Safety Analysis of UML Models

Bruce Powel Douglass, Ph.D.
Chief Evangelist
IBM/Rational

Abstract

The UML has been successfully been used in many real-time and embedded domains, including aerospace, military, and medical markets. Many of these systems within these markets are used within safety critical contexts. So far, disparate tools and environments have been used for capturing requirements, creating designs, and analyzing system safety. However, UML is an extremely powerful, extensible language. To this end, I have created a UML profile that support capturing requirements, creating designs, and analyzing system safety all within the same Rhapsody tool environment.

This paper will discuss the use of Fault Tree Analysis (FTA) for safety analysis, and use of the UML profiling mechanism to create a safety analysis profile, including the definition of its normative metamodel. This profile enables developers and analysts to capture safety-related requirements, perform FTA and other safety analyses, create designs that meet those safety concerns, and provide reports showing the relations between the safety analysis, requirements, and design model elements.

What is Safety?

The paucity of material on safety critical systems has lead to widespread misunderstanding of the various terms used to discuss safety. The most basic term is *safety*. Safety is defined to be *freedom from accidents or losses.* An *accident* is an event in time in which an undesirable consequence occurs, such as death, injury, equipment damage, or financial loss. A *safety-critical system* in a system, which may contain electronic, mechanical, and software aspects, that presents an opportunity for accidents to occur. For many people, safety-critical systems are only those that present the opportunity for injury or loss of life, but this omits from consideration other systems which might benefit from the techniques and approaches common in safety analysis. Therefore, I prefer to designate a safety critical system to be any system in which the cost of use of a system due to an accident is potentially high.

A *hazard* is system state that when combined with other environmental conditions inevitably leads to an accident [1]. Hazards are normally classified as to severity. For example, there is a hazard of being shocked when jumping the 12-volt battery in your car, but this is a less severe risk than slamming into a mountainside at 600 knots while riding in a commercial aircraft. Different standards use different categories for hazard severity. For example, the FDA[2] uses major (irreversible injury or death), moderate (injury), and minor (no injury) levels of concern for device safety. The German standard DIN 19250

identifies 8 categories, along with required safety measures for each category while the more recent IEC 61508 [3] identifies 4 safety integrity levels (SILs): catastrophic, critical, marginal, and negligible, although the text notes that the severity of system-presented hazards is actually a continuum.

The *risk* of a hazard is defined to be the product of the probability of the occurrence of the hazard and its severity:

$$risk_{hazard} = probability_{hazard} \times severity_{hazard}$$

Being shocked by your car battery is relatively high but when combined with the low severity, the overall risk is low. Similarly, while the consequences of an abrupt release of the kinetic energy of a commercial aircraft are quite severe, its probability is low – again resulting in a low risk. The various standards also identify different risk levels based on both the severity of the hazard and its likelihood of occurrence.

In the process of system design, hazards must be identified and safety measures must be put in place to reduce the risk.

Faults and Failures.

A *safety fault* is a nonconformance of a system that leads to a hazard. Faults come in two flavors: failure states and errors. A *failure* is an event that occurs when a component no longer functions properly, and leads to a *failed state*. A *soft failure* is a temporary failure that may be corrected (or correct itself) without replacing the failed component. A *hard failure* is one in which the component must be replaced to repair the defect. Failures are distinct from *errors.* An error is a design or implementation defect. Failures are events that occur at some point in time while errors are omnipresent conditions or states. Errors may not always be apparent; when they become apparent, they are said to *manifest*.

Mechanical or electronic hardware may have both failures and errors, while software can only have errors. In addition, many (but by no means all) systems have a condition that is known to be always safe – this is called the *fail-safe state*. In many systems, this state is with the device turned off or power removed. For example, the fail-safe state for a microwave oven is *off*. Many systems do not have such a fail-safe state.

Faults may be tolerated for a period of time before they lead to an accident. For example, a patient ventilator failure can be tolerated for about five minutes before death occurs. Overpressure can be tolerated for about 250 ms before it causes irreversible lung damage. A failure in the control of aircraft ailerons and elevators in many modern aircraft must be corrected within 50 ms or less to maintain stability. The period of time the system can tolerate a fault is called the *fault tolerance time*. To ensure safety, the system must both detect and handle the fault before the fault tolerance time has elapsed. Also, note that the mean time between failures (MTBF) of the component must be (much) longer than the fault tolerance time. Figure 1 shows the relevant times related to the handling of the fault.

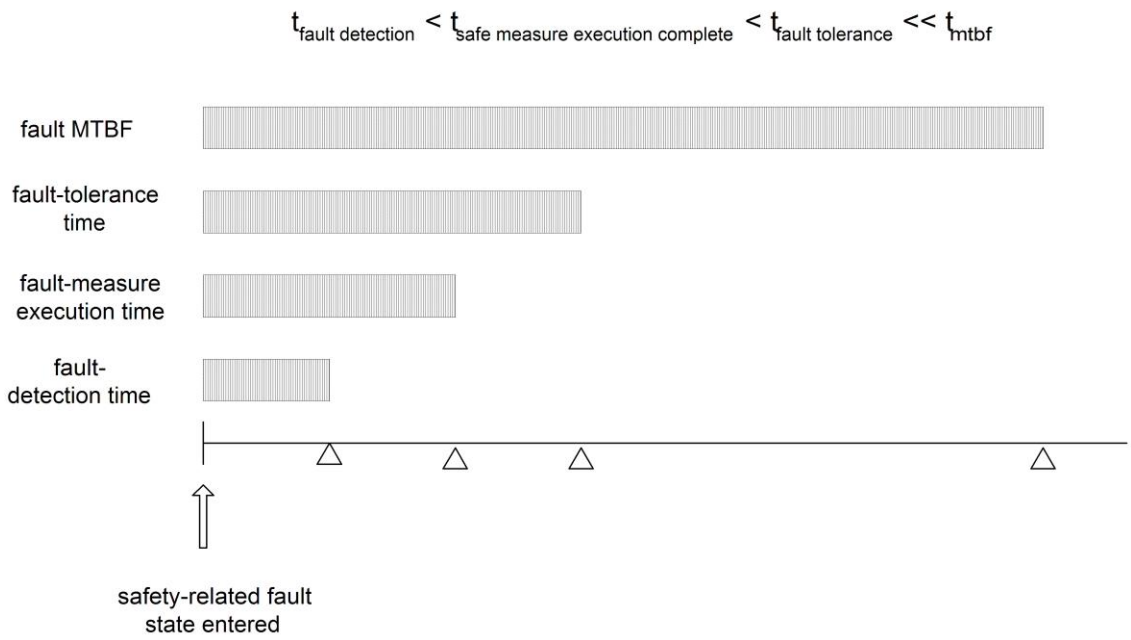


Figure 1: Fault Timeline

These timeframes have ramifications on the kinds of safety detection and correction measures to be applied. If the detection is to be done with periodic or continuous background testing, then the time to complete the test (including the time to perform the normal device operation during that time) is called the *fault detection time*. In many systems, there simply isn't enough processor bandwidth to complete the tests in software in addition to the normal system execution to detect the faults in timely fashion. When this is true, other means must be added to detect the fault. For example, a periodic RAM test, such as the Abraham walking bit test, can detect various kinds of hard memory failures. However, in a system with several megabytes of memory and short fault tolerance time, the detection of a safety-relevant fault cannot be guaranteed to occur within the fault tolerance time. A possible solution is to add mirrored memory with built-in parity checking eliminates the need for a periodic RAM test.

Reliability and Safety

Reliability and safety are mostly independent concerns. Reliability refers to the probability that a system or component will meet its functional and quality of service (e.g. timeliness) requirements within a specified timeframe. While this sounds similar to our previous definition of safety, but the two concepts are importantly different. A safe system is one which does not lead to accidents. It may fail all the time and still be safe. A reliable system may fail infrequently but when it does fail it does so with catastrophic consequences – such a system is not safe. A handgun, for example, is a very reliable piece of equipment, but can easily lead to accidents even in the absence of a system fault. On the other hand, my old Plymouth station wagon refuses turn on at all, therefore it is very safe even though it is unreliable.

In general, reliability is a separate concern from safety, and it is important to maintain the distinction. For the most part, in systems that have a fail-safe state, reliability is an opposing concern to safety. Reliability is improved when the system continues to provide services even it creates a hazardous situation. If the system is creating a hazardous situation, and there is a fail-safe state, then entering the fail-safe state improves system safety but decreases system reliability.

Consider a medical treatment laser. If a memory cell in the controller seems faulty, the safest thing the system can do is to shut down with the laser is de-energized (its fail-safe state), even if it is relatively unlikely that the detected fault could lead to a hazard. This decreases the system reliability. In such systems, a pessimistic policy is likely to be safer than an optimistic policy.

Many systems don't have a fail-safe state. If you're flying at 600 knots and 35,000 feet, it is not safe to shut off the jet engine if it is suspected of having a fault. Similarly, in a drive-by-wire car, the *last* thing I want to see is an "Abort, Retry, Ignore" message appear on my dashboard when I'm driving down the freeway at 85 (ah, excuse me, 55) mph. In such systems, increasing reliability (such as by adding redundant delivery channels) also improves the system safety.

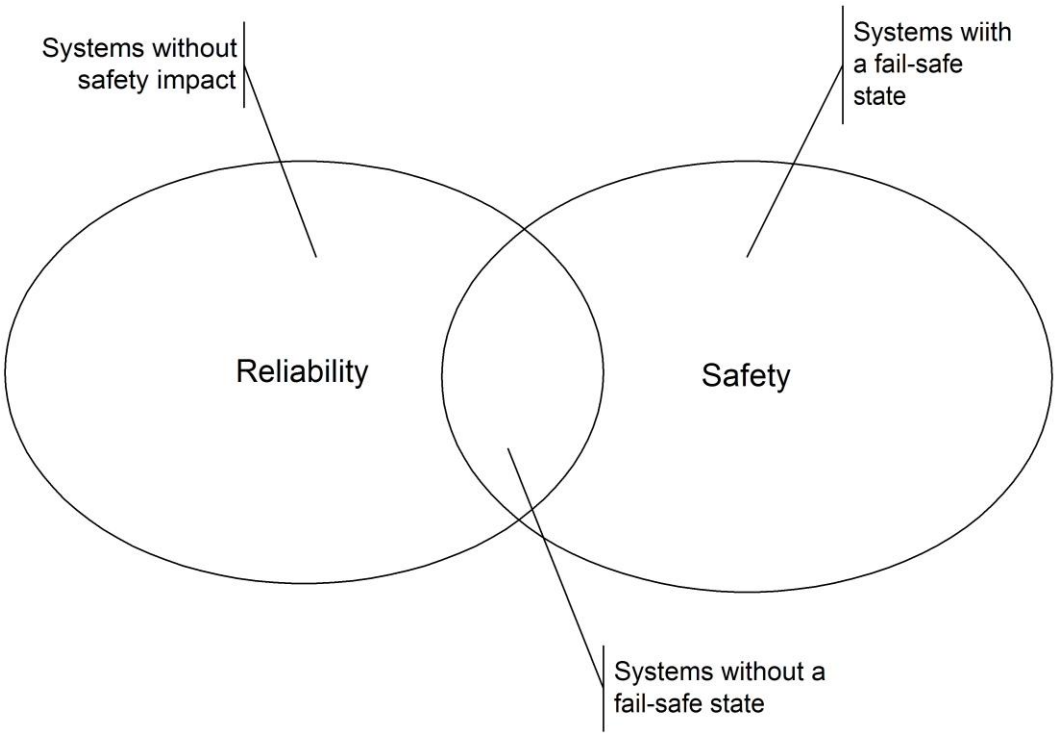


Figure 2: Safety vs. Reliability

Types of safety measures

There are several different kinds of things one can do about faults:

- Obviation
  For example, the use of mechanically incompatible fasteners can remove the hazard of connecting a patient oxygen intake to a nitrogen source.
- Education
  The hazard can be handled by educating the users so that they won't create hazardous conditions through equipment misuse. This is a relatively weak safety measure that depends on the sophistication of the user and may not be appropriate in many circumstances.
- Alarming
  Announcing the hazard to the user when it appears so that they can take appropriate action. This approach requires a fault tolerance time that can take into account the reaction time of monitoring personnel. For example, an ECG monitor can notify an attending physician of an asystole condition.
- Interlocks
  The hazard can be removed by using secondary devices and/or logic to intercede when a hazard presents itself. For example, a medical treatment laser might automatically disconnect power to the laser when its cover is off.
- Transitioning to a fail-safe state
  The hazard can be handled by ensuring that a system can detect faults prior to an accident and enter a state which is known to be safe. For example, a cruise control system can shut off, returning to manual control when a fault is detected.
- Switch to a redundant channel
  The hazard can be handled by engaging another actuation channel to perform the system action correctly. This approach is generally preferred when the system has no fail-safe state.
- Use of additional safety equipment
  For example, the use of a drill press may require a light curtain to ensure the user doesn't place his or her limbs in harm's way.
- Restricting access
  Using passwords to prevent users from inadvertently invoking "service mode" in which safety checks are turned off.
- Labeling
  The hazard can be handled by labeling, e.g. *High Voltage -- DO NOT TOUCH*

`

Each of these different approaches may be appropriate in different circumstances. Obviation is usually safest, but not always achievable. Going to a fail-safe state requires both a means for detecting a fault and the presence of a system condition which is both known and achievable.

How Can the UML Help?

UML is a modeling language that is commonly applied to both software and systems development. It provides a semantic basis of fundamental concepts and views (diagrams) that depicts the interaction of elements of interest. UML can aid the development of safety critical systems in a number of ways:

- By providing design clarity
- By modeling architectural redundancy
- By modeling low-level redundancy
- By creating safety-relevant views of the requirements and design
- By aiding in safety analysis

First, UML can provide design clarity by exposing the design of the system in class diagrams (AKA internal block diagrams in SysML[1]) and by explicating showing the traceability to requirements. If all you have is source code, then it can be extremely difficult to identify the redundant safety measures, traceability to requirements, and other safety relevant aspects of the design.

The fundamental building blocks of a UML model is the notion of a *class* (*Block* in SysML). It contains features such as data ("attributes"), services ("operations"), logic ("state machines"), algorithms ("activity diagrams"), quality of services aspects ("constraints"), interactions ("sequence diagrams"), and connection points ("ports"). When a class has safety relevance, it is possible to add low-level redundancy, such as using CRC on the class attributes, data replication, precondition and post condition checking, etc., to ensure that safety-relevant faults are identified and handled appropriately.

One of the big benefits that UML provides is the ability to construct views (diagrams) that focus on narrow aspects of the system structure or design. The same elements can be depicted in many different views and the underlying model repository will ensure that all the views are consistent. The Harmony/ESW ("embedded software") process [4,5] identifies five key views of architecture, of which the Safety and Reliability View is one. It typically shows the structurally redundant elements and their interaction that achieves the safety goals of the system, and can do this at different levels of abstraction. This allows the engineering and safety staff to understand how faults propagate through the system, how safety measures interrupt that fault propagation, and to perform safety analysis of the designs.

Fault Tree Analysis (FTA), an analytic approach discussed later in the paper, is a common technique for analyzing how faults lead to hazards and how to add safety measures to address these concerns. While there are a few FTA tools available, it is possible to create a safety-critical profile[2] that permits the capturing of fault metadata for analysis. The advantage of this is that the requirements, design model and safety analysis are all co-located and all interconnected. This allows developers to reliably navigate between these three kinds of views with ease.

[1] SysML is a profile (specialized version) of UML used in system engineering.
[2] As we shall see shortly, a profile is a specialized version of the UML, consistent with the underlying UML semantics, to meet a specialized need. The Systems Modeling Language (SysML) is one such profile.

Safety Analysis with Fault Tree Analysis

Elsewhere I have described a best practice called "8 Steps to Safety[4]". This practice is meant to be added on top of a general development process such as a traditional waterfall lifecycle or spiral model, such as the Harmony/ESW process[4,5]. The basic practice steps are simple to understand, and relatively straightforward to implement.

Harmony/ESW's Eight Steps to Safety

1. Identify the hazards
2. Determine the risks
3. Define the safety measures
4. Create safety requirements
5. Create safe designs
6. Implement safety
7. Assure the safety process
8. Test, Test, Test

Safety analysis (steps 1-3) is performed early in the development lifecycle and elaborated frequently throughout development. The safety analysis identifies the hazards presented by a system used in its execution context. This feeds back into the system requirements specification in order to ensure that the system, as specified, is safe. The safety analysis results in a *hazard analysis* document which lists the hazards presented by the system, the faults that can lead to the hazard, the fault tolerance time of the hazard, the safety measure used to mitigate the hazard, and the necessary fault handling response time.

Safety design specified the means for detecting and extenuating the faults in the design. This is most commonly done by identifying the architectural and detail design redundancy in such a way that the safety requirements are met. The Harmony/ESW recommends this be done with the application of fault tree analysis to link the safety measures with the faults to ensure fault coverage.

Safety testing is then performed to ensure that the safety requirements are met. This typically involves not only the primary functionality and quality of service testing used for non-safety critical systems, but also seeding the system with faults. Seeded faults may be simulated or they may be done by actually inducing the faults in the running system. It is common, for example, to cut wires, discontinue power, and pull chips from sockets during fault seeding tests.

As mentioned above, a common and useful analytic technique applied to safety-critical systems is called *fault tree analysis* (FTA). In FTA, conditions leading up to hazards are logically analyzed for cause-effect relations using standard logical operators AND, OR, XOR, and NOT. Figure 3 shows the basic symbols used in the FTA diagrams. FTA allows you to analyze what are the preconditions of hazardous conditions and how they

combine with faults to result in hazards. Once these relations are identified, you can add safety measures whose faults must be ANDed with the original fault in order to lead to the hazardous condition – that is, to arrive at the hazardous condition, the original fault must occur AND there must be a fault in the safety measure as well. There is normally an assumption of *single fault independence* – that is, the primary and safety measure faults are independent[3].

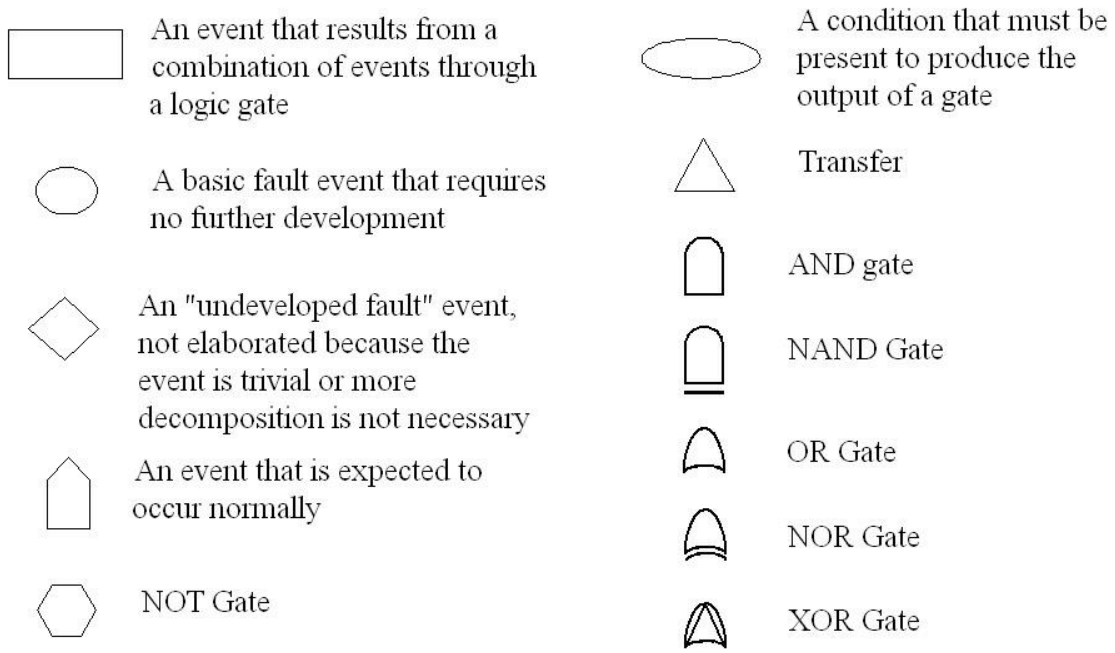


Figure 3: Fault Tree Analysis (FTA) Symbols

Consider the patient ventilator in Figure 4. This system uses the Monitor-Actuator design pattern[4] to achieve safety against two different hazardous conditions – hypoventilation and overpressure. This pattern creates two channels (sets of sequential processing elements); the actuation channel delivers the therapy, and the monitoring channel checks on how well the therapy is delivered

Figure 5 shows an FTA for these two hazards. In an unprotected system, a fault occurring in the breathing circuit, gas supply or the ventilator can lead to hypoventilation. Note that intubating the esophagus (rather than the trachea – unfortunately, a rather common physician error) can also lead to hypoventilation. Our architectural redundancy has added a number of sensors that can detect these conditions, and a failure of *all* of these or a failure of the alarm system is required in addition to the original fault, for the hazard to be realized. In this case, the fault tolerance time for the fault is about five minutes, leaving an adequate amount of time for the attending physician to correct the fault.

[3] When the faults are not independent, this is called a *common mode fault* and usually means that the safety measure is inadequate for the need.
[4] A design pattern is a generalized solution to a commonly occurring design problem. A set of safety-related design patterns are published in [6].

The other hazard protected against is overpressure. In an unprotected system, a fault in the breathing circuit, gas supply, or ventilator presents the hazard. Since the fault tolerance time for this fault is about 250 ms, alarming is an inadequate safety measure. Therefore, we added a relief valve that responds in < 5ms to an over pressure situation. Because of our additional safety measure, the original fault must occur in addition to a failure in the safety measure before the hazard is realized.

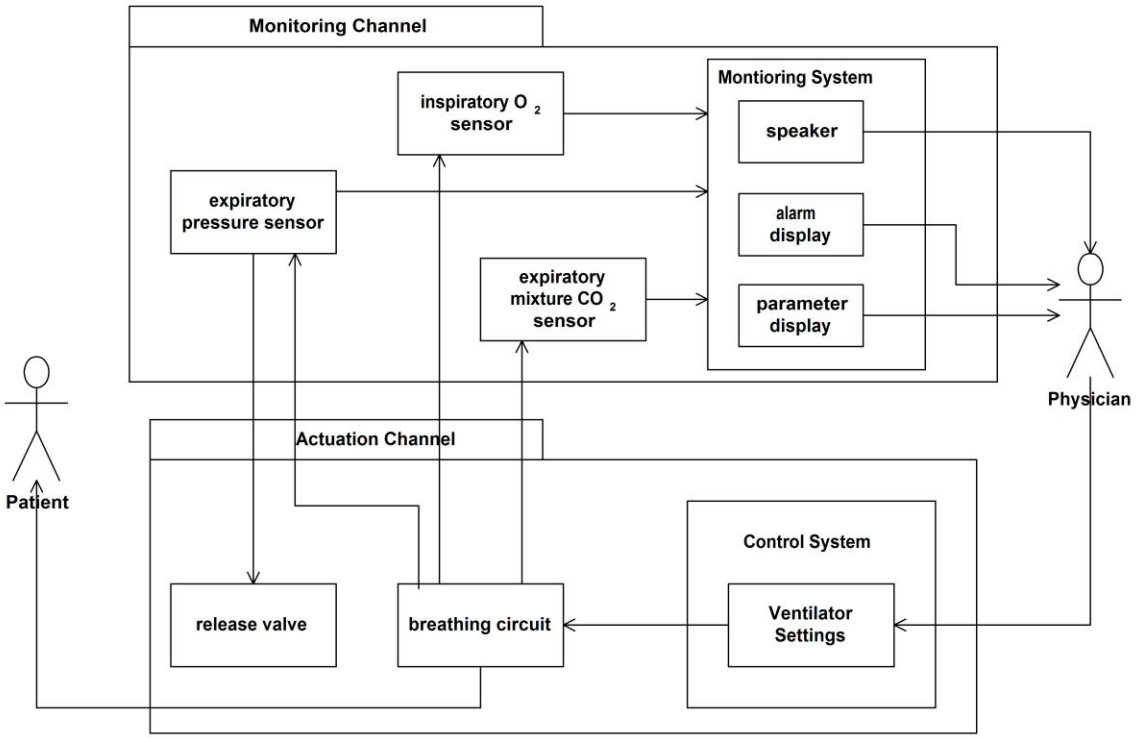


Figure 4: Patient Ventilator Simplified Model

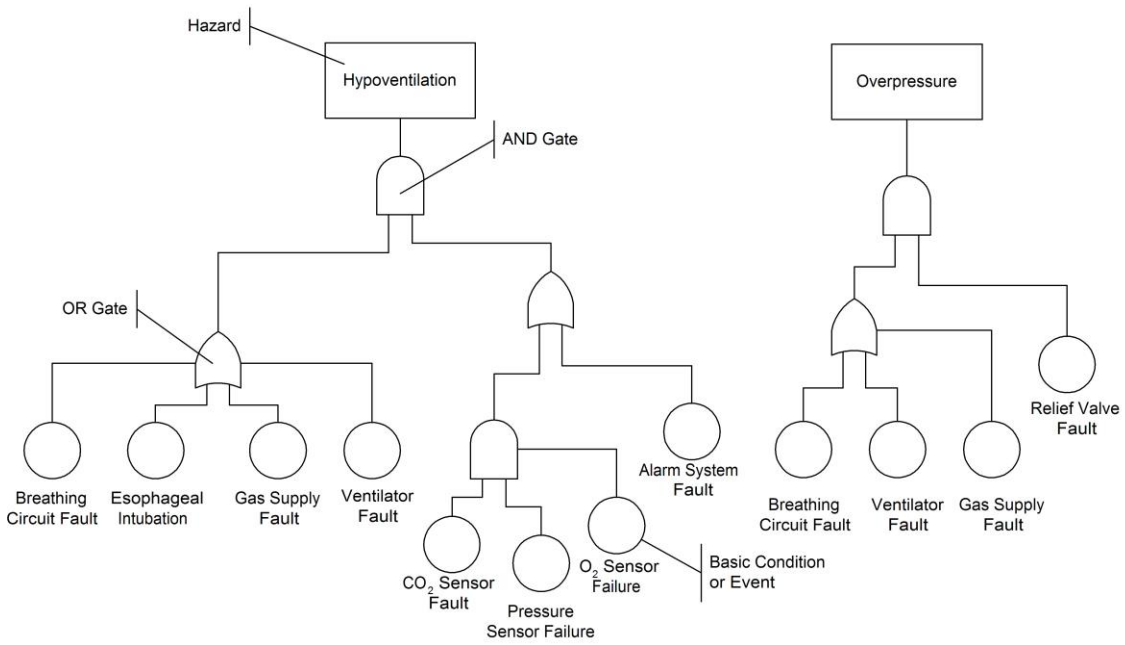


Figure 5: Patient Ventilator Simplified FTA

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 Rhapsody tool to create a safety analysis profile that adds new stereotyped elements to create 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.

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 6 shows the metamodel for the Safety Analysis Profile

(the metasubtypes of Logical Operator are detailed in Figure 7). The attributes of the metaclasses will end up as tags on our defined stereotypes. The colored boxes are the relations between the core metaclasses.

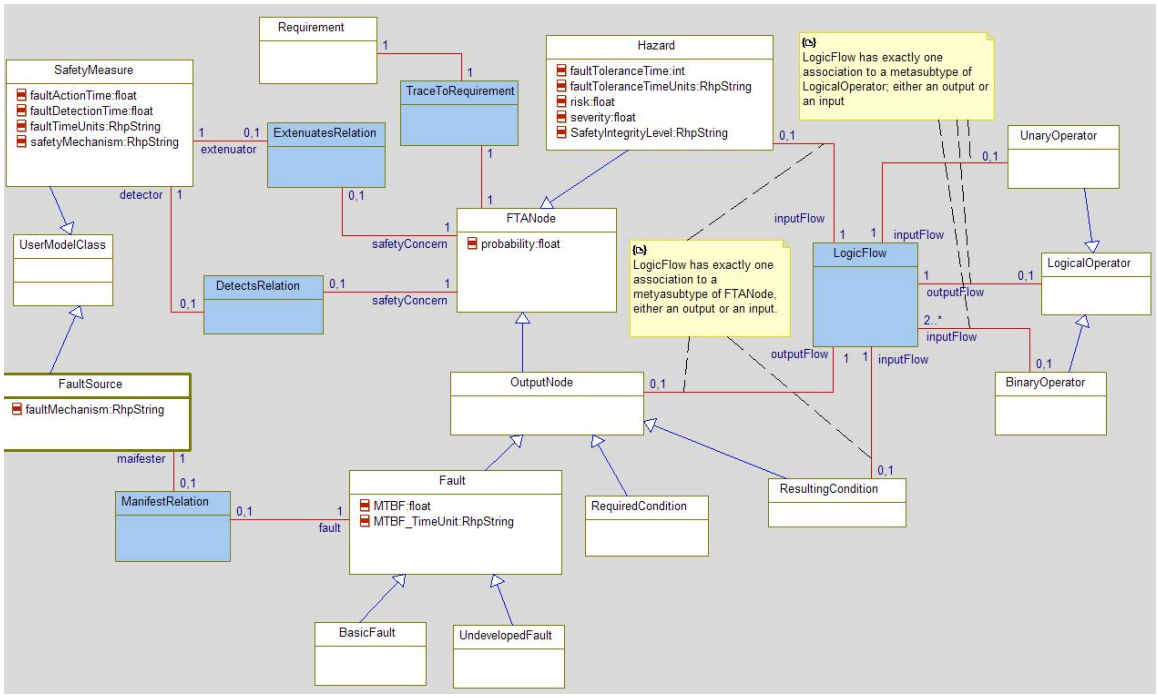


Figure 6: Safety Profile Metamodel

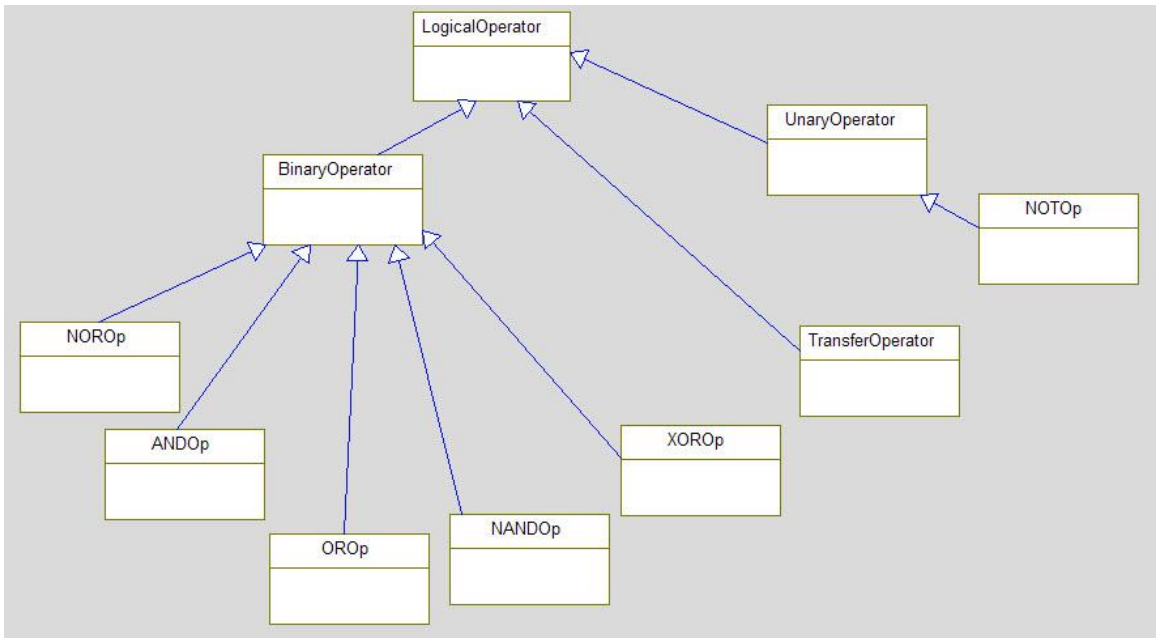


Figure 7: Metasubtypes of Logical Operators

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

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.

Table or Matrix	Format	Description
Fault Table	Rhapsody Table View	This lists[5] the faults and all their metadata
Hazard Table	Rhapsody Table View	This lists the hazards and all their metadata
Fault Source Matrix	Rhapsody Matrix View	This shows a fault x fault source matrix, as defined by the Manifests relations
Fault Detection Matrix	Rhapsody Matrix View	This shows a fault x safety measure matrix, as defined by the Detects relations
Fault Extenuation Matrix	Rhapsody 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 Hazard analysis is generated as an external file with a helper macro. This macro scans the entire model and generates the tab-separated value file[6] that can be loaded into

[5] Tables and matrices are always within a defined scope that is specified in the Features dialog for the table or matrix.

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

most spreadsheet programs. 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.

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.

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 UML and Rhapsody 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.

Medical Example: Anesthesia Machine Patient Ventilator

To illustrate the use of the profile, I created a model for a surgical anesthesia machine. This system the delivers inhalant anesthetic drugs, mixes gases, delivers gas/drug mixture to the patient via ventilation, scavenges exhaled CO_2, and monitors both patient and machine status, including blood O_2 saturation (SpO_2), inspiratory limb O_2 concentration, expiratory limb CO_2 concentration, inspiratory limb agent (drug) concentration, gas flow, and breathing circuit gas pressures.

Figure 8 shows a schematic for the SleepyTime anesthesia machine breathing circuit and important sensors.[7] The primary elements are:

[7] If the reader is interested in learning more about anesthesia machine operation, visit http://www.simanest.org/vfgs3.html. This is an interesting tutorial/simulation of the Drager Fabius GS Anesthesia Workstation.

- Gas supplies – these may be wall supplies or tanks but supply medically certified gases (O_2, N_2, N_2O, He, or Air)
- Gas Mixer – formally a part of the ventilator, this device mixes the input gases from the gas supplies as directed and outputs a mixed gas to the breathing Circuit
- Ventilator – this device shapes the breath delivered to the patient. It's primary parameters include the respiration rate (breaths/minute), tidal volume (volume per breath), I:E ratio (Inspiration time:Expiration time ratio), Inspiratory Time (seconds), and an optional inspiratory pause (delay between breaths).
- Vaporizer – this device vaporizes an anesthetic drug, such as Suprane, Halothane, or Enflourane, and delivers it to the breathing circuit in the concentration demanded.

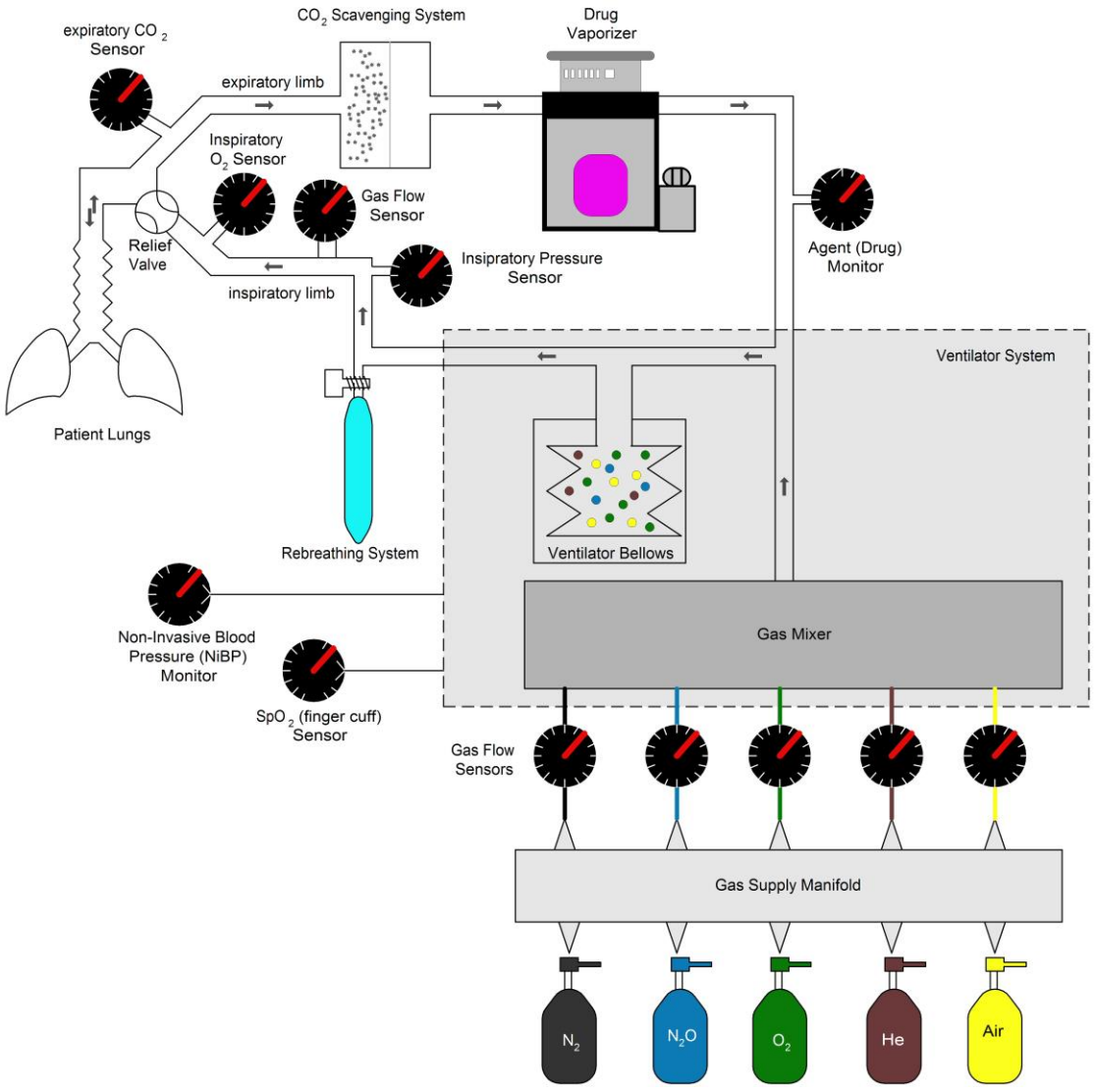


Figure 8: Anesthesia Machine Schematic

The overall use case model for the *SleepyTime Anesthesia Machine* is shown in Figure 9. Certain functionality, such as CO_2 scavenging, while important, doesn't involve the software and so is not included in this model.[8]

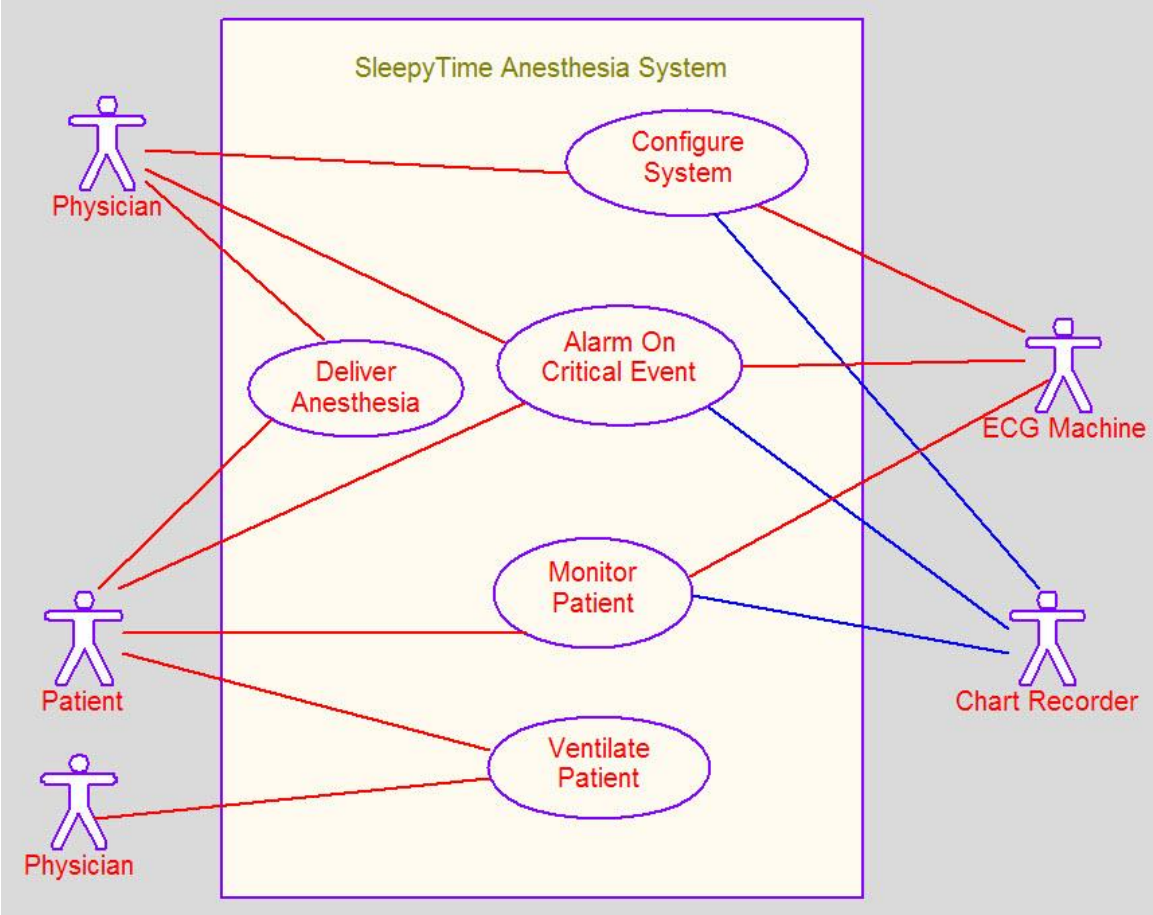


Figure 9: SleepyTime Anesthesia Machine Use Cases

Most of our discussion will focus on the patient ventilator – a device responsible for mixing incoming gases to appropriate concentrations, and shaping and delivering breaths via the breathing circuit inspiratory limb to the patient. The delivery of ventilation involves the timing of the inspiratory time and expiratory time, delivering a specified volume per breath (known as *Tidal Volume*), with either a specified Inspiration-to-expiration time ratio (known as *I:E Ratio*), or, alternatively, a specified time for inspiration (*Inspiration Time*). Additionally, a pause between breaths may be specified (*Inspiratory Pause*) as well as a rate of respiration (*Respiration Rate*). In the context of this discussion, the ventilator is a subsystem of the anesthesia machine. Figure 10 shows the use cases for the ventilator subsystem. The use of the stereotype «InternalActor» indicates that the element used as an actor in this context is actually part of the system but within another scope (e.g. another subsystem).

[8] This model is focused on the development of embedded software. If this was a system engineering model, then gas scavenging would most definitely be included.

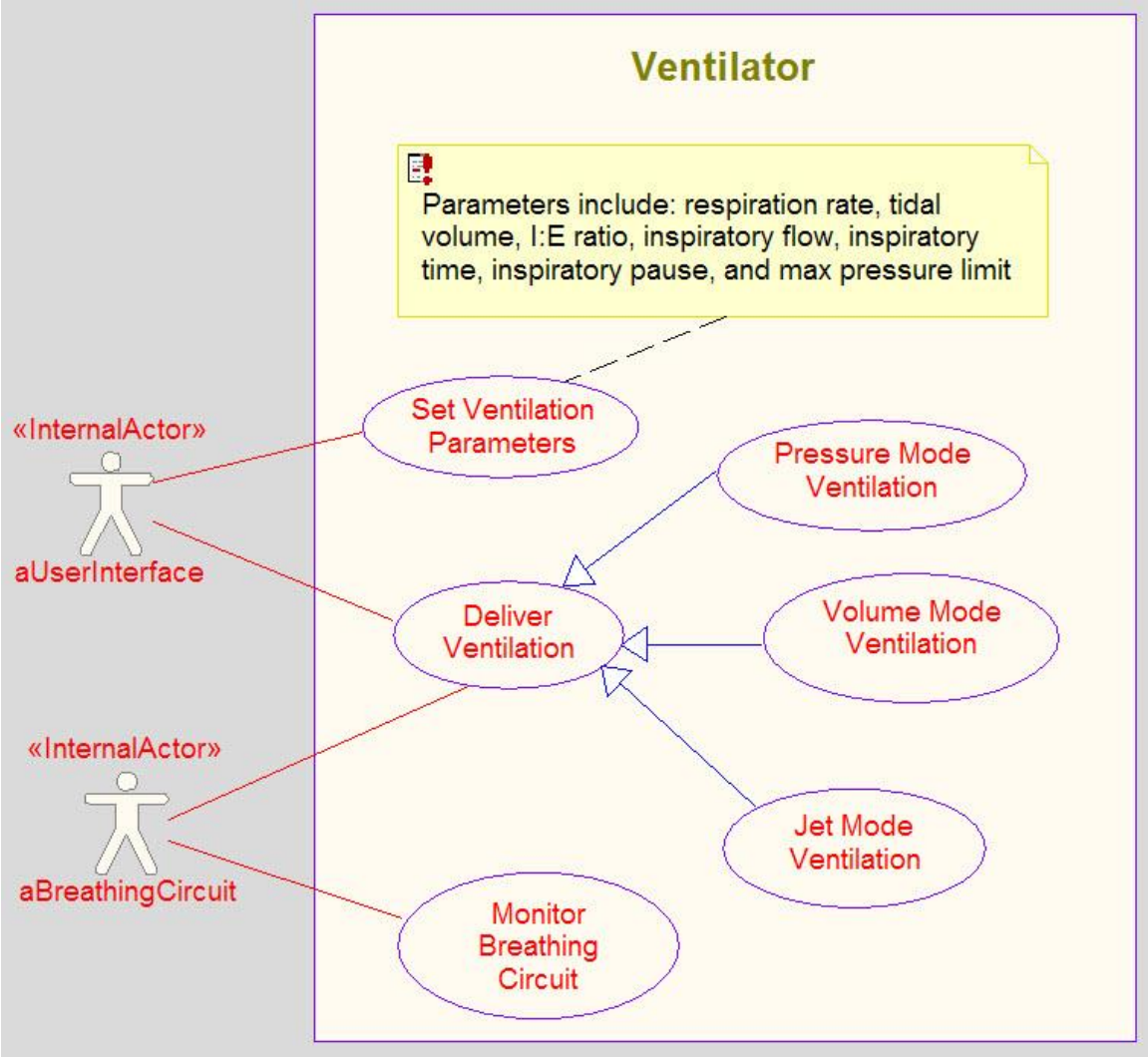


Figure 10: Ventilator Use Cases

Of course, there are a large number of requirements bound to each of the use cases. These requirements are typically managed with requirements traceability tools such as DOORS™, and can then be imported to the model and attached to the use cases and design elements for traceability. For example, you can see in Figure 11 the requirements having to do with setting the different ventilator parameters. Figure 12 and Figure 13 show similar requirements for two other use cases.

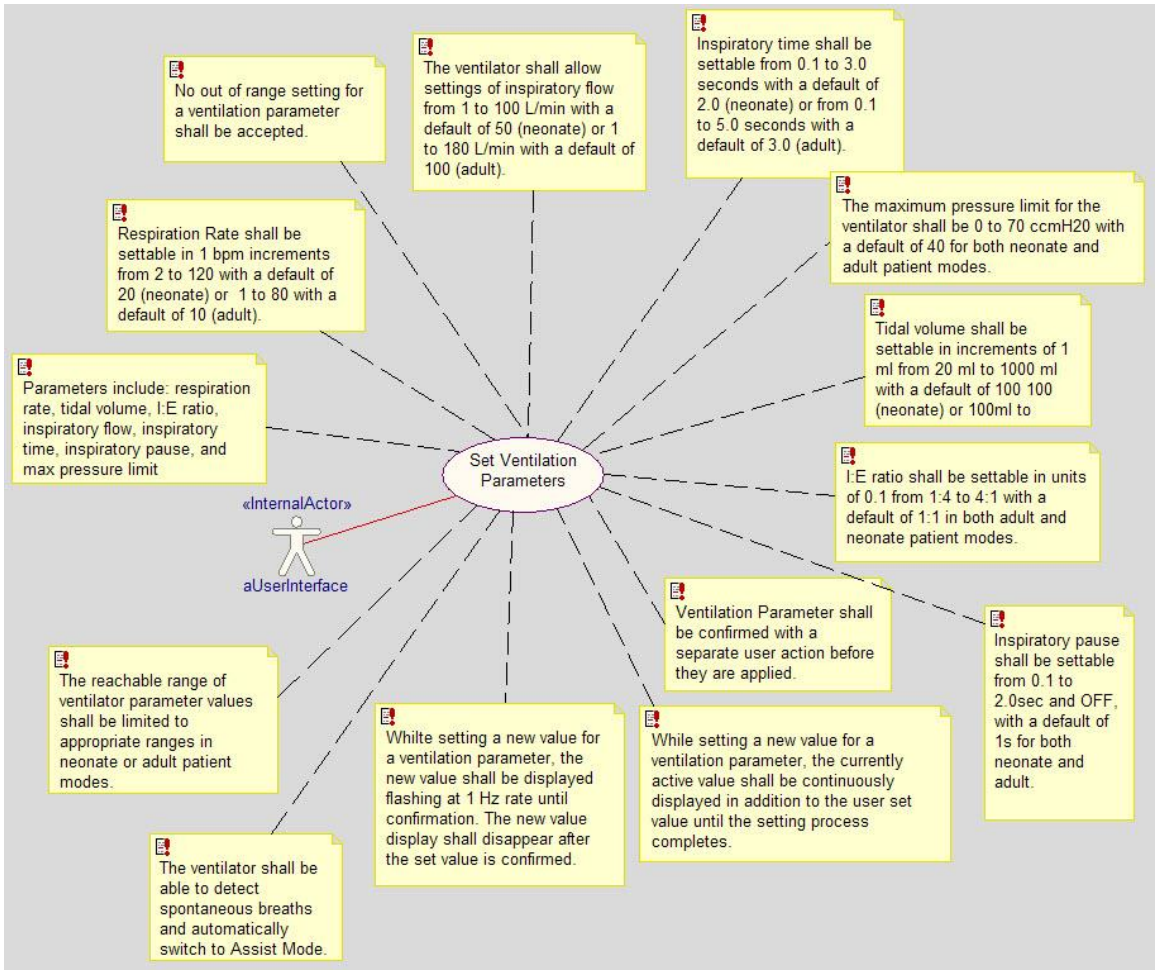


Figure 11: Requirements for use case Set Ventilation Parameters

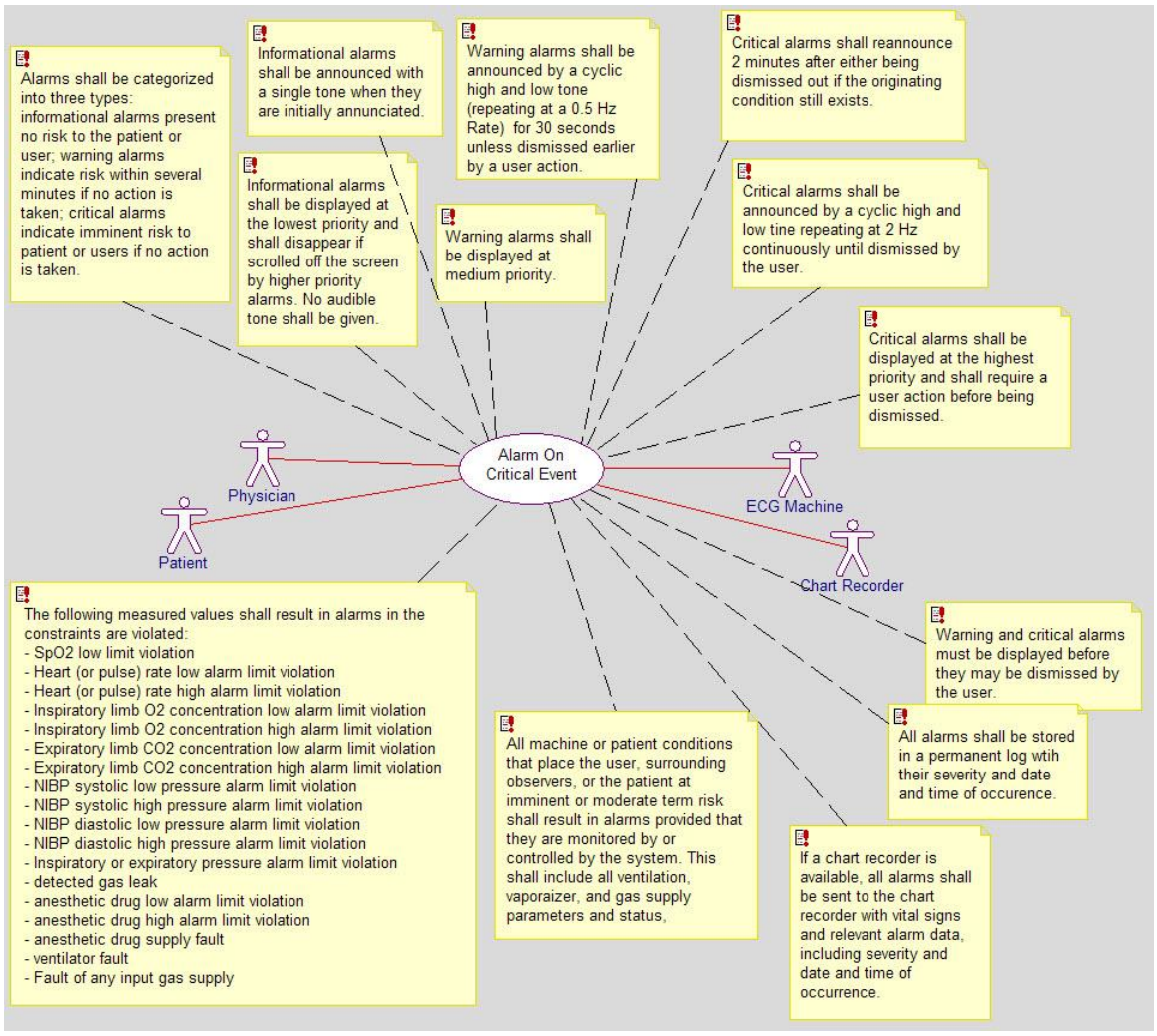


Figure 12: Requirements for use case Alarm on Critical Event

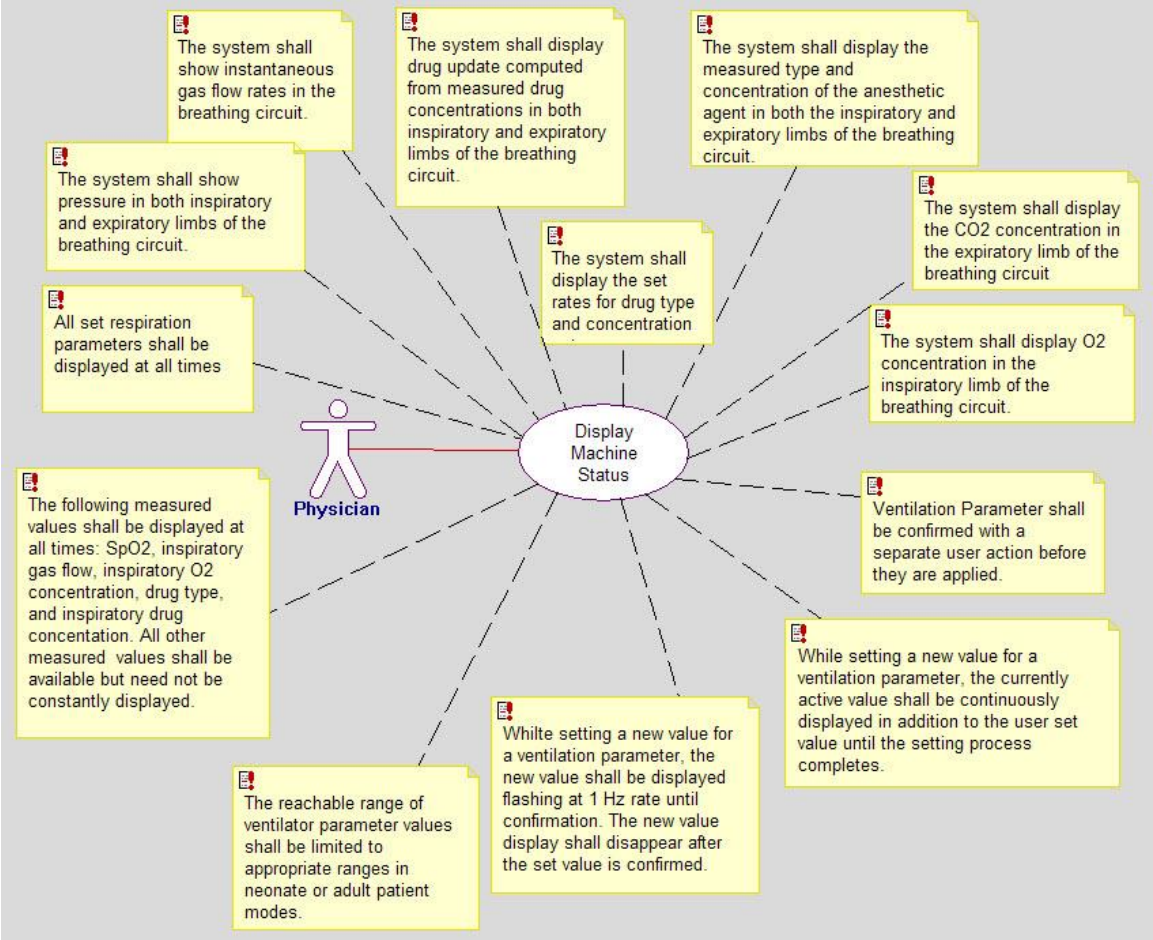


Figure 13: Requirements for use case Display Machine Status

Once we have an understanding of the requirements, we can begin the process of analyzing the system for safety. One of the expected results of such an analysis is the identification of additional requirements needed to ensure the safe operation of the system.

Let's consider a single hazard – Hypoxia. This is a very fundamental hazard for a ventilator but although just one of many[9]. We would now create a new FTA diagram and draw something like Figure 14.

[9] Others include hyperoxia (a problem that can lead to blindness is neonates), over pressure, inadequate anesthesia, over anesthesia, and leaking drugs into the operating room environment.

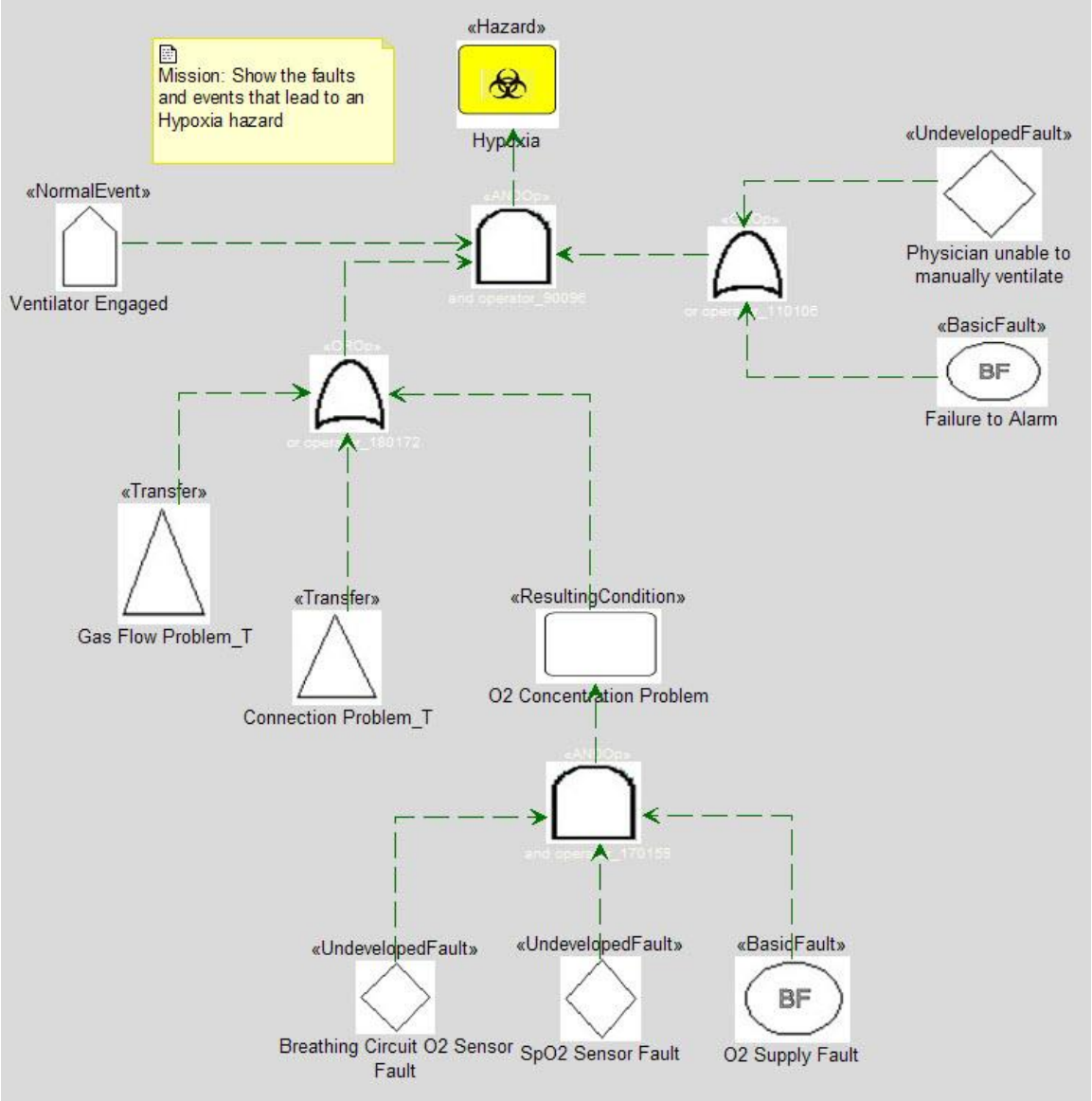


Figure 14: FTA for hazard Hypoxia

Figure 14 shows the basic structure of the FTA for the Hypoxia hazard. The Resulting Condition *O_2 Concentration Problem* occurs only if all of the following faults occur: the breathing circuit O_2 sensor fault, the SpO_2 sensor fault, and the O_2 Supply fault. The last of these is the primary fault while the other two are faults in what are known as safety measures – elements and behaviors added to the system specifically to address safety concerns. In this case, we add both a breathing circuit sensor and an SpO_2 finger cuff sensor to improve safety. The hazardous condition due to a fault in the O_2 gas supply can only occur if *both* the SpO_2 and breathing circuit O_2 sensor have faults. This is what we mean by AND-ing redundancy. For the hazardous condition to occur both the original fault AND the fault in the safety measure must occur.

Furthering this analysis, we see that for an O_2 concentration problem to result in the Hypoxia hazard, other conditions must be true. Specifically, the ventilator must be in use and the attending physician doesn't take proper action – either because he or she doesn't know they should (*Failure to Alarm* fault) or because for some reason they are unable to (*Physician unable to manually ventilate*).

Besides an O_2 concentration problem, other faults could also result in Hypoxia, subject to the same overall conditions discussed in the previous paragraph. There could be a problem delivering the gas (*Gas Flow Problem*) or the patient could be disconnected from the breathing circuit (*Connection Problem*). These latter concerns are somewhat involved and including them in this diagram would make it difficult to read. For this reason, transfer operators connect them logically into their appropriate logical position within this diagram even though they are drawn on a separate diagram.

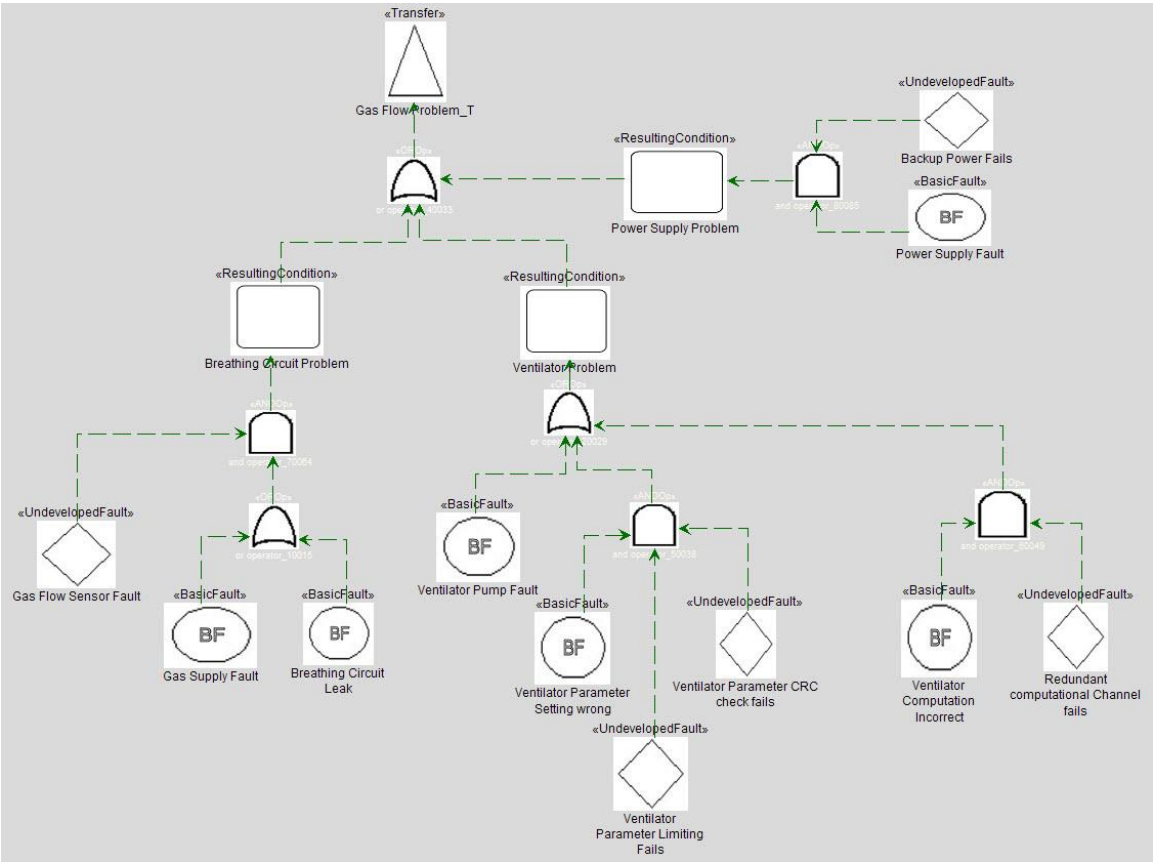


Figure 15: FTA for Gas Flow Problem subdiagram

Figure 15 elaborates the FTA for the Resulting Condition "Gas Flow Problem". Here again we see that basic faults must be ANDed with the faults of safety measures for the hazard to be realized. This analysis leads to the identification of new requirements for the safety measures – such as gas flow sensors, CRCs on parameter settings, and redundant computation of ventilator control.

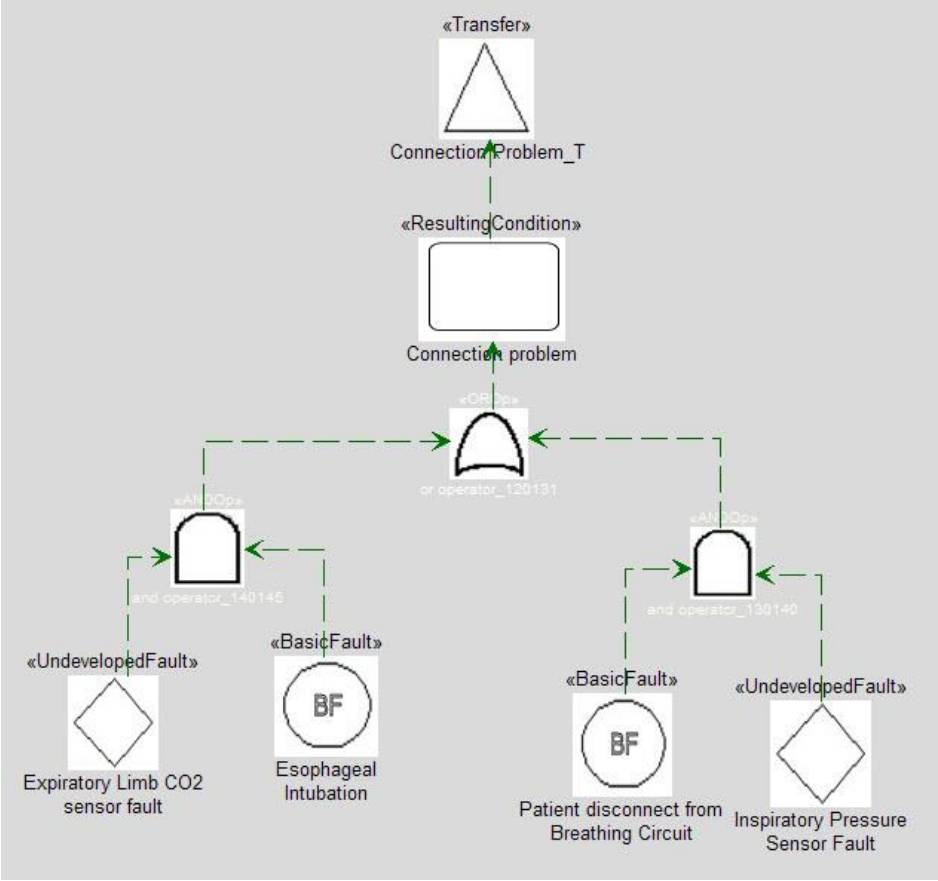


Figure 16: FTA for Connection Problem subdiagram

Figure 16 points out the last concern for the Hypoxia hazard we will consider – the problem of misconnection or disconnection of the patient from the breathing circuit. A very common problem is improper intubation[10]. The most common physician fault is intubation of the esophagus. Esophageal intubation cannot be detected with gas flow sensors, pressure sensors, or even O_2 sensors. The selected safety measure is to add a CO_2 sensor on the expiratory limb. Since the only thing in the entire system that inserts CO_2 into the breathing circuit is the patient's lungs[11], so if the expiratory limb doesn't see an increased CO_2 concentration, then either the patient isn't producing CO_2 (a very bad sign) or the expiratory limb of the breathing circuit isn't connected to the patient's lungs.

The identified faults can be characterized with the fault metadata by filling in the identified tag fields. Figure 17 shows a portion of the generated fault table summarizing the faults.

[10] Intubation is the insertion of the breathing circuit connection into the patient trachea.
[11] In Great Britain, it is common practice for anesthesia machines to actually drive CO_2 into the patient lungs when trying to wake them after the completion of a surgical procedure, so this is not strictly true in the UK.

Name	Description	MTBF	MTBF_TimeUnits	Probability
Gas Supply Fault	This fault occurs when gas from a required source (e.g. O2 air N2 or He). This may be to any number of root causes such as a stuck or closed valve, running out of gas, a leak_	1e6	minutes	1e-6
Breathing Circuit Leak	This fault occurs when a significant amount of gas leaks from the breathing circuit into the	1e3	minutes	1e-3
Ventilator Pump Fault	This fault occurs when the pump internal to the ventilator no longer functions to shape the	1e6	seconds	1e-6
Ventilator Parameter Setting wrong	This fault occurs when a ventilator parameter is out of range. This includes: I:E ratio Tidal Volume Respiration Rate Inspiratory Pause Maximum inspiratory pressure Inspiration time	1e4	seconds	1e-4
Ventilator Computation Incorrect	This fault occurs when an error in the software or a fault in a necessary resource (e.g.	1e5	seconds	1e-5
Esophageal Intubation	This is a user-fault, but is common. This is mitigated by a CO2 sensor on the expiratory	1e5	minutes	1e-4
Patient disconnect from Breathing Circuit	This fault can occur as a result of jostling the breathing circuit during a surgical procedure.	1e4	minutes	1e-4
Power Supply Fault	The mains can fail because of a source power supply fault or if the power cord becomes	1e5	minutes	1e-5
Failure to Alarm	The alarm system is a system that exists solely for safety reasons. Therefore, it need not	1e5	minutes	1e-5
O2 Supply Fault	The O2 supply fault can occur because of a exhaustion of the supply itself, stuck or	1e4	seconds	1e-4
Breathing Circuit Problem				
Ventilator Problem				
Power Supply Problem				
Connection problem				
O2 Concentration Problem				
Redundant computational Channel fails	The redundant computational channel uses a heterogeneous algorithm to compute the	1e5	seconds	1e-5
Ventilator Parameter Limiting Fails	This fault occurs if the limit checks on the setting of ventilator parameters fail, i.e. allow a	1e6	seconds	1e-6
Gas Flow Sensor Fault	This fault occurs if the gas flow sensor fails to correctly measure the gas flow in the	1e-7	minutes	1e-7
Ventilator Parameter CRC check fails	Ventilator parameters are protected with a 32-bit CRC algorithm. This is specifically	1e5	seconds	1e-5
Backup Power Fails	The battery backup exists as a safety means to enable the system to continue to provide	1e4	minutes	1e-4
Physician unable to manually ventilate	The anesthesiologist is required to have a manual ventilation system available in the case	1e10	minutes	1e-10
SpO2 Sensor Fault	The SpO2 sensor is a fingercuff O2 sensor. This fault occurs if the sensor does not	1e7	seconds	1e-7
Breathing Circuit O2 Sensor Fault	The breathing circuit O2 sensor is provided to ensure that the O2 delivered from the	1e7	seconds	1e-7
Inspiratory Pressure Sensor Fault	The inspiratory pressure sensor is used to determine that the pressures delivered to the	1e7	seconds	1e-7
Expiratory Limb CO2 sensor fault	The expiratory limb CO2 sensor exists to ensure that the breathing circuit is properly	1e7	seconds	1e-7

Figure 17: Fault Table

One of the results for the safety analysis is the discovery of additional requirements to enhance safety. We can tie the requirements to the faults identified in the FTA. For example, Figure 18 shows the FTA for the Connection Problem linked to the relevant requirements with the Trace To Requirement relation. This is important because it binds the safety analysis directly to the requirements. The result of this process is captured in the generated Fault-Requirement Matrix, a portion of which is shown in Figure 19.

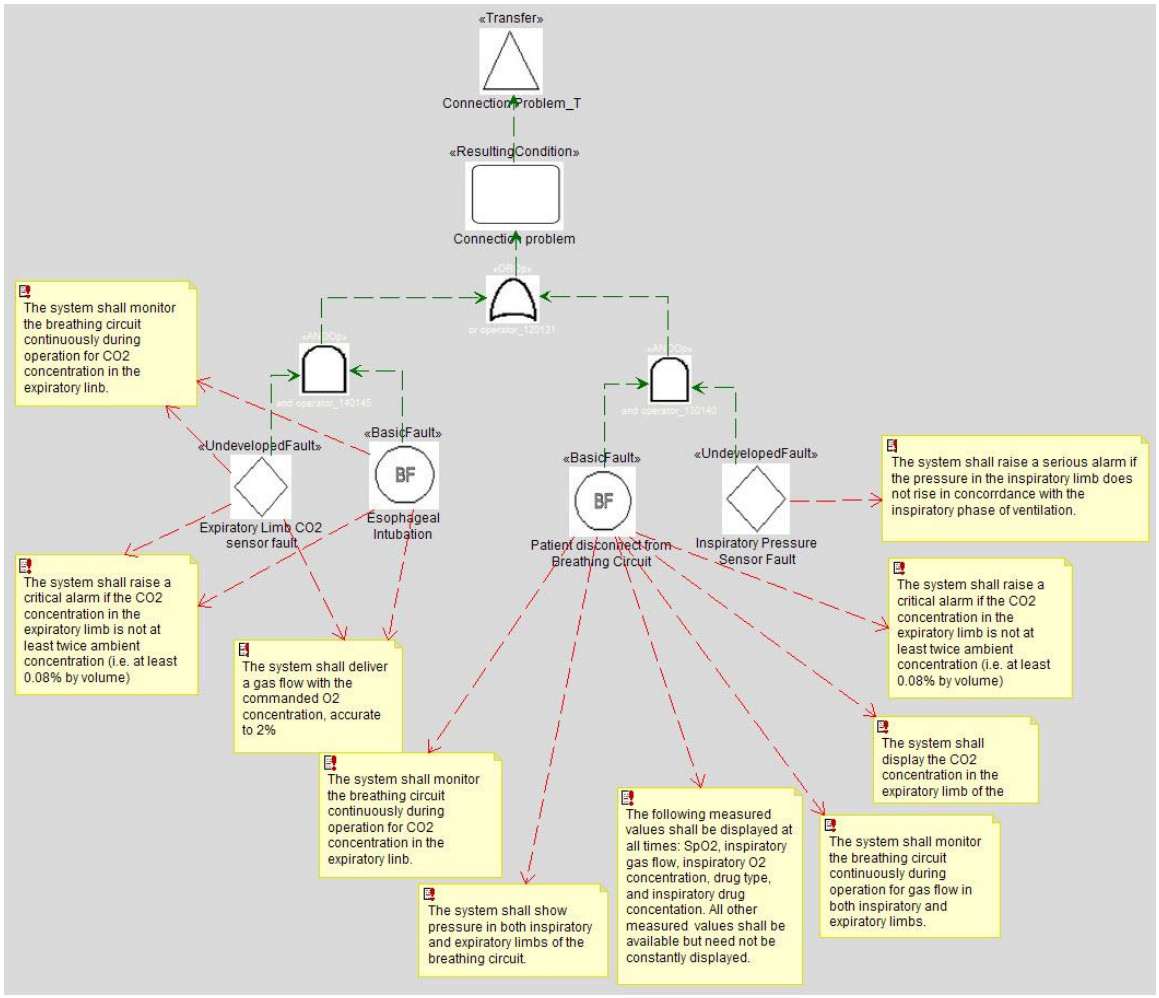
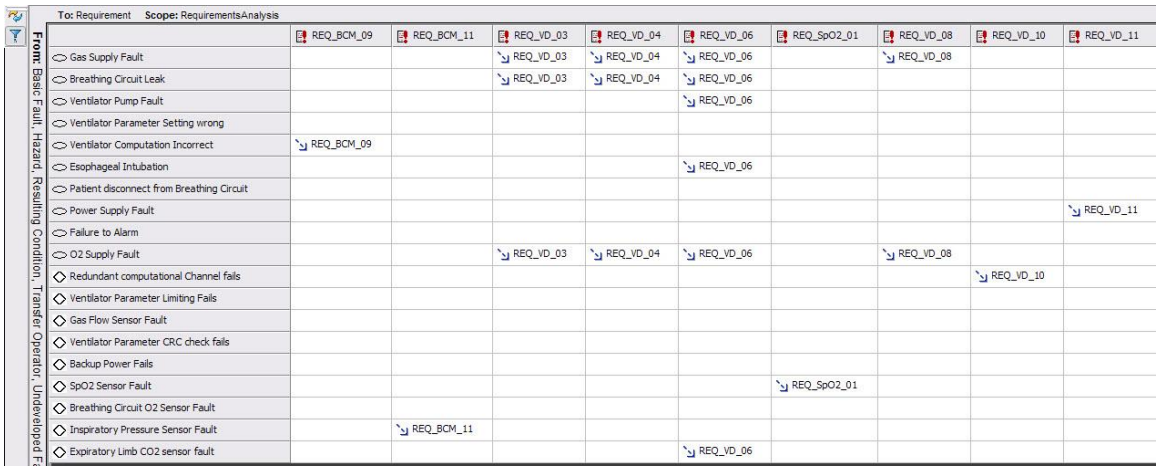


Figure 18: Faults linked to Requirements



To: Requirement Scope: RequirementsAnalysis

From: Basic Fault, Hazard, Resulting Condition, Transfer Operator, Undeveloped Fa

	REQ_BCM_09	REQ_BCM_11	REQ_VD_03	REQ_VD_04	REQ_VD_06	REQ_SpO2_01	REQ_VD_08	REQ_VD_10	REQ_VD_11
Gas Supply Fault			REQ_VD_03	REQ_VD_04	REQ_VD_06		REQ_VD_08		
Breathing Circuit Leak			REQ_VD_03	REQ_VD_04	REQ_VD_06				
Ventilator Pump Fault					REQ_VD_06				
Ventilator Parameter Setting wrong									
Ventilator Computation Incorrect	REQ_BCM_09								
Esophageal Intubation					REQ_VD_06				
Patient disconnect from Breathing Circuit									
Power Supply Fault									REQ_VD_11
Failure to Alarm									
O2 Supply Fault			REQ_VD_03	REQ_VD_04	REQ_VD_06		REQ_VD_08		
Redundant computational Channel fails								REQ_VD_10	
Ventilator Parameter Limiting Fails									
Gas Flow Sensor Fault									
Ventilator Parameter CRC check fails									
Backup Power Fails									
SpO2 Sensor Fault						REQ_SpO2_01			
Breathing Circuit O2 Sensor Fault									
Inspiratory Pressure Sensor Fault		REQ_BCM_11							
Expiratory Limb CO2 sensor fault					REQ_VD_06				

Figure 19: Fault-Requirement Matrix

During subsequent analysis and design, the UML model of the design elaborates. Figure 20 shows a model system emphasizing the vaporizer and their relations with other elements while Figure 21 shows the primary classes within the Ventilator subsystem.

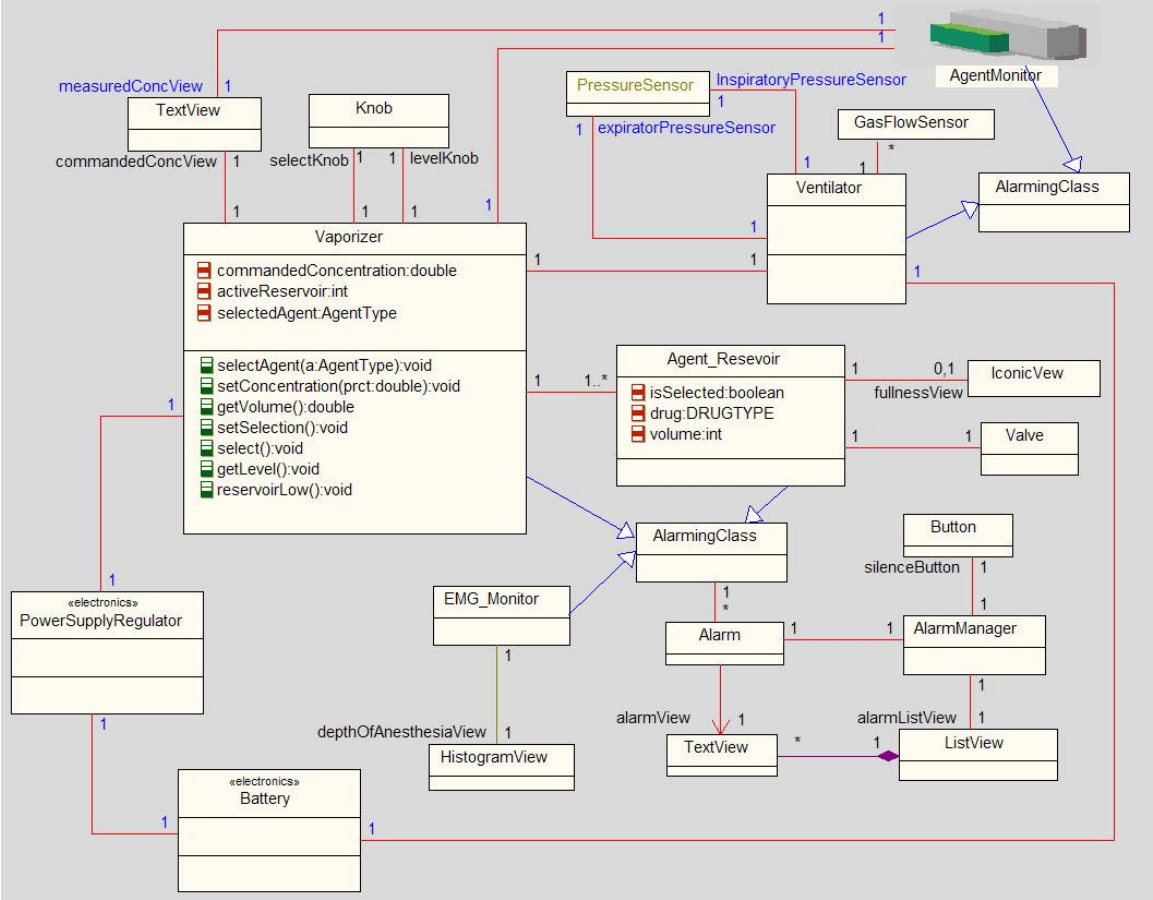


Figure 20: SleepyTime Subsystems

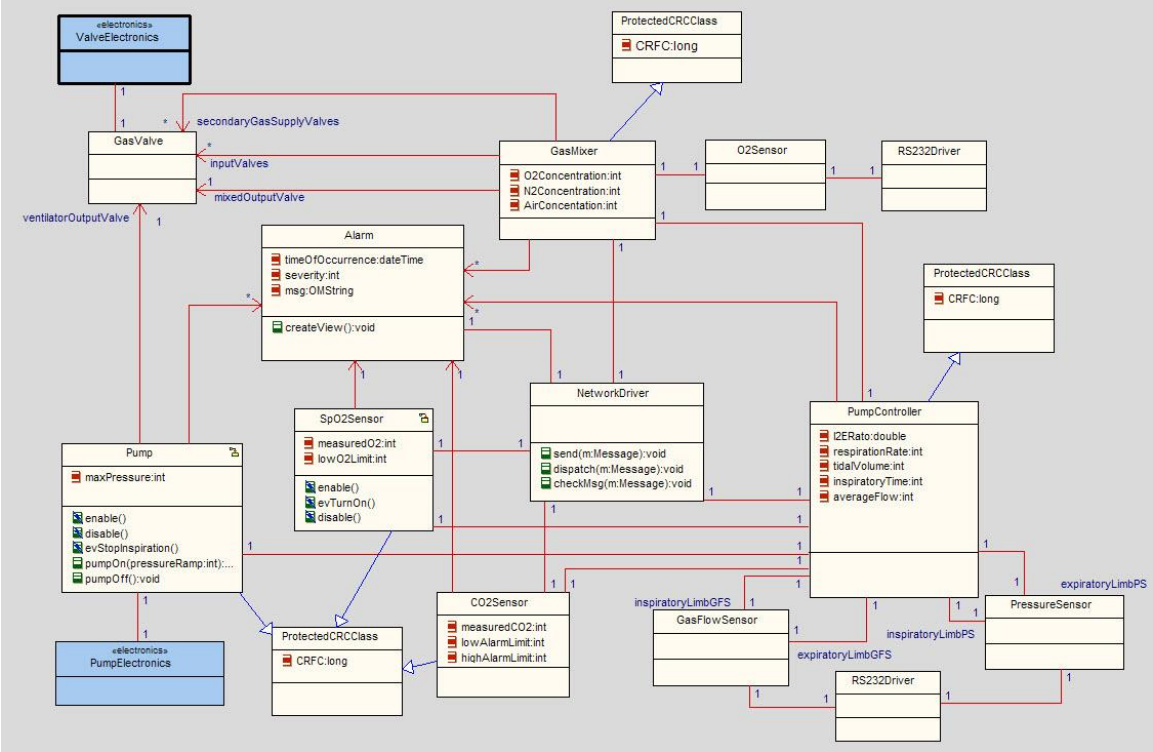


Figure 21: Ventilator Design Model

Figure 21 shows some electronic components with the stereotype «electronics» and colored in blue. The other elements are all software classes that collaborate together to realize the ventilator use cases.

During the design and development work, new hazards may be added – for example, the selection of bottled O_2 might result in a pressure explosion hazard – as well as questions as to whether or not the design appropriately addresses the safety concerns. At this point, we can elaborate the FTA model with that information and we can add specific links from the profile from the faults to the elements in the model that can manifest the faults or that detect or extenuate the faults.

The safety analysis profile also supports linking the analysis and design elements to the faults using the Manifests (to Fault Sources), Detects and Extenuates (to Safety Measures) relations. Figure 22 and Figure 23 show examples of such elaborated FTA diagrams.

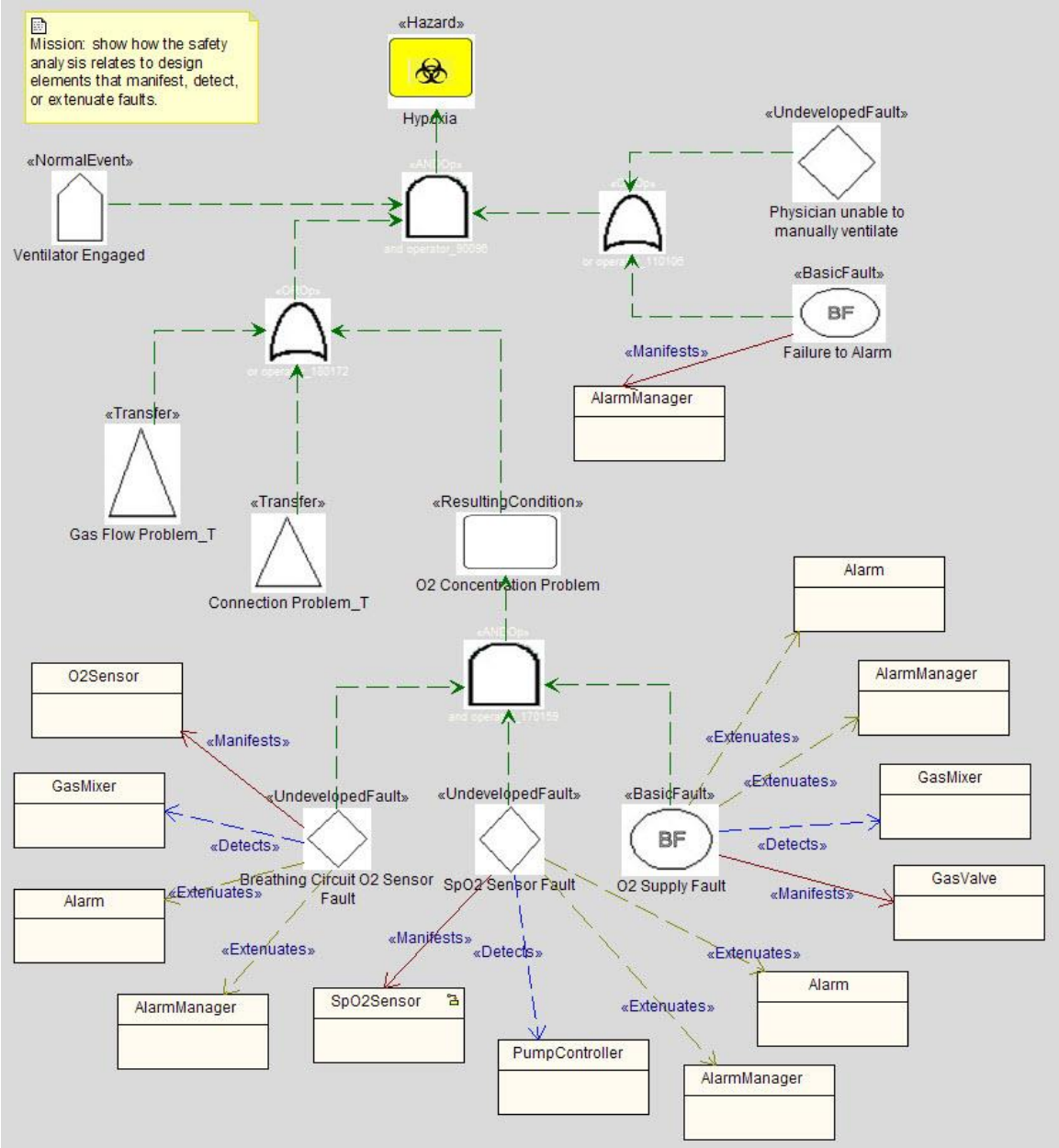


Figure 22: FTA with Design Element Links

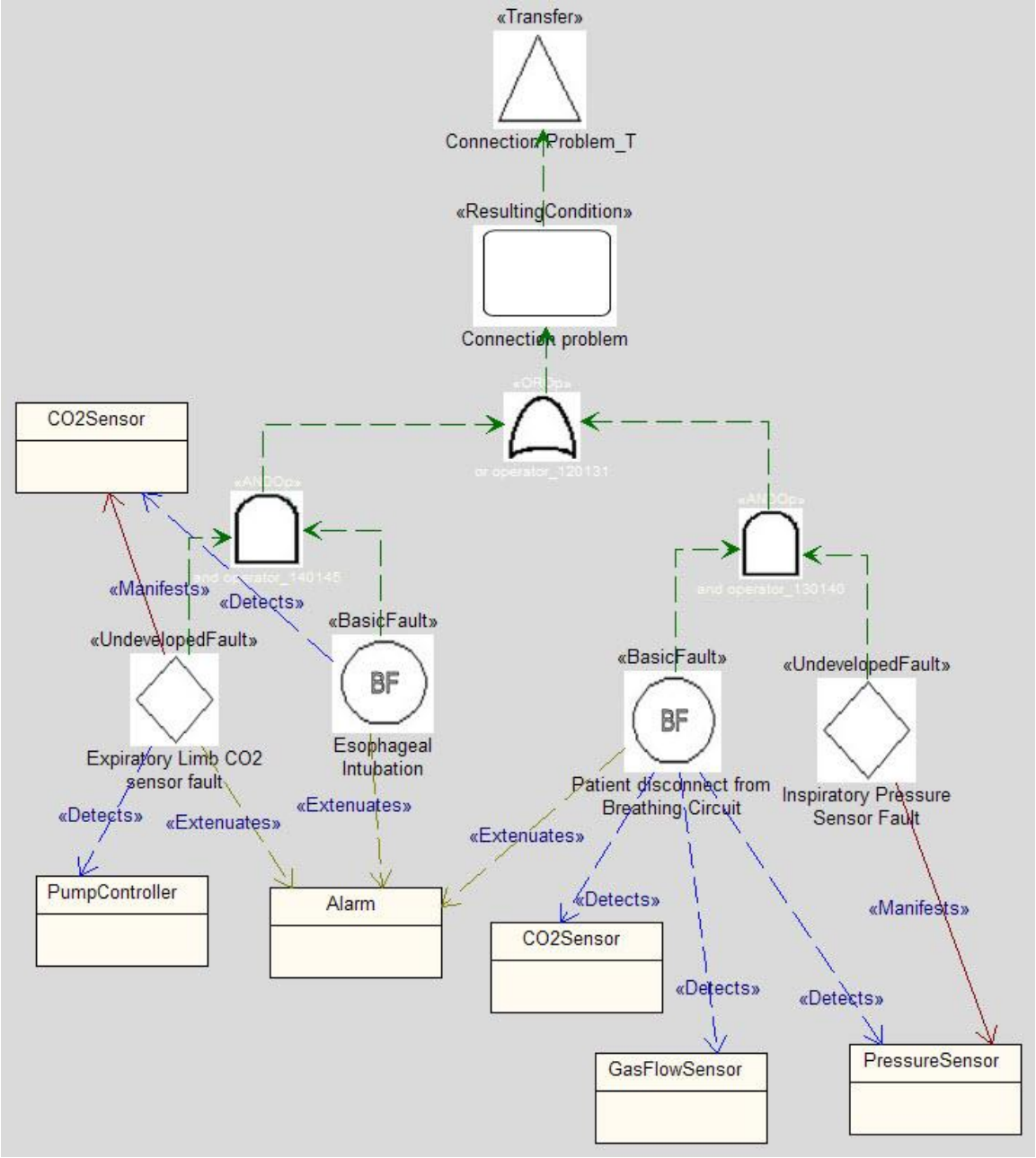


Figure 23: FTA with Design Elements 2

Not only is this information visually apparent. It is also represented in the Fault Source Matrix, Fault Detection Matrix, and Fault Extenuation Matrix. For example, Figure 24 shows a portion of the matrix of the faults and the design elements that can manifest them while Figure 25 shows the same view for the faults and the design elements that detect them.

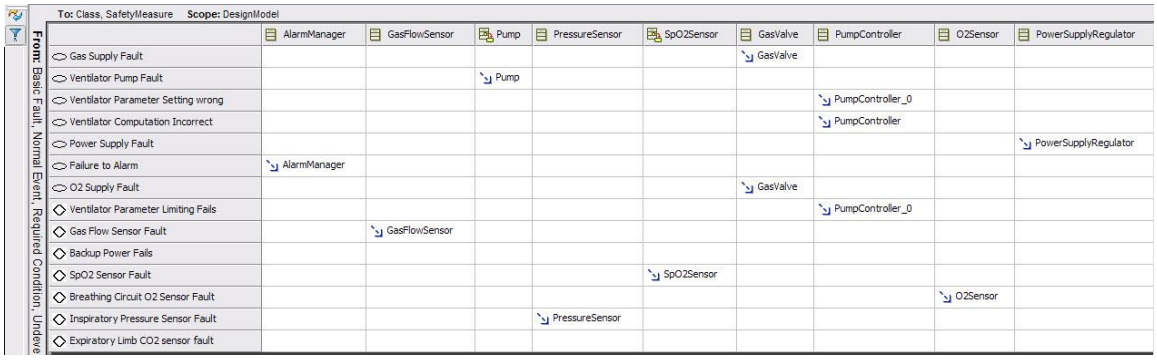


Figure 24: Fault Source Matrix

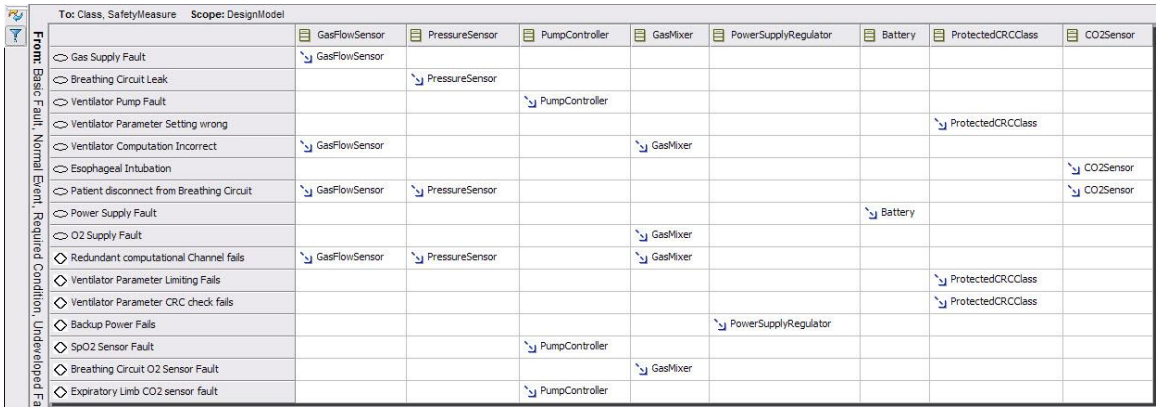


Figure 25: Fault Detection Matrix

In addition, a complete hazard analysis can be generated from the annotated fault model. This is generated as an external Tab-Separated Value (.tsv) file. This file is placed automatically in the main directory of the model and may be added as a controlled file into the model. This file can be read by most spreadsheet programs, although you may have to customize the registry to open the appropriate application if the spreadsheet program doesn't do that for you automatically.

The hazard analysis consists of three sections. The first shows the hazards and the metadata from the safety model. The output from this model is shown in Table 3.

Hazard	Description	Fault Tolerance Time	Fault Tolerance Time Units	Probability	Severity	Risk	Safety Integrity Level
Hypoxia	The hypoxia hazard occurs when the brain and other organs receive insufficient oxygen. In a normal 21% O2 environment, death or irreversible injury occurs after 5 minutes of no oxygen. If the patient is breathing 100% for a significant period of	5	minutes	1.00E-02	8	8.00E-02	3

	time, this time is about 10 minutes.						
Overpressre	Overpressure can damage the lungs. This is an especially severe trauma, possibly fatal, to neonates.	200	milliseconds	1.00E+04	4	3.00E+04	3
Hyperoxia	Hyperoxia problems are usually limited to neonates, where it can cause blindness.	10	minutes	1.00E+05	4	4.00E+05	4
Inadequate Anesthesia	In adequate anesthesia leads to patient discomfort and memory retention of the surgical procedures. This is normally not life threatening but can be severely discomforting.	5	minutes	1.00E+04	2	2.00E+04	2
Over anesthesia	Over anesthesia can lead to death.	3	minutes	1.00E+03	4	4.00E+03	4
Anesthesia leak into ER	Anesthesia leak can lead to short or, in smaller doses, to long-term poisoning of medical staff.	10	minutes	1.00E+05	5	4.00E+05	5

Table 3: Hazard Analysis Part 1 - Hazard Metadata

The second part of the hazard analysis summarizes the relations between the faults and the hazards. This involves the tracing of multiple levels of logic flows connecting the faults with the hazards. The output for this model is shown in Table 4

Hazard	Fault or Event	Fault Type	Fault Description	MTBF	MTBF Time Units	Probability
Hypoxia	Ventilator Engaged	NormalEvent				1
Hypoxia	Gas Supply Fault	BasicFault	This fault occurs when gas from a required source (e.g. O2 air N2 or He). This may be to any number of root causes such as a stuck or closed valve, running out of gas, a leak_ etc.	1.00E+06		1.00E-06
Hypoxia	Breathing Circuit Leak	BasicFault	This fault occurs when a significant amount of gas leaks from the breathing circuit into the surrounding environment. This can lead to a poisoning hazard when the gas contains anesthetic drugs.	1.00E+03		1.00E-03
Hypoxia	Ventilator Pump Fault	BasicFault	This fault occurs when the pump internal to the ventilator no longer functions to shape the breath and push gas into	1.00E+06		1.00E-06

			the breathing circuit.		
Hypoxia	Ventilator Parameter Setting wrong	BasicFault	This fault occurs when a ventilator parameter is out of range. This includes: -I:E ratio -Tidal Volume - Respiration Rate - Inspiratory Pause - Maximum inspiratory pressure -Inspiration time	1.00E+04	1.00E-04
Hypoxia	Ventilator Computation Incorrect	BasicFault	This fault occurs when an error in the software or a fault in a necessary resource (e.g. memory) results in an incorrect computation that in turn results in incorrect delivery of ventilation.	1.00E+05	1.00E-05
Hypoxia	Redundant computational Channel fails	UndevelopedFault	The redundant computational channel uses a heterogeneous algorithm to compute the output values as a check on the primary. Since there are only two computational channels, if one is in error, the system cannot determine which channel is in error, only that an error has occurred.	1.00E+05	1.00E-05
Hypoxia	Ventilator Parameter Limiting Fails	UndevelopedFault	This fault occurs if the limit checks on the setting of ventilator parameters fail, i.e. allow a value to be entered that is out of the allowed range, given the mode (neonate or adult) of the system.	1.00E+06	1.00E-06
Hypoxia	Gas Flow Sensor Fault	UndevelopedFault	This fault occurs if the gas flow sensor fails to correctly measure the gas flow in the breathing circuit limb to which it is attached, or if it fails to send that information to the system.	1.00E-07	1.00E-07
Hypoxia	Ventilator Parameter CRC check fails	UndevelopedFault	Ventilator parameters are protected with a 32-bit CRC algorithm. This is specifically designed to identify situations in which the value has been changed through inappropriate means (e.g. memory cell fault). A fault here means that the CRC fails to identify the corruption of the parameter.	1.00E+05	1.00E-05
Hypoxia	Esophageal Intubation	BasicFault	This is a user-fault, but is common. This is mitigated by a CO2 sensor on the expiratory limb of the breathing circuit.	1.00E+05	1.00E-04
Hypoxia	Patient disconnect from Breathing Circuit	BasicFault	This fault can occur as a result of jostling the breathing circuit during a surgical procedure.	1.00E+04	1.00E-04

Hypoxia	Power Supply Fault	BasicFault	The mains can fail because of a source power supply fault or if the power cord becomes unplugged.	1.00E+05	1.00E-05
Hypoxia	Backup Power Fails	UndevelopedFault	The battery backup exists as a safety means to enable the system to continue to provide therapy and monitoring when mains fail. This fault means that the backup system is unable to provide that backup.	1.00E+04	1.00E-04
Hypoxia	Physician unable to manually ventilate	UndevelopedFault	The anesthesiologist is required to have a manual ventilation system available in the case of an unrecoverable system failure. This fault may occur because that manual system is missing or nonfunctional or if the system has alarmed but the physician is unaware of the alarm or of the need for immediate action.	1.00E+10	1.00E-10
Hypoxia	Failure to Alarm	BasicFault	The alarm system is a system that exists solely for safety reasons. Therefore, it need not be extenutated by another system since it exists solely to address safety issues of the primary systems. It must, however, be tested as a part of system start up.	1.00E+05	1.00E-05
Hypoxia	SpO2 Sensor Fault	UndevelopedFault	The SpO2 sensor is a fingercuff O2 sensor. This fault occurs if the sensor does not accurately determine the blood concentration of O2 or if the sensor is unable to communicate its readings to the system.	1.00E+07	1.00E-07
Hypoxia	Breathing Circuit O2 Sensor Fault	UndevelopedFault	The breathing circuit O2 sensor is provided to ensure that the O2 delivered from the system matches expectations. This fault means that it is unable to either determine the O2 concentration or unable to communicate that information.	1.00E+07	1.00E-07
Hypoxia	Inspiratory Pressure Sensor Fault	UndevelopedFault	The inspiratory pressure sensor is used to determine that the pressures delivered to the patient lungs are within min and max limits and that they match the expectations of the system based on the delivery of the shaped breath. This fault means that the sensor is either unable to determine pressure accurately or that it cannot communicate these values to the system.	1.00E+07	1.00E-07

Hypoxia	Expiratory Limb CO2 sensor fault	UndevelopedFault	The expiratory limb CO2 sensor exists to ensure that the breathing circuit is properly connected to the patient - if there is inadequate CO2 in the expiratory limb than either the patient isn't generating CO2 or the expiratory limb is disconnected from the patient. This fault means that the sensor is either unable to accurately determine the CO2 concentration or is unable to communicate those values to the system.	1.00E+07	1.00E-07
Hypoxia	O2 Supply Fault	BasicFault	The O2 supply fault can occur because of a exhaustion of the supply itself, stuck or incorrectly commanded valves, or a problem in the supply line to the ventilator.	1.00E+04	1.00E-04

Table 4: Hazard analysis Part 2 - Hazard Fault Matrix

Lastly, the hazard analysis contains the relations between all faults and the elements of the model, including requirements, and classes that manifest, detect, or extenuate faults. This view is crucial for a detailed understanding of the correctness and safety of a design model. Table 5 shows the output for the example model.

Fault or Event	Requirements	Manifestors	Detectors	Extenuators
Gas Supply Fault	REQ_BCM_01	GasValve	GasFlowSensor	Alarm
Gas Supply Fault	REQ_VD_06			
Gas Supply Fault	REQ_VD_03			
Gas Supply Fault	REQ_VD_04			
Gas Supply Fault	REQ_VD_08			
Breathing Circuit Leak	REQ_VD_03		PressureSensor	Alarm
Breathing Circuit Leak	REQ_VD_04			
Breathing Circuit Leak	REQ_VD_06			
Ventilator Pump Fault	REQ_VD_06	Pump	PumpController	PumpController
Ventilator Parameter Setting wrong	REQ_vent_limit_range_on_patient_mode	PumpController	ProtectedCRCClass	Alarm
Ventilator Parameter Setting wrong	REQ_vent_parameter_out_of_range_setting			
Ventilator Parameter Setting wrong	REQ_Vent_confirmation			
Ventilator Computation Incorrect	REQ_BCM_06	PumpController	GasFlowSensor	Alarm
Ventilator Computation Incorrect	REQ_BCM_07		GasMixer	
Ventilator Computation Incorrect	REQ_BCM_08			
Ventilator Computation Incorrect	REQ_BCM_09			

Redundant computational Channel fails	REQ_VD_10		PressureSensor	Alarm
Redundant computational Channel fails			GasFlowSensor	
Redundant computational Channel fails			GasMixer	
Ventilator Parameter Limiting Fails	REQ_vent_parameter_out_of_range_setting	PumpController	ProtectedCRCClass	Alarm
Ventilator Parameter Limiting Fails	REQ_vent_limit_range_on_patient_mode			
Gas Flow Sensor Fault	REQ_BCM_03	GasFlowSensor		Alarm
Ventilator Parameter CRC check fails	REQ_vent_parameter_out_of_range_setting		ProtectedCRCClass	Alarm
Ventilator Parameter CRC check fails	REQ_vent_limit_range_on_patient_mode			
Esophageal Intubation	REQ_BCM_02		CO2Sensor	Alarm
Esophageal Intubation	REQ_BCM_07			
Esophageal Intubation	REQ_VD_06			
Patient disconnect from Breathing Circuit	REQ_Display_Pressures		CO2Sensor	Alarm
Patient disconnect from Breathing Circuit	REQ_BCM_02		GasFlowSensor	
Patient disconnect from Breathing Circuit	REQ_Display_Status_constantly		PressureSensor	
Patient disconnect from Breathing Circuit	REQ_Display_CO2			
Patient disconnect from Breathing Circuit	REQ_BCM_03			
Patient disconnect from Breathing Circuit	REQ_BCM_07			
Power Supply Fault	REQ_VD_11	PowerSupplyRegulator	Battery	Battery
Backup Power Fails	REQ_VD_12	Battery	PowerSupplyRegulator	PowerSupplyRegulator
Failure to Alarm	REQ_Chart_recorder_alarms	AlarmManager		
Failure to Alarm	REQ_Alarm_retention			
Failure to Alarm	REQ_Alarm_categories			
Failure to Alarm	REQ_Warning_sounds			
Failure to Alarm	REQ_Dismiss_alarms			
Failure to Alarm	REQ_Critical_reannounciation			
Failure to Alarm	REQ_Critical_alarms			
Failure to Alarm	REQ_Alarm_condition			
Failure to Alarm	REQ_informational_alarms			
Failure to Alarm	REQ_Critical_alarm_sounds			
Failure to Alarm	REQ_warning_alarms			
Failure to Alarm	REQ_Informational_alarms			
Failure to Alarm	REQ_patient_alarms			
SpO2 Sensor Fault	REQ_SpO2_01	SpO2Sensor	PumpController	AlarmManager
SpO2 Sensor Fault				Alarm

Breathing Circuit O2 Sensor Fault	REQ_BCM_01	O2Sensor	GasMixer	Alarm
Breathing Circuit O2 Sensor Fault	REQ_BCM_05			AlarmManager
Breathing Circuit O2 Sensor Fault	REQ_BCM_06			
Inspiratory Pressure Sensor Fault	REQ_BCM_11	PressureSensor		
Expiratory Limb CO2 sensor fault	REQ_BCM_02	CO2Sensor	PumpController	Alarm
Expiratory Limb CO2 sensor fault	REQ_BCM_07			
Expiratory Limb CO2 sensor fault	REQ_VD_06			
O2 Supply Fault	REQ_VD_03	GasValve	GasMixer	AlarmManager
O2 Supply Fault	REQ_VD_04			Alarm
O2 Supply Fault	REQ_VD_08			
O2 Supply Fault	REQ_VD_06			
O2 Supply Fault	REQ_BCM_01			

Table 5: Hazard Analysis Part 3 - Fault - Model Element Relations

Summary

This paper has introduced an approach for using the UML to aid the requirements analysis, safety analysis, and design of safety critical systems. 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.

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