Profiles.mdzip	N	Y

2. Conformance

RAAML specifies two types of conformance.

- Type 1 Conformance: RAAML model interchange conformance. A tool demonstrating model interchange conformance can import and export conformant XMI for all valid RAAML models.
- Type 2 Conformance: RAAML View specification conformance. A tool demonstrating view specification conformance shall implement the views specified in RAAML specification.

A tool vendor may choose to implement one method supported by the specification (FMEA, FTA, STPA, GSN or ISO 26262) and claim conformance to it.

3. References

3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

3.2 OMG Documents (Normative References)

- Unified Modeling Language (UML), 2.5.1, December 2017, http://www.omg.org/spec/UML
- Object Constraint Language (OCL), 2.4, February 2014, http://www.omg.org/spec/OCL
- System Modeling Language (SysML) ,1.6, December 2019, http://www.omg.org/spec/SysML
- XMI Metadata Interchange (XMI), 2.5.1, June 2015, https://www.omg.org/spec/XMI

3.3 Other Normative References

- IEC 60812 for FMEA, https://webstore.iec.ch/publication/26359 [accessed on October 28, 2020]
- IEC 61025 for FTA, https://webstore.iec.ch/publication/4311 [accessed on October 28, 2020]
- IEC 61508:2010 for Functional safety of electrical/electronic/programmable electronic safety-related systems, https://webstore.iec.ch/publication/22273 [accessed on October 28, 2020]
- International Standardization Organization. ISO PAS 21448:2019(en) Road vehicles – Safety of the intended functionality, https://www.iso.org/standard/70939.html [accessed on October 28, 2020]
- International Standardization Organization. ISO 26262-1:2011(en) Road vehicles Functional safety - Part 1, Part 3. https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-1:v1:en [accessed on October 28, 2020]
- N. Leveson and J. Thomas, STPA Handbook, Boston, MA: MIT, March 2018, https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf [accessed on October 28, 2020]
- GSN specification, http://www.goalstructuringnotation.info [accessed on October 28, 2020]

3.4 Informative References

- ISO/IEC 15288:2015, Systems Engineering - Systems Life Cycle Processes, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=63711 [accessed on October 28, 2020]
- International Council On Systems Engineering (INCOSE), Systems Engineering Handbook V4, 2015, https://www.incose.org/products-and-publications/se-handbook [accessed on October 28, 2020]

4. Acknowledgements

The following companies and organizations submitted or supported parts of the original version of this standard:

Industry

- Dassault Systemes
- Ford Motor Company
- The Aerospace Corporation

Government

- NASA/Jet Propulsion Laboratory
- Commissariat à l'énergie atomique
- German Aerospace Center
- National Institute of Advanced Industrial Science and Technology (AIST)

Vendors

- No Magic owned by Dassault Systemes

Academia

- Massachusetts Institute of Technology

Liaisons

- Gesellschaft für Systems Engineering
- MACE
- Assystem

The following persons were members of the team that designed and wrote this International Standard: Achim Weiss, Andreas Knapp, Andrius Armonas, Annelisa Sturgeon, Axel Berres, Christian Lalitsch-Schneider, Christoph Barchanski, Christopher Davey, Damun Mollahassani, Dave Banham, Edith Holland, Geoffrey Biggs, George Walley, Ilse Adamek, Jean-Francois Castet, Jianlin Shi, John Thomas, Kyle Post, Laura Hart, Manfred Koethe, Mark Sampson, Matthias Nagorni, Myron Hecht, Nataliya Yakymets, Rajiv Murali, Regis Casteran, Sarra Yako, Stephan Boutenko, Thomas Krynicki, Tim Weilkiens, Tomas Juknevicius, Vanessa Sehon, Victor Arcos Barraquero, Yan Liu.

For the final edition of the standard, the following people contributed: Andrius Armonas, Axel Berres, Dave Banham, Kyle Post, George Walley, Tomas Juknevicius.

5. Terms and Definitions

New terms and definitions have been required to create this specification. They are listed in the table below.

Table 5.1 – Description of terms and definitions used in this specification

Situation	A situation describes a set of situation occurrences of some type. The system, place, time and state parameters are described by classifiers rather than individual descriptions. A situation occurrence is a system being in a given place at given time and in a given state. For example, "Boeing 747 with S/N 12305 is being refueled at Gate 7 of Amsterdam Schiphol at 11:45 on Monday, 30th of July 2018."
Causality	Identifies cause-effect relationship between two situations. Causality could be direct (non-conditional), conditional, probabilistic or any other inter-situation relationship, defined by the user. Multiple situations can cause one situation and vice versa - one situation can cause multiple other situations. For example, a car in frequent contact with salt, causing safety-critical parts to corrode, which causes leaks in the brake line, causing the brakes to fail, causing a car accident, causing a passenger injury.
Relevant To	The Relevant To relationship is used to link situations to system model elements to provide context and relevance for the Situation. For example, in an insulin pump, a Situation where the insulin pump cannot be charged would be related to the main battery element in the system model.
Controlling Measure	A measure taken to address (mitigate severity, reduce probability of occurrence, increase probability of detection) a potential or real adverse situation.

6. Acronyms and Abbreviations

For the purposes of this specification, the following List of acronyms and abbreviations apply.

Table 6.1 – Description of acronyms used in this specification

ASIL	Automotive Safety Integrity Level
DET	Detectability
FMEA	Failure Mode and Effect Analysis
FTA	Fault Tree Analysis
GSN	Goal Structuring Notation
HARA	Hazard Analysis and Risk Assessment
HAZOP	A hazard and operability study
MBSE	Model-Based Systems Engineering
ISO	International Standardization Organization
OCC	Occurrence
OMG	Object Management Group
RAAML	Risk Analysis and Assessment Modeling Language
RPN	Risk priority number
SEV	Severity
STPA	Systems Theoretic Process Analysis
SysML	Systems Modeling Language
UAF	Universal Architecture Framework
UML	Unified Modeling Language

7. Additional Information (non-normative)

7.1 Language Architecture

The RAAML specification reuses a subset of UML 2.5.1 and SysML 1.6 and provides additional extensions needed to address the Safety and Reliability for UML RFP (ad/2017-03-05) requirements. Those requirements form the basis for this specification. This document specifies the language architecture in terms of UML 2.5.1 and SysML 1.6. It explains the design principles and how they are applied to implement RAAML.

7.2 Philosophy

The RAAML working group uses a library approach heavily with a light UML profile support. Using model libraries has several significant benefits compared with implementing everything in a profile:

- It makes use of the full UML structural modeling capabilities instead of just using metamodeling, which are further limited by the UML prescriptions for stereotyping. The tools with good support for UML/SysML class and composite structure diagrams can make use of their existing generic functionality for modeling safety and reliability aspects of a system.
- It enables end users to extend the libraries and profiles provided by the specification because safety and reliability practices vary across domains (automotive, aerospace, nuclear, etc.) and organizations.
- Finally, it is typically easier to make modifications and extensions to model libraries than to profiles, as extensions occur at lower metalevels.

The RAAML development uses a model-driven approach. A simple description of the work process is:

- The specification is generated from the UML model used to describe RAAML. This approach allows the working group members to concentrate on architecture issues rather than documentation production. The UML tool automatically maintains consistency.

7.3 Principles of Creating, Editing, and Displaying of Composite Situations in Diagrammatic and Tabular Views

This standard uses UML/SysML structural modeling capabilities to capture safety and reliability data. The safety and reliability data are captured by a collection of scenarios and situations as shown in

Figure 7.1.

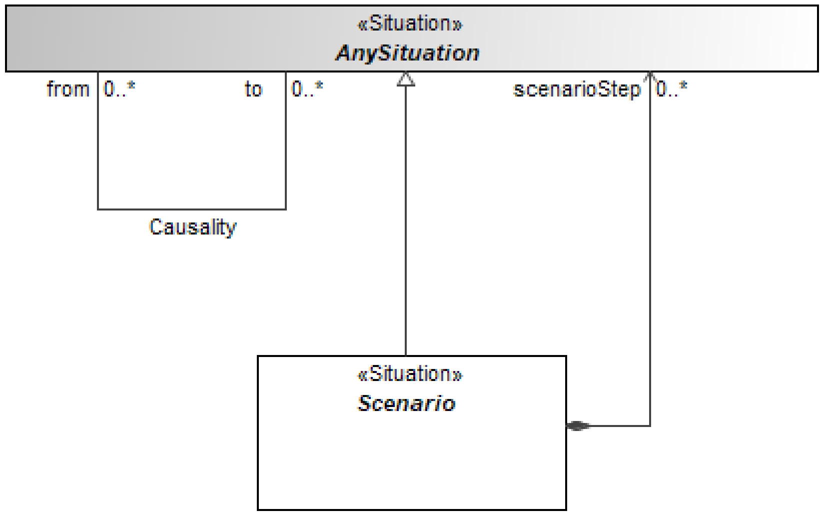


Figure 7.1 – Fundamental situation modeling principles

Complex scenarios can be built by inheriting from other scenarios and composing other situations as parts. Scenarios defined in libraries of this standard provide template scenarios from which to be inherited from. This way multilevel composite situations can be built.

- Situations are UML Classes, SysML Blocks.
- Scenario steps are captured using SysML parts - UML Properties with aggregation set to composite, and type set to sub-situation (which is UML Class, SysML Block); usually an association is also created for this property.
- Situation attribute values are captured using value properties - UML properties with type describing possible values (which is UML DataType, SysML ValueType) with the value specified in the defaultValue field.

When inheriting from library situations the properties of the user defined situations redefine or subset the properties of the library situation.

Note that user's model can have additional properties (including sub situations, and attributes and other kinds of properties), beyond those defined in the library. However, from the viewpoint of this standard, they carry user-specific extensions and are not relevant.

Situation in the user model can be inherited from the situation in the standard library indirectly through intermediate situations. This can be used to capture generality/specificity between the real-world situations being described and introduce user-specific library extensions.

Creation and Displaying of situation and scenario models can be done in diagrams, usual for UML/SysML tools, e.g., Class or Block Definition and Composite Structure or Internal Block diagrams. This suits rather well for the safety and reliability domains, which are used to graphical information input such as Fault Tree Analysis. However, users of many safety and reliability domains such as FMEA, STPA or ISO26262 are accustomed to tabular information input. Therefore, the principles of how these models can be described in a tabular format are explained in section §7.3.2.

7.3.1 Diagrammatic Situation Specification

Taking the operational situation TypicalAutomotiveSituation from ISO26262 library as an example, here is how the situation "Highway Driving Straight as Speed" would be defined in a diagram.

The ISO26262 library shown in (Figure 7.2) stipulates, that TypicalAutomotiveSituation is described by specifying trafficAndPeople, vehicleUsage, roadCondition, location, and environmentalCondition sub-situations and an Exposure attribute.

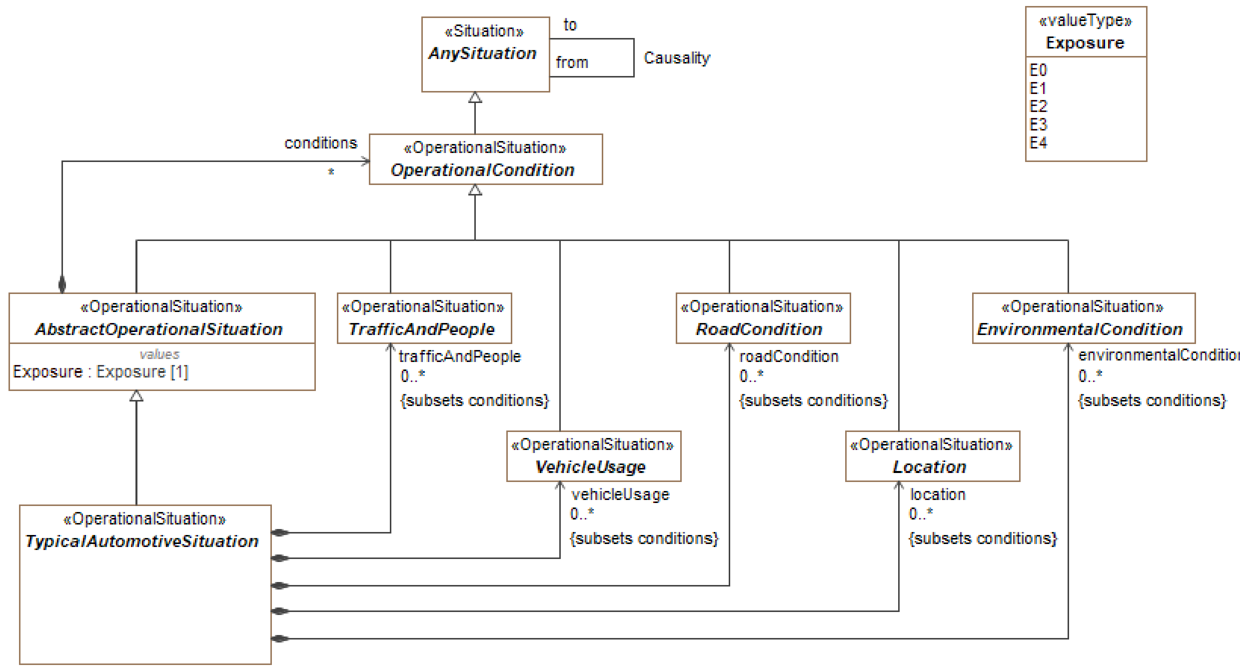


Figure 7.2 – Typical Automotive Situation definition in the ISO26262 Library

The "Highway Driving Straight at Speed" situation, in the user model (Figure 7.3) specifies, that Exposure level is E4 (chosen from the level enumeration defined in the library), trafficAndPeople is "Traffic Free Flow" (another situation defined by the user or coming from a library of operational conditions), the vehicleUsage is "Driving at Speed", location is "City Roads" and "Highway" (two values), while roadCondition and environmentalCondition are left unspecified.

Note that:

a) The scenario and sub-situations are inherited from the situations defined in the library.

b) Exposure, which is a value attribute (i.e., an attribute, whose type is not a situation, but some data type instead a numeric or enumerated value) is specified by redefining a library attribute and specifying a default value.

c) The trafficAndPeople and vehicleUsage attributes, which specify sub-situations, are redefining corresponding library attributes, and specifying a different type. The normal rules for UML attribute redefinition apply, i.e., redefined attribute type must be narrower that the parent attribute type.

d) The roadCondition and environmentalCondition are not redefined, therefore they are left unspecified. The attributes type remains the maximally wide, library type ("RoadCondition" and "EnvironmentalCondition" library types)

e) Two values are being specified for location attribute. Therefore, two attributes location1 and location2 are defined in the situation. These attributes are sub-setting the parent location attribute instead of redefining, as in case 3 above. Note that, according to UML rules, names of the sub-setting attributes are not regulated and therefore they can be anything. However, it is strongly recommended, that the tool vendor adopt some intuitive, user-friendly naming scheme like parent_attribute_name+number.

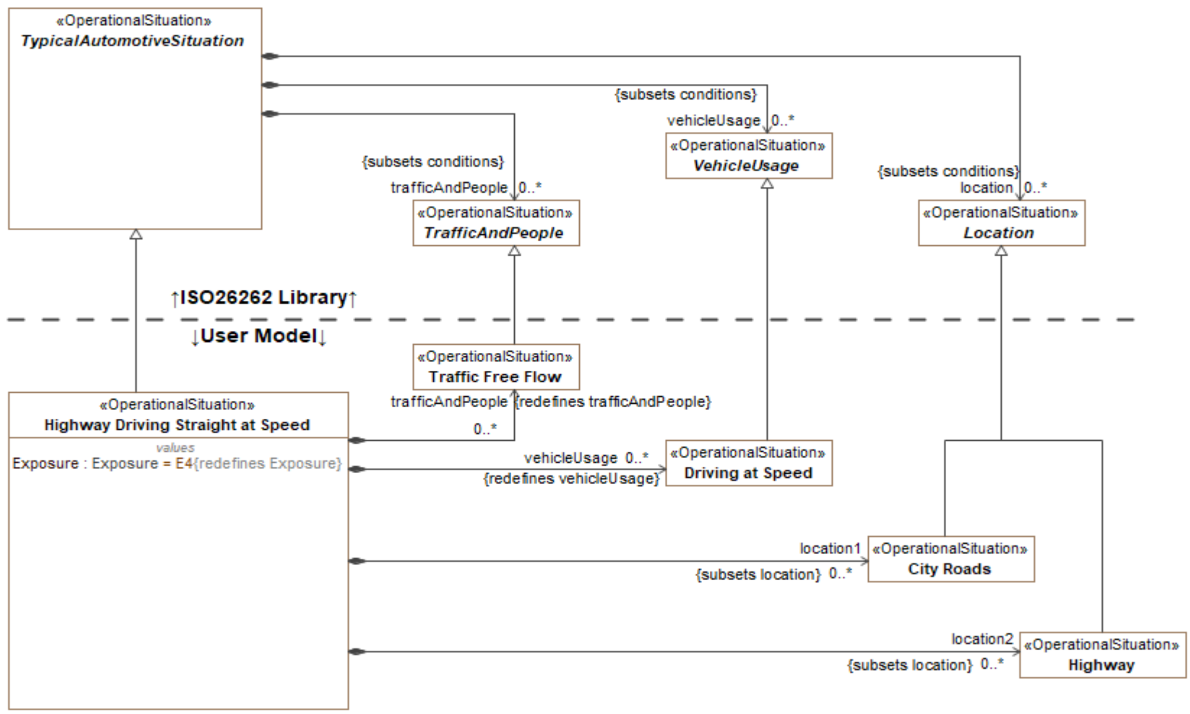


Figure 7.3 – User Model Defining Operational Situation "Highway Driving Straight at Speed"

7.3.2 Tabular Situation Specification

The same TypicalAutomotiveSituation, defined by the ISO26262 library and again shown in Figure 7.2, can also define a table format for entering automotive situation user model data in a tabular format.

The table for specifying typical automotive situations comprises the main Name column for defining the situation itself, plus one column per each attribute. A table for typical automotive situations, as defined by TypicalAutomotiveSituation library situation class would then have columns for Exposure, vehicleUsage, trafficAndPeople, location, roadCondition, and environmentalCondition. The column's name does not need to follow library attributes strictly. They can be beautified, for the sake of user-friendliness. It is important that when the user adds or edits rows in this table, the underlying model data must be created in accordance to the chapters above.

The table below (Table 7.1) shows the same "Highway Driving Straight as Speed" situation defined in tabular format as in the previous chapter. Therefore, the underlying UML model structures must be the same as those shown in diagrammatic format (

Figure 7.1).

Table 7.1 – Table for Specifying Operational Situations with Situation "Highway Driving Straight at Speed" Defined

#	Name	Exposure	Vehicle Usage	Traffic and People	Location	Road Condition	Environmental Condition
1	Highway Driving Straight at Speed	E4	Driving at Speed	Traffic Free Flow	Highway, City Roads		
1.1	Highway Driving Straight at Speed, Dangerous Conditions	E3	Driving at Speed	Traffic Free Flow	Highway, City Roads	Wet, Ice	Reduced Visibility

A typical safety and reliability domain such as ISO26262 will then use multiple tables, one for each of the structures defined in the library for that domain.

The tables can have additional columns, at the vendor's discretion, for specifying additional data about the situation, being described in a row. An example of such data could be a description (realized by e.g., UML Comment) of the situation.

Sub-classing by using a generalization relationship between situations can be expressed in tabular format, using hierarchical indented text in table row. In the above table, the "Highway Driving Straight at Speed, Dangerous Conditions" situation is a subclass of the "Highway Driving Straight at Speed" situation. Therefore, a generalization relationship is created between the two in the model. Note that the more specific situation can narrow down the field types of the parent. In this example, the sub-classing situation provides additional data for road and environmental conditions by using attributes and redefining attributes from the library. Using UML redefinition overrides the parent exposure to E3. The vehicle use, traffic and people, and location settings are inherited from the parent and do not require additional model elements.

In case of multiple composition levels between the situations defined the in the library, it is possible to show multi-level composite situation data in a single table instead of the multiple interrelated tables by using hierarchical grouped column approach.

An example of using this hierarchical approach is shown for the main situation - HazardousEvent - in the library for ISO26262 standard (Figure 7.4):

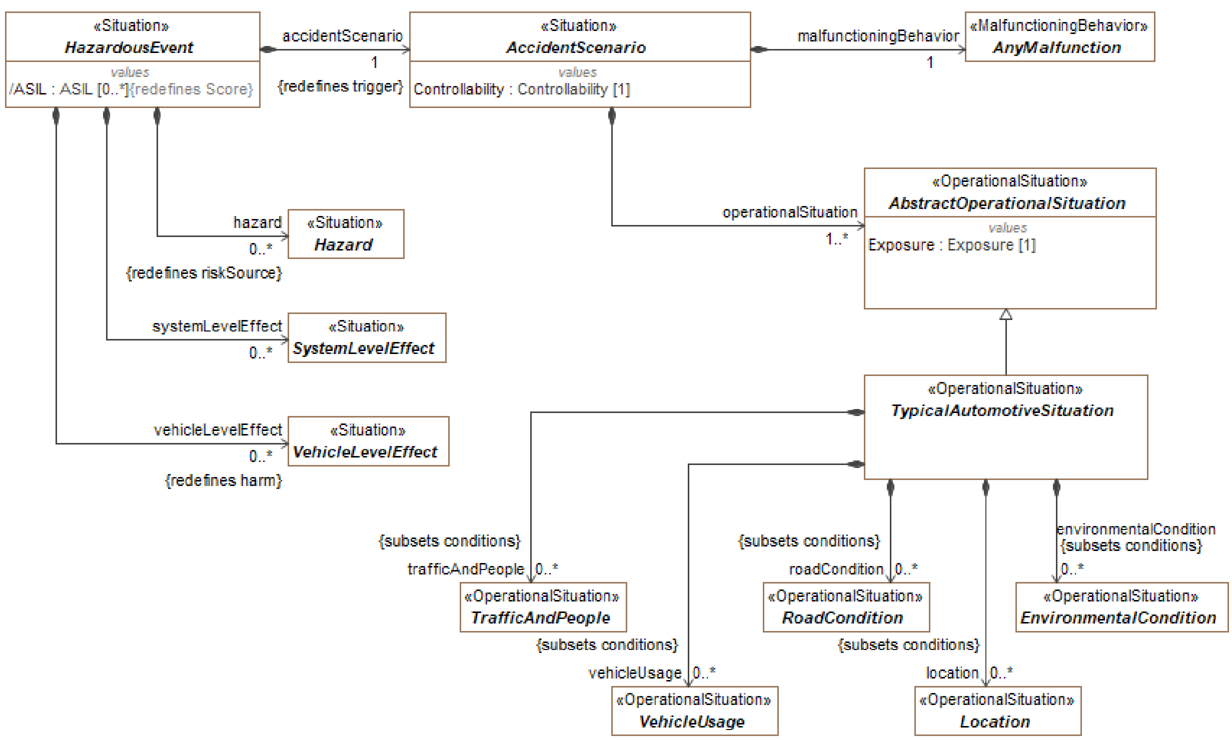


Figure 7.4 – HazardousEvent Definition in the ISO26262 Library

The HazardousEvent comprises sub-situations hazard, systemLevelEffect, vehicleLevelEffect which are elementary and an accidentScenario which is a composite sub-situation. AccidentScenario is composed of the elementary malfunctioningBehavior and operationalSituation. OperationalSituation is composed of a multitude of operational condition sub-situations vehicleUsage, trafficAndPeople, location, roadCondition, and environmentalCondition.

If tabular format is used for entering this information, there could be 3 simple tables:

1. Table for operational situations, having columns for **vehicleUsage**, **trafficAndPeople**, **location**, **roadCondition**, and **environmentalCondition**.

2. Table for accident scenarios, having columns for **malfunctioningBehavior** and **operationalSituation**.

3. Table for hazardous events, having columns for **hazard**, **systemLevelEffect**, **vehicleLevelEffect**, and **accidentScenario**.

Alternatively, all this data can be entered in a single table, as shown in Table 7.2:

1. Table for hazardous events, having columns for **hazard**, **systemLevelEffect**, **vehicleLevelEffect**, and an **accidentScenario**.

    1.1. Accident scenario is a column group, comprising of columns **malfunctioningBehavior** and **operationalSituation**.

        1.1.1. Operational situation is a column group comprising of columns **vehicleUsage**, **trafficAndPeople**, **location**, **roadCondition**, and **environmentalCondition.**

Table 7.2 – Hazardous Event Table with Grouped Columns

Name	Hazard	Accident Scenario								System Level Effect	Vehicle Level Effect
		Malfunctioning Behavior	Operational Situation						Controllability		
			Vehicle Usage	Traffic and People	Location	Road Condition	Environmental Condition	Exposure			

Note – some columns (like ASIL level, or names of accident scenario, operational situation) have been skipped in the table for compactness reasons; in the actual tool that is not limited by page width they would be present.

8. Diagram Legend (non-normative)

The section 9 is comprised of diagrams that represent elements from the RAAML 1.0 specification. The diagrams are color-coded to help the reader to understand the model easier. Please refer to the legend in Figure 8.1 to understand the diagrams.

Diagram shapes color-coded using gray color represent elements belonging to other packages than the one being specified in the current diagram

Diagram shapes color-coded using white color represent elements belonging to packages that are being specified in the current diagram

Figure 8.1 – Legend of color codes

An example in Figure 8.2 demonstrates how legends are used. Elements that belong to FTA (Fault Tree Analysis) library will be represented in white color in diagrams which belong to FTA method specification. Other elements like DysfunctionalEvent will be represented in gray since they belong to the General part of the specification.

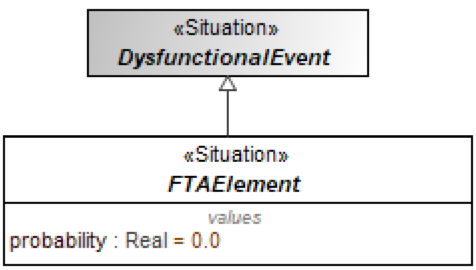


Figure 8.2 – An example of using a legend

9. Risk Analysis and Assessment Modeling Language (RAAML) Library and Profile

The RAAML library and profile imports the entire SysML profile. The use of this import is intended to provide more seamless integration with system modeling using SysML and to be able to fully leverage the capabilities of SysML.

9.1 Core

The core concepts domain model is depicted in Figure 9.1. The submission team uses this domain model to derive the CoreLibrary and CoreProfile packages (specified in sections 9.1.1 and 9.1.2 respectively). The other libraries and profiles of the specification are based on the CoreLibrary and CoreProfile packages, and contain elements and relationships representing concepts common across safety and reliability analysis methods.

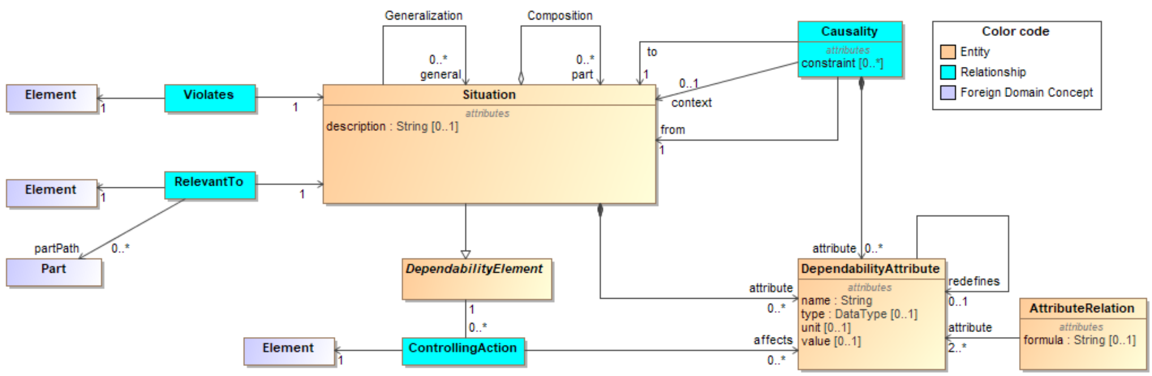


Figure 9.1 – Core concepts domain model

The central element in the core concepts domain model is the "Situation" concept. A situation occurrence is defined as a system being in a given place at given time and in a given state. For example, "Boeing 747 with S/N 12305 is being refueled at Gate 7 of Amsterdam Schiphol at 11:45 on Monday, 30th of July 2018." An elementary situation is a classifier. It describes a set of situation occurrences of some type. The system, place, time and state parameters are described by classifiers rather than individual descriptions.

When describing a situation, some of its parameters may be omitted if the situation does not need to be specific with respect to that parameter. For example:

- Fire in the engine compartment of the ship.
- Finger injury of the circular saw operator.

Different Situations can have generalization/specialization relationships between them. Generalization between two situations expresses the subset/superset relationship between the sets of occurrences that these situations represent. For example, "bone fracture" may be defined as a subtype of "Injury".

Situations can have quantitative attributes, such as probability of occurrence. These are defined using the DependabilityAttribute class. Quantitative attributes can be related to each other and to attributes of the system by formulae using the AttributeRelation class. Formulae can be expressed in any language that the modeling tool can compute, including OCL and other executable languages. For example:

FMEAItem.RiskPriorityNumber = Cause.Occurrence × FailureMode.Detectability × Effect.Severity

Different Situations can be associated with each other using the Causality class, expressing semantic relationships between situations such as simple causality, conditional causality, and probabilistic connections. These relations may also have quantitative attributes, such as the probability of occurrence of the "to" situation if the "from" situation occurs. For example, a car in frequent contact with salt, causing safety-critical parts to corrode, which causes leaks in the brake line, causing the brakes to fail, causing a car accident, causing a passenger injury.

A non-elementary situation (the "Composition" relationship in Figure 9.1) is a concept encompassing multiple elementary situations: a single system or combination of several systems in a mutable layout, flowing in time through a sequence of states. The choice of whether to use a composite situation with parts described by subsituations, or to use a single situation, is at the discretion of the modeler. It depends on the modeler's needs, such as the depth of analysis required.

Situations can violate requirements, constraints defined/prescribed for the system, or other specifications describing how the system should operate. For example, a Situation where the system can-not detect glucose level violates the requirement that “the insulin pump must work for 1 week without the need to replace batteries”.

The RelevantTo relationship is used to link situations to system model elements to provide context and relevance for the Sitution. For example, in the aforementioned insulin pump, a Situation where the insulin pump cannot be charged would be related to the main battery element in the system model.

Situations can be mitigated, detected, and prevented via the ControllingAction. The use of this relationship introduces new safety requirements.

It was decided early on to reuse as many concepts from the SysML language as possible and only add concepts that are missing in SysML to address safety and reliability aspects of systems. This avoids duplication between two languages that will typically be used together. It also enables tool vendors to implement the new profile and library without requiring new tool capabilities, assuming SysML is supported. This leads to a very small library and profile on top of SysML/UML being sufficient to cover all core concepts. The core domain model is covered by SysML/UML concepts as shown in Table 1. The CoreLibrary package is specified in section 9.1.1. The CoreProfile package is shown in 9.1.2. The Core profile and library are used by all domain-specific methods in the specification.

Table 9.1 – Mapping of core concepts to the SysML/UML language

Core concept	SysML/UML concept
Situation	A specialization of a Block in SysML and a new stereotype «Situation »
DependabilityAttribute	SysML Value Property
AttributeRelation	SysML Constraint Block
Generalization	UML Generalization relationship
Composition	UML Composition relationship
Violates	A stereotyped UML dependency
RelevantTo	A stereotyped UML dependency
Causality	An association/connector combination
ControllingAction	A stereotyped UML dependency

9.1.1 Core::Core Library

AnySituation

Package: Core Library
isAbstract: Yes
Applied Stereotype: «Situation»

Description

AnySituation is the universal root of all situations. All situations inherit from AnySituation. A situation describes a set of situation occurrences of some type. The system, place, time and state parameters are described by classifiers rather than individual descriptions. A situation occurrence is a system being in a given place at given time and in a given state.

For example, “Boeing 747 with S/N 12305 is being refueled at Gate 7 of Amsterdam Schiphol at 11:45 on Monday, 30th of July 2018.”

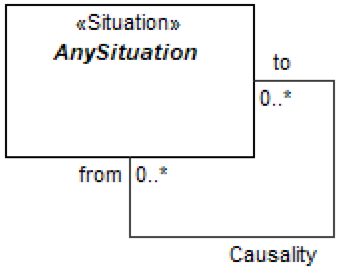


Figure 9.2 - AnySituation

Attributes

from : AnySituation[0..*] (member end of Causality association)	A situation which precedes the one at the other end of the Causality relationship.
to : AnySituation[0..*] (member end of Causality association)	A situation which follows the one at the other end of the Causality relationship.

Causality

Package: Core Library

Description

Universal root relationship between situations. All situation relationships inherit from this relationship. Identifies cause and effect relationship between two situations. Causality could be direct (non-conditional), conditional or probabilistic or any other inter-situation relationship, defined by the user. Multiple situations can cause one situation and vice versa - one situation can cause multiple other situations.

For example, a car in frequent contact with salt, causing safety-critical parts to corrode, which causes leaks in the brake line, causing the brakes to fail, causing a car accident, causing a passenger injury.

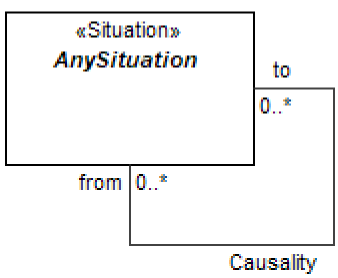


Figure 9.3 - Causality

Association ends

to : AnySituation[0..*] (member end of Causality association)	A situation which follows the one at the other end of the Causality relationship.
from : AnySituation[0..*] (member end of Causality association)	A situation which precedes the one at the other end of the Causality relationship.

9.1.2 Core::Core Profile

Situation

Package: Core Profile
isAbstract: No
Generalization: Block

Extension: Class

Description

A situation is a SysML v1.6 Block. The situation reuses the following functionality from the Block concept: generalizations, parts, value properties, and Parametrics. The situation stereotype is only needed to distinguish situations from other types of blocks. See AnySituation for the definition of a situation concept.

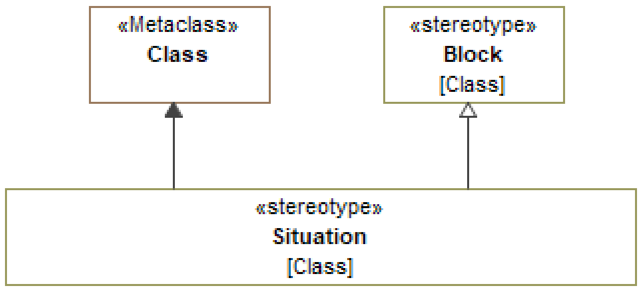


Figure 9.4 - Situation

RelevantTo

Package: Core Profile
isAbstract: No
Generalization: DirectedRelationshipPropertyPath
Extension: Dependency

Description

The RelevantTo relationship is used to link situations to system model elements to provide context and relevance for the Situation. For example, in an insulin pump, a Situation where the insulin pump cannot be charged would be related to the main battery element in the system model. The RelevantTo relationship reuses the following functionality from the DirectedRelationshipPropertyPath concept: targetContext and targetPropertyPath.

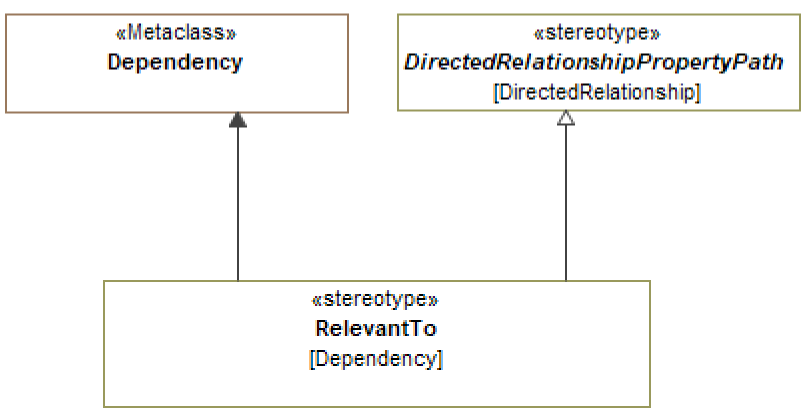


Figure 9.5 - RelevantTo

Constraints

[1] ClientIsSituation	-- client of the RelevantTo must be a Situation Situation.allInstances().base_Class->includesAll(self.base_Dependency.client)

ControllingMeasure

Package: Core Profile
isAbstract: Yes

Generalization: DirectedRelationshipPropertyPath
Extension: Dependency

Description

A measure taken to address (mitigate severity, reduce probability of occurrence, increase probability of detection) a potential or real adverse situation.

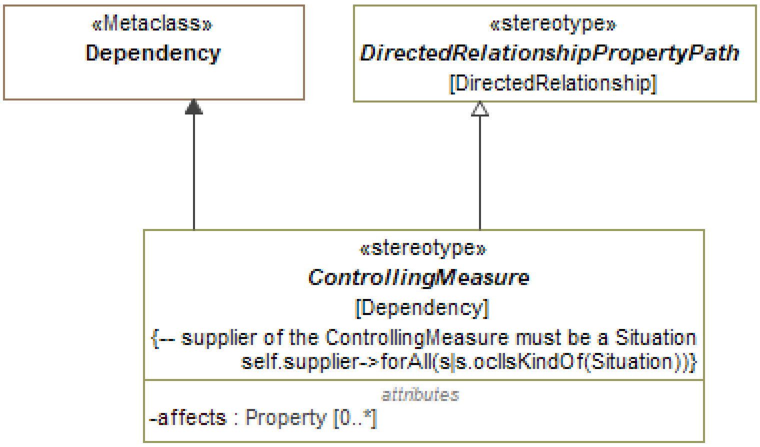


Figure 9.6 - ControllingMeasure

Attributes

affects : Property[0..*]	Indicates that this controlling measure influences (typically improves) a particular quantitative attribute of the situation.

Constraints

[1] SupplierIsSituation	-- supplier of the ControllingMeasure must be a Situation Situation.allInstances().base_Class->includesAll(self.base_Dependency.supplier)

Violates

Package: Core Profile
isAbstract: No
Extension: Dependency

Description

The violates relationship indicates a situation where a system is violating a prescription (requirement, constraint, etc.). It is used to connect situations to requirements, design constraints and any other elements of system models which prescribe a characteristic of the system.

For example, a Situation where the insulin pump drains the battery in 3 days violates the requirement that “The system must work for 1 week without the need to replace batteries”.

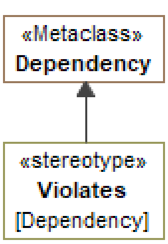


Figure 9.7 - Violates

Constraints

[1] ClientIsSituation	-- client of the Violates must be a Situation Situation.allInstances().base_Class->includesAll(self.base_Dependency.client)

9.2 General

The specification includes a general safety and reliability package that extends the core package. It defines common concepts that are used or extended in the method- and domain-specific reliability and safety packages. The package provides a model library, specified in section 9.2.1, and a profile, specified in section 9.2.2.

The general concepts contained in this package can be used as-is to model the safety and reliability related aspects of a system. However, the intended purposes of the package are as follows.

1. Provide a common base for the method- and domain-specific reliability and safety modeling packages. The same concepts are used in a number of safety and reliability techniques (such as FMEA and FTA), so the role of this package is to prevent duplication of common concepts in other packages. This also enables movement of information between domains for cross-domain issues. This is particularly important as different domains may use the same concepts with different vocabulary. A common foundation provides a way to translate between these.
2. Provide traceability links between safety and reliability artefacts across the system life cycle. For example, the failure modes defined during Hazard Analysis and Risk Assessment (HARA, defined in the ISO 26262 package) and in an FMEA could be traced and considered during an FTA.
3. Provide a foundation on which additional methods, techniques and domains with safety and reliability concerns not currently included in the profile can be built by users. For example, a tool vendor could build an additional package for the railway domain by building on the general safety and reliability foundation. This both reduces effort to introduce an additional domain and allows additional domain packages to be compatible with the existing specification content.

9.2.1 General::General Concepts Library

AbstractEvent

Package: General Concepts Library
isAbstract: Yes
Generalization: AnySituation
Applied Stereotype: «Situation»

Description

Anything that causes a change in a system under analysis or environment. Event has an identifiable starting point in time.

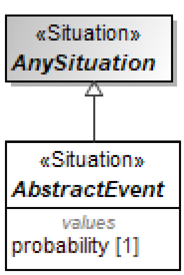


Figure 9.8 - AbstractEvent

Attributes

probability : [1]	A placeholder attribute for indicating probability of occurrence of an event. It is intentionally left without a type. Method developers can derive more specialized ways to characterize probability.

AbstractCause

Package: General Concepts Library
isAbstract: Yes
Generalization: AbstractEvent
Applied Stereotype: «Situation»

Description

An AbstractCause is a precursor event that activates other events. The AbstractCause is a root class for all kinds of causes; method developers should derive from it more specific kinds of causes with specific types for occurrence property. One case is demonstrated in the Cause element that redefines the occurrence property of the AbstractCause with the type Real.

See the diagram GeneralConceptsLibrary.

See also: fault association end of the Activation association.

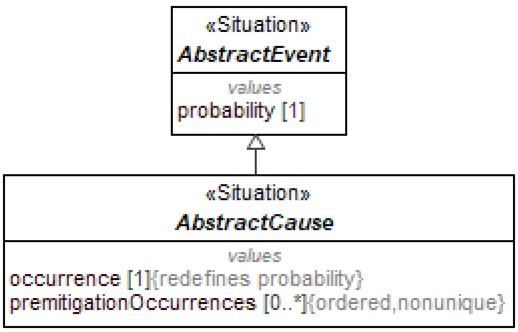


Figure 9.9 - AbstractCause

Attributes

occurrence : [1], redefines probability	A placeholder attribute without a type declared, for indicating how often this situation occurs. It is a redefinition of probability.
premitigationOccurrences : [0..*]	A placeholder attribute for indicating how often this situation occurred prior to mitigation. This property can have more than one value.

Cause

Package: General Concepts Library
isAbstract: Yes
Generalization: AbstractCause
Applied Stereotype: «Situation»

Description

A Cause is a specific implementation of AbstractCause that defines occurrence property with the type Real.

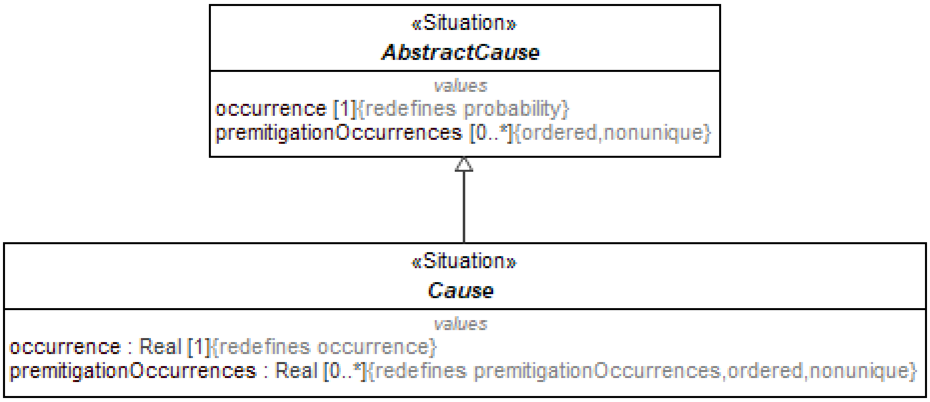


Figure 9.10 - Cause

Attributes

occurrence : Real[1], redefines occurrence	An attribute with the type Real, for indicating how often this situation occurs.
premitigationOccurrences : Real[0..*], redefines premitigationOccurrences	An attribute for indicating how often this situation occurred prior to mitigation. This property can have more than one value.

DysfunctionalEvent

Package: General Concepts Library
isAbstract: Yes
Generalization: AbstractEvent
Applied Stereotype: «Situation»

Description

An event whose occurrence can cause a dysfunctional behavior of a system or a part of the system.

The DysfunctionalEvent concept is a generalization of such concepts as failure, feared event, etc. that are considered in the domain-specific safety standards. It might be extended for introducing new safety and reliability methods and techniques.

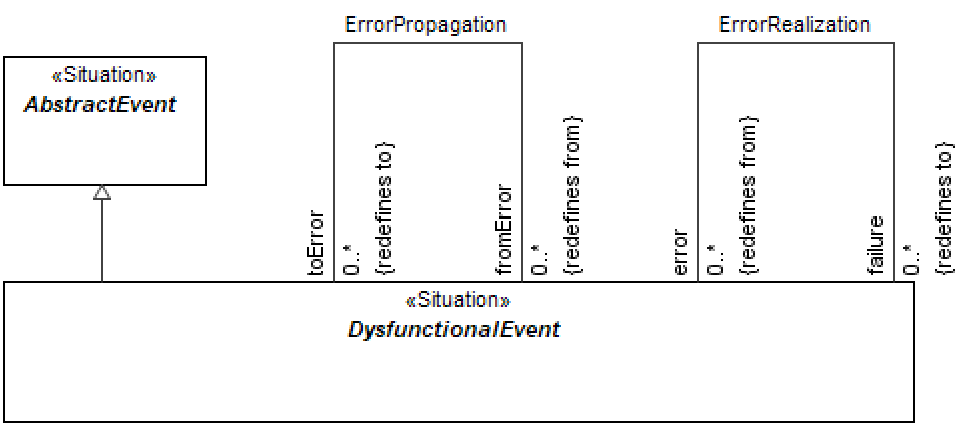


Figure 9.11 - DysfunctionalEvent

AbstractFailureMode

Package: General Concepts Library
isAbstract: Yes

Generalization: UndesiredState
Applied Stereotype: «FailureMode»

Description

The manner in which a system or part of a system (e.g. functions, components, hardware, software, hardware parts, software units), can fail (ISO 26262-1:2018, definition 3.51, modified).

The AbstractFailureMode is a root class for all failure modes; method developers should derive more specific kinds of failure modes with specific types for the detectability property. One case is demonstrated in the FailureMode element that redefines the detectability property of the AbstractFailureMode with the type Real.

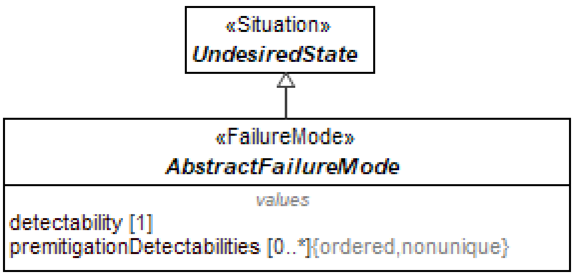


Figure 9.12 - AbstractFailureMode

Attributes

detectability : [1]	A placeholder attribute without a type declared, for indicating how easy it is to detect this failure mode.
premitigationDetectabilities : [0..*]	A placeholder attribute for indicating how easy it would have been to detect the situation with the previous design iteration. This property can have more than one value.

FailureMode

Package: General Concepts Library
isAbstract: Yes
Generalization: AbstractFailureMode
Applied Stereotype: «FailureMode»

Description

FailureMode is a specific implementation of AbstractFailureMode that defines the detectability property with the type Real.

A failure is an instance of a FailureMode.

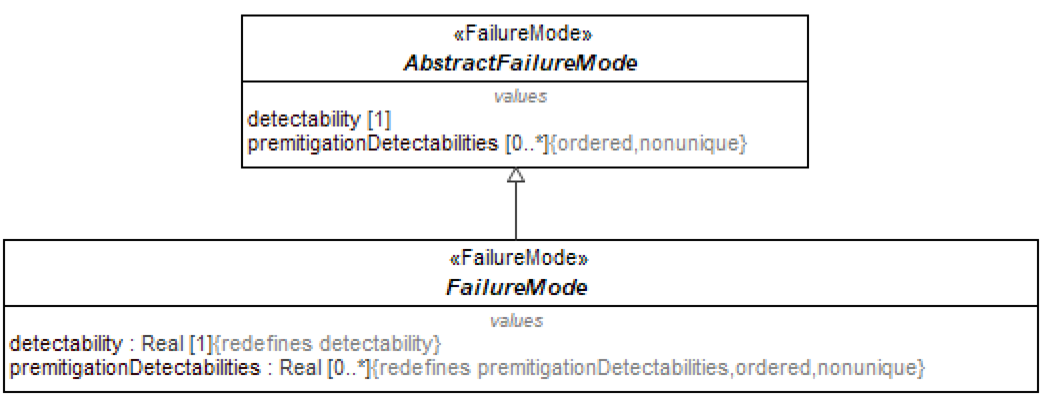


Figure 9.13 - FailureMode

Attributes

detectability : Real[1], redefines detectability	An attribute with the type Real, for indicating how easy it is to detect the situation.
premitigationDetectabilities : Real[0..*], redefines premitigationDetectabilities	An attribute for indicating how easy it would have been to detect the situation with the previous design iteration. This property can have more than one value.

AbstractEffect

Package: General Concepts Library
isAbstract: Yes
Generalization: DysfunctionalEvent
Applied Stereotype: «Situation»

Description

An AbstractEffect is a DysfunctionalEvent that is a result or a consequence of another Situation. The AbstractEffect is a root class for all effects; method developers should derive more specific kinds of effects with specific types for the severity property.

One case is demonstrated in the Effect element that redefines the severity property of the AbstractEffect with the type Real.

See the diagram GeneralConceptsLibrary.

See also: ErrorPropagation, ErrorRealization associations.

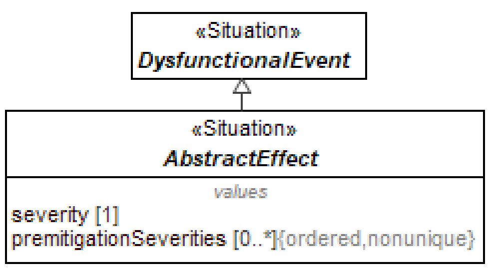


Figure 9.14 - AbstractEffect

Attributes

severity : [1]	A placeholder attribute without a type declared, for indicating the estimate of the extent of harm.
premitigationSeverities : [0..*]	A placeholder attribute for indicating the estimate of the extent of harm that would have resulted from the previous design iterations. This property can have more than one value.

Effect

Package: General Concepts Library
isAbstract: Yes
Generalization: AbstractEffect
Applied Stereotype: «Situation»

Description

An Effect is a specific implementation of AbstractEffect that defines the severity property with the type Real.

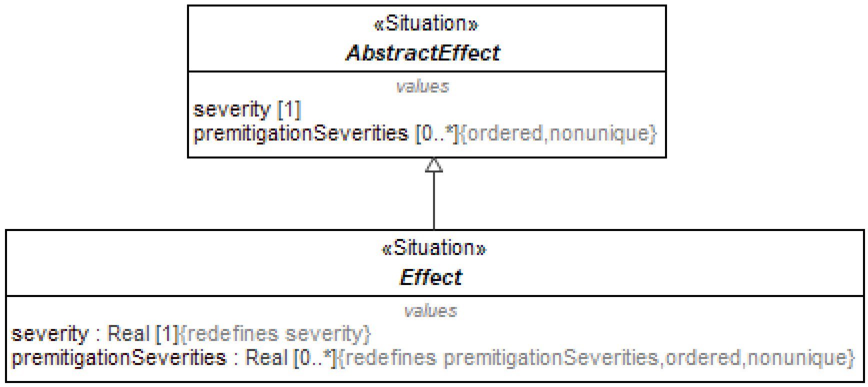


Figure 9.15 - Effect

Attributes

severity : Real[1], redefines severity	An attribute with the type Real, for indicating the estimate of the extent of harm.
premitigationSeverities : Real[0..*], redefines premitigationSeverities	An attribute for indicating the estimate of the extent of harm that would have resulted from the previous design iterations. This property stores more than one value.

Activation

Package: General Concepts Library
Generalization: Causality

Description

A causal relationship describing the propagation of the initial AbstractCause situation to the DysfunctionalEvent situation in the system.

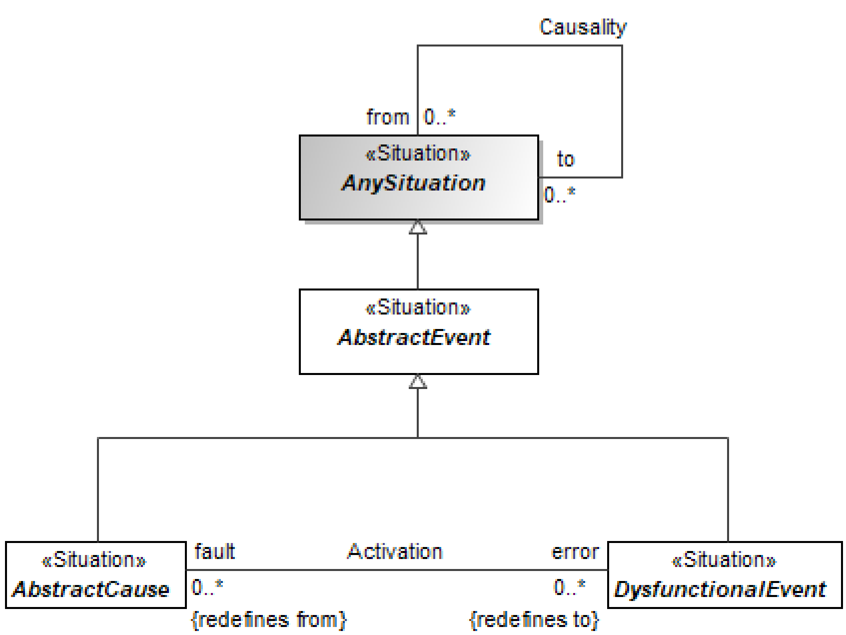


Figure 9.16 - Activation

Association ends

error : DysfunctionalEvent[0..*] (member end of Activation association, redefines to)	The dysfunctional situation (error) of the system.
fault : AbstractCause[0..*] (member end of Activation association, redefines from)	The causal fault.

ErrorPropagation

Package: General Concepts Library
Generalization: Causality

Description

A causal relationship describing the propagation of errors (one error leading to another) throughout the system.

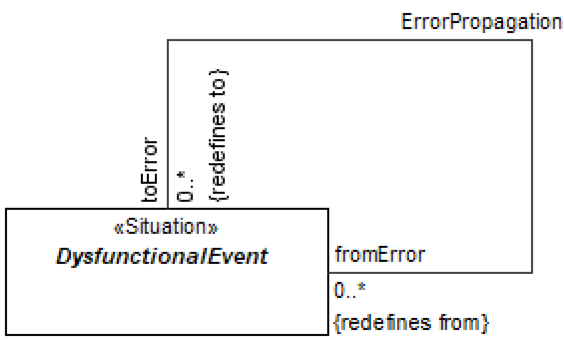


Figure 9.17 - ErrorPropagation

Association ends

toError : DysfunctionalEvent[0..*] (member end of ErrorPropagation association, redefines to)	The successor error.
fromError : DysfunctionalEvent[0..*] (member end of ErrorPropagation association, redefines from)	The predecessor error.

ErrorRealization

Package: General Concepts Library
Generalization: Causality

Description

A causal relationship describing the propagation of an error to a failure.

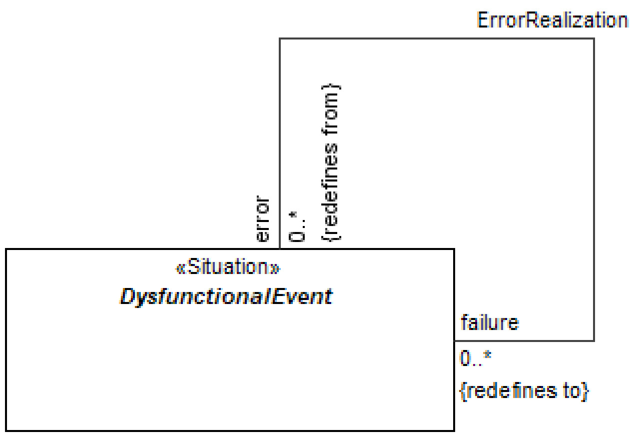


Figure 9.18 - ErrorRealization

Association ends

failure : DysfunctionalEvent[0..*] (member end of ErrorRealization association, redefines to)	The resulting failure.
error : DysfunctionalEvent[0..*] (member end of ErrorRealization association, redefines from)	The predecessor error.

HarmPotential

Package: General Concepts Library
isAbstract: Yes
Generalization: AnySituation
Applied Stereotype: «Situation»

Description

A state where there is the potential of harm. This includes all types of harm arising from malicious or non-malicious causes.

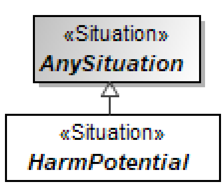


Figure 9.19 - HarmPotential

Hazard

Package: General Concepts Library
isAbstract: Yes
Generalization: HarmPotential
Applied Stereotype: «Situation»

Description

A potential source of harm (IEC 61508-4, 3.1.2). Source of harm is non-malicious.

The term includes danger to persons arising within a short time scale (for example, fire and explosion) and also those that have a long-term effect on a person's health (for example, release of a toxic substance).

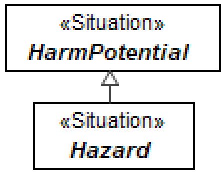


Figure 9.20 - Hazard

Scenario

Package: General Concepts Library
isAbstract: Yes
Generalization: AnySituation
Applied Stereotype: «Situation»

Description

A composite situation, consisting of multiple steps (that are themselves situations). Steps should have causal ordering, indicated by Causality relationships or sub-types thereof.

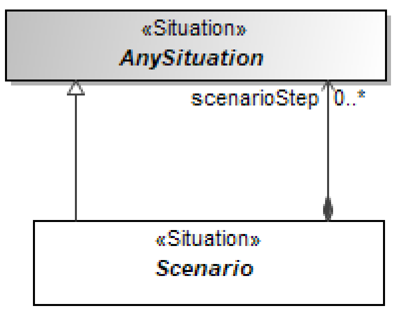


Figure 9.21 - Scenario

Attributes

scenarioStep : AnySituation[0..*] (member end of association)	A situation which is a part of a bigger situation - scenario.

AbstractRisk

Package: General Concepts Library
isAbstract: Yes
Generalization: Scenario
Applied Stereotype: «Situation»

Description

An AbstractRisk is a Scenario - combination of harm potential (Hazard or Vulnerability), triggering event (AbstractEvent), and resulting harm (AbstractEffect).

The AbstractRisk is a placeholder to enable modelers to specify methodology-specific kinds of risks.

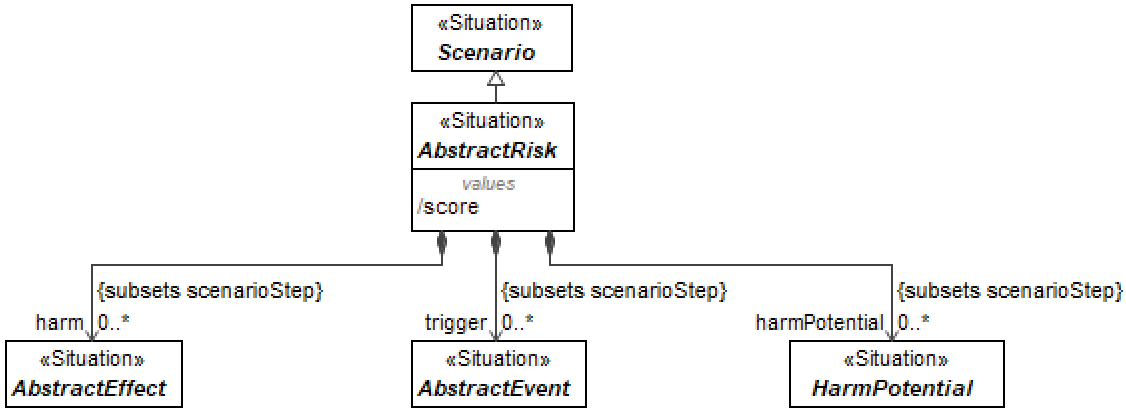


Figure 9.22 - AbstractRisk

Attributes

score :	Combination of the probability of occurrence of abstract event resulting from abstract harm and the severity of that harm (IEC 61508-4, 3.1.5, modified). An example could be risk priority number (RPN) in FMEA analysis.
trigger : AbstractEvent[0..*] (member end of association, subsets scenarioStep)	Triggering event (AbstractEvent) which causes harm to materialize.
harm : AbstractEffect[0..*] (member end of association, subsets scenarioStep)	Resulting harm (AbstractEffect).
harmPotential : HarmPotential[0..*] (member end of association, subsets scenarioStep)	Pre-existing risk (HarmPotential).

UndesiredState

Package: General Concepts Library
isAbstract: Yes
Generalization: DysfunctionalEvent
Applied Stereotype: «Situation»

Description

An element's condition as a specific time which represents an unintended situation.

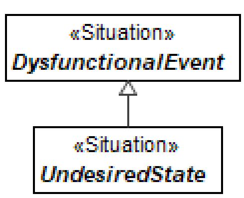


Figure 9.23 - UndesiredState

9.2.2 General::General Concepts Profile

FailureMode

Package: General Concepts Profile
isAbstract: No
Generalization: Situation
Extension: Class

Description

See FailureMode library class for the definition of a situation concept.

The FailureMode stereotype is only needed to distinguish FailureModes from other types of situations.

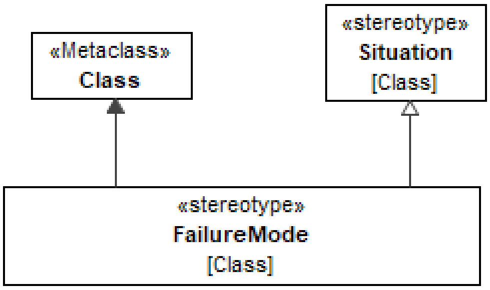


Figure 9.24 - FailureMode

Error

Package: General Concepts Profile
isAbstract: No
Generalization: Situation
Extension: Class

Description

The discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition. [IEC 61508-4, 3.6.11].

The Error stereotype is needed to distinguish this type of situations.

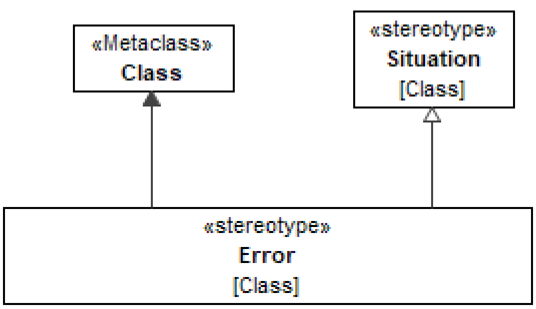


Figure 9.25 - Error

Fault

Package: General Concepts Profile
isAbstract: No
Generalization: Situation
Extension: Class

Description

Abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function. [IEC 61508-4, 3.6.1].

Abnormal or undesired condition that can cause an element or a system to fail. [ISO 26262-1:2018, 3.54, modified]

The Fault stereotype is needed to distinguish this type of situations.

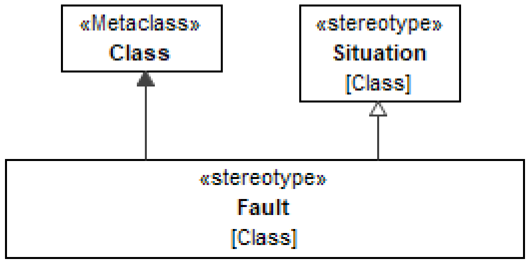


Figure 9.26 - Fault

Detection

Package: General Concepts Profile
isAbstract: No
Generalization: ControllingMeasure
Extension: Dependency

Description

A kind of ControllingMeasure taken to increase probability of detecting the situation under analysis. In hardware these measures may include built-in diagnostic tests, or physical inspection and manual tests.

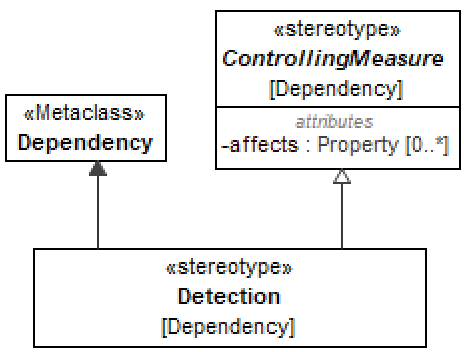


Figure 9.27 - Detection

Prevention

Package: General Concepts Profile
isAbstract: No
Generalization: ControllingMeasure
Extension: Dependency

Description

A kind of ControllingMeasure taken to reduce probability of occurrence of the situation under analysis.

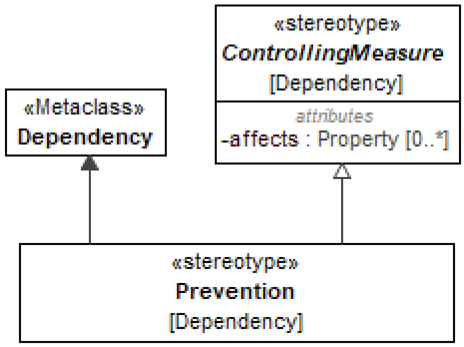


Figure 9.28 - Prevention

Mitigation

Package: General Concepts Profile
isAbstract: No
Generalization: ControllingMeasure
Extension: Dependency

Description

A kind of ControllingMeasure taken to reduce severity of the situation under analysis.

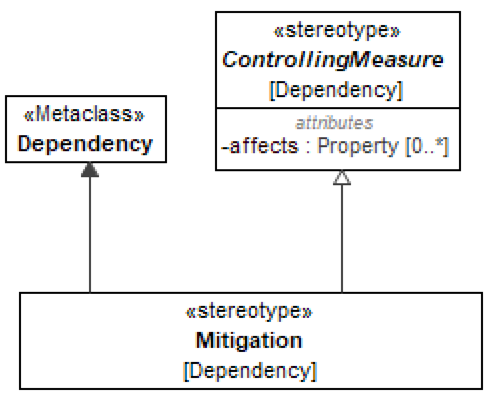


Figure 9.29 - Mitigation

Recommendation

Package: General Concepts Profile
isAbstract: No
Generalization: ControllingMeasure
Extension: Dependency

Description

Recommendation is used to connect the situation to an action item.

An action item is normally a Requirement but it can be a less "strong" type of advice - comment, rationale, etc.

The requirement is further managed by the requirements management system - it can have responsible persons, due date, verification properties etc.

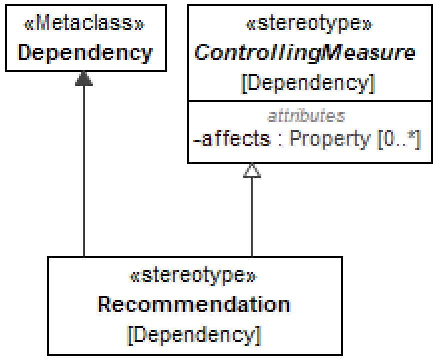


Figure 9.30 - Recommendation

FailureState

Package: General Concepts Profile
isAbstract: No
Extension: State

Description

State, which the system or a part of the system enters after occurrence of FailureMode (failure).

The Failure state concept might be used in various formal safety and reliability analysis methods based on the state machine notation. Failure states could be tied to FailureModes via the RelevantTo dependency.

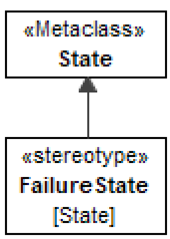


Figure 9.31 - FailureState

Undeveloped

Package: General Concepts Profile
isAbstract: No
Extension: Element

Description

Undeveloped stereotype can be applied in combination with Goal or Strategy stereotype to express the fact that the goal or strategy is not fully developer, and therefore may lack crucial details.

9.3 Methods::FMEA

The Failure Mode and Effects Analysis (FMEA) is a method of inspecting a system to analyze potential failures. Therefore, as many components, assemblies and subsystems as possible are examined in order to identify these failure modes in a system and their causes and effects.

The FMEA package contains all required elements to implement a Failure Model and Effects Analysis. Thus, for each component, the failure modes and their resulting effects on the rest of the system are defined in a SysML BDD and IBD.

9.3.1 Methods::FMEA::FMEALibrary

AbstractFMEAItem

Package: FMEALibrary
isAbstract: Yes
Generalization: AbstractRisk
Applied Stereotype: «FMEAItem»

Description

An AbstractFMEAItem is a scenario (more specifically - AbstractRisk scenario) composed of a failure mode, a cause and (potentially multiple) effect(s). It stores assessed and mitigated risk priority numbers.

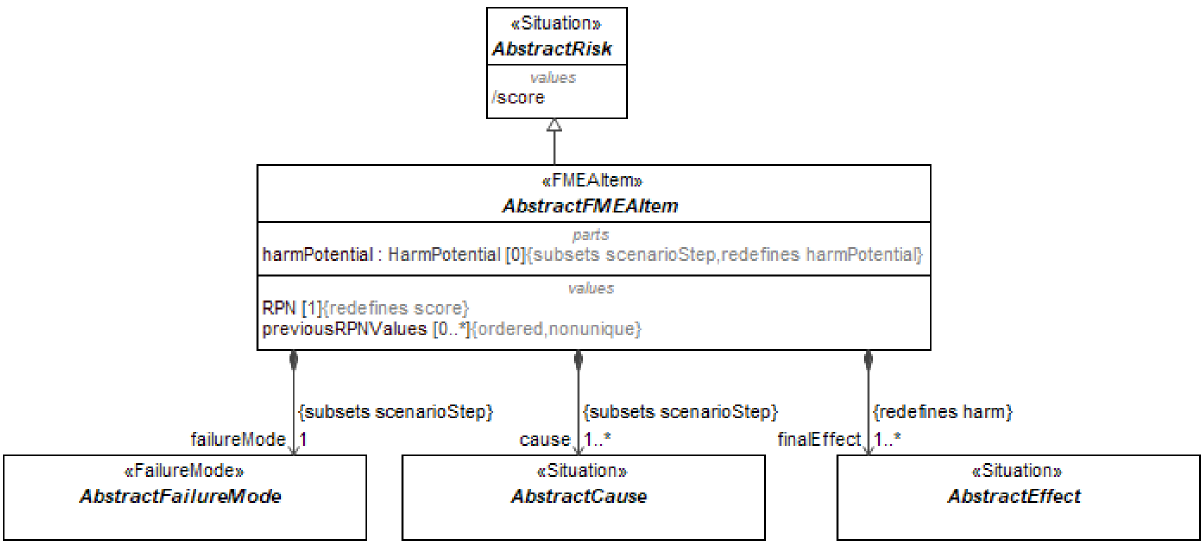


Figure 9.32 - AbstractFMEAItem

Attributes

RPN : [1], redefines score	The risk priority number ranks the risk of the FMEA item. It is a specialization of AbstractRisk::score.
failureMode : AbstractFailureMode[1] (member end of association, subsets scenarioStep)	Represents the failure mode which is reached if a system element fails.
cause : AbstractCause[1..*] (member end of association, subsets scenarioStep)	Represents the cause of the failure of a system element.
finalEffect : AbstractEffect[1..*] (member end of association, redefines harm)	Represents the effect which occurs on the system border.
previousRPNValues : [0..*]	Represents the assessed risk priority number before mitigating the risk of a failure.
harmPotential : HarmPotential[0] (member end of association, redefines harmPotential, subsets scenarioStep)	Pre-existing risk. Not used in FMEA method, therefore redefined in this library with multiplicity [0]

FMEAItem

Package: FMEALibrary
isAbstract: Yes
Generalization: AbstractFMEAItem
Applied Stereotype: «FMEAItem»

Description

A FMEAItem is a specialization of AbstractFMEAItem with the Real implementation of quantitative attributes.

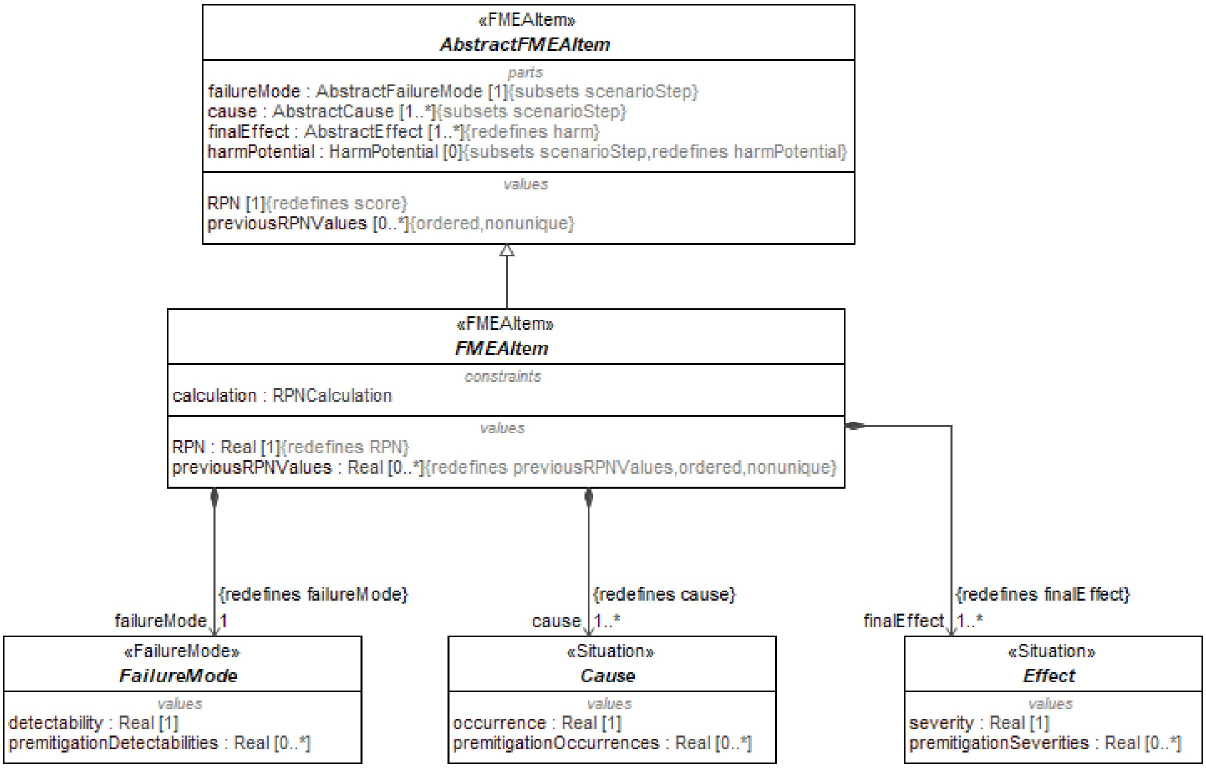


Figure 9.33 - FMEAItem

Attributes

finalEffect : Effect[1..*] (member end of association, redefines finalEffect)	The specialization of AbstractFMEAItem :: finalEffect with the implementation of Effect with Real severity.
cause : Cause[1..*] (member end of association, redefines cause)	The specialization of AbstractFMEAItem :: cause with the implementation of Cause with Real occurrence.
RPN : Real[1], redefines RPN	The specialization of AbstractFMEAItem :: RPN with the type Real.
failureMode : FailureMode[1] (member end of association, redefines failureMode)	The specialization of AbstractFMEAItem :: failureMode with the implementation of FailureMode with Real detectability.
calculation : RPNCalculation	Link to a formula for RPN calculation.
previousRPNValues : Real[0..*], redefines previousRPNValues	The specialization of AbstractFMEAItem :: previousRPNValues with the type Real.

RPNCalculation

Package: FMEALibrary
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description

A formula for RPN calculation. This implementation uses multiplication of Occurrence x Detectability x Severity to calculate RPN.

Attributes

RPN :	Risk priority number
SEV :	Severity
OCC : Real	Occurrence
DET :	Detectability

Constraints

[1] Reduced priority number is calculated by simple multiplication of Severity, Detectability and Occurrence.

LossOfFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing loss of function e.g., the function is inoperable, or suddenly fails.

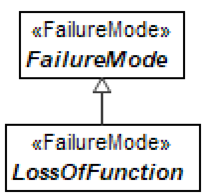


Figure 9.34 - LossOfFunction

DegradationOfFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing a degradation of function or loss of function over time.

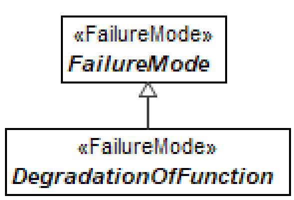


Figure 9.35 - DegradationOfFunction

IntermittentFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing an intermittent function or the random stops and starts of a function.

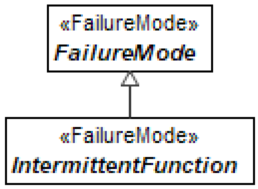


Figure 9.36 - IntermittentFunction

PartialFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing a partial function or loss of performance.

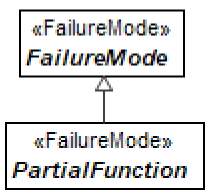


Figure 9.37 - PartialFunction

UnintendedFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing an unintended function, function operating at the wrong time, with unintended direction, or unequal performance.

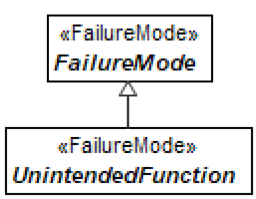


Figure 9.38 - UnintendedFunction

ExceedingFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing a function exceeding the acceptable operational performance.

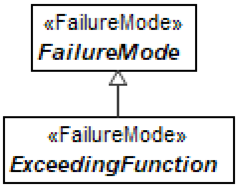


Figure 9.39 - ExceedingFunction

DelayedFunction

Package: FMEALibrary
isAbstract: Yes
Generalization: FailureMode
Applied Stereotype: «FailureMode»

Description

A failure mode representing a delayed function or function operating after an unintended time interval.

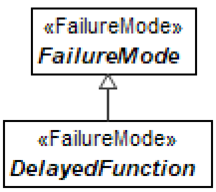


Figure 9.40 - DelayedFunction

9.3.2 Methods::FMEA::FMEAProfile

FMEAItem

Package: FMEAProfile
isAbstract: No
Generalization: Block
Extension: Class

Description

See AbstractFMEAItem library class for the definition of a FMEA Item concept.

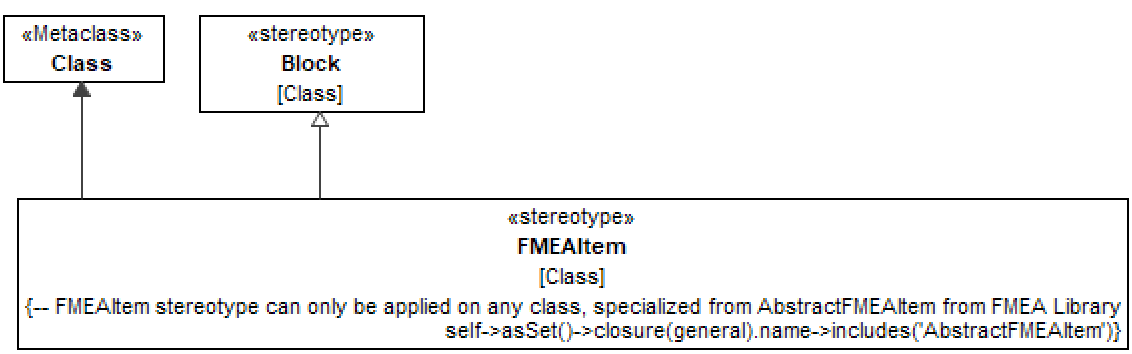


Figure 9.41 - FMEAItem

Constraints

[1] FMEAItemIsAbstractFMEAItem	-- FMEAItem stereotype can only be applied on any class, specialized from AbstractFMEAItem from FMEA Library self.base_Class->asSet()->closure(general).name->includes('AbstractFMEAItem')

9.4 Methods::FTA

Fault Tree Analysis (FTA) is a top-down failure analysis in which an undesired state of a system is analyzed using Boolean logic to combine a series of lower-level (basic) events. This analysis method is used to understand how systems can fail, to identify the best ways to reduce risk and to determine event rates of a safety accident or a functional failure.

The FTA package contains all required elements to implement this analysis. Support for Fault Tree Analysis (FTA) modeling is based on the IEC 61025:2006 standard. Using this standard ensures that the specification offers a form of FTA that is based on best practices and accepted by practitioners. It is also possible for a user to extend the capabilities of the FTA package to enable, for example, dynamic fault tree analysis and component fault tree modeling while still remaining compatible with other information modeled using the specification.

In order to combine FMEA and FTA analysis, a connection between a failure mode and a fault tree event needs to be made. Therefore, the Cause of an FMEAItem can be interpreted as the event which leads to a failure of a system item. By combining FMEAs and FTAs, both analyses can be used to verify the analysis results. This may lead to a better understanding of the behavior of a system during erroneous behavior.

9.4.1 Methods::FTA::FTALibrary

FTAElement

Package: FTALibrary
isAbstract: Yes
Generalization: DysfunctionalEvent
Applied Stereotype: «Situation»

Description

Any of the Events and Gates needed for the evaluation of the TopEvent probability.

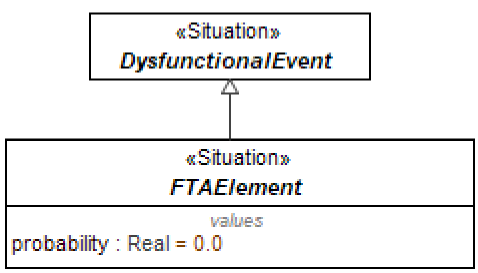


Figure 9.42 - FTAElement

Attributes

probability : Real	The probability that the event represented by the owning FTA element occurs. Probability is a Real value between 0 and 1.

FTATree

Package: FTALibrary
isAbstract: No
Generalization: FTAElement, Scenario
Applied Stereotype: «Tree»

Description

A collection of FTAElements and their interrelationships for the evaluation of the top event probability.

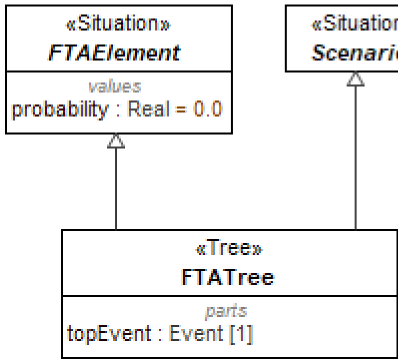


Figure 9.43 - FTATree

Attributes

topEvent : Event[1] (member end of association)	Undesired event which lead to the failure of the system.

Methods::FTA::FTALibrary::Events

Package of events for building fault trees.

Event
Package: Events
isAbstract: Yes
Generalization: FTAElement
Applied Stereotype: «Situation»

Description

The Event is a base class for all types fault tree events. It is a kind of DysfunctionalEvent.

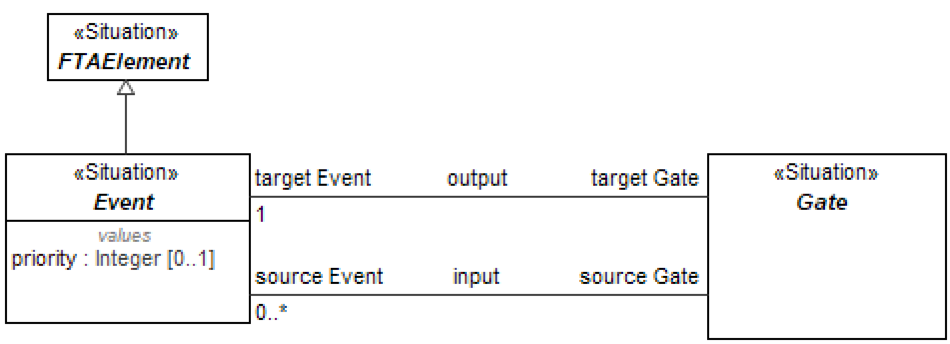


Figure 9.44 - Event

Attributes

source Gate : Gate (member end of input association)	
priority : Integer[0..1]	The priority field is only used to indicate the order of this event when multiple events are inputs of Priority AND (SEQ) gate.
target Gate : Gate (member end of output association)	

BasicEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «BasicEvent»

Description

A basic initiating failure requiring no further development.

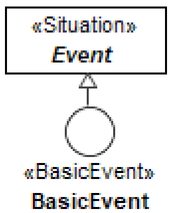


Figure 9.45 - BasicEvent

IntermediateEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «IntermediateEvent»

Description

An intermediate event is a failure which occurs because of one or more antecedent events acting through logic gates.

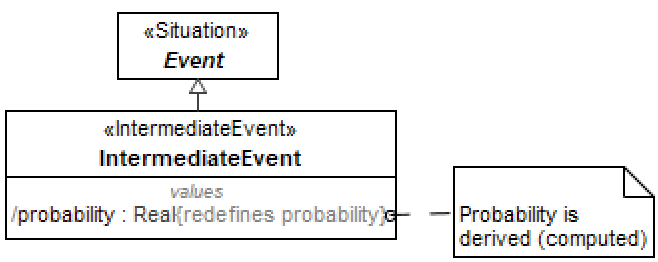


Figure 9.46 - IntermediateEvent

Attributes

probability : Real, redefines probability	Probability of the intermediate event is derived. It is calculated by the gate from the probabilities of the more basic events.

TopEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «TopEvent»

Description

Undesired event - failure or effect - at the top of the fault tree.

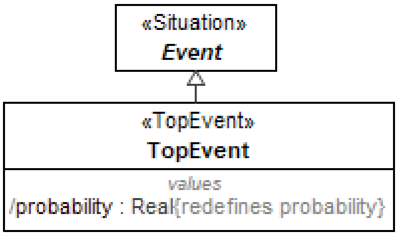


Figure 9.47 - TopEvent

Attributes

probability : Real, redefines probability — The (derived) probability of the top event is the result of the fault tree calculation.

ConditionalEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «ConditionalEvent»

Description

Specific conditions or restrictions that apply to any logic gate (used primarily with PRIORITY AND and INHIBIT gates).

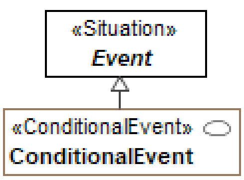


Figure 9.48 - ConditionalEvent

DormantEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «DormantEvent»

Description

The dormant event is similar to BasicEvent but indicates the latent failure which is discovered by periodical tests.

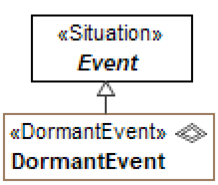


Figure 9.49 - DormantEvent

UndevelopedEvent

Package: Events
isAbstract: No

Generalization: Event
Applied Stereotype: «BasicEvent», «Undeveloped»

Description

An event which is not further developed either because it is of insufficient consequence or because information is unavailable.

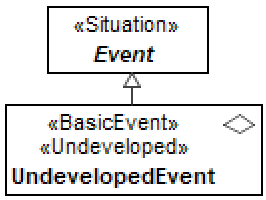


Figure 9.50 - UndevelopedEvent

HouseEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «HouseEvent»

Description
An event which can be set to occur or not occur.

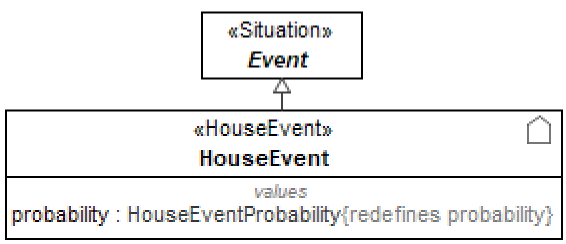


Figure 9.51 - HouseEvent

Attributes

probability : HouseEventProbability, redefines probability	Probability of the house event is 0 or 1. It is set before doing a fault tree evaluation.

ZeroEvent

Package: Events
isAbstract: No
Generalization: Event
Applied Stereotype: «ZeroEvent»

Description
An event which represents a condition or an event that will never occur.

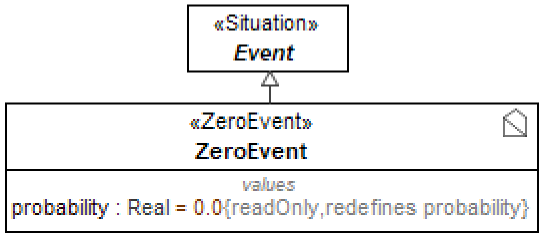


Figure 9.52 - ZeroEvent

Attributes

probability : Real, redefines probability	The probability of zero event is always 0.

Methods::FTA::FTALibrary::Gates

Package of logical conditions for building fault trees.

Gate

Package: Gates
isAbstract: Yes
Applied Stereotype: «Situation»

Description

An FTAElement that combines input Event probabilities in a prescribed manner to determine output Event probability. The output event occurs if the combination of input events is satisfied. The gate subtypes specify the necessary combination.

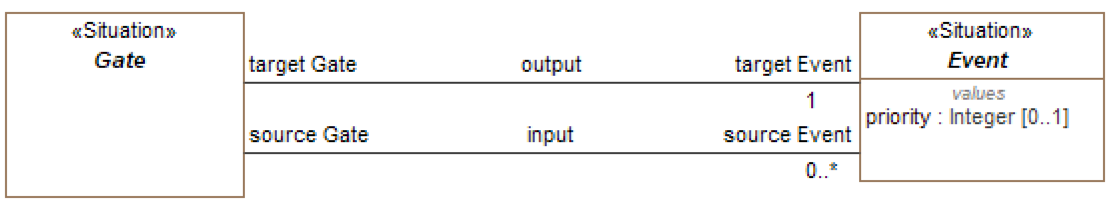


Figure 9.53 - Gate

Attributes

source Event : Event[0..*] (member end of input association)

target Event : Event[1] (member end of output association)

AND

Package: Gates
isAbstract: No
Generalization: Gate
Applied Stereotype: «Block», «AND»

Description

The output event occurs only if all input events occur.

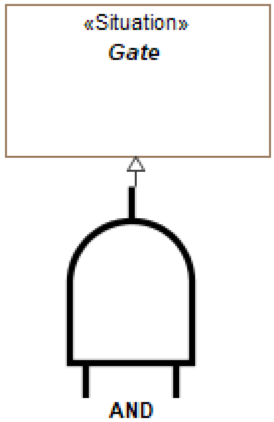


Figure 9.54 - AND

OR

Package: Gates
isAbstract: No
Generalization: Gate
Applied Stereotype: «Block», «OR»

Description
The output event occurs if at least one of input event occurs.

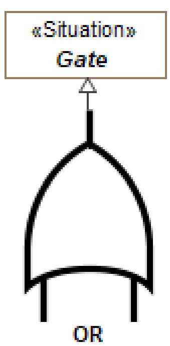


Figure 9.55 - OR

NOT

Package: Gates
isAbstract: No
Generalization: Gate
Applied Stereotype: «Block», «NOT»

Description
The output event occurs if the input event does not occur.

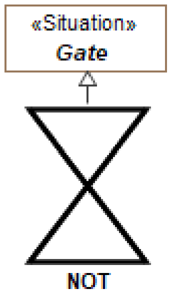


Figure 9.56 - NOT

XOR

Package: Gates
isAbstract: No
Generalization: Gate
Applied Stereotype: «Block», «XOR»

Description
The output event occurs if exactly one of the input events occurs.

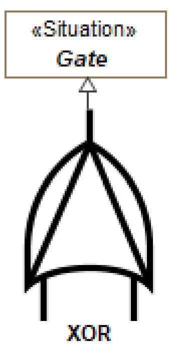


Figure 9.57 - XOR

SEQ

Package: Gates
isAbstract: No
Generalization: Gate
Applied Stereotype: «Block», «SEQ»

Description
The output event occurs if all of the input events occur in a specific sequence.

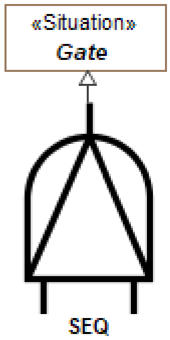


Figure 9.58 - SEQ

INHIBIT

Package: Gates
isAbstract: No
Generalization: Gate
Applied Stereotype: «INHIBIT», «Block»

Description
The output event occurs if the (single) input event occurs in the presence of an enabling condition.

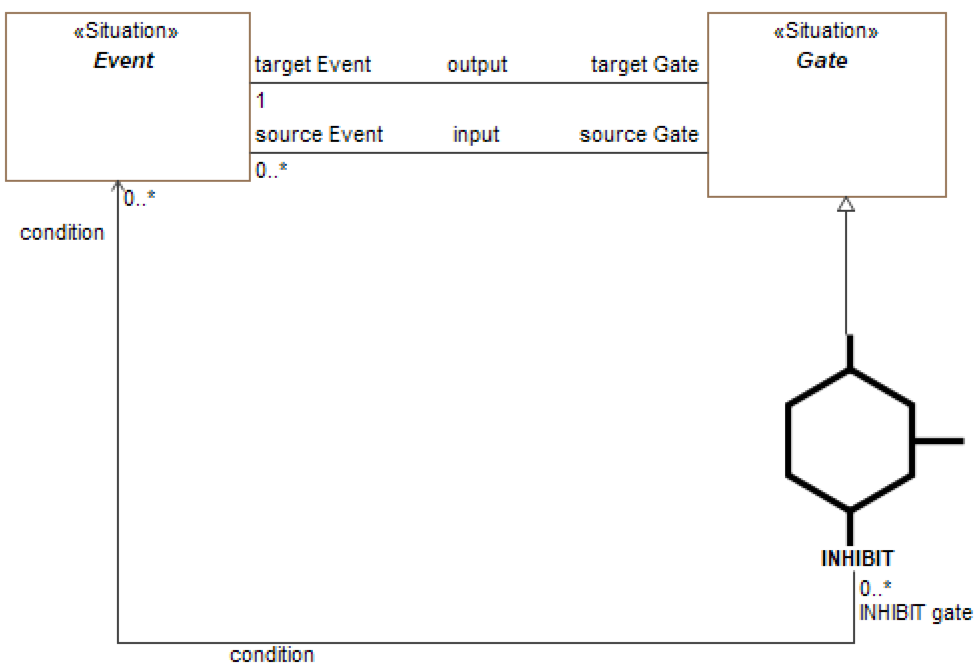


Figure 9.59 - INHIBIT

Attributes

condition : Event[0..*] (member end of condition association)

MAJORITY_VOTE

Package: Gates
isAbstract: No

Generalization: Gate
Applied Stereotype: «Block», «MAJORITY_VOTE»

Description
The output event occurs if the majority of the input events occurs. It has a threshold parameter m.

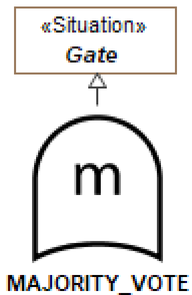


Figure 9.60 - MAJORITY_VOTE

Attributes

m : Integer	The m parameter defines the number of input events that form a majority. It is not necessarily ceil(number_of_inputs / 2). It is possible to stipulate that e.g. 5 (or 2) input events have to occur out of total of 7 events for majority gate to fire.

Methods::FTA::FTALibrary::Gates::ConstraintBlocks
Reference implementation for the FTA gates.

ANDConstraintBlock
Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description
Reference implementation for the AND gate.

Attributes
output :
input : [0..*]

Constraints

[1]	Probability of AND node is simply a multiplication of probabilities of incoming nodes. Note - this simplistic calculation assumes that incoming node events are mutually independent.

ORConstraintBlock
Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»
Description

Reference implementation for the OR gate.

Attributes

output :

input : [0..*]

Constraints

[1] Probability of OR node is calculated as opposite probability of the event where neither of the input events happen.

This follows De Morgan's theorem - OR(input1, input2, input3...) is equal to NOT AND (NOT input1, NOT input2, NOT input3...).

Note - this simplistic calculation assumes that incoming node events are mutually independent.

SEQConstraintBlock

Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description

Reference implementation for the SEQ gate.

Attributes

output :

input : Real[0..*]

Constraints

[1] Probability of SEQ node is calculated the same way as AND node - it is simply a multiplication of probabilities of incoming nodes.

This simplistinc calculation cannot capture time-dependency of the events; only more complex simulations can estimate this probability.

XORConstraintBlock

Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description

Reference implementation for the XOR gate.

Attributes

output :

input : [0..*]

Constraints

[1] In case of two inputs, XOR probability is calculated by ORing of two event combintation probabilities -

probability that first event happened and second did not ORed with probability that second event happened while first did not.

Input1 XOR Input2 = Input1 AND NOT Input2 OR Input2 AND NOT Input1

Since combinations are mutually exclusive, simple (+) operation can be used for ORing them. Therefore

Input1 XOR Input2 = Input1 AND NOT Input2 + Input2 AND NOT Input1

Further expanding ANDs and NOTs using their corresponding formulas, we get

Input1 XOR Input2 = Input1*(1 - Input2) + Input2*(1 - Input1) = Input1 + Input2 - 2 * Input1 * Input2

This formula can be iteratively applied for the case with number of inputs greater than two.

Note - this simplistic calculation assumes that incoming node events are mutually independent.

INHIBITConstraintBlock

Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description

Reference implementation for the INHIBIT gate.

Attributes

output :

input : [0..*]

condition : Real

Constraints

[1] Probability of INHIBIT node is calculated the same way as AND node - it is simply a multiplication of probabilities of input nodes and condition nodes.

Note - this simplistic calculation assumes that incoming node events and conditions are mutually independent.

MAJORITY_VOTEConstraintBlock

Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description

Reference implementation for the MAJORITY_VOTE gate.

Attributes

output :

input : [0..*]

m :

Constraints

[1] Majority Vote probability can be calculated by iteratively examining all the combinations of input events,

taking those combinations that satisfy the condition that at least m input events happen,

then calculating probability of each combination using AND formula (multiplying all individual event probabilities in that combination) and then calculating cumulative probability of all combinations by ORing them.

Note - this simplistic calculation assumes that incoming node events are mutually independent.

NOTConstraintBlock

Package: ConstraintBlocks
isAbstract: No
Applied Stereotype: «ConstraintBlock»

Description

Reference implementation for the NOT gate.

Attributes

output :

input : [1]

Constraints

[1] Probability of NOT node is calculated as probability of the event opposite to the input event.

Thereby it is unity minus probability of input event.

9.4.2 Methods::FTA::FTAProfile

Tree

Package: FTAProfile
isAbstract: No
Generalization: Situation
Extension: Class

Description

A marker stereotype for fault trees. See FTATree library class for definition.

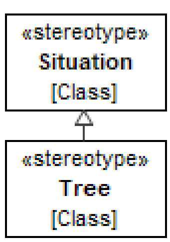


Figure 9.61 - Tree

Constraints

[1] TreeIsFTATree -- Tree stereotype can only be applied on any class specialized from FTATree from FTA Library

self.base_Class->asSet()->closure(general).name->includes('FTATree')

Gate

Package: FTAProfile
isAbstract: Yes
Extension: Class

Description

A marker stereotype for fault tree gates. See Gate library class for definition.

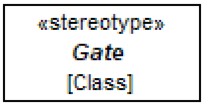


Figure 9.62 - Gate

Event

Package: FTAProfile
isAbstract: Yes
Generalization: Situation
Extension: Class

Description

A marker stereotype for fault tree events. See Event library class for definition.

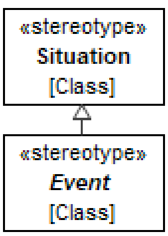


Figure 9.63 - Event

DormantEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for dormant events. See DormantEvent library class for definition.

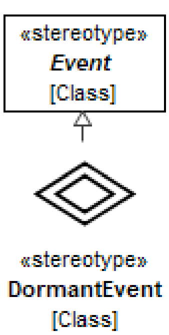


Figure 9.64 - DormantEvent

Constraints

[1] DormantEventIsDormantEvent	--DormantEvent stereotype can only be applied on any class specialized from DormantEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('DormantEvent')

BasicEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for basic events. See BasicEvent library class for definition.

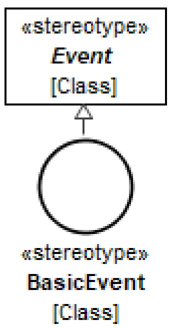


Figure 9.65 - BasicEvent

Constraints

[1] BasicEventIsBasicEvent	--BasicEvent stereotype can only be applied on any class specialized from BasicEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('BasicEvent')
[2] UndevelopedEventIsUndevelopedEvent	--BasicEvent + Undeveloped stereotype combination can be applied on any class specialized from UndevelopedEvent from FTA Library Undeveloped.allInstances().base_Element->includesAll(self.base_Class) implies self.base_Class->asSet()->closure(general).name->includes('UndevelopedEvent')

ConditionalEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for conditional events. See ConditionalEvent library class for definition.

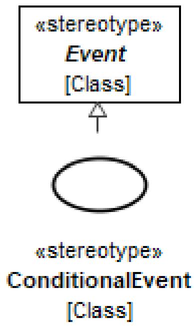


Figure 9.66 - ConditionalEvent

Constraints

[1] ConditionalEventIsConditionalEvent	--ConditionalEvent stereotype can only be applied on any class specialized from ConditionalEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('ConditionalEvent')

ZeroEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for zero events. See ZeroEvent library class for definition.

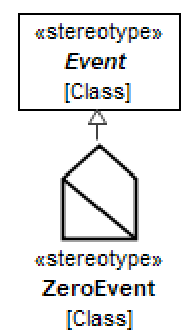


Figure 9.67 - ZeroEvent

Constraints

[1] ZeroEventIsZeroEvent	--ZeroEvent stereotype can only be applied on any class specialized from ZeroEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('ZeroEvent')

HouseEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for house events. See HouseEvent library class for definition.

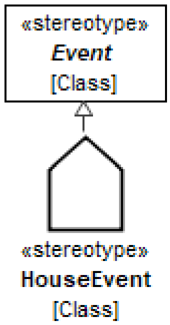


Figure 9.68 - HouseEvent

Constraints

[1] HouseEventIsHouseEvent	--HouseEvent stereotype can only be applied on any class specialized from HouseEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('HouseEvent')

AND

Package: FTAProfile
isAbstract: No
Generalization: Gate
Extension: Class

Description

A marker stereotype, carrying icon for AND gates. See AND library class for definition.

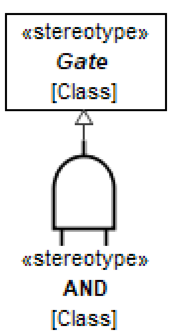


Figure 9.69 - AND

Constraints

[1] ANDIsAND	--AND stereotype can only be applied on any class specialized from AND gate from FTA Library self.base_Class->asSet()->closure(general).name->includes('AND')

OR

Package: FTAProfile
isAbstract: No
Generalization: Gate

Extension: Class

Description

A marker stereotype, carrying icon for OR gates. See OR library class for definition..

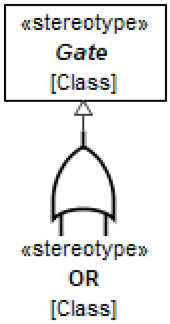


Figure 9.70 - OR

Constraints

[1] ORIsOR	--OR stereotype can only be applied on any class specialized from OR gate from FTA Library self.base_Class->asSet()->closure(general).name->includes('OR')

SEQ

Package: FTAProfile
isAbstract: No
Generalization: Gate
Extension: Class

Description

A marker stereotype, carrying icon for SEQ gates. See SEQ library class for definition.

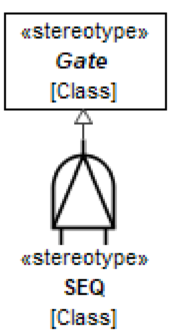


Figure 9.71 - SEQ

Constraints

[1] SEQIsSEQ	--SEQ stereotype can only be applied on any class specialized from SEQ gate from FTA Library self.base_Class->asSet()->closure(general).name->includes('SEQ')

XOR

Package: FTAProfile

isAbstract: No
Generalization: Gate
Extension: Class

Description

A marker stereotype, carrying icon for XOR gates. See XOR library class for definition.

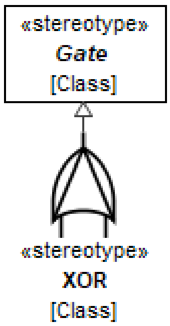


Figure 9.72 - XOR

Constraints

[1] XORIsXOR	--XOR stereotype can only be applied on any class specialized from XOR gate from FTA Library self.base_Class->asSet()->closure(general).name->includes('XOR')

INHIBIT

Package: FTAProfile
isAbstract: No
Generalization: Gate
Extension: Class

Description

A marker stereotype, carrying icon for INHIBIT gates. See INHIBIT library class for definition.

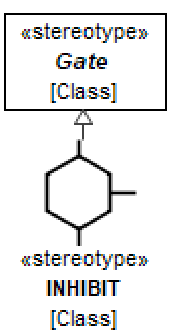


Figure 9.73 - INHIBIT

Constraints

[1] INHIBITIsINHIBIT	--INHIBIT stereotype can only be applied on any class specialized from INHIBIT gate from FTA Library self->asSet()->closure(general).name->includes('INHIBIT')

MAJORITY_VOTE

Package: FTAProfile
isAbstract: No
Generalization: Gate
Extension: Class

Description

A marker stereotype, carrying icon for MAJORITY_VOTE gates. See MAJORITY_VOTE library class for definition.

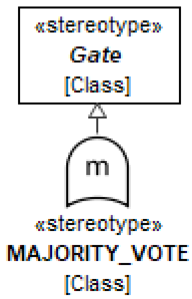


Figure 9.74 - MAJORITY_VOTE

Constraints

[1] MAJORITY_VOTEIsMAJORITY_VOTE	--MAJORITY_VOTE stereotype can only be applied on any class specialized from MAJORITY_VOTE gate from FTA Library self.base_Class->asSet()->closure(general).name->includes('MAJORITY_VOTE')

NOT

Package: FTAProfile
isAbstract: No
Generalization: Gate
Extension: Class
Description

A marker stereotype, carrying icon for NOT gates. See NOT library class for definition.

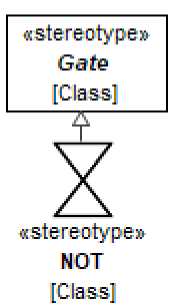


Figure 9.75 - NOT

Constraints

[1] NOTIsNOT	--NOT stereotype can only be applied on any class specialized from NOT gate from FTA Library self.base_Class->asSet()->closure(general).name->includes('NOT')

IntermediateEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for intermediate events. See IntermediateEvent library class for definition.

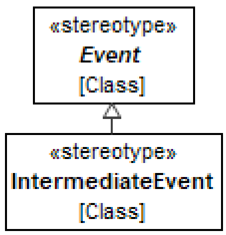


Figure 9.76 - IntermediateEvent

Constraints

[1] IntermediateEventIsIntermediateEvent	--IntermediateEvent stereotype can only be applied on any class specialized from IntermediateEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('IntermediateEvent')

TopEvent

Package: FTAProfile
isAbstract: No
Generalization: Event
Extension: Class

Description

A marker stereotype, carrying icon for top events. See TopEvent library class for definition.

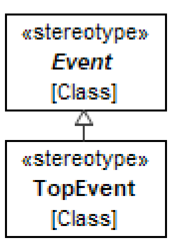


Figure 9.77 - TopEvent

Constraints

[1] TopEventIsTopEvent	--TopEvent stereotype can only be applied on any class specialized from TopEvent from FTA Library self.base_Class->asSet()->closure(general).name->includes('TopEvent')

TransferIn

Package: FTAProfile
isAbstract: No

Extension: Property

Description

The node of the current fault tree that indicates that the tree is developed further as a separate fault tree - TransferOut.

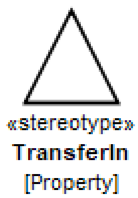


Figure 9.78 - TransferIn

Constraints

[1] TypeIsTransferOut -- type of TransferIn property must be TransferOut FTA Tree
TransferOut.allInstances().base_Class->includesAll(self.base_Property.type)

TransferOut

Package: FTAProfile
isAbstract: No
Generalization: Tree
Extension: Class

Description

A marker stereotype for partial fault trees. It indicates that this tree is used as a part of another fault tree through TransferIn. The computed probability of the top event of the TransferOut tree is used as a probability of the TransferIn node.

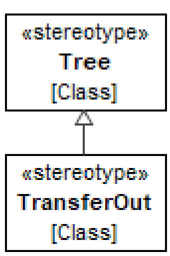


Figure 9.79 - TransferOut

9.5 Methods::STPA

The System Theoretical Process Analysis (STPA) is a hazard analysis technique based on control and system theory. In comparison, most existing hazard analysis techniques are based on reliability theory. In STPA, however, the easy goals are pursued as in any hazard analysis, i.e., collecting information on how hazards may occur. For further information on this approach the handbook[1] describes the method and show the application.

[1] https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

9.5.1 Methods::STPA::STPA Library

OutOfSequence

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing kind of control.

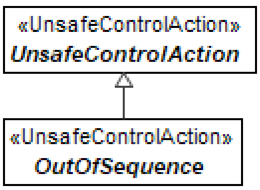


Figure 9.80 - OutOfSequence

Late

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing kind of control.

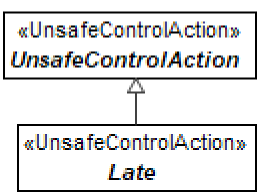


Figure 9.81 - Late

Early

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing kind of control.

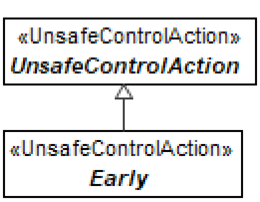


Figure 9.82 - Early

TooLong

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing kind of control.

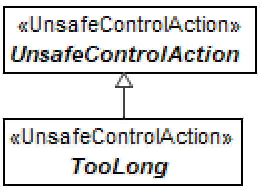


Figure 9.83 - TooLong

TooShort

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing kind of control.

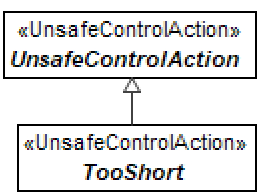


Figure 9.84 - TooShort

Provided

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing a kind of control.

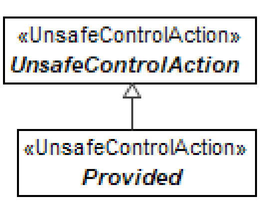


Figure 9.85 - Provided

NotProvided

Package: STPA Library
isAbstract: Yes
Generalization: UnsafeControlAction
Applied Stereotype: «UnsafeControlAction»

Description

STPA Guideword, describing kind of control.

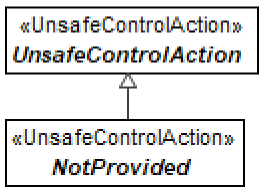


Figure 9.86 - NotProvided

LossScenario

Package: STPA Library
isAbstract: Yes
Generalization: Scenario
Applied Stereotype: «Situation»

Description

A sequence of situations starting from Factors, that (through Process Model deficiencies) leads to an UnsafeControlAction (which further leads to risks and possibly losses).

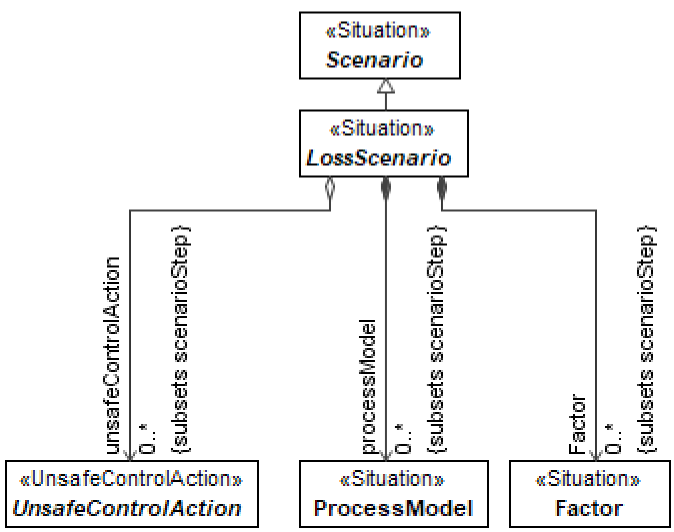


Figure 9.87 - LossScenario

Attributes

Factor : Factor[0..*] (member end of association, subsets scenarioStep)

unsafeControlAction : UnsafeControlAction[0..*] (member end of association, subsets scenarioStep) processModel : ProcessModel[0..*] (member end of association, subsets scenarioStep)

ProcessModel

Package: STPA Library
isAbstract: No
Applied Stereotype: «Situation»

Description

A ProcessModel describes a process / control loop model that may lead to an Unsafe Control Action. The four high level kinds of process model deficiencies can be used to specify the section of the control loop.

Process model deficiencies are often called (high level) Scenario in STPA theory.

Attributes

Factor : Factor[0..*] (member end of ProcessModelFactor association, redefines from)

unsafeControlAction : UnsafeControlAction[0..*] (member end of ProcessModelConsequence association, redefines to)

Inadequate Controller Decisions

Package: STPA Library
isAbstract: Yes
Generalization: ProcessModel
Applied Stereotype: «Situation»

Description

A kind of ProcessFlaw.

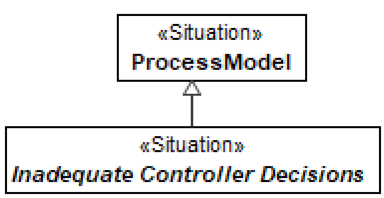


Figure 9.88 - Inadequate Controller Decisions

Inadequate Control Execution

Package: STPA Library
isAbstract: Yes
Generalization: ProcessModel
Applied Stereotype: «Situation»

Description

A kind of ProcessFlaw.

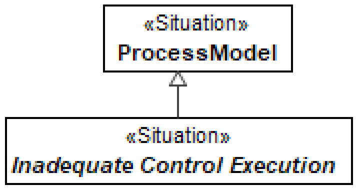


Figure 9.89 - Inadequate Control Execution

Inadequate Process Behavior

Package: STPA Library
isAbstract: Yes
Generalization: ProcessModel
Applied Stereotype: «Situation»

Description

A kind of ProcessFlaw.

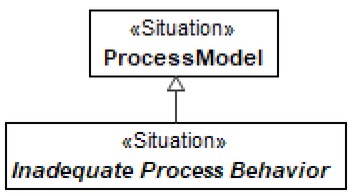


Figure 9.90 - Inadequate Process Behavior

Inadequate Feedback and Inputs

Package: STPA Library
isAbstract: Yes
Generalization: ProcessModel
Applied Stereotype: «Situation»

Description

A kind of ProcessFlaw.

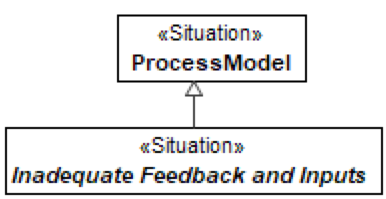


Figure 9.91 - Inadequate Feedback and Inputs

UnsafeControlAction

Package: STPA Library
isAbstract: Yes
Generalization: UndesiredState
Applied Stereotype: «UnsafeControlAction»

Description

An Unsafe Control Action (UCA), used in STPA, describes in what context providing / not providing a Control Action might lead to an undesired result.

A UCA generally consist of four parts:

- Controller (Subject) that issues the Control Action - inferred from Control Action and model of the system (block/part producing the control action).
- Guideword (provides, does not provide, etc.) - indicated using Generalization relationship
- Control Action - connected with RelevantTo relationship.
- Context in which Control Action leads to undesired outcome - sub situation of (part of) UCA situation.

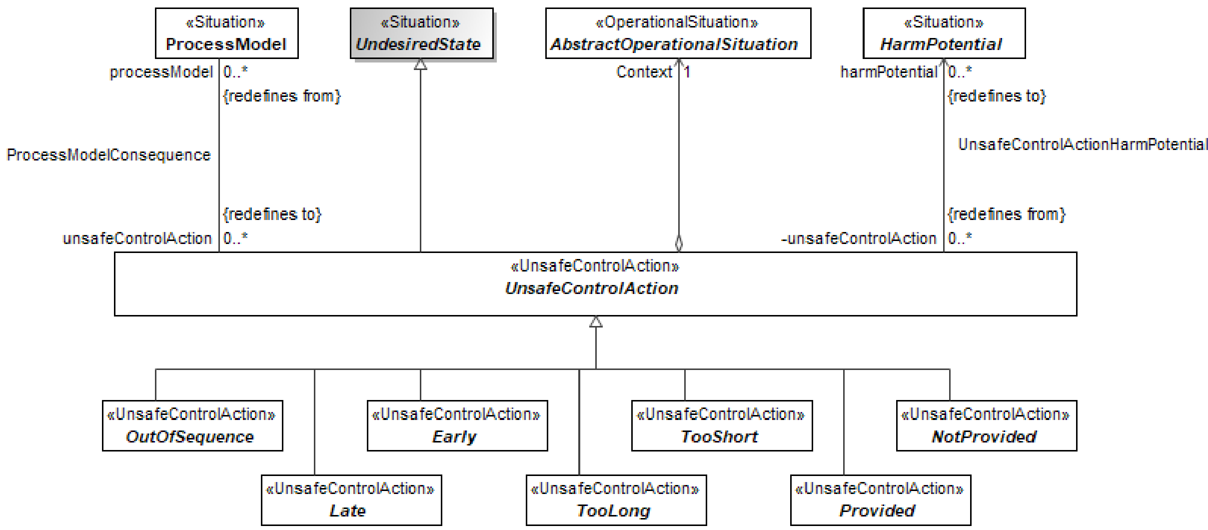


Figure 9.92 - UnsafeControlAction

Attributes

Context : AbstractOperationalSituation[1]
(member end of association)

processModel : ProcessModel[0..*]
(member end of
ProcessModelConsequence association,
redefines from)

harmPotential : HarmPotential[0..*]
(member end of
UnsafeControlActionHarmPotential
association, redefines to)

Factor

Package: STPA Library
isAbstract: No
Generalization: AbstractCause
Applied Stereotype: «Situation»

Description

A Factor (F) can be used to further refine Process Model inadequacies - specifying causes of deficiencies in the process model and/or other contributing factors.

Attributes

processModel : ProcessModel[0..*]
(member end of ProcessModelFactor
association, redefines to)

Loss

Package: STPA Library
isAbstract: Yes
Generalization: AbstractEffect
Applied Stereotype: «Situation»

Description

In STPA, is any effect that is unacceptable and should be prevented. Some factors such as environmental conditions may contribute to a loss but are outside our control.

Examples for losses are:

- Loss of human life or injury
- Vehicle/property damage
- Mission loss (inadequate transportation)
- Loss of customer satisfaction
- Financial loss
- Loss of public image
- Environmental pollution

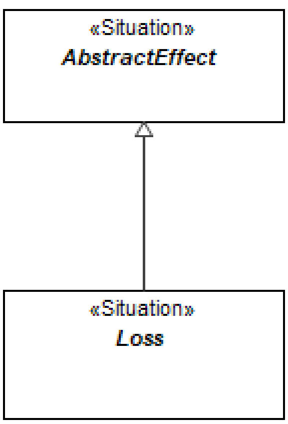


Figure 9.93 - Loss

RiskRealization

Package: STPA Library
isAbstract: No
Generalization: AbstractRisk, Causality
Applied Stereotype: «Block»

Description

Association between the Loss and Hazard (potential harm).

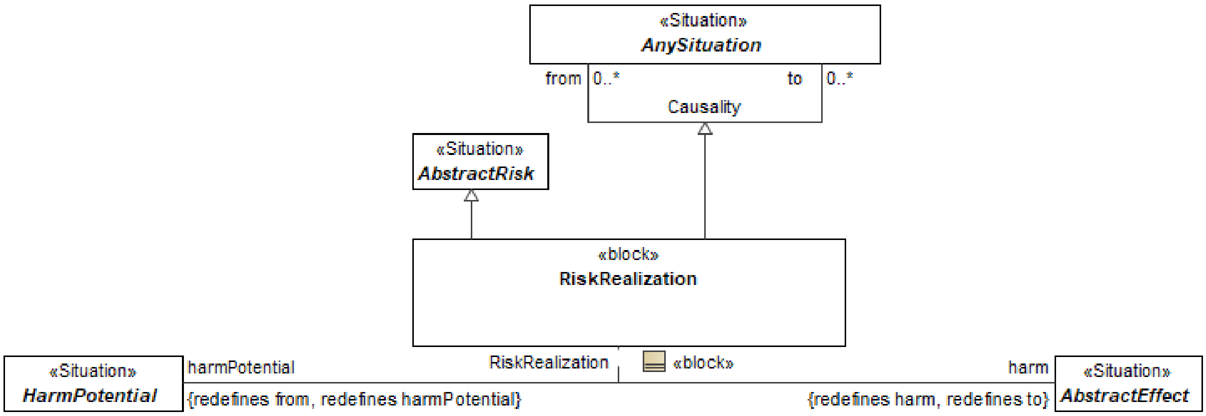


Figure 9.94 - RiskRealization

ProcessModelFactor

Package: STPA Library
Generalization: Causality

Description

Causal relationship between CausalFactor and ProcessFlaw

Association ends

processModel : ProcessModel[0..*] (member end of ProcessModelFactor association, redefines to)

Factor : Factor[0..*] (member end of ProcessModelFactor association, redefines from)

ProcessModelConsequence

Package: STPA Library
Generalization: Causality

Description

Causal relationship between ProcessFlaw and UnsafeControlAction

Association ends

unsafeControlAction : UnsafeControlAction[0..*] (member end of ProcessModelConsequence association, redefines to)

processModel : ProcessModel[0..*] (member end of ProcessModelConsequence association, redefines from)

UnsafeControlActionHarmPotential

Package: STPA Library
Generalization: Causality

Description

Causal relationship between UnsafeControlAction and RiskSource

Association ends

harmPotential : HarmPotential[0..*] (member end of UnsafeControlActionHarmPotential association, redefines to)

unsafeControlAction : UnsafeControlAction[0..*] (member end of UnsafeControlActionHarmPotential association, redefines from)

9.5.2 Methods::STPA::STPA Profile

ControlAction

Package: STPA Profile
isAbstract: No
Extension: Signal, Class, DataType

Description

A Control Action (CA) is an output signal from a functional / logical Controller to a ControlledProcess (via the Actuator), that determines the receiving process behaviour.

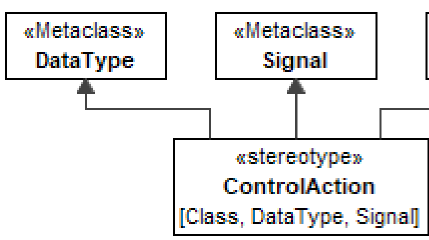


Figure 9.95 - ControlAction

Feedback

Package: STPA Profile
isAbstract: No
Extension: Signal, Class, DataType

Description

A Feedback is an input signal to a functional / logical Controller from a ControlledProcess (via the Sensor), that characterizes the current processes behavior (or the environment).

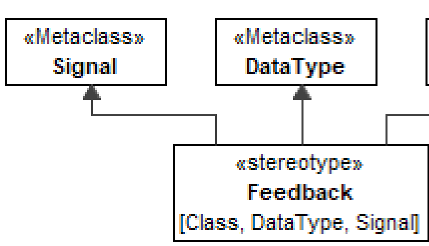


Figure 9.96 - Feedback

UnsafeControlAction

Package: STPA Profile
isAbstract: No
Generalization: FailureMode
Extension: Class

Description

Stereotype used to demarcate all the UnsafeControlActions.

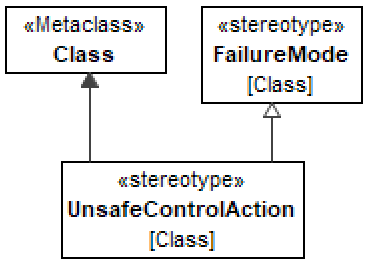


Figure 9.97 - UnsafeControlAction

ControlledProcess

Package: STPA Profile
isAbstract: No
Extension: Property, Class

Description

An abstract representation of the system and it's behaviours that need to be supervised and governed. Controller is controlling this process through the ControlAction via the Actuator.

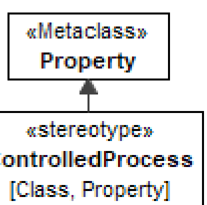


Figure 9.98 - ControlledProcess

Actuator

Package: STPA Profile
isAbstract: No
Extension: Property, Class

Description

Actuator receives ControlActions from Controller and influences the ControlledProcess in some way.

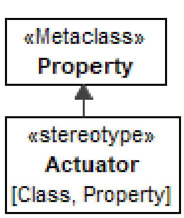


Figure 9.99 - Actuator

Sensor

Package: STPA Profile
isAbstract: No
Extension: Property, Class

Description

Sensor assesses the ControlledProcess (also environment or other controllers) and gives Feedback to the Controller.

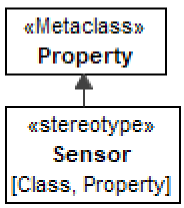


Figure 9.100 - Sensor

Controller

Package: STPA Profile
isAbstract: No
Extension: Property, Class

Description

Controller sends the ControlActions and receives Feedback.

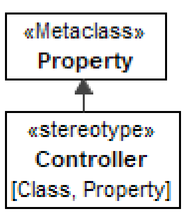


Figure 9.101 - Controller

ControlStructure

Package: STPA Profile
isAbstract: No
Generalization: Block
Extension: Class

Description

ControlStructure is a system-of-systems composed of ControlledProcess, Controller and their functional relationships - ControllActions, Feedbacks, describing feedback control loops.

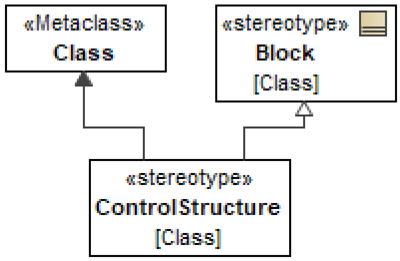


Figure 9.102 - ControlStructure

9.6 GSN

9.6.1 GSN::GSN Profile

Notation

Most of the stereotypes in GSN profile have stereotype images specified. Displaying the stereotyped GSN elements in UML Class diagram may follow the UML standard prescription (UML 2.5.1, Chapter 12.3.4.1 Icon presentation) for displaying elements having stereotypes with icons, namely:

- Showing model element as an image with element name below
- Showing model element as a box with the iconic form image inside the box at the top left

Figure 9.103 - Standard UML notation for stereotyped elements (from UML 2.5.1, Figure 12.25)

However, in addition to the notation described in UML standard, this standard allows additional notation. Namely – using stereotype image as a (resizable) outline/shape of the box, with the same compartments that are prescribed by the UML standard (including name/stereotype/tag values compartment) inside. This notation is recommended i.e. preferred over the standard UML notation.

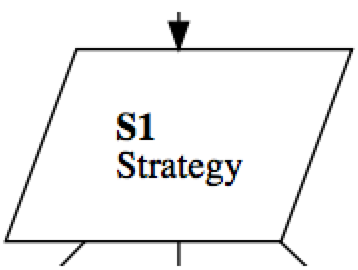


Figure 9.104 - Recommended notation

Combined Stereotype Notation

GSN standard allows combination of several stereotypes applied on the model element. Namely – the combination of Goal+Undeveloped stereotypes and Strategy+Undeveloped stereotypes is being used.

In that case, recommended notation is a combination of image shapes for Goal (or Strategy) and Undeveloped.

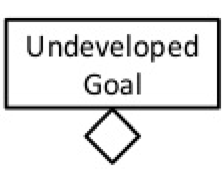


Figure 9.105 - Stereotype combination notation

GSNNode

Package: GSN Profile
isAbstract: Yes
Extension: Element

Description

Root type for all the different kinds of nodes in GSN.

Note: name versus human-readable ID

GSN domain elements frequently have both a short phrase, describing the element and human-readable identifier. For example:

G1 Control System is acceptably safe to operate

In this example "Control System is acceptably safe to operate" is a short phrase, describing the goal, while G1 is a human-readable identifier of the goal.

In this standard, the short phrase shall be captured as UML model element name –NamedElement::name field. Human-readable identifier shall be stored in a separate tag, defined in this profile – GSNNode::id.

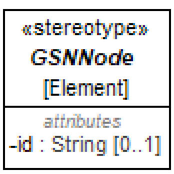


Figure 9.106 - GSNNode

Attributes

id : String[0..1]

GSNArgumentNode

Package: GSN Profile
isAbstract: Yes
Generalization: GSNNode
Extension: Element

Description

A Goal or a Strategy.

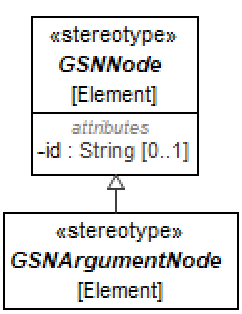


Figure 9.107 - GSNArgumentNode

Solution

Package: GSN Profile
isAbstract: No
Generalization: GSNNode
Extension: Class

Description

A solution presents a reference to an evidence item or items.

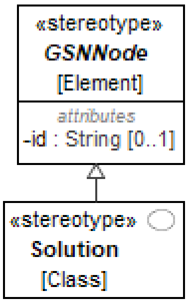


Figure 9.108 - Solution

Goal

Package: GSN Profile
isAbstract: No
Generalization: GSNArgumentNode
Extension: Class

Description

A goal presents a claim forming part of the argument.

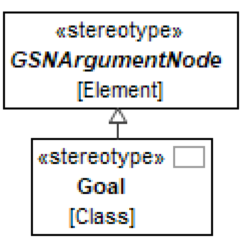


Figure 9.109 - Goal

Strategy

Package: GSN Profile
isAbstract: No
Generalization: GSNArgumentNode
Extension: Class

Description

A strategy describes the nature of the inference that exists between a goal and its supporting goal(s).

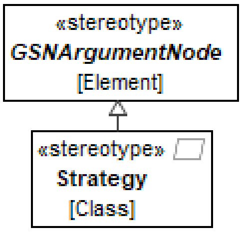


Figure 9.110 - Strategy

SupportingInformation

Package: GSN Profile
isAbstract: Yes
Extension: Element

Description

A ContextStatement or an Assumption or a Justification.

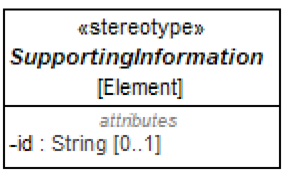


Figure 9.111 - SupportingInformation

Attributes

id : String[0..1]

ContextStatement

Package: GSN Profile
isAbstract: No
Generalization: SupportingInformation
Extension: Class

Description

A context presents a contextual artefact. This can be a reference to contextual information, or a statement.

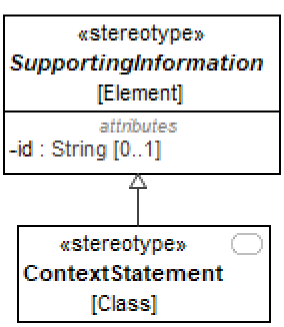


Figure 9.112 - ContextStatement

Assumption

Package: GSN Profile

isAbstract: No
Generalization: SupportingInformation
Extension: Class

Description

An assumption presents an intentionally unsubstantiated statement.

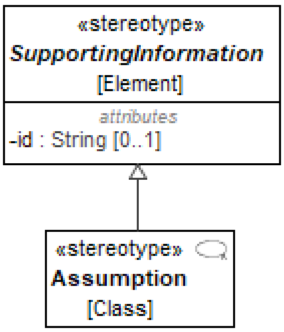


Figure 9.113 - Assumption

Justification

Package: GSN Profile
isAbstract: No
Generalization: SupportingInformation
Extension: Class

Description

A justification presents a statement of rationale.

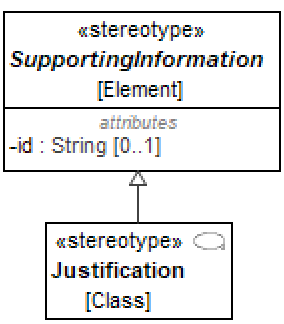


Figure 9.114 - Justification

InContextOf

Package: GSN Profile
isAbstract: No
Extension: Dependency

Description

InContextOf declares a contextual relationship.
Permitted connections are: goal-to-context, goal-to-assumption, goal-to-justification, strategy-to-context, strategy-to-assumption and strategy-to-justification.

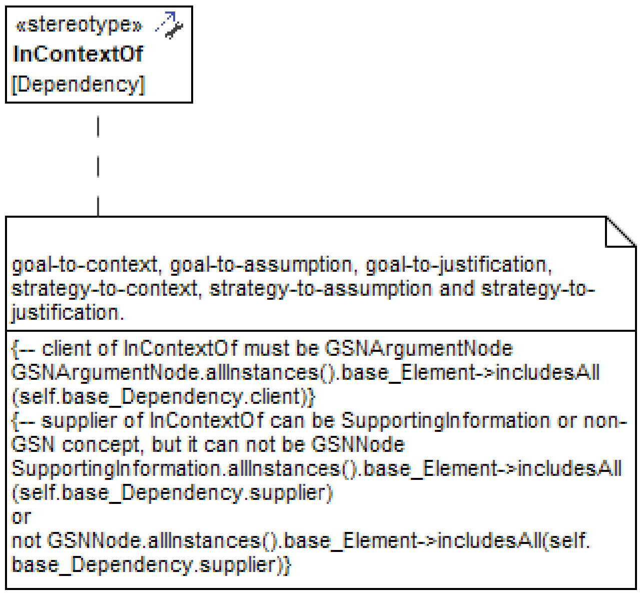


Figure 9.115 - InContextOf

Constraints

[1] ClientIsArgumentNode -- client of InContextOf must be GSNArgumentNode

GSNArgumentNode.allInstances().base_Element->includesAll(self.base_Dependency.client)

[2] SupplierIsNotGSNNode -- supplier of InContextOf can be SupportingInformation or non-GSN concept, but it can not be GSNNode

SupportingInformation.allInstances().base_Element->includesAll(self.base_Dependency.supplier)

or

not GSNNode.allInstances().base_Element->includesAll(self.base_Dependency.supplier)

SupportedBy

Package: GSN Profile
isAbstract: No
Extension: Dependency

Description

SupportedBy allows inferential or evidential relationships to be documented. Inferential relationships declare that there is an inferencebetween goals in the argument. Evidential relationships declare the link between a goal and the evidence used to substantiate it. Permitted supported by connections are: goal-to-goal, goal-to-strategy,goal-to-solution, strategy to goal.

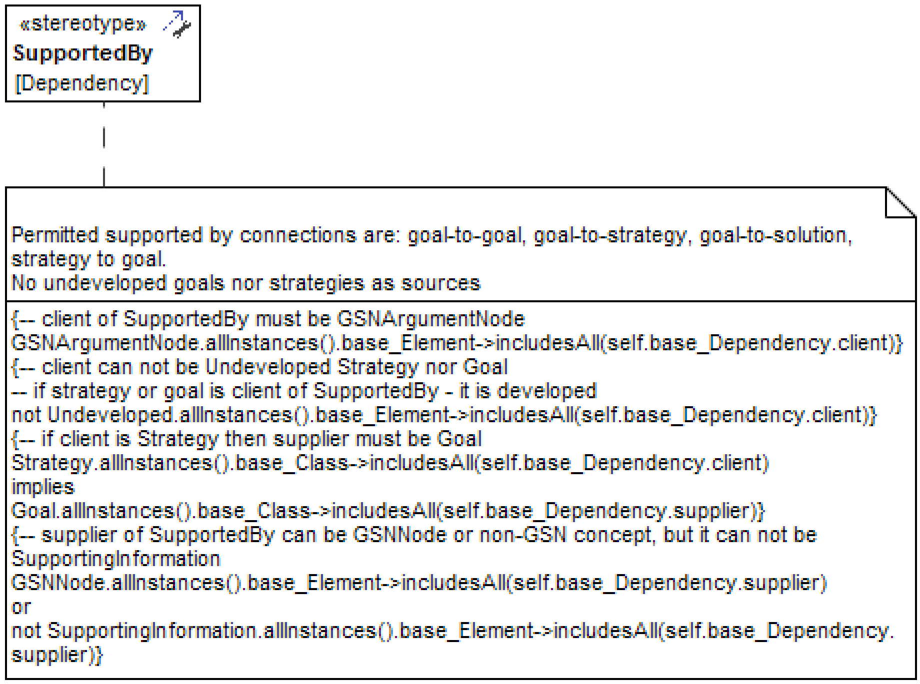


Figure 9.116 - SupportedBy

Constraints

[1] ClientIsGSNArgumentNode	-- client of SupportedBy must be GSNArgumentNode GSNArgumentNode.allInstances().base_Element->includesAll(self.base_Dependency.client)
[2] StrategyToGoal	-- if client is Strategy then supplier must be Goal Strategy.allInstances().base_Class->includesAll(self.base_Dependency.client) implies Goal.allInstances().base_Class->includesAll(self.base_Dependency.supplier)
[3] SupplierIsNotSupportingInformation	-- supplier of SupportedBy can be GSNNode or non-GSN concept, but it can not be SupportingInformation GSNNode.allInstances().base_Element->includesAll(self.base_Dependency.supplier) or not SupportingInformation.allInstances().base_Element->includesAll(self.base_Dependency.supplier)
[4] ClientIsNotUndeveloped	-- client can not be Undeveloped Strategy nor Goal -- if strategy or goal is client of SupportedBy - it is developed not Undeveloped.allInstances().base_Element->includesAll(self.base_Dependency.client)

9.7 Methods::ISO 26262

The ISO 26262 package contains elements supporting the analysis and requirement specification aspects of Functional Safety, as specified by ISO 26262 standard for automotive applications. ISO 26262 is a risk based standard derived from IEC 61508. The ISO 26262 package redefines or extends concepts from the Core concepts package and the General Concepts package.

The ISO 26262 package enables modeling a HAZOP, which is typically used to identify malfunctioning behaviors. The failure modes concept is used from the General Concepts and specialized as a malfunctioning behavior. This allows the malfunctioning behavior to be related to the system behaviors through the HAZOP guidewords for construction of the HAZOP table. The risk analysis is performed by identifying Hazards that could result from the MalfunctioningBehavior, which in combination with a particular OperationalSituation could result in an AccidentScenario. This information is contained in the HazardousEvent which provides the risk level assessment for the event. Each of these concepts are modeled using elements defined in the ISO 26262 package as extensions of the Core and General concepts. This means that the same elements can be used in other analyses in the model, such as in an FMEA.

9.7.1 Methods::ISO 26262::ISO 26262 Library

TrafficAndPeople

Package: ISO 26262 Library
isAbstract: Yes
Generalization: OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

TrafficAndPeople extends the <<situation>> class, and is used to describe the presence and behaviour of any motorists or non-motorists considered in a hazardous event.

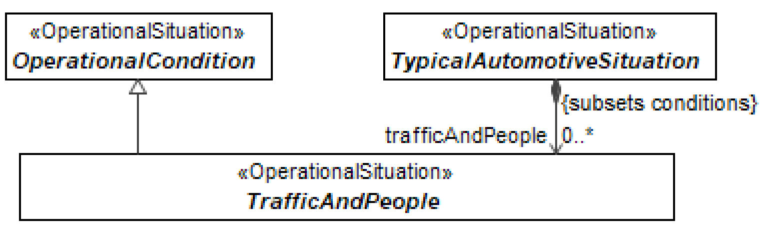


Figure 9.117 - TrafficAndPeople

VehicleUsage

Package: ISO 26262 Library
isAbstract: Yes
Generalization: OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

VehicleUsage extends the <<situation>> class, and is used to describe the usage of a vehicle during a hazardous event.

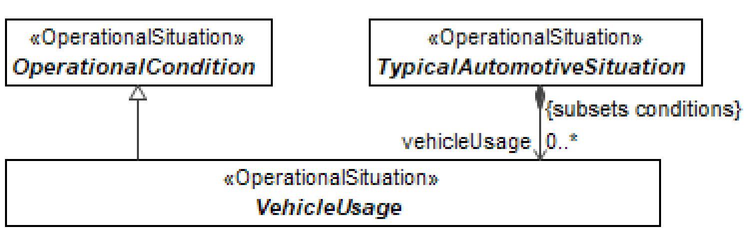


Figure 9.118 - VehicleUsage

RoadCondition

Package: ISO 26262 Library
isAbstract: Yes
Generalization: OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

RoadConditions extends the <<situation>> class, and is used to describe the conditions or state of the surface a vehicle is driving on (Low-traction, Grade(Slope), etc.) during a hazardous event.

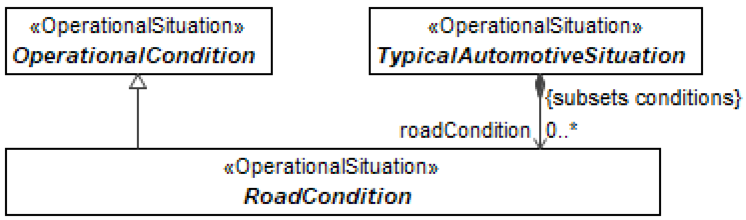


Figure 9.119 - RoadCondition

Location

Package: ISO 26262 Library
isAbstract: Yes
Generalization: OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

VehicleLocation extends the <<situation>> class, and is used to describe the physical location (high speed road, intersection, parking lot, etc.) of a vehicle during a hazardous event.

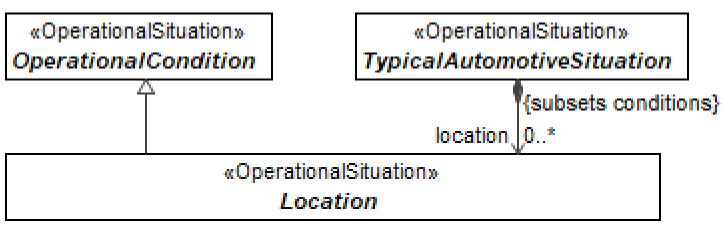


Figure 9.120 - Location

EnvironmentalCondition

Package: ISO 26262 Library
isAbstract: Yes
Generalization: OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

EnvironmentalConditions extends the <<situation>> class, and is used to describe the environmental conditions at the time of vehicle operation in a hazardous event.

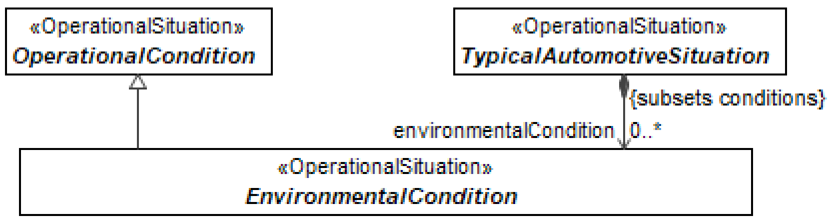


Figure 9.121 - EnvironmentalCondition

OperationalCondition

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnySituation
Applied Stereotype: «OperationalSituation»

Description

Component/part of operational situation.

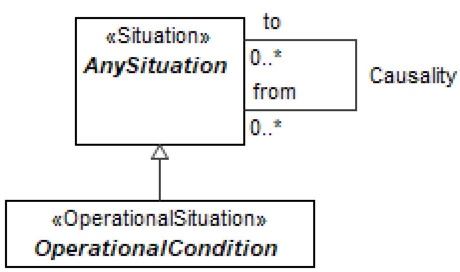


Figure 9.122 - OperationalCondition

AbstractOperationalSituation

Package: ISO 26262 Library
isAbstract: Yes
Generalization: OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

Operational situation is a scenario that can occur in vehicle's life.

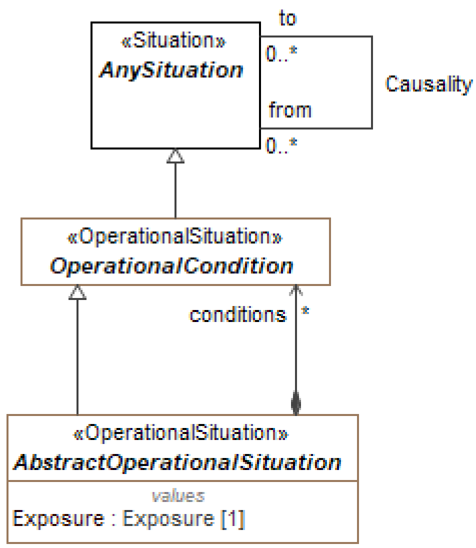


Figure 9.123 - AbstractOperationalSituation

Attributes

conditions : OperationalCondition[*] (member end of association)	
Exposure : Exposure[1]	Likelihood of being in a particular operational situation. Must have a Rationale attached.

TypicalAutomotiveSituation

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AbstractOperationalSituation
Applied Stereotype: «OperationalSituation»

Description

A grouping of operational conditions, including traffic and people, vehicle usage, road conditions, location, and environmental conditions.

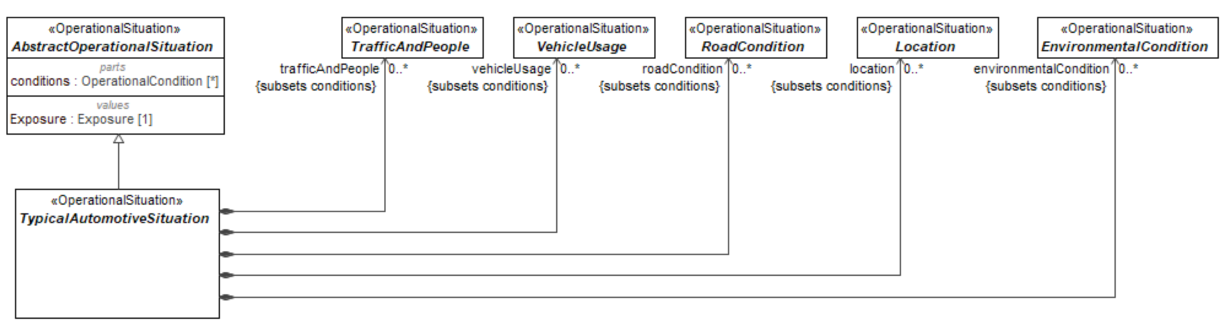


Figure 9.124 - TypicalAutomotiveSituation

Attributes

trafficAndPeople : TrafficAndPeople[0..*]
(member end of association, subsets conditions)

vehicleUsage : VehicleUsage[0..*] (member end of association, subsets conditions)

roadCondition : RoadCondition[0..*] (member end of association, subsets conditions)

location : Location[0..*] (member end of association, subsets conditions)

environmentalCondition : EnvironmentalCondition[0..*] (member end of association, subsets conditions)

Exposure

Package: ISO 26262 Library
isAbstract: No
Applied Stereotype: «ValueType»

Description

Possible values of exposure.

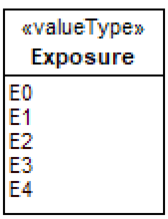


Figure 9.125 - Exposure

Severity

Package: ISO 26262 Library
isAbstract: No
Applied Stereotype: «ValueType»

Description

Possible values for severity.

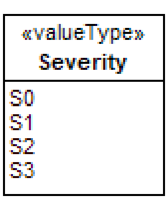


Figure 9.126 - Severity

ASIL

Package: ISO 26262 Library
isAbstract: No
Applied Stereotype: «ValueType»

Description

Possible ASIL values.

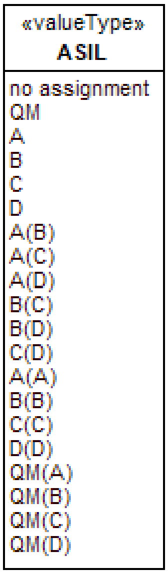


Figure 9.127 - ASIL

Controllability

Package: ISO 26262 Library
isAbstract: No
Applied Stereotype: «ValueType»

Description

Possible values of controllability.

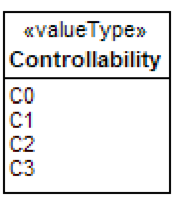


Figure 9.128 - Controllability

Less

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from providing less output/behaviour than required.

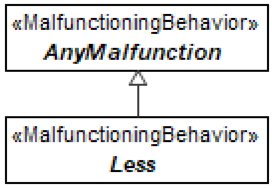


Figure 9.129 - Less

More

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from providing more output/behaviour than required.

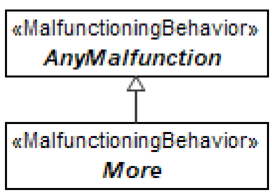


Figure 9.130 - More

No

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from the behaviour not being performed when required.

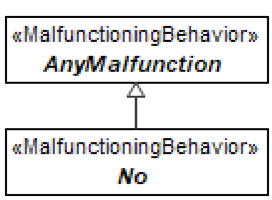


Figure 9.131 - No

Intermittent

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure from the behaviour being performed intermittently.

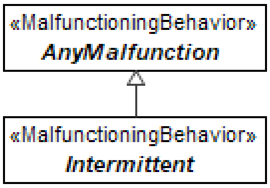


Figure 9.132 - Intermittent

Unintended

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from the behaviour being provided when not required.

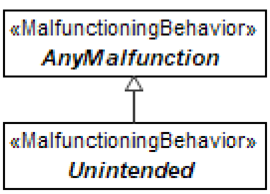


Figure 9.133 - Unintended

Early

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from the behaviour being performed earlier than required.

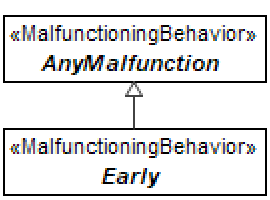


Figure 9.134 - Early

Late

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from the behaviour being performed later than required.

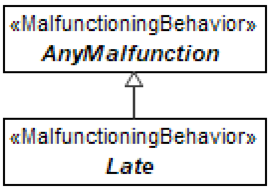


Figure 9.135 - Late

Inverted

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AnyMalfunction
Applied Stereotype: «MalfunctioningBehavior»

Description

A subclass of malfunctioning behaviour used for classification purposes. Must be connected to a behavioural element (Use Case or Function). This kind of malfunctioning behaviour represents a failure resulting from the behaviour providing an inverted output.

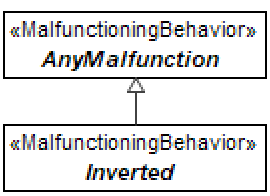


Figure 9.136 - Inverted

HazardousEvent

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AbstractRisk
Applied Stereotype: «Situation»

Description

Combination of hazard and operational situation to identify automotive safety integrity level.

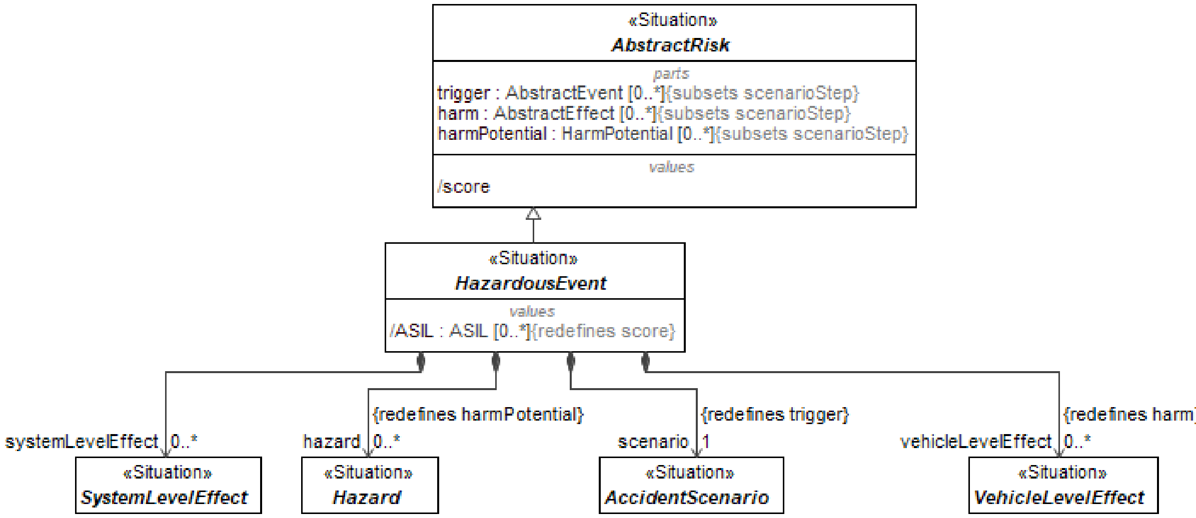


Figure 9.137 - HazardousEvent

Attributes

scenario : AccidentScenario[1] (member end of association, redefines trigger)	
hazard : Hazard[0..*] (member end of association, redefines harmPotential)	
systemLevelEffect : SystemLevelEffect[0..*] (member end of association)	
vehicleLevelEffect : VehicleLevelEffect[0..*] (member end of association, redefines harm)	
ASIL : ASIL[0..*], redefines score	Automotive Safety Integrity Level value - one of four levels to specify necessary requirements for ISO-26262 and safety measures for avoiding unreasonable risks.

AnyMalfunction

Package: ISO 26262 Library
isAbstract: Yes
Generalization: UndesiredState
Applied Stereotype: «MalfunctioningBehavior»

Description

Root of all malfunctioning behaviours.

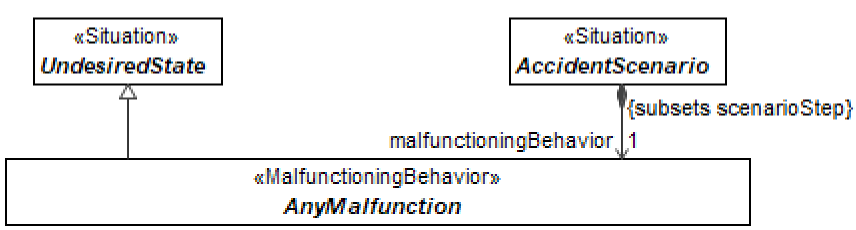


Figure 9.138 - AnyMalfunction

AutomotiveEffect

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AbstractEffect
Applied Stereotype: «Situation»

Description

System- or vehicle-level effect which is or could result in harm.

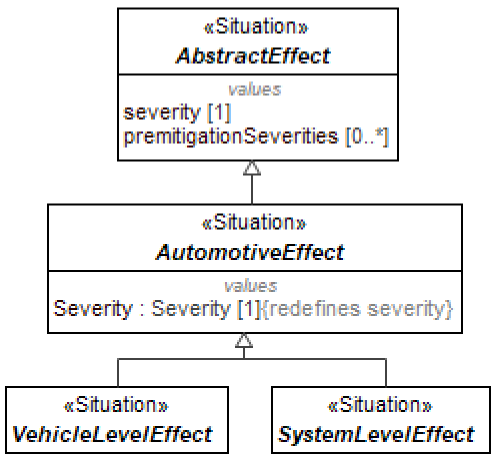


Figure 9.139 - AutomotiveEffect

Attributes

Severity : Severity[1], redefines severity	Estimate of the extent of harm. Must have a Rationale attached.

ISO26262SafetyRequirementTemplate

Package: ISO 26262 Library
isAbstract: No
Applied Stereotype: «DependabilityRequirement»

Description

A template for dependability requirements.

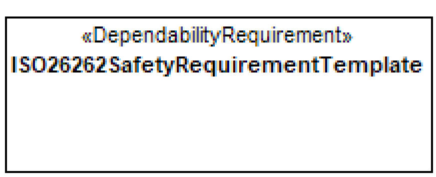


Figure 9.140 - ISO26262SafetyRequirementTemplate

Attributes

ASIL : ASIL[1]	ASIL value of the requirement.
FTTI : time[1]	Fault Tolerant Time Interval.

AccidentScenario

Package: ISO 26262 Library
isAbstract: Yes

Generalization: DysfunctionalEvent, Scenario
Applied Stereotype: «Situation»

Description

A combination of operational situation and malfunctioning behaviour.

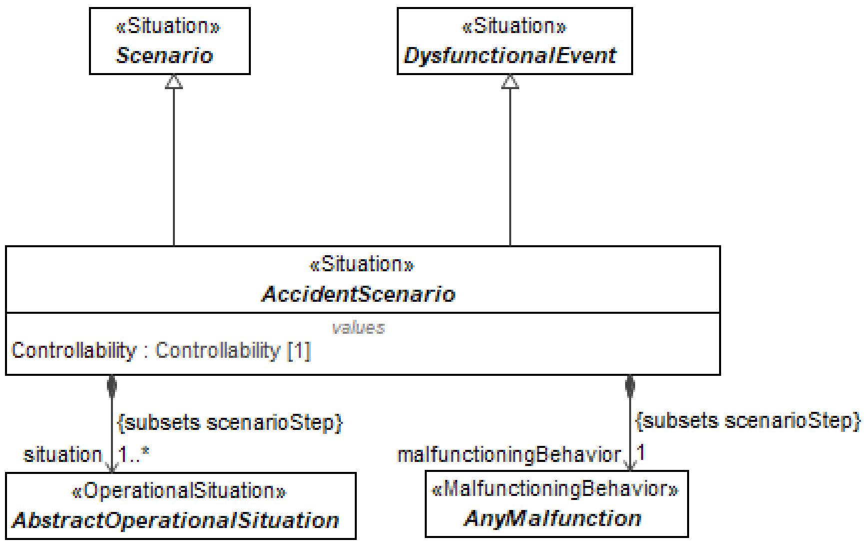


Figure 9.141 - AccidentScenario

Attributes

situation : AbstractOperationalSituation[1..*] (member end of association, subsets scenarioStep)	
Controllability : Controllability[1]	Ability to avoid a specified harm or damage through timely reactions of individuals involved in the scenario. Must have a Rationale attached.
malfunctioningBehavior : AnyMalfunction[1] (member end of association, subsets scenarioStep)	

AnyTrafficAndPeople

Package: ISO 26262 Library
isAbstract: No
Generalization: OperationalCondition, TrafficAndPeople
Applied Stereotype: «OperationalSituation»

Description

TrafficAndPeople extends the <<situation>> class, and is used to describe the presence and behaviour of any motorists or non-motorists considered in a hazardous event.

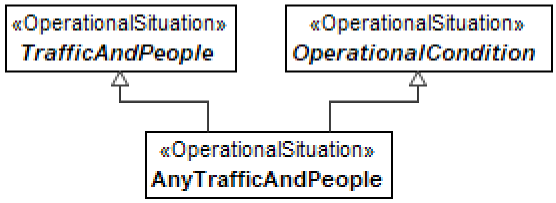


Figure 9.142 - AnyTrafficAndPeople

AnyVehicleUse

Package: ISO 26262 Library
isAbstract: No
Generalization: OperationalCondition, VehicleUsage
Applied Stereotype: «OperationalSituation»

Description

TrafficAndPeople extends the <<situation>> class, and is used to describe the presence and behaviour of any motorists or non-motorists considered in a hazardous event.

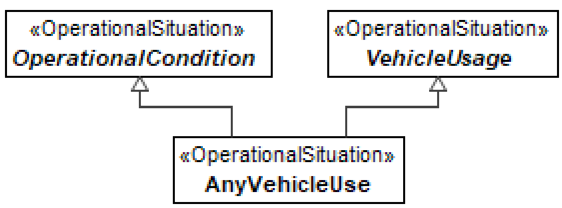


Figure 9.143 - AnyVehicleUse

AnyRoadCondition

Package: ISO 26262 Library
isAbstract: No
Generalization: OperationalCondition, RoadCondition
Applied Stereotype: «OperationalSituation»

Description

TrafficAndPeople extends the <<situation>> class, and is used to describe the presence and behaviour of any motorists or non-motorists considered in a hazardous event.

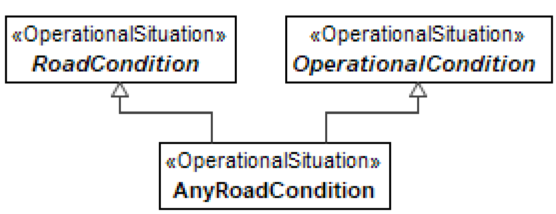


Figure 9.144 - AnyRoadCondition

AnyLocation

Package: ISO 26262 Library
isAbstract: No
Generalization: Location, OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

TrafficAndPeople extends the <<situation>> class, and is used to describe the presence and behavior of any motorists or non-motorists considered in a hazardous event.

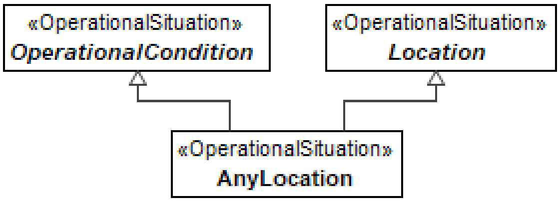


Figure 9.145 - AnyLocation

AnyEnvironmentalCondition

Package: ISO 26262 Library
isAbstract: No
Generalization: EnvironmentalCondition, OperationalCondition
Applied Stereotype: «OperationalSituation»

Description

TrafficAndPeople extends the <<situation>> class, and is used to describe the presence and behaviour of any motorists or non-motorists considered in a hazardous event.

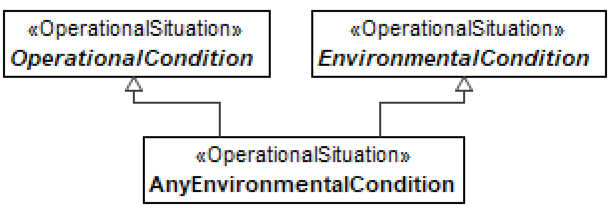


Figure 9.146 - AnyEnvironmentalCondition

SystemLevelEffect

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AutomotiveEffect
Applied Stereotype: «Situation»

Description

System- or vehicle-level effect which is or could result in harm.

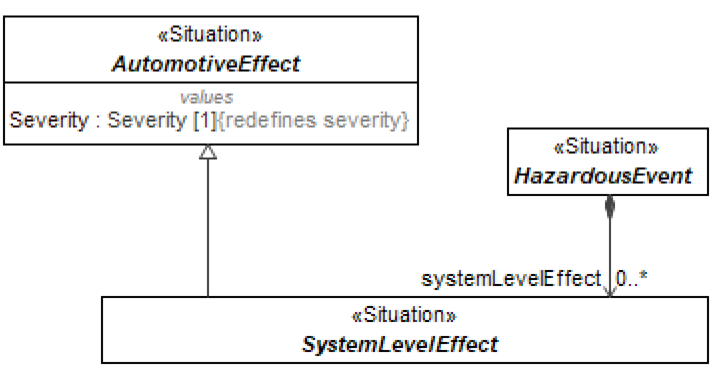


Figure 9.147 - SystemLevelEffect

VehicleLevelEffect

Package: ISO 26262 Library
isAbstract: Yes
Generalization: AutomotiveEffect
Applied Stereotype: «Situation»

Description

System- or vehicle-level effect which is or could result in harm.

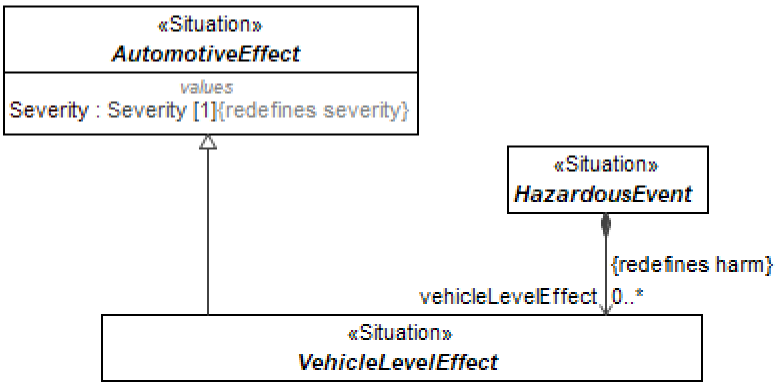


Figure 9.148 - VehicleLevelEffect

Methods::ISO 26262::ISO 26262 Library::Diagrams by elements

9.7.2 Methods::ISO 26262::ISO 26262 Profile

OperationalSituation

Package: ISO 26262 Profile
isAbstract: No
Generalization: Situation
Extension: Class

Description

A situation describes the operational scenario or driving scenario which is considered in a hazardous event, as part of the Hazard Analysis and Risk Assessment process.

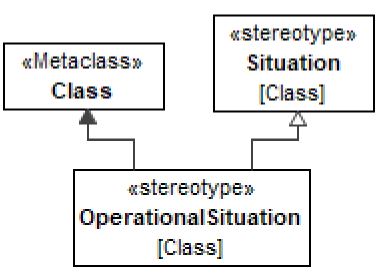


Figure 9.149 - OperationalSituation

MalfunctioningBehavior

Package: ISO 26262 Profile
isAbstract: No
Generalization: FailureMode
Extension: Class

Description

A malfunctioning behaviour describes a failure or unintended behaviour of an item with respect to its design intent. It is a subtype of failure mode.

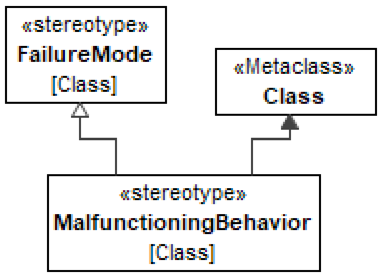


Figure 9.150 - MalfunctioningBehavior

Methods::ISO 26262::ISO 26262 Profile::RequirementManagement

IndependenceRequirement

Package: RequirementManagement
isAbstract: No
Generalization: DeriveReqt
Extension: Abstraction

Description

A relationship between requirement elements indicating that the child requirement specifies an independence criterion that needs to be satisfied in order for an ASIL decomposition to be valid. The decomposition between the parent requirement and 2 other children requirements.

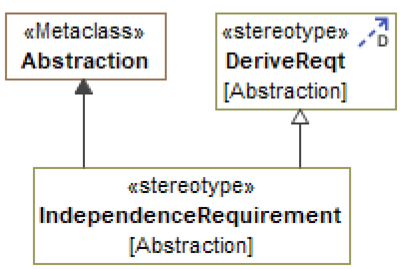


Figure 9.151 - IndependenceRequirement

ASILDecompose

Package: RequirementManagement
isAbstract: No
Generalization: DeriveReqt
Extension: Abstraction

Description

An ASIL decompose relation is used to connect two safety requirements for the purposes of performing ASIL decomposition. The target requirement (supplier) should be of a higher abstraction than the source (client). ASIL decompose relations shall be applied in pairs (e.g. a requirement cannot be the supplier of a single ASIL decompose relation).

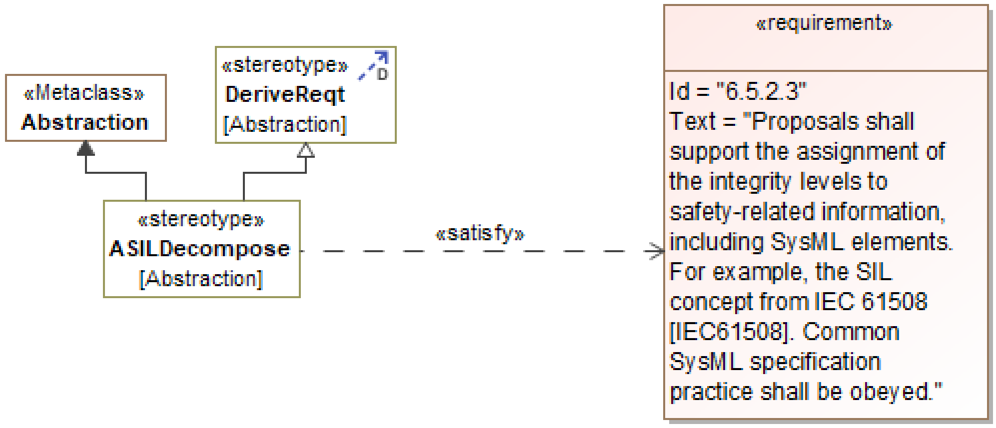


Figure 9.152 - ASILDecompose

SafeState

Package: RequirementManagement
isAbstract: No
Extension: Dependency

Description

A state of function realized by one or more architectural components. May be composed of serval subfunctions or called by other functions. Associated with safety specific behaviours, typically (but not necessarily) triggered by a failure mode.

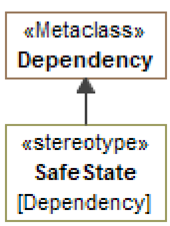


Figure 9.153 - SafeState

UserInfoRequirement

Package: RequirementManagement
isAbstract: No
Generalization: Satisfy
Extension: Abstraction

Description

A UserInfoRequirement relationship is a dependency which links a State to a requirement. The arrow direction points from a state (client) to a FSR or TSR (supplier). Linked requirements specify information that must be presented to vehicle occupants when the vehicle enters a safe state.

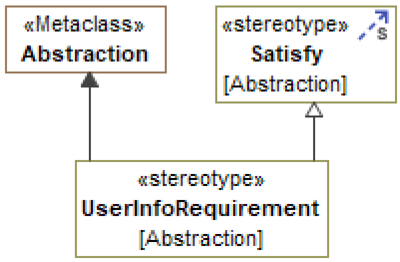


Figure 9.154 - UserInfoRequirement

RecoveryRequirement

Package: RequirementManagement
isAbstract: No
Generalization: Satisfy
Extension: Abstraction

Description

A RecoveryRequirement relationship is a dependency between a safe state and requirement where the requirement indicates the criteria to recover from the safe state to another operational mode.

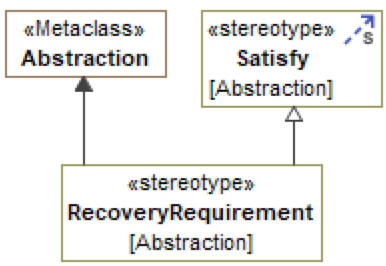


Figure 9.155 - RecoveryRequirement

OperatingMode

Package: RequirementManagement
isAbstract: No
Extension: Dependency

Description

A state of function realized by one or more architectural components. May be composed of serval subfunctions or called by other functions. Associated with specific behaviours.

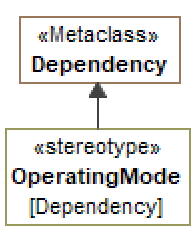


Figure 9.156 - OperatingMode

FunctionalSafetyRequirement

Package: RequirementManagement
isAbstract: No

Generalization: DependabilityRequirement, Requirement
Extension: Class

Description

A functional safety requirement specifies an implementation independent safety behaviour, or an implementation independent safety measure, required for achievement of a safety goal from which it is derived.

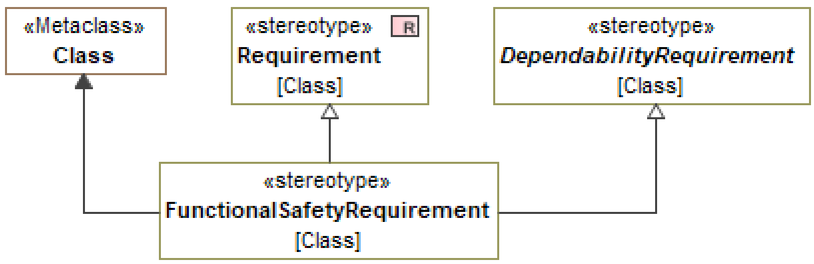


Figure 9.157 - FunctionalSafetyRequirement

SoftwareSafetyRequirement

Package: RequirementManagement
isAbstract: No
Generalization: DependabilityRequirement, Requirement
Extension: Class

Description

A software safety requirement provides implementation details for software. They can express behaviours or specific software mechanisms which realize the technical safety requirements from which they are derived.

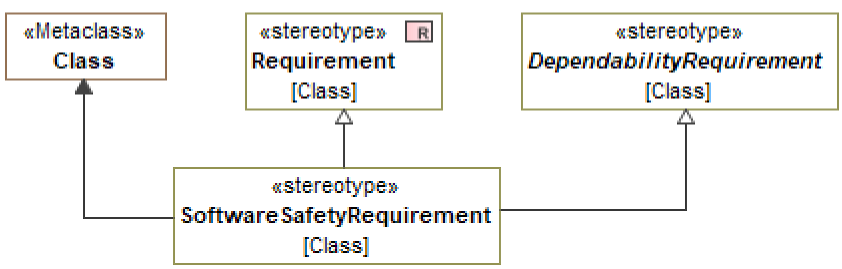


Figure 9.158 - SoftwareSafetyRequirement

HardwareSafetyRequirement

Package: RequirementManagement
isAbstract: No
Generalization: DependabilityRequirement, Requirement
Extension: Class

Description

A hardware safety requirement specifies hardware behaviours or hardware specific details necessary for implementing the safety concept. Hardware safety requirements are implementation specific and assigned to components or subcomponents.

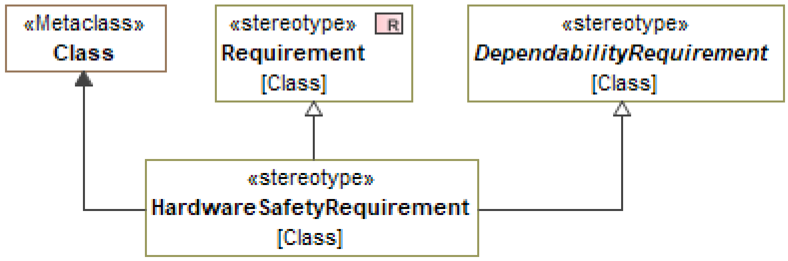


Figure 9.159 - HardwareSafetyRequirement

TechnicalSafetyRequirement

Package: RequirementManagement
isAbstract: No
Generalization: DependabilityRequirement, Requirement
Extension: Class

Description

A technical safety requirement specifies the implementation of the functional safety requirement(s) from which it is derived. Technical safety requirements express the behaviours and details necessary to realize the safety aspects of the item at the system level. Additional details that do not act at the system level can be specified in the hardware safety requirements or software safety requirements.

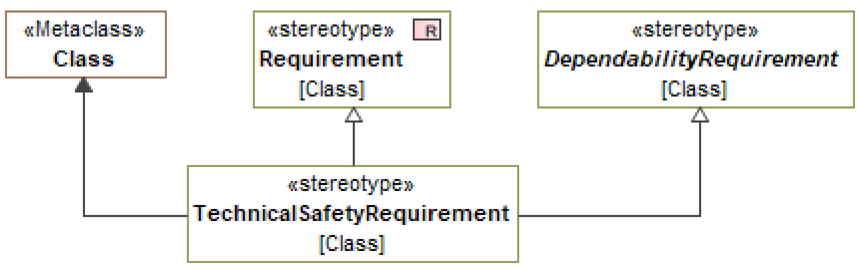


Figure 9.160 - TechnicalSafetyRequirement

SafetyGoal

Package: RequirementManagement
isAbstract: No
Generalization: DependabilityRequirement, Requirement
Extension: Class

Description

A safety goal extends the SysML <<Requirement>> stereotype. It represents a top-level safety requirement, defined as a result of the Hazard Analysis and Risk Assessment process.

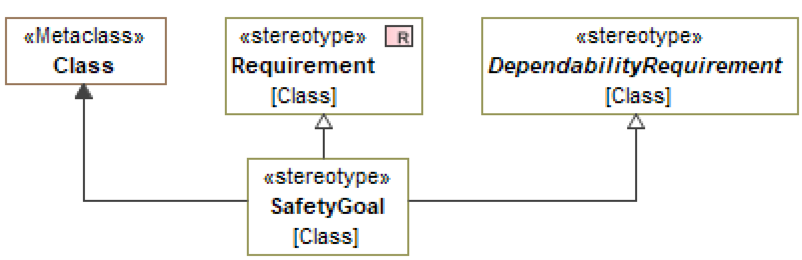


Figure 9.161 - SafetyGoal

DependabilityRequirement

Package: RequirementManagement
isAbstract: Yes
Generalization: AbstractRequirement, Block
Extension: Class

Description

Parent type of all subtypes of safety requirements

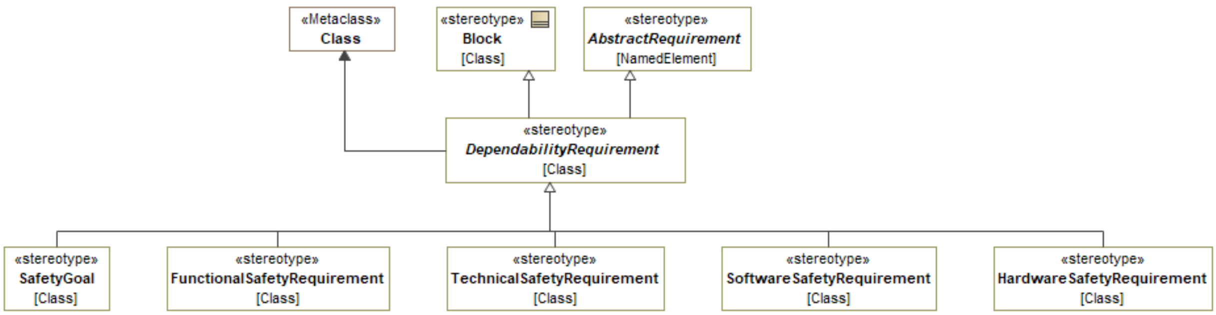


Figure 9.162 - DependabilityRequirement

Verified

Package: ISO 26262 Profile
isAbstract: No
Extension: Class

Description

Marker, indicating that hazardous event has been verified.

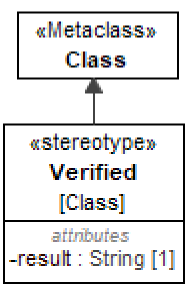


Figure 9.163 - Verified

Attributes

result : String[1]	Verification result

Confirmed

Package: ISO 26262 Profile
isAbstract: No
Extension: Class

Description

Marker, indicating that hazardous event has been confirmed.

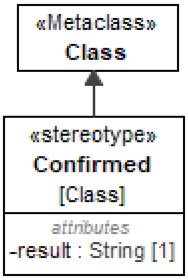


Figure 9.164 - Confirmed

Attributes

result : String[1]	Confirmation result

HazardAndRiskAssessment

Package: ISO 26262 Profile
isAbstract: No
Extension: Package

Description

Grouping package for storing hazardous events.

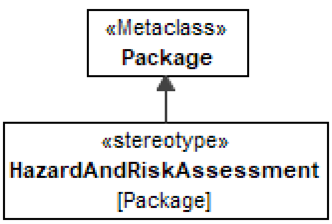


Figure 9.165 - HazardAndRiskAssessment

IDCarrier

Package: ISO 26262 Profile
isAbstract: No
Extension: Element

Description

Additional stereotype for carrying human-readable identification data.

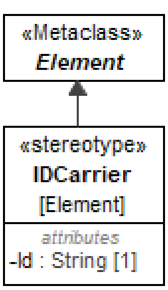


Figure 9.166 - IDCarrier

Attributes

Id : String[1]	Human readable identifier.

LessonLearned

Package: ISO 26262 Profile
isAbstract: No
Extension: Comment

Description

Comments about lessons learned from hazard and risk assessment.

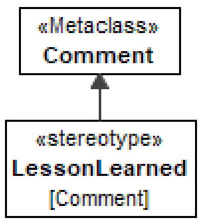


Figure 9.167 - LessonLearned

ASILAssignment

Package: ISO 26262 Profile
isAbstract: No
Extension: Element

Description

Stereotype for assigning ASIL values on system design elements.

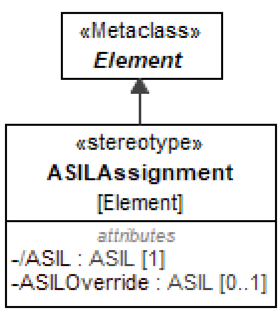


Figure 9.168 - ASILAssignment

Attributes

ASIL : ASIL[1]	The associated ASIL value of the system design element.
ASILOverride : ASIL[0..1]	An ASIL value which does not follow from the normal ASIL derivation rules, but is exceptional. This exceptional value needs to have an associated rationale.

ASILOverrideRationale

Package: ISO 26262 Profile
isAbstract: No
Generalization: Rationale
Extension: Comment

Description

A rationale specifically justifying ASIL Override value.

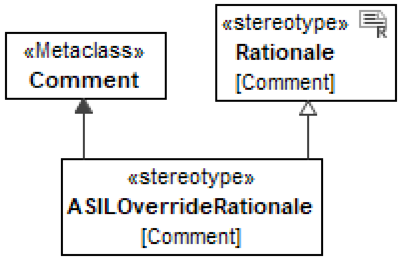


Figure 9.169 - ASILOverrideRationale

10. Views

10.1 Core

10.1.1 Core::Core Library

View Core::Core Library::Core Library

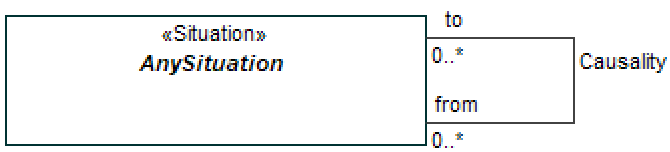


Figure 10.1 – Core Library

Elements

- AnySituation
- Causality

10.1.2 Core::Core Profile

View Core::Core Profile::CoreProfile

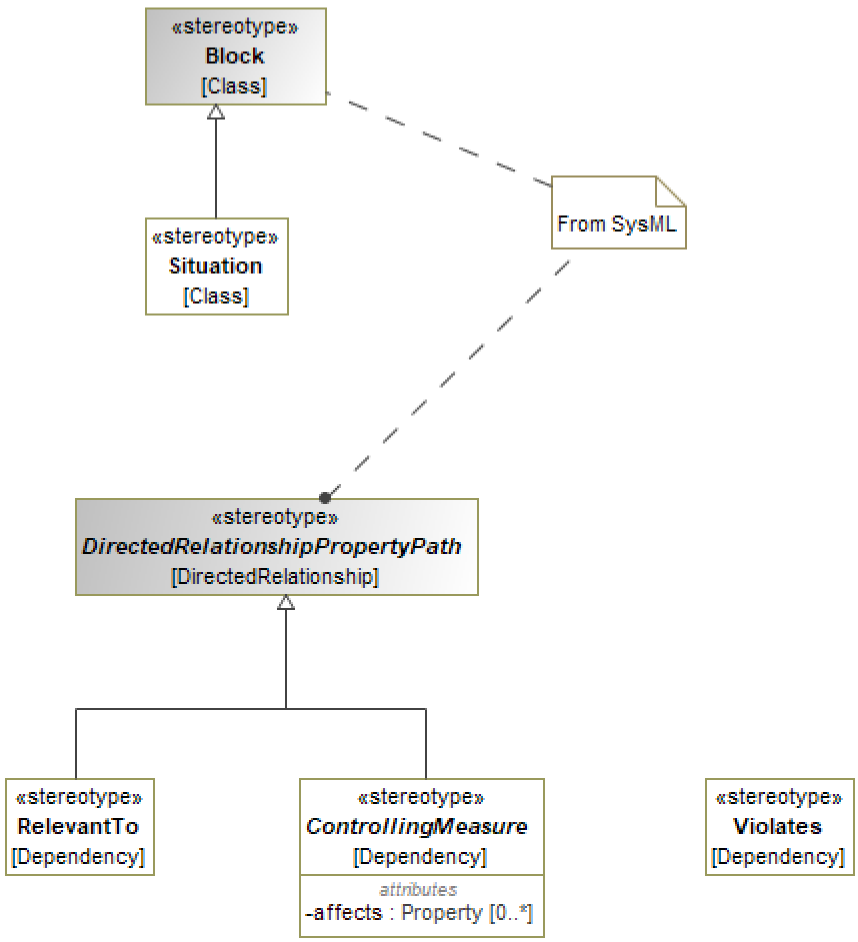


Figure 10.2 - CoreProfile

Elements

- ControllingMeasure
- RelevantTo
- Situation
- Violates

10.2 General

10.2.1 General::General Concepts Library

View General::General Concepts Library::General Concepts Library

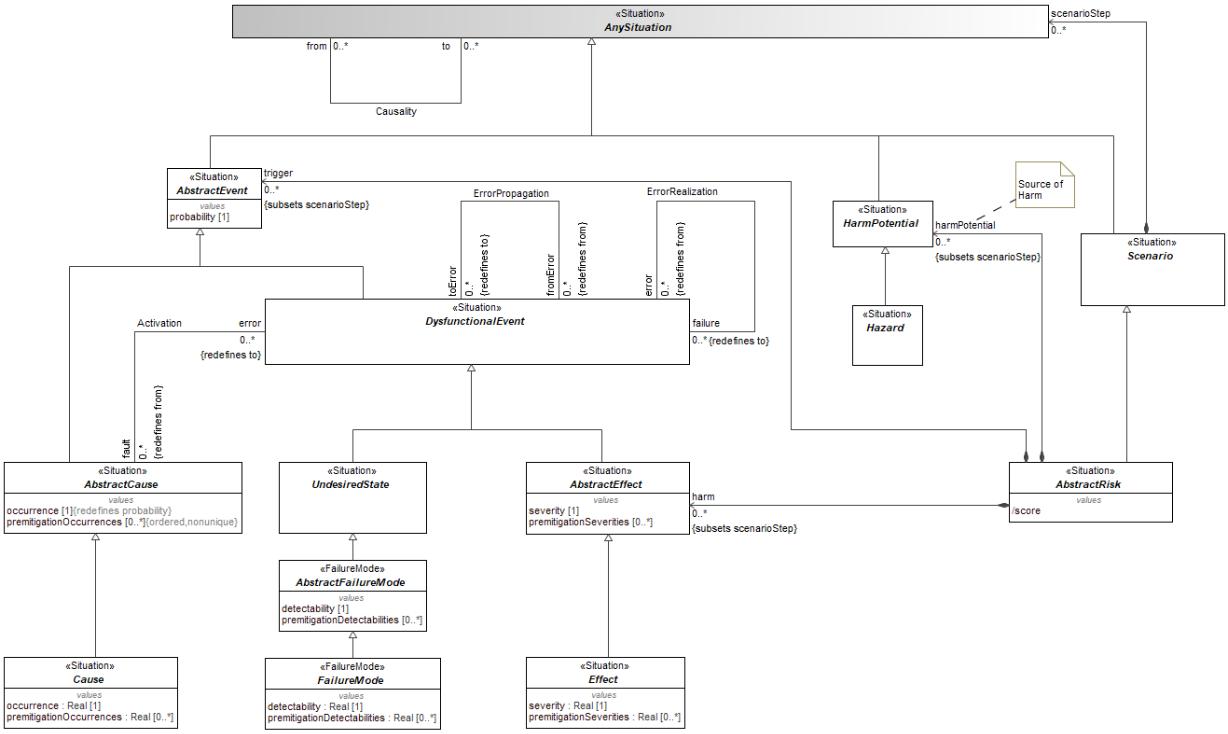


Figure 10.3 - General Concepts Library

Elements

- AbstractCause
- AbstractEffect
- AbstractEvent
- AbstractFailureMode
- AbstractRisk
- Activation
- AnySituation
- Causality
- Cause
- DysfunctionalEvent
- Effect
- ErrorPropagation
- ErrorRealization
- FailureMode
- HarmPotential
- Hazard
- Scenario
- UndesiredState

10.2.2 General::General Concepts Profile

View General::General Concepts Profile::General Concepts Profile

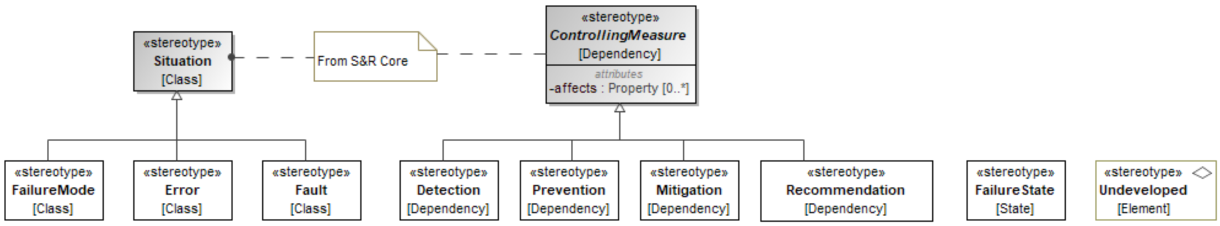


Figure 10.4 - General Concepts Profile

Elements

- ControllingMeasure
- Detection
- Error
- FailureMode
- FailureState
- Fault
- Mitigation
- Prevention
- Recommendation
- Situation
- Undeveloped

10.3 Methods::FMEA

10.3.1 Methods::FMEA::FMEA Library

View Methods::FMEA::FMEA Library::FMEA Library

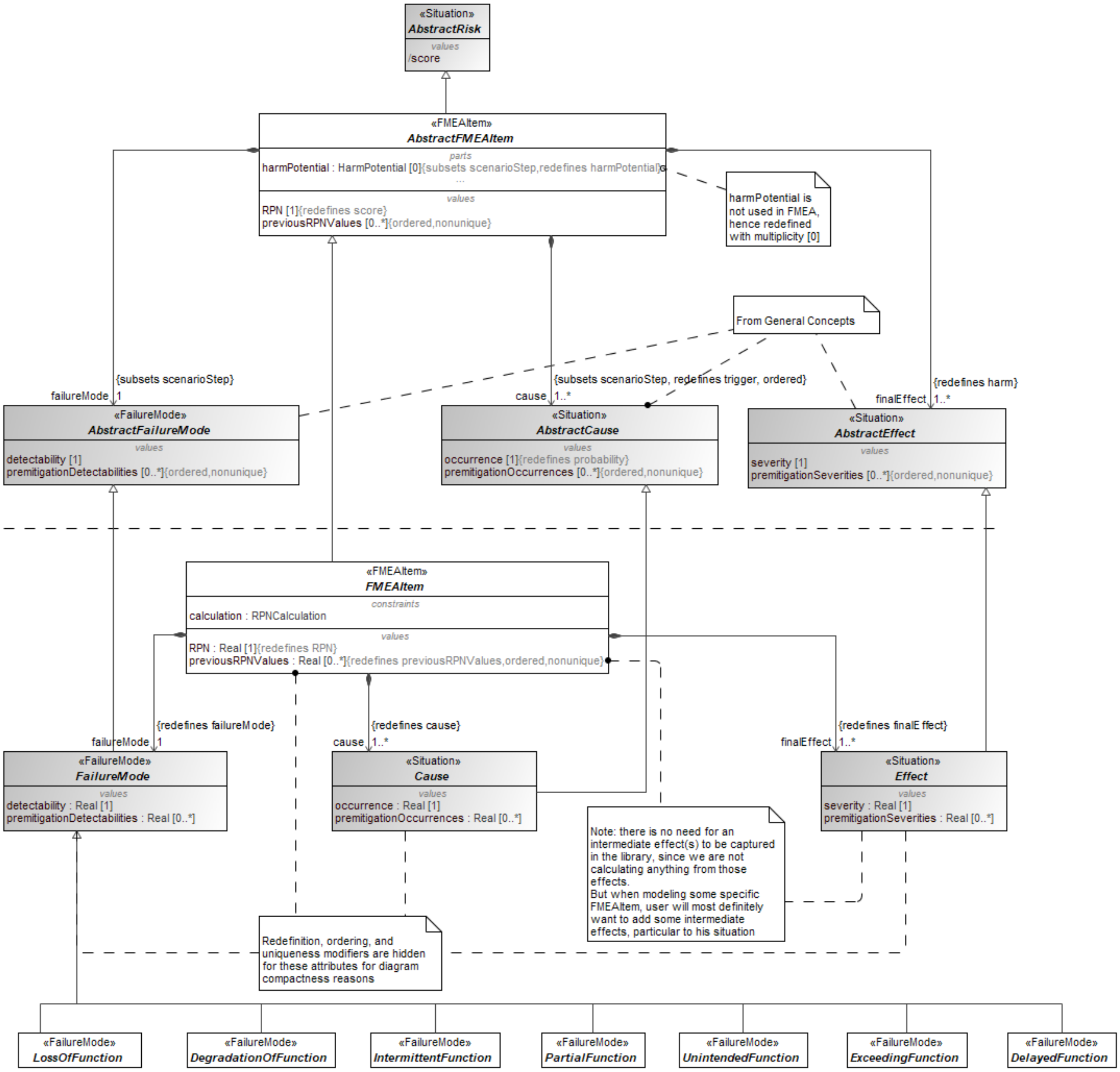


Figure 10.5 - FMEA Library

Elements

- AbstractCause
- AbstractEffect
- AbstractFailureMode
- AbstractFMEAItem
- AbstractRisk
- Cause
- DegradationOfFunction
- DelayedFunction
- Effect

- ExceedingFunction
- FailureMode
- FMEAItem
- IntermittentFunction
- LossOfFunction
- PartialFunction
- UnintendedFunction

10.3.2 Methods::FMEA::FMEA Profile

View Methods::FMEA::FMEA Profile::FMEA Profile

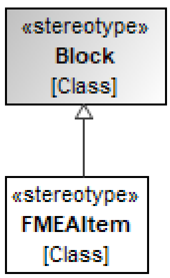


Figure 10.6 - FMEA Profile

Elements

- FMEAItem

10.4 Methods::FTA

10.4.1 Methods::FTA::FTALibrary

Methods::FTA::FTALibrary::Events

View Methods::FTA::FTALibrary::Events::Events

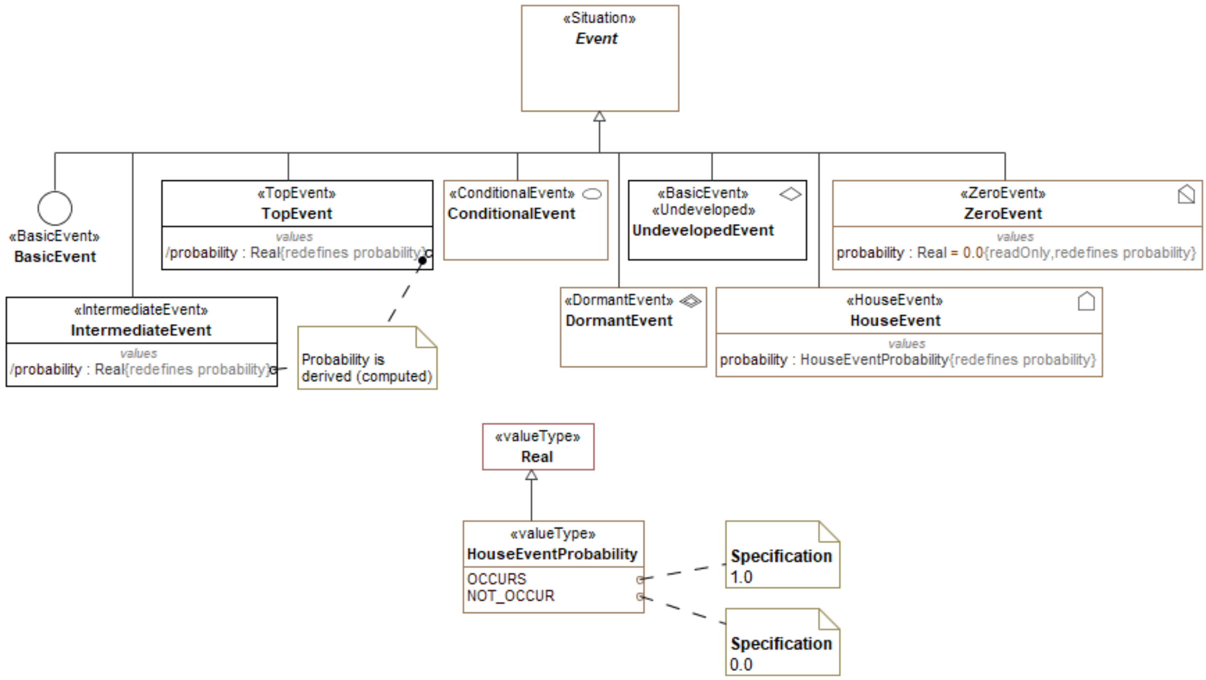


Figure 10.7 - Events

Elements

- BasicEvent
- ConditionalEvent
- DormantEvent
- Event
- HouseEvent
- IntermediateEvent
- TopEvent
- UndevelopedEvent
- ZeroEvent

View Methods::FTA::FTALibrary::FTA Library

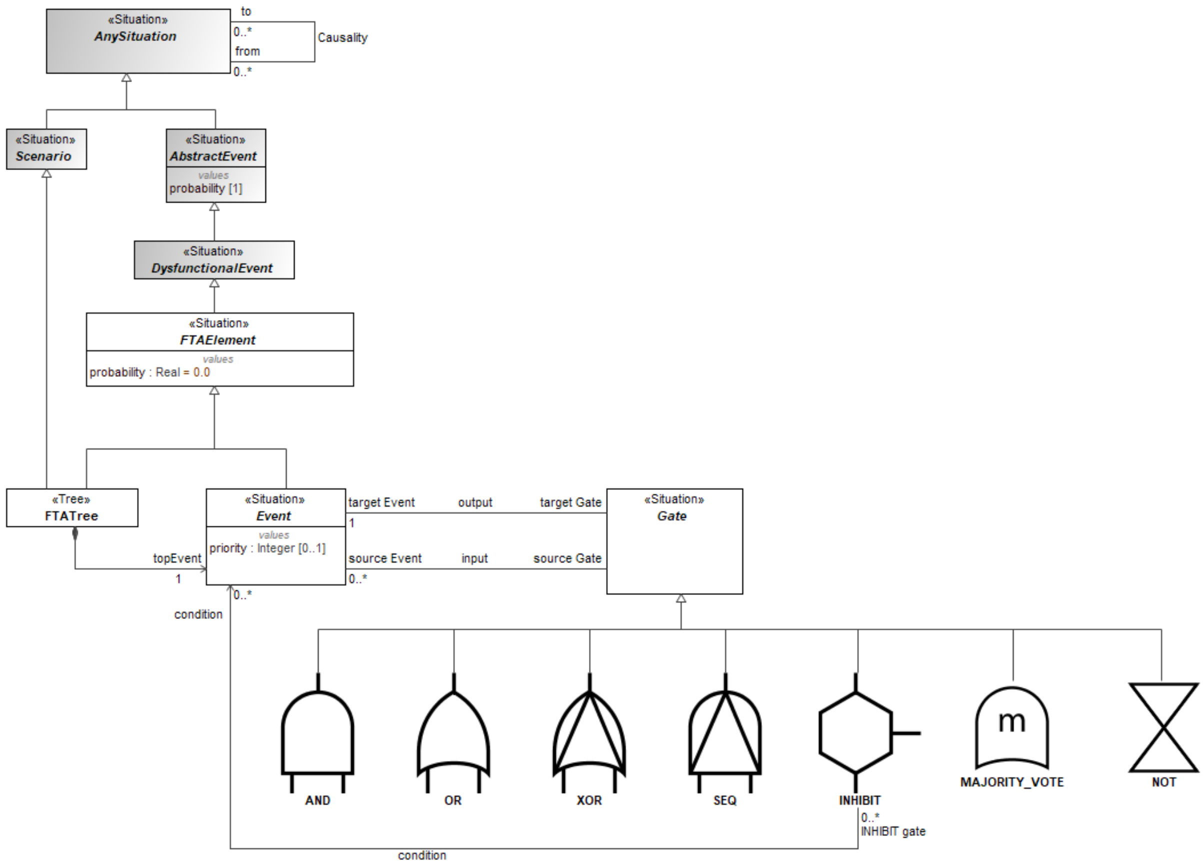


Figure 10.8 - FTA Library

Elements

- AbstractEvent
- AND
- AnySituation
- Causality
- DysfunctionalEvent
- Event
- FTAElement
- FTATree
- Gate
- INHIBIT
- MAJORITY_VOTE
- NOT
- OR
- Scenario
- SEQ
- XOR

10.4.2 Methods::FTA::FTAProfile

Methods::FTA::FTAProfile::Diagrams by elements

View Methods::FTA::FTAProfile::FTA Profile

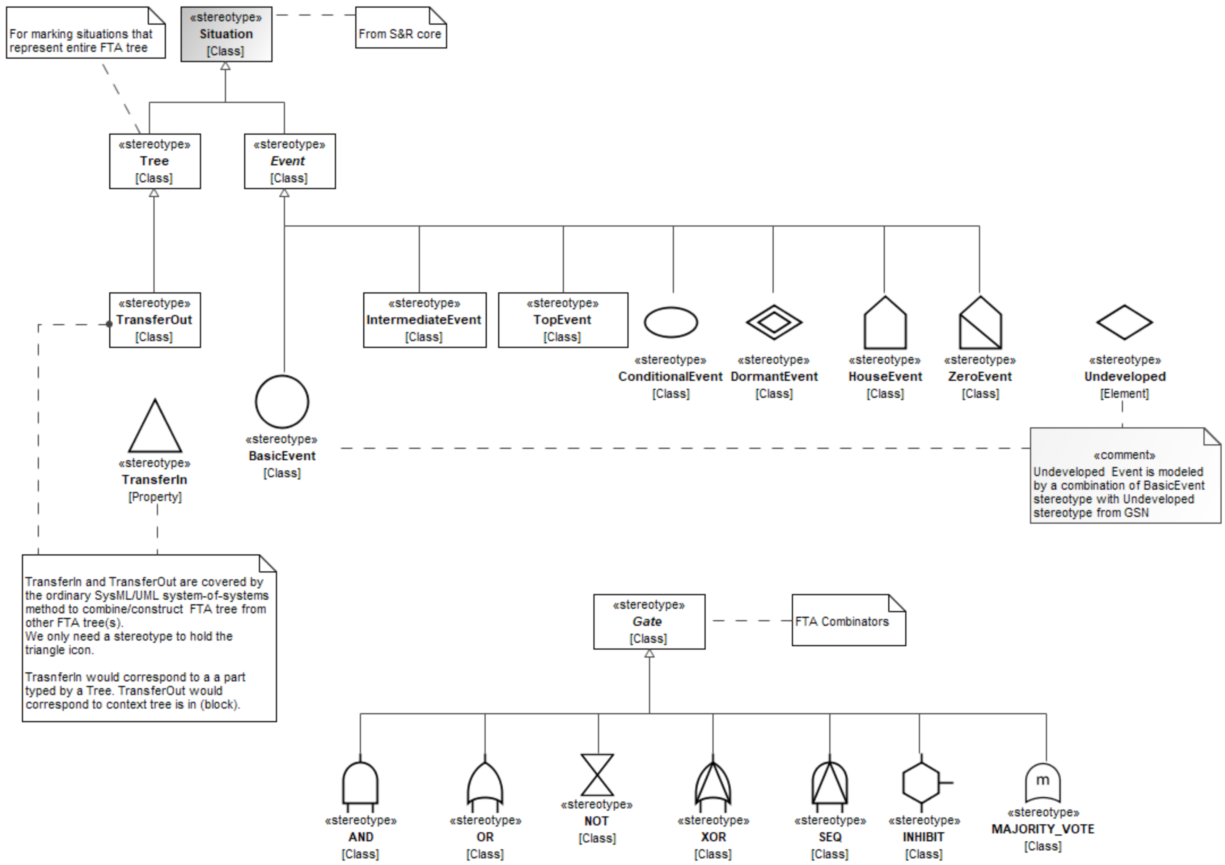


Figure 10.9 - FTA Profile

Elements

- AND
- BasicEvent
- ConditionalEvent
- DormantEvent
- Event
- Gate
- HouseEvent
- INHIBIT
- IntermediateEvent
- MAJORITY_VOTE
- NOT
- OR
- SEQ
- Situation

- TopEvent
- TransferIn
- TransferOut
- Tree
- Undeveloped
- XOR
- ZeroEvent

10.5 Methods::STPA

10.5.1 Methods::STPA::STPA Library

View Methods::STPA::STPA Library::STPA Library

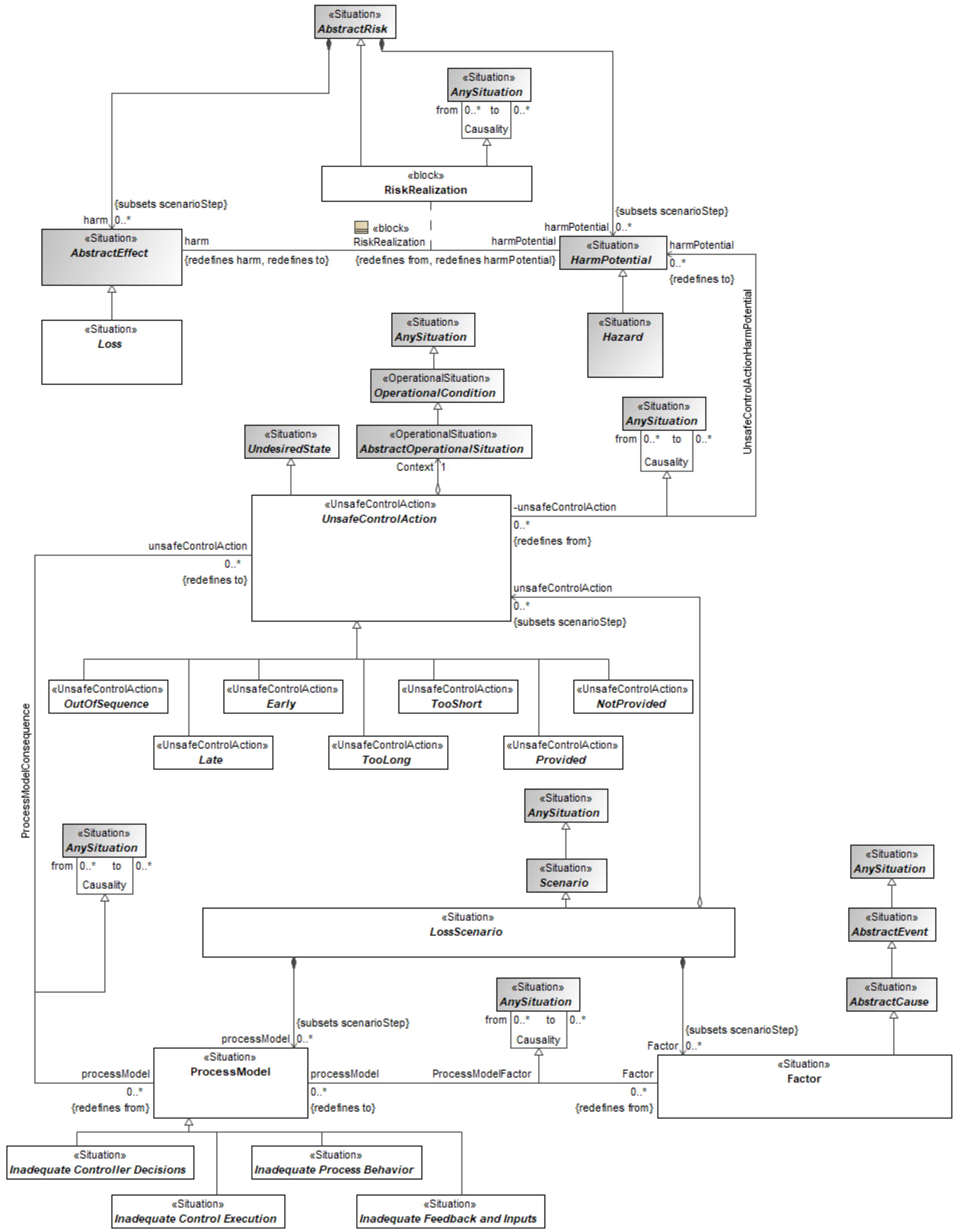


Figure 10.10 - STPA Library

Elements

- AbstractCause
- AbstractEffect
- AbstractEvent
- AbstractOperationalSituation
- AbstractRisk
- AnySituation
- Causality
- Early
- Factor
- HarmPotential
- Hazard
- Inadequate Control Execution
- Inadequate Controller Decisions
- Inadequate Feedback and Inputs
- Inadequate Process Behavior
- Late
- Loss
- LossScenario
- NotProvided
- OperationalCondition
- OutOfSequence
- ProcessModel
- ProcessModelConsequence
- ProcessModelFactor
- Provided
- RiskRealization
- Scenario
- TooLong
- TooShort
- UndesiredState
- UnsafeControlAction
- UnsafeControlActionHarmPotential

10.5.2 Methods::STPA::STPA Profile

View Methods::STPA::STPA Profile::STPA Profile

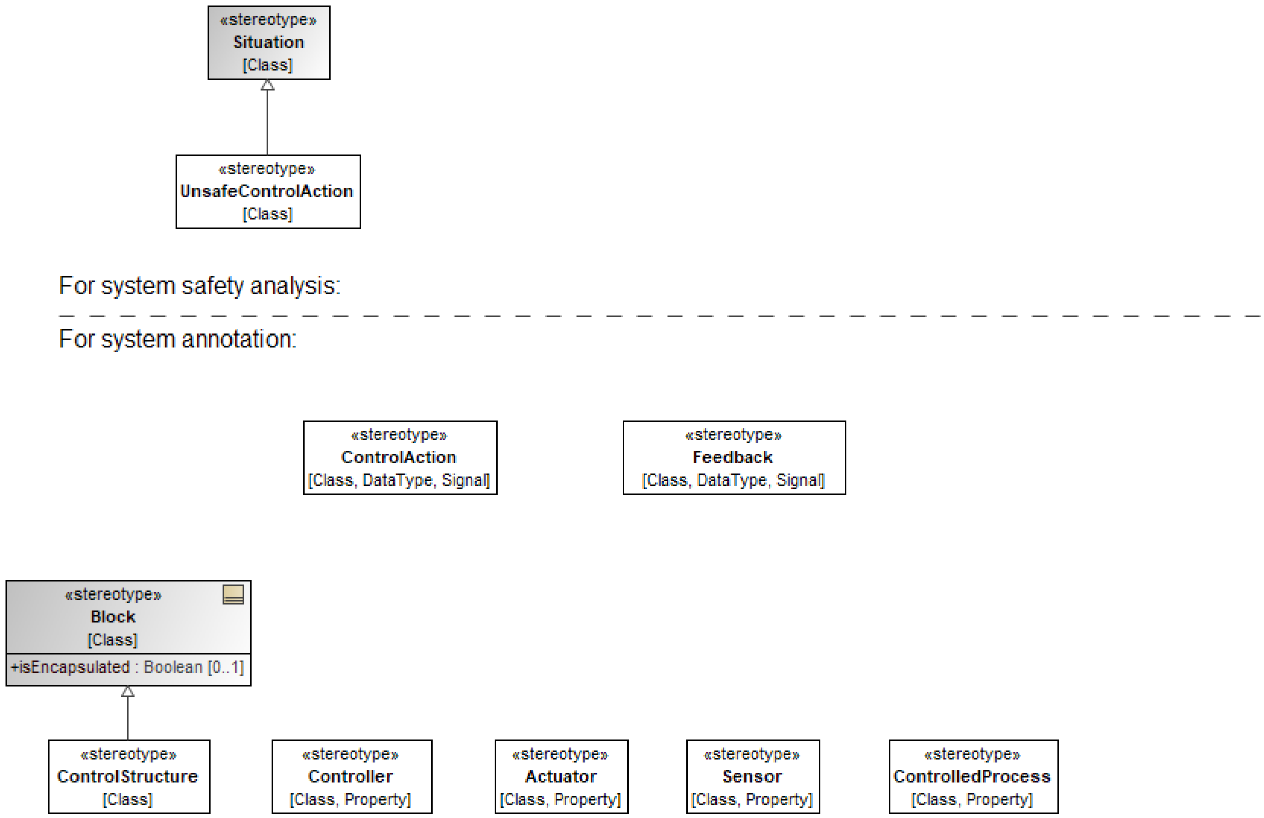


Figure 10.11 - STPA Profile

Elements

- Actuator
- ControlAction
- ControlledProcess
- Controller
- ControlStructure
- FailureMode
- Feedback
- Sensor
- UnsafeControlAction

10.6 GSN

10.6.1 GSN::GSN Profile

View GSN::GSN Profile::GSN Profile

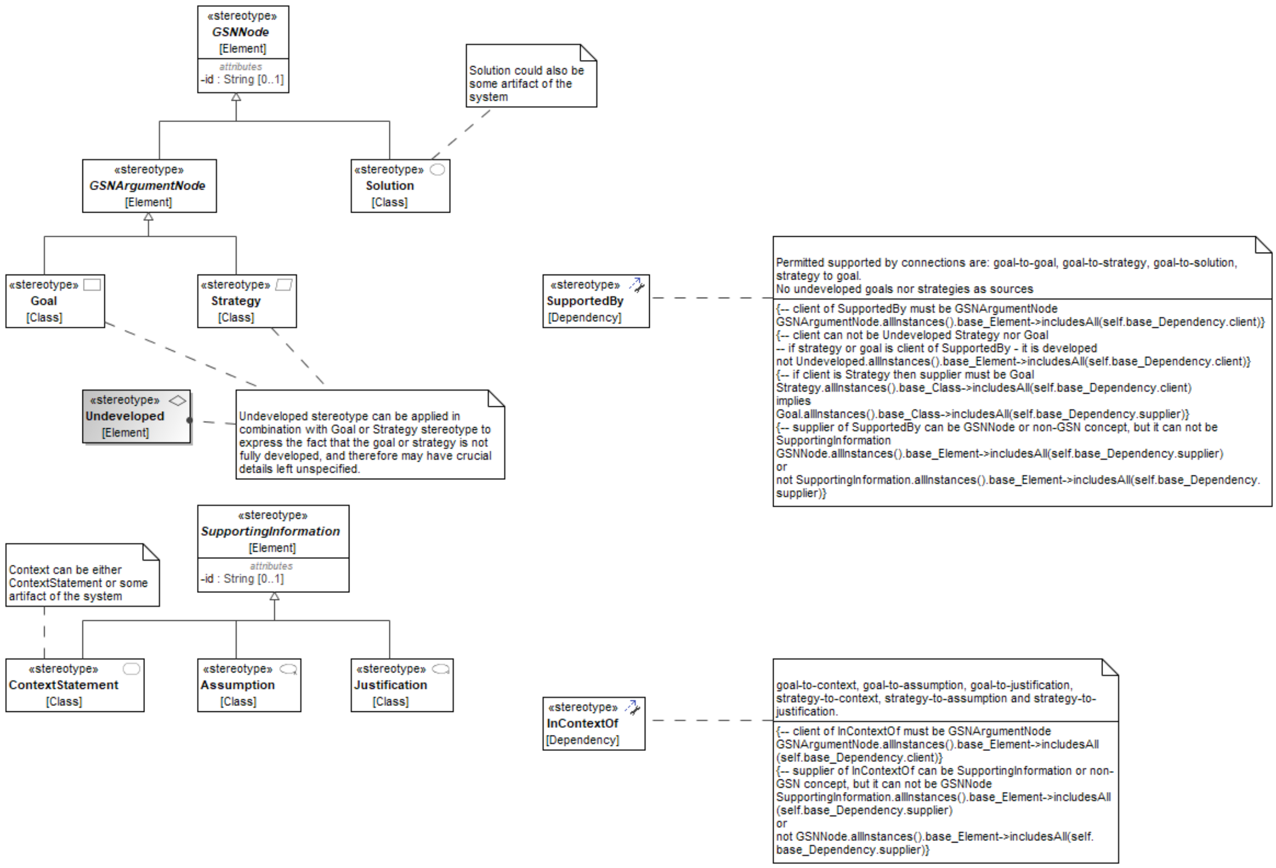


Figure 10.12 - GSN Profile

Elements

- Assumption
- ContextStatement
- Goal
- GSNArgumentNode
- GSNNode
- InContextOf
- Justification
- Solution
- Strategy
- SupportedBy
- SupportingInformation
- Undeveloped

10.7 Methods::ISO 26262

10.7.1 Methods::ISO 26262::ISO 26262 Library

View Methods::ISO 26262::ISO 26262 Library::ISO26262 Library

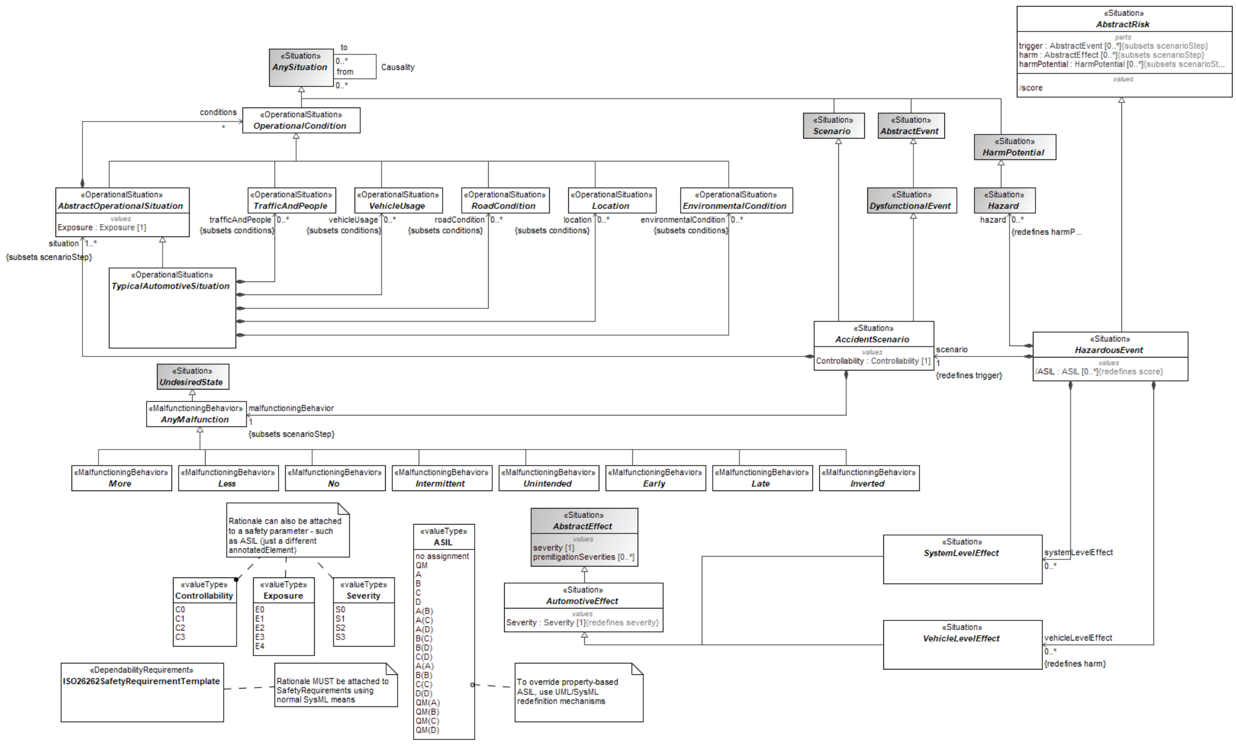


Figure 10.13 - ISO26262 Library

Elements

- AbstractEffect
- AbstractEvent
- AbstractOperationalSituation
- AbstractRisk
- AccidentScenario
- AnyMalfunction
- AnySituation
- ASIL
- AutomotiveEffect
- Causality
- Controllability
- DysfunctionalEvent
- Early
- EnvironmentalCondition
- Exposure
- HarmPotential
- Hazard
- HazardousEvent
- Intermittent
- Inverted
- ISO26262SafetyRequirementTemplate
- Late

- Less
- Location
- More
- No
- OperationalCondition
- RoadCondition
- Scenario
- Severity
- SystemLevelEffect
- TrafficAndPeople
- TypicalAutomotiveSituation
- UndesiredState
- Unintended
- VehicleLevelEffect
- VehicleUsage

View Methods::ISO 26262::ISO 26262 Library::All-Encompasing Operational Situations

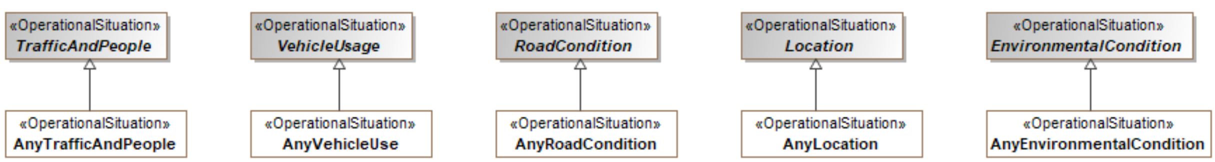


Figure 10.14 - All-Encompasing Operational Situations

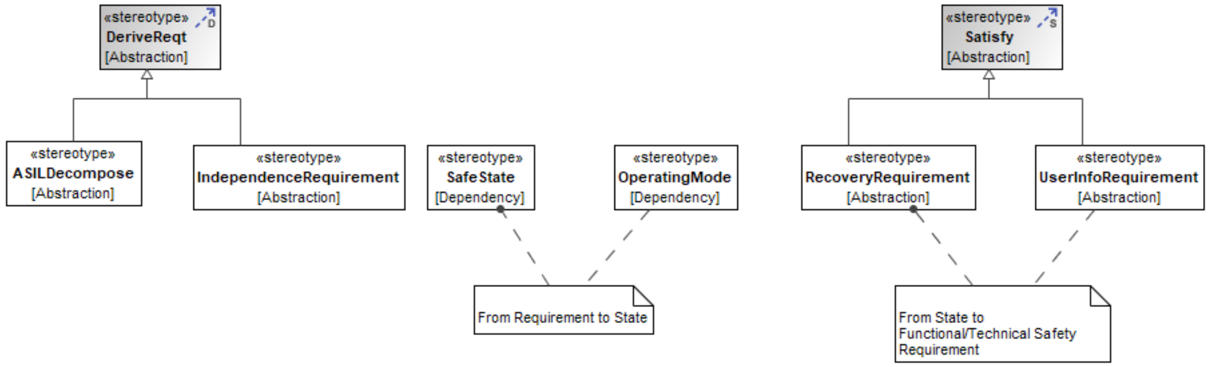
Elements

- AnyEnvironmentalCondition
- AnyLocation
- AnyRoadCondition
- AnyTrafficAndPeople
- AnyVehicleUse
- EnvironmentalCondition
- Location
- RoadCondition
- TrafficAndPeople
- VehicleUsage

10.7.2 Methods::ISO 26262::ISO 26262 Profile

Methods::ISO 26262::ISO 26262 Profile::RequirementManagement

View Methods::ISO 26262::ISO 26262 Profile::RequirementManagement::RequirementManagement



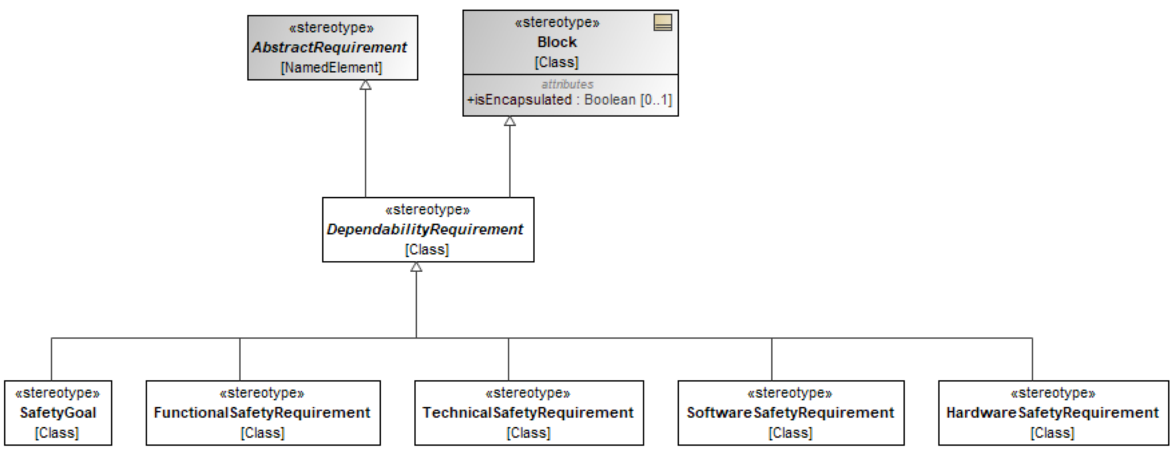


Figure 10.15 - RequirementManagement

Elements

- ASILDecompose
- DependabilityRequirement
- FunctionalSafetyRequirement
- HardwareSafetyRequirement
- IndependenceRequirement
- OperatingMode
- RecoveryRequirement
- SafeState
- SafetyGoal
- SoftwareSafetyRequirement
- TechnicalSafetyRequirement
- UserInfoRequirement

View Methods::ISO 26262::ISO 26262 Profile::ISO26262 Profile

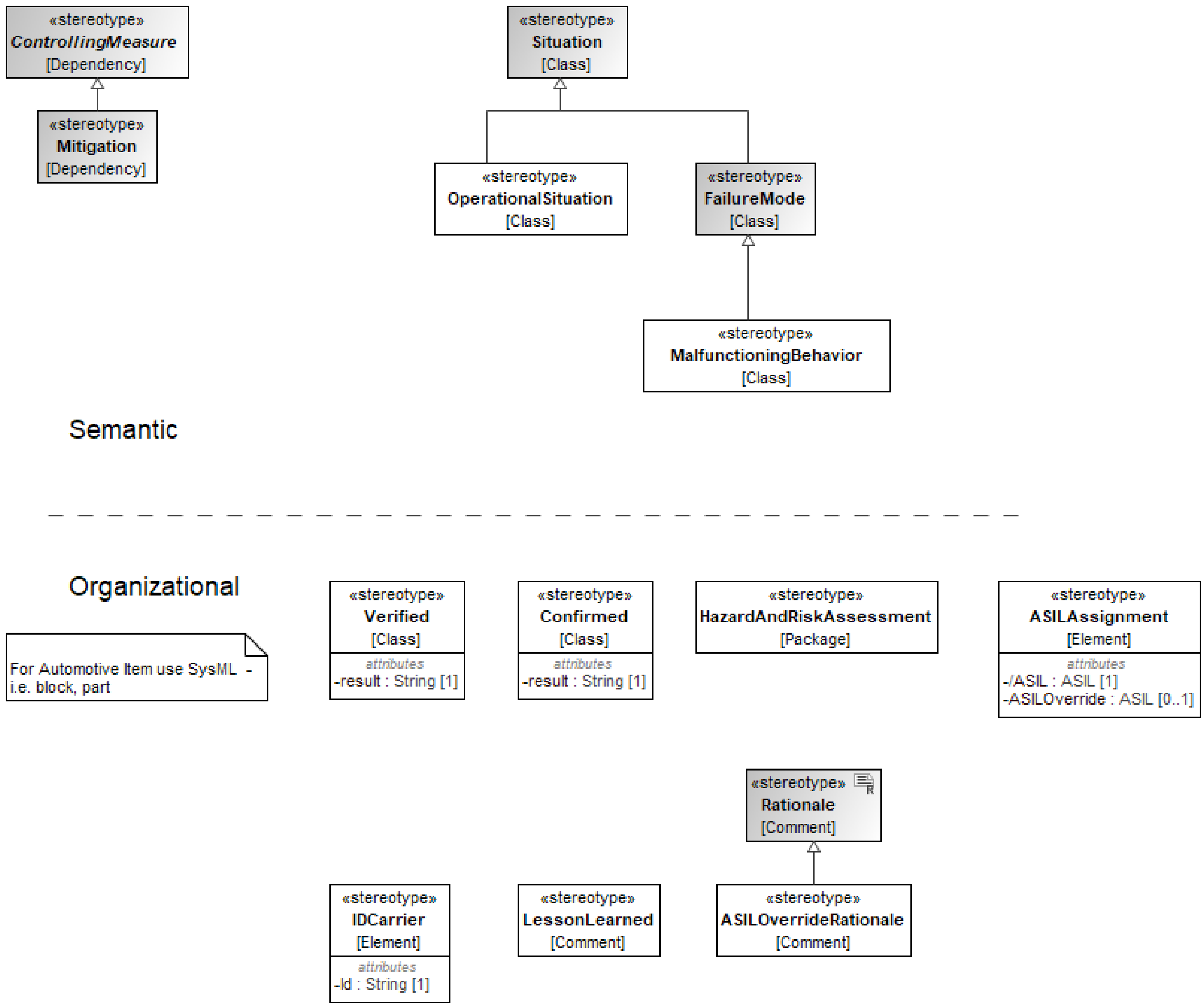


Figure 10.16 - ISO26262 Profile

Elements

- ASILAssignment
- ASILOverrideRationale
- Confirmed
- ControllingMeasure
- FailureMode
- HazardAndRiskAssessment
- IDCarrier
- LessonLearned
- MalfunctioningBehavior
- Mitigation
- OperationalSituation
- Situation
- Verified