Build Safety-Critical Designs with UML-based Fault Tree Analysis - Defining a Profile

Bruce Powel Douglass, IBM/Rational

4/28/2009 12:15 AM EDT

Bruce Powel Douglass, IBM/Rational

: This three part series describes use of Fault Tree Analysis (FTA) in safety-critical design, taking advantage of UML profiling to create a safety analysis profile, including the definition of its normative metamodel. Part 2: Defining a UML Profile for Safety Analysis. A profile is a coherent set of lightweight extensions to the UML, creating a version of the UML specialized for some purpose or problem domain. A profile can contain a number of things, including stereotypes, tags, constraints, new diagram types, iconic representations, and model libraries.

Each stereotype within a profile must extend a metaclass from the UML metamodel, such as Class, Event, or Association. The stereotype is usually elaborated with metadata stored in named tags, constraints on its usage, and graphical iconic depictions.

For example, in the SysML profile, a flow port is a stereotype of port that applies to flows. New types of diagrams may be created that represent collections of these elements for specific purposes.

For example, in the UML Profile for **DoDAF** and **MoDAF** **(UPDM)**, and **OV-2** product is a diagram based on a UML class diagram that specifically contains stereotyped elements from the UPDM to show operational nodes within an operational architecture and their relations.

In this context, I will use the IBM Rational Rhapsody tool to create a safety analysis profile that adds new stereotyped elements to create fault tree analysis (**FTA**) diagrams and custom matrix and tabular views to summarize the results of the analysis.

In addition, I will extend the typical definition of FTA elements to support traceable links from the FTA model into both requirements and design elements. However, any UML tool that supports UML profiling mechanisms should work.

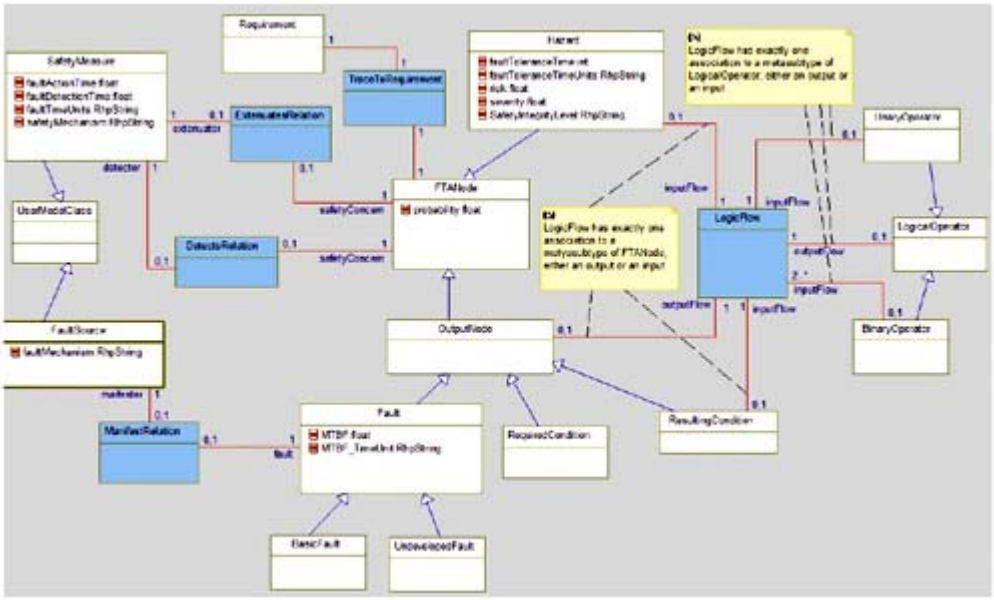

Figure 1: Safety Profile Metamodel

(*To view an expanded version* ***Click Here.***)

Before a profile can be created, it is necessary to characterize the concepts contained within the profile and how these concepts relate to each other. This is usually done with a metamodel. A metamodel is a model of the fundamental concepts and their relations for a domain or subject matter.

Figure 1 above shows the metamodel for the Safety Analysis Profile (*the metasubtypes of Logical Operator are detailed in* ***Figure 2 below***). The attributes of the metaclasses will end up as tags on our defined stereotypes. The colored boxes are the relations between the core metaclasses.

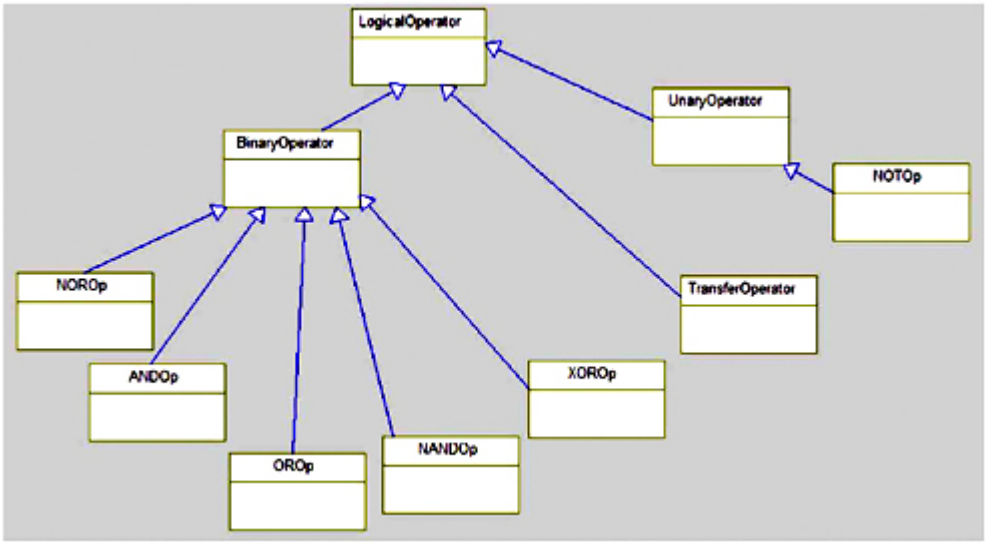

Figure 2: Metasubtypes of Logical Operators

The key elements for the metamodel (*along with their profile realizations*) are:

Hazard - a condition that will lead to an accident or loss. This is usually the top terminal element in an FTA. (Stereotype of Class)

Fault - the non-conformance of an element to its specification or expectation. Faults are further subclassed into Basic Faults and Undeveloped Faults. These are usually the bottom terminal elements in an FTA. (Stereotype of Class)

Resulting Condition - the condition resulting from a combination of faults and conditions, combined with logical operators. (Stereotype of Class)

Required Condition - A condition required for the fault to interact. (Stereotype of Class)

Logical Operator - one of several logic conjunctives, such as AND, NOT, OR, etc. Note that Transfer operator actually has no semantics of its own but is used as a "diagram connector", allowing large FTAs to be broken up across multiple diagrams. (Stereotype of Class)

Logic Flow - the connection of a fault, condition or hazard to a Logical Operator. The logic flow can be an input or an output. For example, in the statement A || B --> C, there is a flow output from A as an input to the || (OR) operator. There is also an output from flow the || operator to the resulting condition C. (Stereotype of Flow).

Fault Source - this is a normal UML element that could manifest a fault, i.e. that could be the source of a fault. (Stereotype of Class)

Safety Measure - this is a normal UML element that could detect or extenuate (i.e. mitigate) a fault. (Stereotype of Class)

Manifest relation - this is a relationship from a Fault to a Fault Source that causes the fault (Stereotype of Dependency)

Detect relation - this is a relation from a Fault or Hazard to a Safety Measure that can detect when the fault has occurred. (Stereotype of Dependency)

Extenuates relation - this is a relation from a Fault or Hazard to a Safety Measure that reduces either the likelihood or severity of the hazard or fault. (Stereotype of Dependency)

Trace To Requirement - this is a relation from a Fault or Hazard to a Requirement. (Stereotype of Dependency)

Faults and Hazards elements have important metadata characterizing them. The important metadata is summarized in **Table 1, below**.

Metaclass	Metadata	Description
Hazard	Fault Tolerance Time	This is the length of time the fault can be tolerated before it leads to an accident.
	Fault Tolerance Time Units	This is the units of time (e.g. ms, seconds, hours, days)
	Risk	Risk is the product of the severity times the probability
	Severity	The degree of damage the accident can cause
	Safety Integrity Level	For standards such as IEC65-1508, this is the identified SIL level
	Probability	The likelihood of occurrence of the hazardous condition, usually computed from the metadata of the faults
Fault	Probability	The likelihood the fault will occur
	MTBF	The Mean Time Between Failure for the element
	MFBF Time Units	The time units expressed in the MTBF meta-attribute
Fault Source	Fault Mechanism	A description of how the fault can occur
Safety Measure	Fault Action Time	The length of time the corrective action requires to complete once initiated
	Fault Detection Time	The length of time, from the occurrence of the fault to its detection
	Fault Time Units	The unit of time used in the Fault Action Time and the Fault Detection Time
	Safety Mechanism	A description of how the detection and/or safety action is performed

Table 1: Safety Metadata

Tables, Matrices and Hazard Analyses

In addition to the elements of the profile, new tables and matrices are added in the profile as well, shown in **Table 2 below**.

The Hazard analysis is generated as an external file with a helper macro. This macro scans the entire model and generates the tab-separated value file1 that can be loaded into most spreadsheet programs.

[*Tab-separated value format was used because Excel has defects in its interpretation of the more-common comma-separated value (CSV) file format.***]**

The macro generates the name from the current date and time so that multiple versions of the hazard analysis can be kept. The output is divided into three sections.

Fault Table	Table View	This lists the faults and all their metadata
Hazard Table	Table View	This lists the hazards and all their metadata
Fault Source Matrix	Matrix View	This shows a fault x fault source matrix, as defined by the Manifests relations
Fault Detection Matrix	Matrix View	This shows a fault x safety measure matrix, as defined by the Detects relations
Fault Extenuation Matrix	Matrix View	This shows a fault x safety measure matrix, as defined by the Extenuates relations
Hazard Analysis	Tab-separated value text file (.tsv) intended to load into Excel	This is an external file generated by the profile helper macros summarizing the hazard and fault information.

Table 2: Tables and Matrix summary views

The first section lists the hazards and their metadata, including the description, fault tolerance time, fault tolerance time units, probability, severity, risk, and safety integrity level.

The second section lists the relations between the faults and the hazards as defined by multiple intervening logical operators and logic flows. Each fault is identified with is name, description and other metadata.

The third section lists the relations between the faults and the "normal" UML model elements " requirements and classes related with the manifests, detects, extenuates, and traceToReqs relations.

The hazard analysis provides a summary with enough information to trace from the safety requirements to the model elements realizing those requirements, as well as from the faults and hazards to the requirements and design.

Creating the Profile

Once the metamodel is understood it can be used as a blueprint for the profile. Metaclasses in the metamodel become stereotypes. Metaattributes become tags. The result is the creation of an integrated set of stereotypes, tags, and constraints such as shown in Figure 3 below.

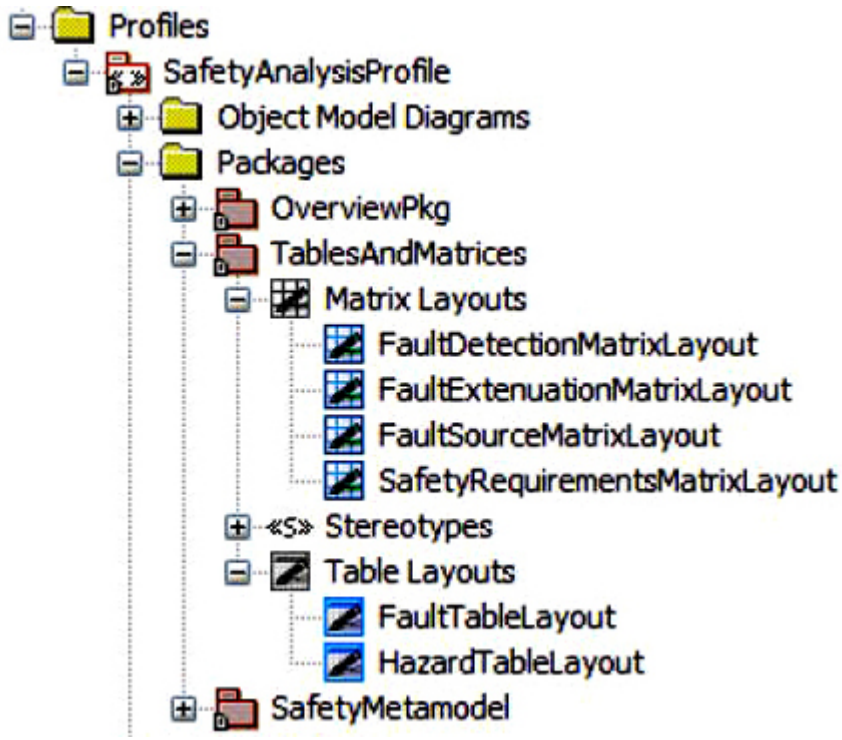

Figure 3: Profile Structure

(*To download a PDF of the long version of this truncated profile structure list, **Click Here.***)

Using the Profile

To use the profile in Rhapsody, you can create a new model of the type Safety Analysis or you can add the profile after the model is created. If you do this, you must select the project in the browser, right-click, and change the type of the model to Safety Analysis Profile.

Once the model is created, a new diagram type is available on the diagram toolbar " the FTA diagram. All of the standard UML features remain available to you. I recommend that you put the safety analysis in a separate package in your model to separate it from your requirements and design elements. Again, if you're using a different tool to create the profile, the exact mechanisms to install and use the profile are likely to differ.

Conclusion

Fault Tree Analysis (FTA) is well established as a useful method for understanding how normal events, conditions and faults combine to create hazardous conditions. The safety analysis profile discussed in this paper adds the ability to create and report on FTA diagrams into a UML tool.

This includes the specification of safety-related metadata, such as hazard severity, risk, probability and safety integrity level, as well as fault probability and **MTBF**.

The profile extends the FTA method by supplying relations from the analysis to normal UML model elements " specifically, requirements, source of faults, and elements that can detect or extenuate the faults. These extensions add value by making the relations between the safety analysis and the UML model elements explicit and traceable.

This profile supports the safety approach identified in the Harmony/ESW (Embedded Software) process [4,5] from IBM/Rational, developed by the author. Through the use of this profile, developers and safety analysts can use a common language and tool environment, improving their collaboration and quality of work.

Next in Part 3: **Fault Tree Analysis of a surgical anesthesia ventilator**

To read Part 1, go to **The basics of safety and capturing of fault metadata for analysis**

Bruce has worked as a software developer in real-time systems for over 25 years and is a well-known

speaker, author, and consultant in the area of real-time embedded systems. He is on the Advisory Board of the Embedded Systems Conference where he has taught courses in software estimation and scheduling, project management, object-oriented analysis and design, communications protocols, finite state machines, design patterns, and safety-critical systems design. He develops and teaches courses and consults in real-time object-oriented analysis and design and project management and has done so for many years. He has authored articles for a many journals and periodicals, especially in the real-time domain.

He is the Chief Evangelist for Rational/IBM, a leading producer of tools for software and systems development. Bruce worked with various UML partners on the specification of the UM, both versions 1 and 2. He is a former co-chairs of the Object Management Group's Real-Time Analysis and Design Working Group. He is the author of several other books on software, including Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns (Addison-Wesley, 1999), Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems (Addison-Wesley, 2002), Real-Time UML 3rd Edition: Advances in the UML for Real-Time Systems (Addison-Wesley, 2004), Real-Time UML Workshop for Embedded Systems (Elsevier Press, 2006) and several others, including a short textbook on table tennis. His latest book on employing agile methods to develop real-time and embedded systems, Real-Time Agiliy, will appear in June, 2009.

References

[1] Leveson, Nancy. Safeware: System Safety and Computers Reading, MA: Addison-Wesley, 1995

[2] Guidance for FDA Reviewers and Industry: Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices Washington, D.C.; FDA, 1998

[3] IEC 65A/1508: Functional Safety: Safety-Related Systems Parts 1-7 IEC 1995

[4] Douglass, Bruce Powel. Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns Reading, MA: Addison-Wesley, 1999

[5] Douglass, Bruce Powel. Real-Time Agility Reading, MA: Addison-Wesley, 2009.

[6] Douglass, Bruce Powel. Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems Addison-Wesley, 2002