Rational® software

Analyze system safety using UML within the IBM Rational Rhapsody environment.

Bruce Powel Douglass, Ph.D., chief evangelist, Rational software, IBM Software Group

Contents

Abstract

The Unified Modeling Language (UML) is a visual language for specifying, constructing and documenting the artifacts of systems. It has been successfully used in many realtime and embedded domains, including aerospace, military and medical marketplaces. Many of the systems within these marketplaces are used within safety-critical contexts. Until now, safety professionals have relied on disparate tools and environments to capture requirements, create designs and analyze system safety. However, UML is an extremely powerful, extensible language that can help safety professionals within a variety of marketplaces. IBM has therefore created a UML profile that enables you to capture requirements, create designs and analyze system safety all within the same IBM Rational® Rhapsody® tool environment.

This paper will discuss the use of the fault tree analysis (FTA) approach to safety analysis and the 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 relationships between the safety analysis, requirements and design model elements.

Highlights

Safety, the most basic term when discussing safety-critical systems, is defined as freedom from accidents or losses.

Classified according to severity, hazards are defined as system states that, when combined with environmental conditions, inevitably lead to accidents.

What is safety?

The paucity of material on safety-critical systems has led to a widespread misunderstanding of the various terms used to discuss safety. The most basic term is *safety*. Safety is defined as freedom from accidents or losses. An accident is an event in which an undesirable consequence occurs, such as death, injury, equipment damage or financial loss. A safety-critical system is therefore a system containing electronic, mechanical and software aspects and 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 that might benefit from the techniques and approaches common in safety analysis. Therefore, for the purposes of this paper, a safety-critical system is defined as any system in which the cost of use of a system because of an accident is potentially high.

A *hazard* is a system state that, when combined with other environmental conditions, inevitably leads to an accident.[1] Hazards are normally classified according 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 hazard 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 U.S. Food and Drug Administration (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 eight categories, along with required safety measures for each category, while the more recent IEC 61508[3] identifies four safety integrity levels (SILs): catastrophic, critical, marginal and negligible, although the text notes that the severity of system-presented hazards is actually a continuum.

Highlights

The risk of a hazard is the product of the probability of the occurrence of the hazard and its severity.

In system design, it's important to identify hazards and put safety measures in place to reduce the risk.

A failure is an event that occurs when components no longer function properly, while errors are design or implementation defects.

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

$$\text{risk}_{\text{hazard}} = \text{probability}_{\text{hazard}} \text{ x } \text{severity}_{\text{hazard}}$$

Being shocked by your car battery is a relatively high risk, 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 the 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, leading to a failed state. A *soft failure* is a temporary failure that can be corrected or that can 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*.

Highlights

Many systems have a fail-safe state, a condition that is known to be always safe.

Faults can be tolerated for a period of time — known as the fault tolerance time — before an accident occurs.

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

Faults can 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 milliseconds 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 milliseconds 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 handling of the fault.

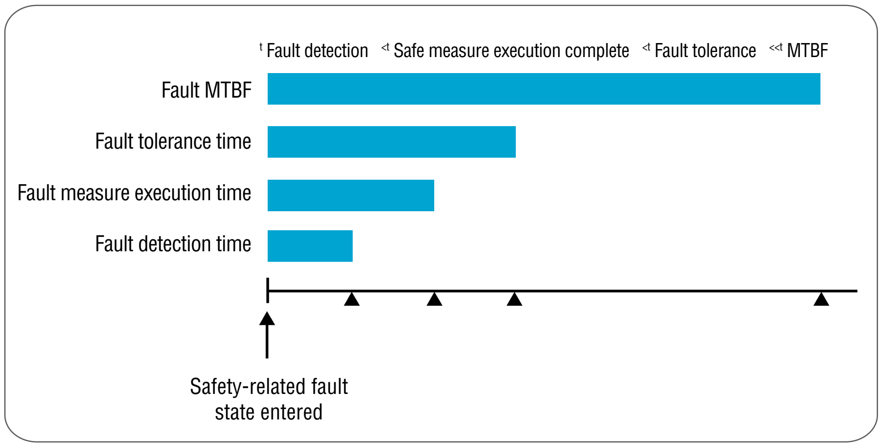


Figure 1: Fault timeline

Highlights

Fault detection time describes the amount of time needed to complete a periodic or continuous background test, which also includes the amount of time needed for the device to operate normally during the test.

These timeframes have ramifications on the kinds of safety detection and correction measures that need to be applied. If the detection is going to be handled with periodic or continuous background testing, then the time to complete the test (including the time to perform the normal device operation during the test) is called the *fault detection time.* In many systems, there simply isn't enough processor bandwidth to complete the test in software—in addition to normal system execution—to detect the faults in a 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 a short fault tolerance time, the detection of a safety-relevant fault will not necessarily occur within the fault tolerance time. A possible solution is to add mirrored memory with built-in parity checking, eliminating the need for a periodic RAM test.

A reliable system or component has a high probability of meeting its functional and quality of service requirements, while a safe system is one that does not lead to accidents.

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 requirements—for example, timeliness—within a specified timeframe. While this sounds similar to our previous definition of safety, the two concepts are importantly different. A safe system is one that 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, an old station wagon that refuses to turn on at all is very safe even though it is unreliable.

Highlights

When a system enters its fail-safe state, its safety improves but its reliability decreases.

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 if 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 to do is shut the system down, leaving the laser de-energized—or put into 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 lack a fail-safe state. For such systems, it's possible to increase both reliability and safety by adding redundant delivery channels.

Many systems don't have a fail-safe state. If you're flying at 600 knots at 35,000 feet, it is not safe to shut off the jet engine if you suspect it has a fault. Similarly, in a drive-by-wire car, the last thing you want to see is an "Abort, Retry, Ignore" message on the dashboard when you're driving down the freeway at 85 miles per hour. In such systems, increasing reliability, such as by adding redundant delivery channels, also improves the system safety.

Highlights

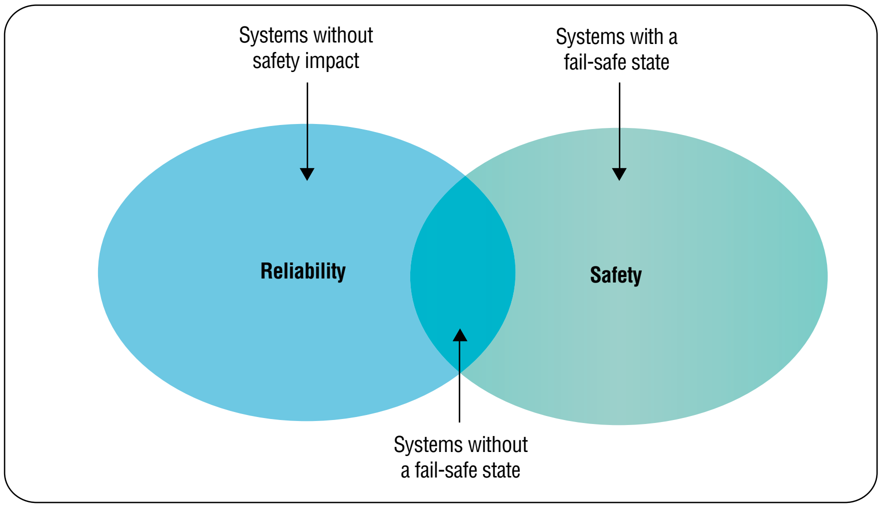

Systems without safety impact
Systems with a fail-safe state
Reliability
Safety
Systems without a fail-safe state

Figure 2: Safety versus reliability

Types of safety measures

There are several ways to handle faults:

There are several ways to handle faults, including obviation, education and alarming.

- ***Obviation.*** *This approach entails preventing the fault by anticipating it and making it difficult for it to occur. For example, using 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 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.*** *This approach announces the hazard to the user when it appears so that the user can take appropriate action. It requires a fault tolerance time that can take into account the reaction time of monitoring personnel. For example, an electrocardiogram (ECG) monitor notifies an attending physician of an asystole or "flatline" condition so that he or she can take corrective action.*

Highlights

Other ways to address faults include interlocks, transitioning to a fail-safe state, use of additional safety equipment and labelling.

- ***Interlocks.*** *The hazard can be removed by using secondary devices or logic to intercede when a hazard presents itself. For example, a medical treatment laser system could automatically disconnect power to the laser when its cover is off.*
- ***Transition 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 that 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 additional safety equipment.*** *For example, the use of a drill press may require a light curtain to ensure that the user doesn't place his or her limbs in harm's way.*
- ***Restrict access.*** *Passwords can prevent users from inadvertently invoking "service mode," in which safety checks are turned off.*
- ***Labels.*** *Hazards can be addressed by labeling; for example, "High voltage—do not touch."*

Circumstances dictate which approach to handling faults is most appropriate.

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

Highlights

UML is a modeling language that can help in the development of safety-critical systems.

By exposing the design of the systems in class diagrams, UML can provide design clarity.

One of the fundamental building blocks in UML is a class, which contains features such as data and services.

How the UML can 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, using diagrams that depict the interaction of elements of interest. UML can aid the development of safety-critical systems in a number of ways, by:

- *Providing design clarity.*
- *Modeling low-level redundancy.*
- *Creating safety-relevant views of the requirements and design.*
- *Aiding in safety analysis.*

Providing design clarity

First, UML can provide design clarity by exposing the design of the system in class diagrams, known as internal block diagrams in Systems Modeling Language (SysML), a profile or specialized version of UML used in system engineering. UML can also specify 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.

Modeling low-level redundancy

One of the fundamental building blocks of a UML model is the notion of a class or block in SysML. It contains features such as data (attributes), services (operations), logic (state machines), algorithms (activity diagrams), quality of service 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 cyclic redundancy codes (CRCs) on the class attributes, data replication, and precondition and postcondition checking to ensure that safety-relevant faults are identified and handled appropriately.

Highlights

UML enables you to construct diagrams that focus on narrow aspects of the system design so you can create safety-relevant views and ensure that all views are consistent.

Creating safety-relevant views of the requirements and design

One of the biggest 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 can ensure that all the views are consistent. The IBM Rational Harmony™ for Embedded RealTime Development process[4,5], a software development process founded on UML, identifies five key views of architecture, including the safety and reliability view. 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 how to perform safety analysis of the designs.

It's possible to create a safety-critical profile, a specialized version of the UML to meet a specialized need. In this way, you can capture fault metadata for analysis.

Aiding in safety 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—a specialized version of the UML that's consistent with the underlying UML semantics to meet a specialized need—that permits the capturing of fault metadata for analysis. The advantage of this is that the requirements, design model and safety analysis are colocated and interconnected. This interconnection allows developers to reliably navigate between these three kinds of views with ease.

Highlights

The Rational Harmony for Embedded RealTime Development process includes a best practice workflow called the "eight steps to safety," which can be added on top of a general development process.

The safety analysis results in a hazard analysis document that describes information about the faults as well as the necessary fault handling response time.

Safety analysis with fault tree analysis

The Rational Harmony for Embedded RealTime Development process includes a best practice workflow called "eight steps to safety."[6] 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 Rational Harmony for Embedded RealTime Development process.[7,8] The basic practice steps are simple to understand and relatively straightforward to implement.

Here are the Rational Harmony for Embedded RealTime Development 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. *Ensure the safety process.*
8. *Test, test, test.*

The safety analysis in 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 to ensure that the system, as specified, is safe. The safety analysis results in a *hazard analysis* document that 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.

Highlights

Safety design specifies 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 Rational Harmony for Embedded RealTime Development process recommends this be done with the application of FTA to link the safety measures with the faults to ensure fault coverage.

Safety testing involves testing the primary functionality and quality of service testing used for nonsafety-critical systems as well as seeding the system with faults.

Safety testing is then performed to ensure that the safety requirements are met. This typically involves testing the primary functionality and quality of service testing used for nonsafety-critical systems as well as 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.

FTA, a common and useful analytic technique applied to safety-critical systems, enables you to logically analyze conditions leading up to hazards for cause-effect relations.

As mentioned above, FTA is a common and useful analytic technique applied to safety-critical systems. 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 the preconditions of hazardous conditions and how they combine with faults to result in hazards. When these relations are identified, you can add safety measures whose faults must be ANDed with the original fault to lead to the hazardous condition. In other words, 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*, which means that the primary and safety-measure faults are independent. 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.

Highlights

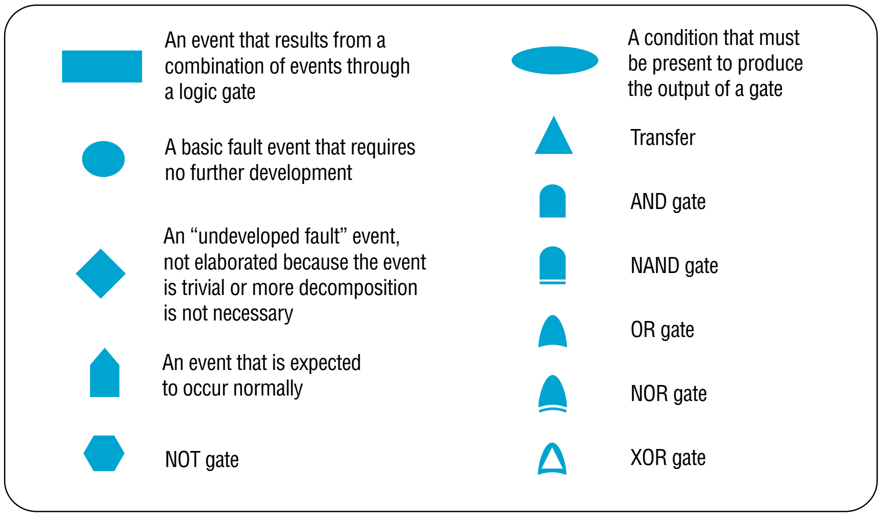

An event that results from a combination of events through a logic gate
A basic fault event that requires no further development
An "undeveloped fault" event, not elaborated because the event is trivial or more decomposition is not necessary
An event that is expected to occur normally
NOT gate
A condition that must be present to produce the output of a gate
Transfer
AND gate
NAND gate
OR gate
NOR gate
XOR gate

Figure 3: FTA symbols

The Monitor-Actuator design pattern is a generalized solution to a commonly occurring design problem — achieving safety against two different hazardous conditions.

Consider the patient ventilator in figure 4. This system uses the Monitor-Actuator design pattern, a generalized solution to a commonly occurring design problem, to achieve safety against two different hazardous conditions: hypoventilation and overpressure. This pattern creates two channels or sets of sequential processing elements: the actuation channel delivers the therapy, and the monitoring channel checks on how well the therapy is delivered.

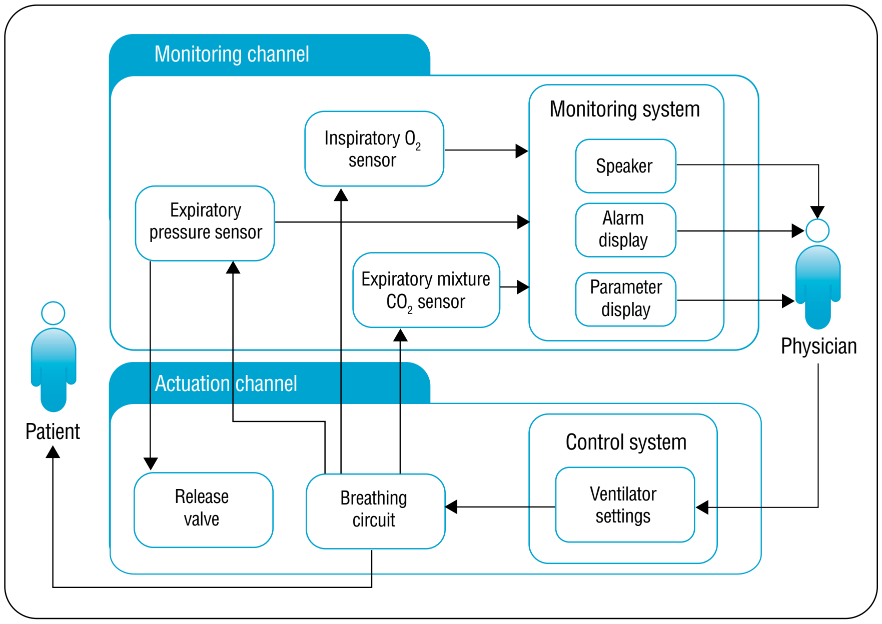


Figure 4: Patient ventilator simplified model

Highlights

Architectural redundancy adds a number of sensors to detect fault conditions. A failure in all of them or a failure of the alarm system is required for the hazard to be realized.

When a system has a short fault tolerance time, alarming is an inadequate safety measure.

Figure 5 shows an FTA for these two hazards. In an unprotected system, a fault occurring in the breathing circuit, gas supply or ventilator can lead to hypoventilation. Note that intubating the esophagus rather than the trachea can also lead to hypoventilation. The 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 occurrence of 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.

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

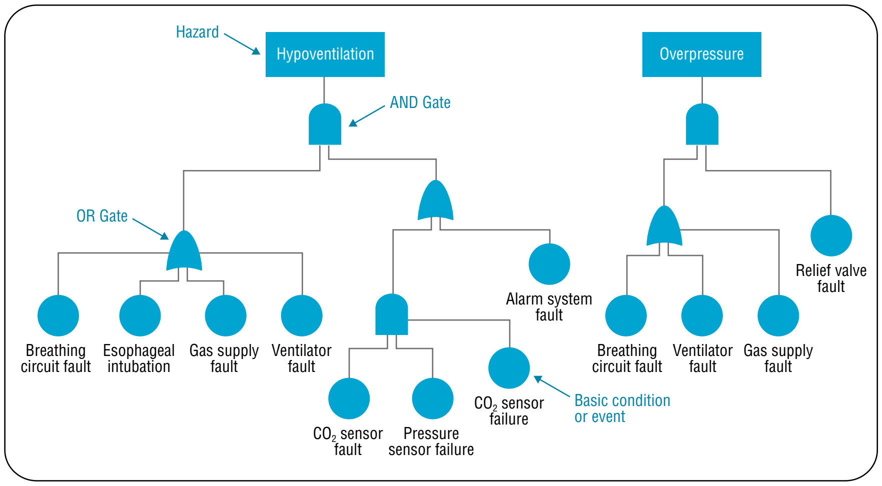


Figure 5: Patient ventilator, simplified FTA

Highlights

A profile is a set of lightweight extensions to the UML that creates a specialized version of UML for a specific purpose.

You can use Rational Rhapsody software to create a safety analysis profile that adds new stereotyped elements to create FTA diagrams.

Before creating a profile, you need to characterize the concepts within the profile and how the concepts relate to one another, usually by creating a metamodel.

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 the U.S. Department of Defense Architecture Framework (DoDAF), U.K. Ministry of Defence Architecture Framework MODAF (also known as the Unified Profile for DoDAF/MODAF [UPDM]) and Operational Node Connectivity Description (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, this paper will show how to use the Rational 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, we 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 one another. 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 (page 17) shows the metamodel for the safety analysis profile (the metasubtypes of the logical operator are detailed in Figure 7 [page 17]). The attributes of the metaclasses will end up as tags on our defined stereotypes. The colored boxes are the relations between the core metaclasses.

Highlights

A metamodel is a model of the fundamental concepts and their relations for a domain or subject matter.

The metasubtypes of a logical operator can be shown using a diagram like figure 7.

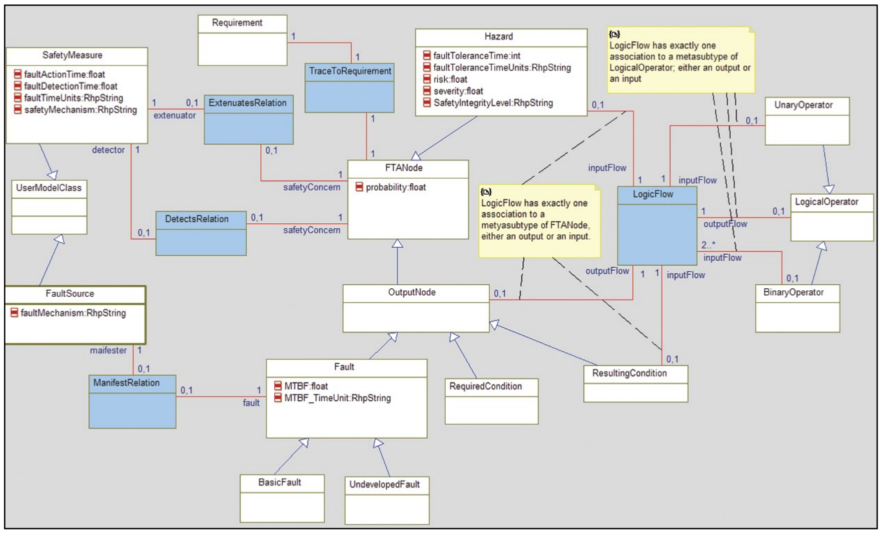


Figure 6: Safety profile metamodel

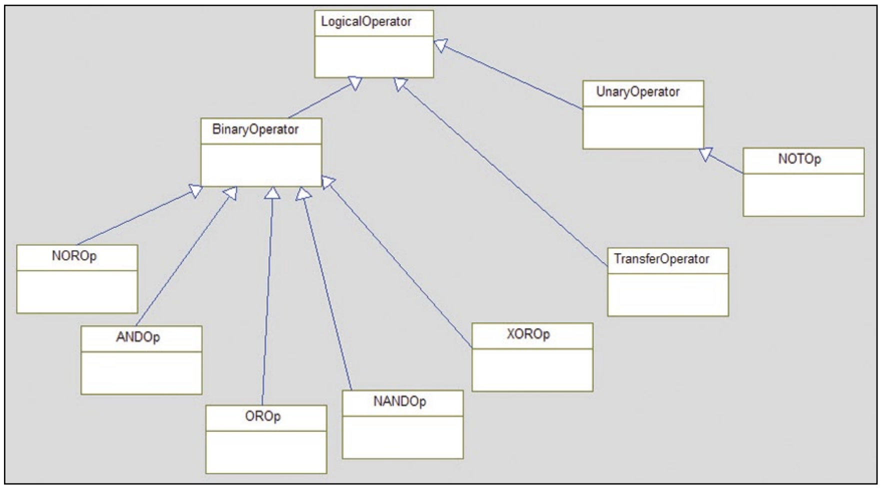


Figure 7: Metasubtypes of logical operators

Highlights

Key elements for a metamodel include hazard, fault, resulting condition, required condition and logical operator.

Additional elements for the metamodel include logic flow, fault source, safety measure, manifest relation, detect relation, extenuates relation and trace to requirement.

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

- ***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 nonconformance 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 OR, NOT or AND. Note that the 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****—a normal UML element that could manifest a fault, for example, or that could be the source of a fault. (Stereotype of class)*
- ***Safety measure****—a normal UML element that could detect or extenuate, or mitigate, a fault. (Stereotype of class)*
- ***Manifest relation****—a relationship from a fault to a fault source that causes the fault. (Stereotype of dependency)*
- ***Detect relation****—a relation from a fault or hazard to a safety measure that can detect when the fault has occurred. (Stereotype of dependency)*
- ***Extenuates relation****—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****—a relation from a fault or hazard to a requirement. (Stereotype of dependency)*

Highlights

Fault and hazard elements have important metadata characterizing them.

Fault and hazard elements have important metadata characterizing them. The important metadata is summarized in table 1.

Table 1: Safety metadata

Metaclass	Metadata	Description
Hazard	Fault tolerance time	The length of time the fault can be tolerated before it leads to an accident.
	Fault tolerance time units	The units of time, such as milliseconds, seconds, hours or days.
	Risk	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, 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 failures for the element.
	MTBF 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.

Highlights

Table 2 shows the tables and matrices added in the profile.

The hazard analysis output is divided into three sections. The first lists hazards and their metadata. The second lists relations between the faults and hazards. And the third lists the relations between the faults and the normal UML model elements.

Tables, matrices and hazard analyses

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

Table 2: Tables and matrix summary views

Table or matrix	Format	Description
Fault table	Rational Rhapsody table view	A list of the faults and all their metadata.
Hazard table	Rational Rhapsody table view	A list of the hazards and all their metadata.
Fault source matrix	Rational Rhapsody matrix view	A fault × fault source matrix, as defined by the *manifests* relations.
Fault detection matrix	Rational Rhapsody matrix view	A fault × safety measure matrix, as defined by the *detects* relations.
Fault extenuation matrix	Rational Rhapsody matrix view	A fault × safety measure matrix, as defined by the *extenuates* relations.
Hazard analysis	Tab-separated value text file (.tsv) intended to load into a commercial spread-sheet program	An external file generated by the profile helper macros summarizing the hazard and fault information.

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 that can be loaded into most spreadsheet programs. The macro generates the name from the current date and time so you can retain multiple versions of the hazard analysis. The output is divided into three sections:

1. *Lists the hazards and their metadata, including the description, fault tolerance time, fault tolerance time units, probability, severity, risk and safety integrity level.*
2. *Lists the relations between the faults and the hazards as defined by multiple intervening logical operators and logic flows. Each fault is identified with its name, description and other metadata.*
3. *Lists the relations between the faults and the normal UML model elements, such as requirements and classes related with the* manifests, detects, extenuates *and* traceToReqs *relations.*

Highlights

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 Rational Rhapsody, you can create a new safety analysis model or add the profile after the model is created.

To use the profile in Rational Rhapsody, you can create a new safety analysis model 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 then change the type of the model to Safety Analysis Profile.

Once the model is created, a new diagram type—the FTA diagram—is available on the diagram toolbar. All of the UML and Rational Rhapsody features remain available to you. It is recommended to put the safety analysis in a separate package in your model to separate it from your requirements and design elements.

A medical example: the anesthesia machine patient ventilator

A model for a surgical anesthesia machine shows how to use the profile.

To illustrate the use of the profile, we'll use a model for a surgical anesthesia machine. This system delivers inhalant anesthetic drugs, mixes gases, delivers a gas/drug mixture to the patient via ventilation, collects 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.

Highlights

The primary elements of the surgical anesthesia model include gas supplies, a gas mixer, a ventilator and a vaporizer.

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

- *Gas supplies, which may be wall supplies or tanks that supply medically certified gases (O_2, N_2, N_2O, He or air).*
- *A gas mixer, which is formally a part of the ventilator and mixes the input gases from the gas supplies as directed and outputs a mixed gas to the breathing circuit.*
- *A ventilator, which shapes the breath delivered to the patient. Its primary parameters include the respiration rate of breaths/minute, tidal volume of volume per breath, the ratio of inspiration time to expiration time (I:E ratio), inspiratory time in seconds and an optional inspiratory pause that measures the delay between breaths.*
- *A vaporizer, which vaporizes an anesthetic drug, such as Suprane, Halothane or Enflourane, and delivers it to the breathing circuit in the concentration specified.*

The schematic for the anesthesia machine shows all of its primary elements.

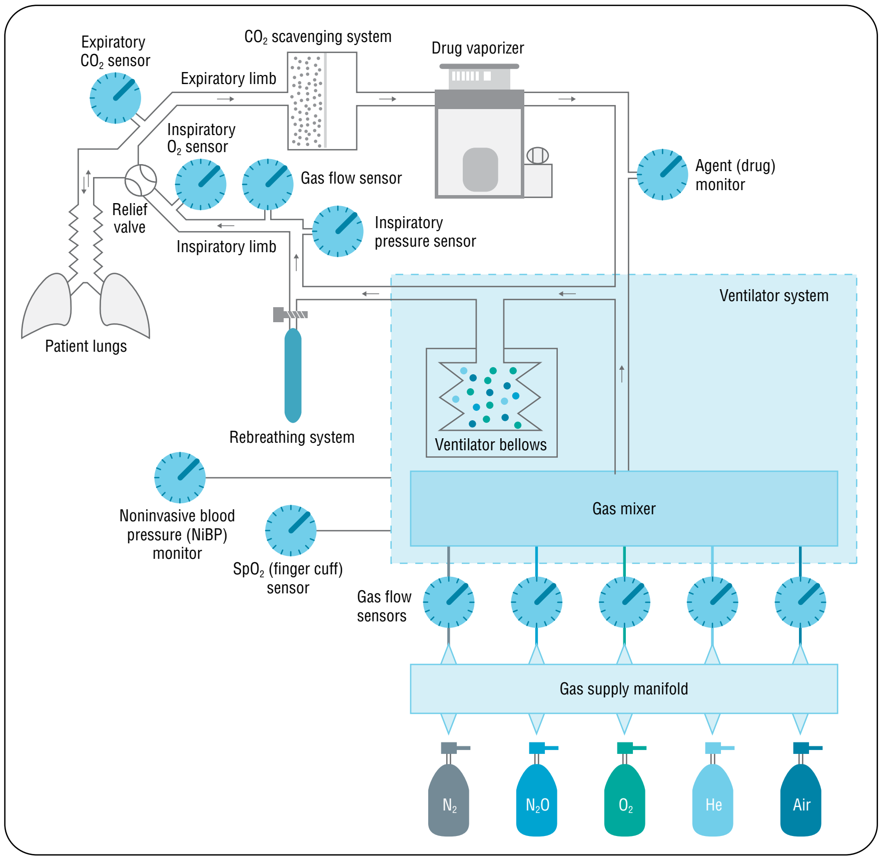


Figure 8: Anesthesia machine schematic

Highlights

The overall use case model for the SleepyTime anesthesia system 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. If this were a system engineering model, then gas scavenging would be included.

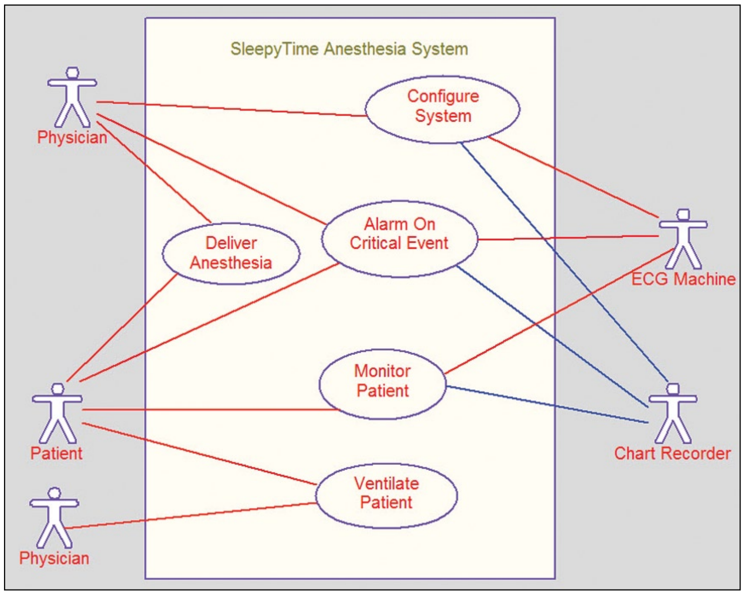


Figure 9: SleepyTime anesthesia system use cases

In this example, the focus is mainly on the patient and how ventilation affects him or her.

Most of our discussion will focus on 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 I:E ratio or, alternatively, a specified time for inspiration (inspiration time). Additionally, an inspiratory pause may be specified as well as a 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, or another subsystem.

Highlights

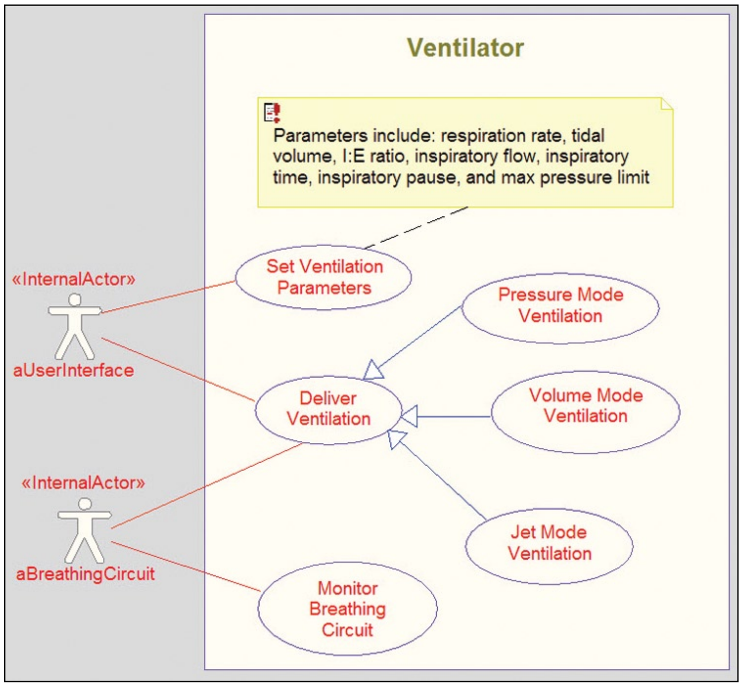

Ventilator
Parameters include: respiration rate, tidal volume, I:E ratio, inspiratory flow, inspiratory time, inspiratory pause, and max pressure limit
«InternalActor»
aUserInterface
«InternalActor»
aBreathingCircuit
Set Ventilation Parameters
Pressure Mode Ventilation
Deliver Ventilation
Volume Mode Ventilation
Jet Mode Ventilation
Monitor Breathing Circuit

Figure 10: Ventilator use cases

Requirements bound to each of the use cases can be managed with a requirements traceability tool, such as Rational DOORS.

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 IBM Rational DOORS® software, and can then be imported to the model and attached to the use cases and design elements for traceability. For example, figure 11 (page 25) shows the requirements for setting the different ventilator parameters. Figure 12 (page 25) and figure 13 (page 26) show similar requirements for two other use cases.

Highlights

Use case requirements can be shown using diagrams like figures 11 and 12.

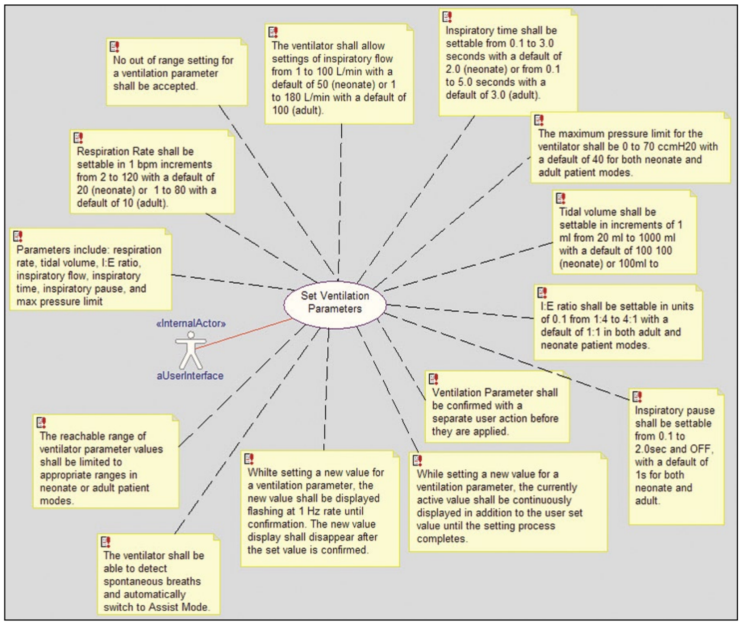

No out of range setting for a ventilation parameter shall be accepted.
The ventilator shall allow settings of inspiratory flow from 1 to 100 L/min with a default of 50 (neonate) or 1 to 180 L/min with a default of 100 (adult).
Inspiratory time shall be settable from 0.1 to 3.0 seconds with a default of 2.0 (neonate) or from 0.1 to 5.0 seconds with a default of 3.0 (adult).
The maximum pressure limit for the ventilator shall be 0 to 70 ccmH20 with a default of 40 for both neonate and adult patient modes.
Respiration Rate shall be settable in 1 bpm increments from 2 to 120 with a default of 20 (neonate) or 1 to 80 with a default of 10 (adult).
Tidal volume shall be settable in increments of 1 ml from 20 ml to 1000 ml with a default of 100 100 (neonate) or 100ml to
Parameters include: respiration rate, tidal volume, I:E ratio, inspiratory flow, inspiratory time, inspiratory pause, and max pressure limit
Set Ventilation Parameters
«InternalActor»
aUserInterface
I:E ratio shall be settable in units of 0.1 from 1:4 to 4:1 with a default of 1:1 in both adult and neonate patient modes.
Ventilation Parameter shall be confirmed with a separate user action before they are applied.
Inspiratory pause shall be settable from 0.1 to 2.0sec and OFF, with a default of 1s for both neonate and adult.
The reachable range of ventilator parameter values shall be limited to appropriate ranges in neonate or adult patient modes.
Whilte setting a new value for a ventilation parameter, the new value shall be displayed flashing at 1 Hz rate until confirmation. The new value display shall disappear after the set value is confirmed.
While setting a new value for a ventilation parameter, the currently active value shall be continuously displayed in addition to the user set value until the setting process completes.
The ventilator shall be able to detect spontaneous breaths and automatically switch to Assist Mode.

Figure 11: Requirements for use case set ventilation parameters

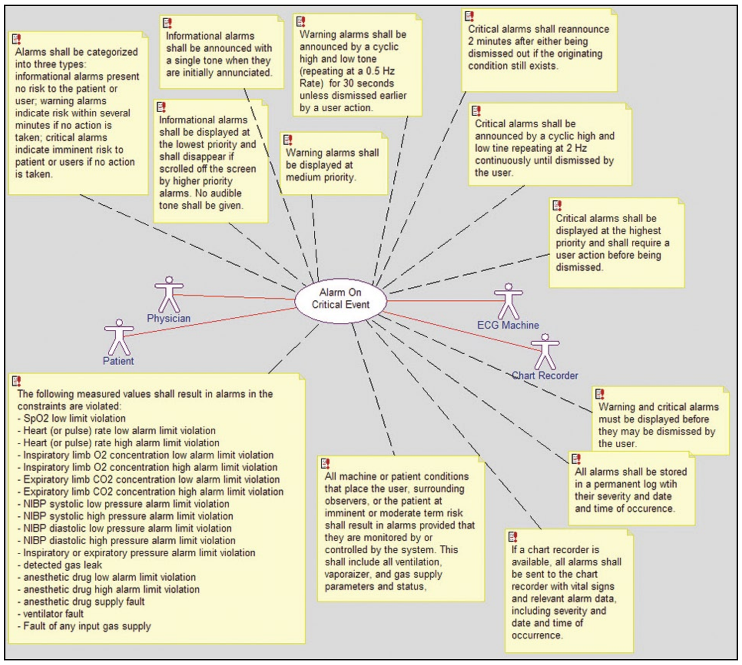
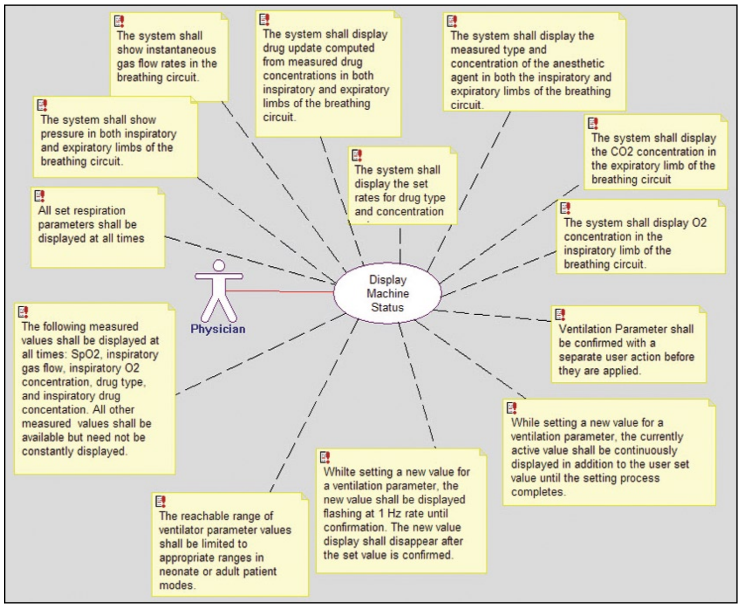

Alarms shall be categorized into three types: informational alarms present no risk to the patient or user; warning alarms indicate risk within several minutes if no action is taken; critical alarms indicate imminent risk to patient or users if no action is taken.
Informational alarms shall be announced with a single tone when they are initially annunciated.
Warning alarms shall be announced by a cyclic high and low tone (repeating at a 0.5 Hz Rate) for 30 seconds unless dismissed earlier by a user action.
Critical alarms shall reannounce 2 minutes after either being dismissed out if the originating condition still exists.
Informational alarms shall be displayed at the lowest priority and shall disappear if scrolled off the screen by higher priority alarms. No audible tone shall be given.
Warning alarms shall be displayed at medium priority.
Critical alarms shall be announced by a cyclic high and low tine repeating at 2 Hz continuously until dismissed by the user.
Critical alarms shall be displayed at the highest priority and shall require a user action before being dismissed.
Physician
Patient
Alarm On Critical Event
ECG Machine
Chart Recorder
The following measured values shall result in alarms in the constraints are violated:
- SpO2 low limit violation
- Heart (or pulse) rate low alarm limit violation
- Heart (or pulse) rate high alarm limit violation
- Inspiratory limb O2 concentration low alarm limit violation
- Inspiratory limb O2 concentration high alarm limit violation
- Expiratory limb CO2 concentration low alarm limit violation
- Expiratory limb CO2 concentration high alarm limit violation
- NIBP systolic low pressure alarm limit violation
- NIBP systolic high pressure alarm limit violation
- NIBP diastolic low pressure alarm limit violation
- NIBP diastolic high pressure alarm limit violation
- Inspiratory or expiratory pressure alarm limit violation
- detected gas leak
- anesthetic drug low alarm limit violation
- anesthetic drug high alarm limit violation
- anesthetic drug supply fault
- ventilator fault
- Fault of any input gas supply
All machine or patient conditions that place the user, surrounding observers, or the patient at imminent or moderate term risk shall result in alarms provided that they are monitored by or controlled by the system. This shall include all ventilation, vaporaizer, and gas supply parameters and status,
Warning and critical alarms must be displayed before they may be dismissed by the user.
All alarms shall be stored in a permanent log wtih their severity and date and time of occurence.
If a chart recorder is available, all alarms shall be sent to the chart recorder with vital signs and relevant alarm data, including severity and date and time of occurrence.

Figure 12: Requirements for use case alarm on critical event

Highlights



The system shall show instantaneous gas flow rates in the breathing circuit.

The system shall display drug update computed from measured drug concentrations in both inspiratory and expiratory limbs of the breathing circuit.

The system shall display the measured type and concentration of the anesthetic agent in both the inspiratory and expiratory limbs of the breathing circuit.

The system shall show pressure in both inspiratory and expiratory limbs of the breathing circuit.

The system shall display the set rates for drug type and concentration

The system shall display the CO2 concentration in the expiratory limb of the breathing circuit

All set respiration parameters shall be displayed at all times

The system shall display O2 concentration in the inspiratory limb of the breathing circuit.

The following measured values shall be displayed at all times: SpO2, inspiratory gas flow, inspiratory O2 concentration, drug type, and inspiratory drug concentation. All other measured values shall be available but need not be constantly displayed.

Ventilation Parameter shall be confirmed with a separate user action before they are applied.

While setting a new value for a ventilation parameter, the currently active value shall be continuously displayed in addition to the user set value until the setting process completes.

Whilte setting a new value for a ventilation parameter, the new value shall be displayed flashing at 1 Hz rate until confirmation. The new value display shall disappear after the set value is confirmed.

The reachable range of ventilator parameter values shall be limited to appropriate ranges in neonate or adult patient modes.

Figure 13: Requirements for use case display machine status

A safety analysis identifies additional requirements needed to ensure the safe operation of the system.

After we have an understanding of the requirements, we can begin to analyze 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.

Consider a single hazard, hypoxia. This is a fundamental hazard for a ventilator, but just one of many. Other hazards for a ventilator include hyperoxia, overpressure, inadequate anesthesia, overanesthesia and leaking drugs into the operating room environment. We would now create a new FTA diagram and draw something like figure 14 (page 27).

Highlights

Safety measures are elements and behaviors added to the system to address safety concerns.

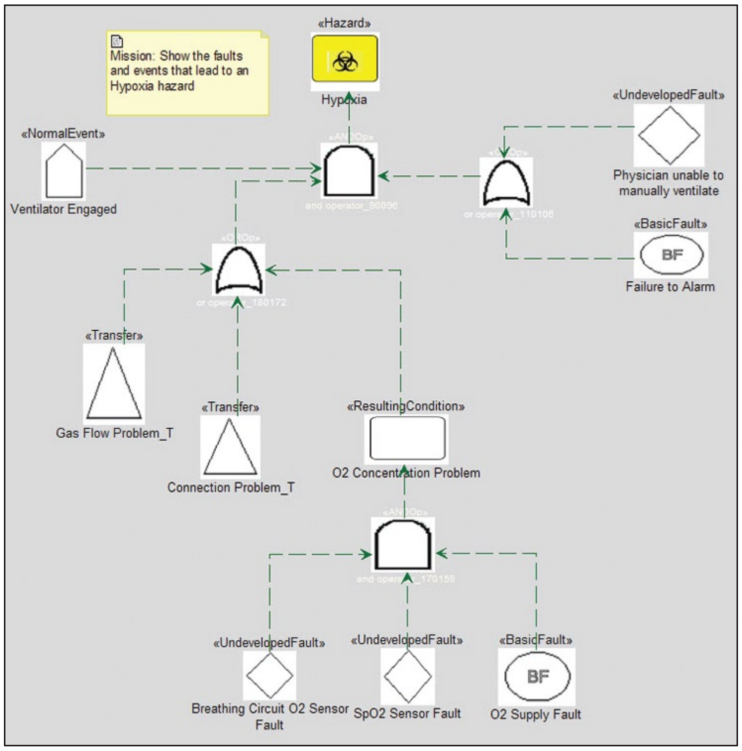


Figure 14: FTA for the hazard hypoxia

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.

Highlights

If both the original fault and the fault in the safety measure must occur for the hazardous condition to take place, it is known as ANDing redundancy.

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 ANDing 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 must fail to take proper action, either because he or she doesn't know that action is required (failure to alarm fault) or because for some reason he or she is unable to take action (physician unable to manually ventilate).

Transfer operators can be used to logically connect issues into the appropriate position within a diagram, even when they are drawn on a separate diagram.

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 position within this diagram even though they are drawn on a separate diagram.

Highlights

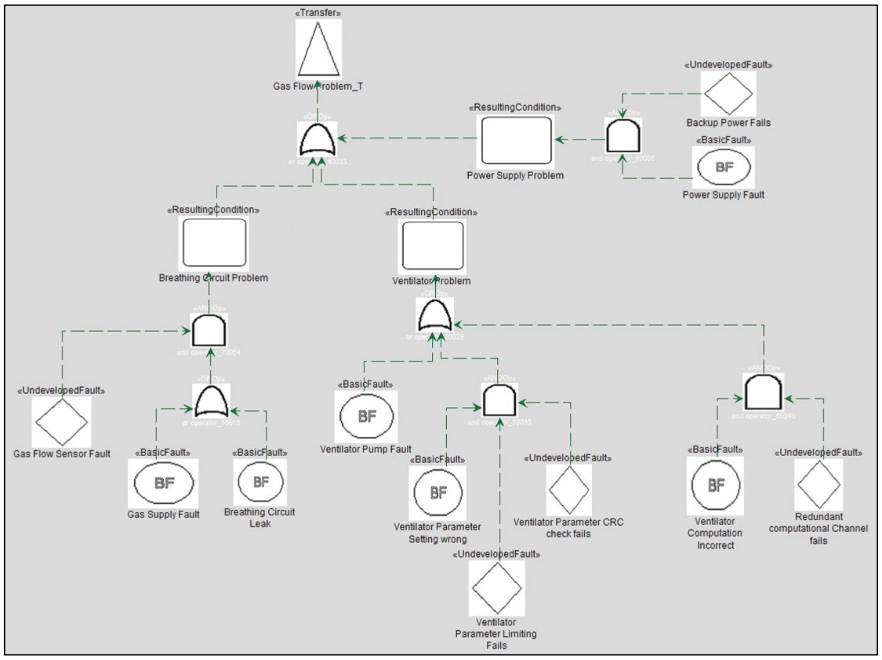


Figure 15: FTA for gas flow problem subdiagram

A more elaborate FTA can lead to the identification of new safety requirements.

Figure 15 elaborates the FTA for the gas flow problem condition. 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.

Highlights

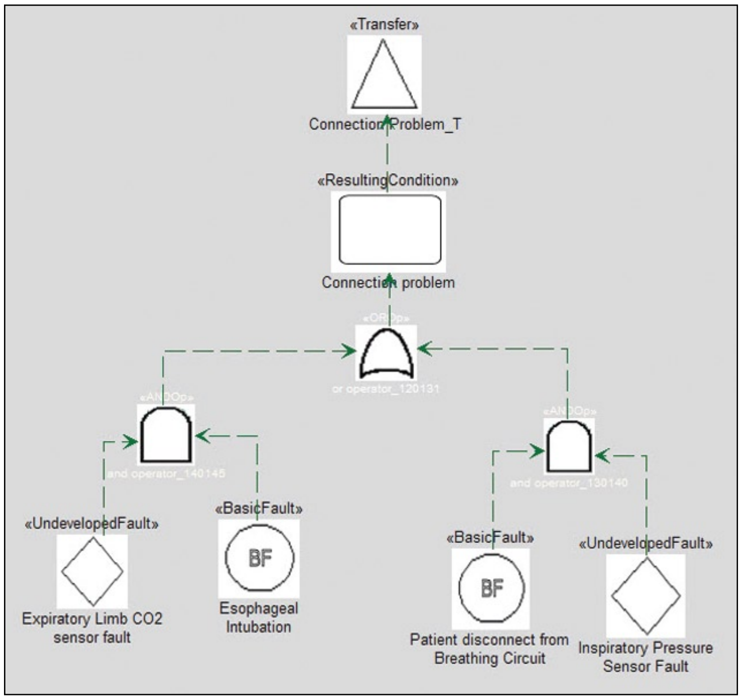


Figure 16: FTA for the connection problem subdiagram

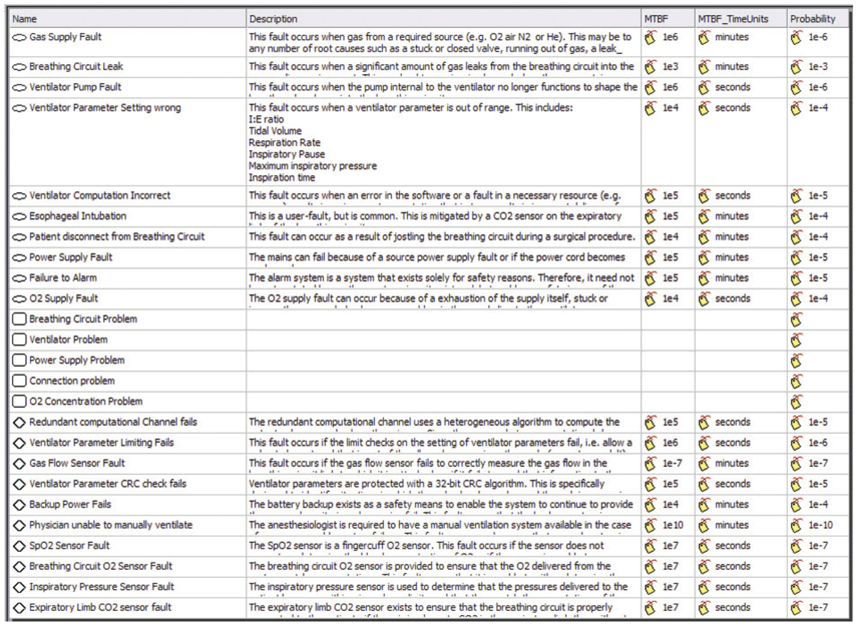
In the example, another concern that can cause the hypoxia hazard to occur is the misconnection or disconnection of the patient from the breathing circuit.

Figure 16 illustrates another concern for the hypoxia hazard, the problem of misconnection or disconnection of the patient from the breathing circuit. A very common problem is improper intubation, that is, the insertion of the breathing circuit connection into the patient's trachea. 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, if the expiratory limb doesn't see an increased CO_2 concentration, then either the patient isn't producing CO_2, which is a very bad sign, or the expiratory limb of the breathing circuit isn't connected to the patient's lungs.

Highlights

Identified faults can be characterized with the fault metadata by filling in the identified tag fields.

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.



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

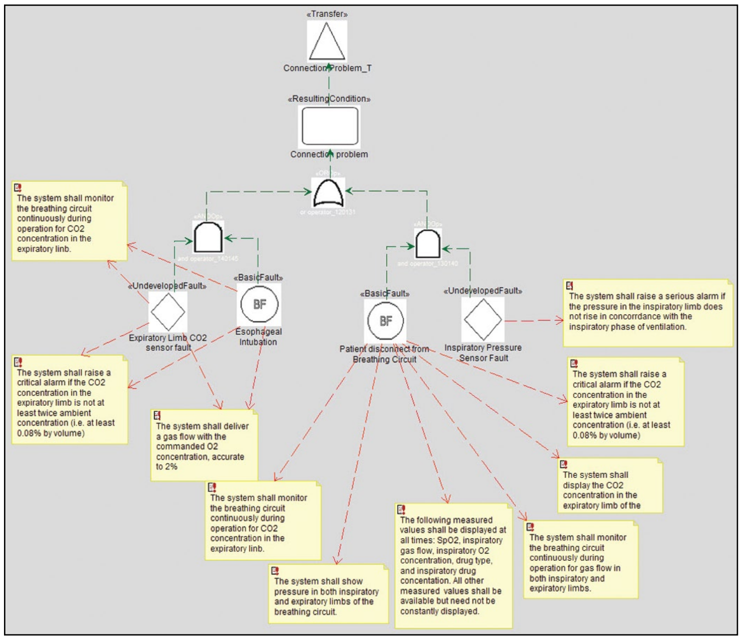
Figure 17: Fault table

Performing a safety analysis can lead to the discovery of additional requirements to enhance safety.

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 (page 32).

Highlights

Figure 18 shows the FTA for the connection problem linked to the relevant requirements with the trace to requirement relation. This binds the safety analysis directly to the requirements.



The system shall monitor the breathing circuit continuously during operation for CO2 concentration in the expiratory linb.

The system shall raise a critical alarm if the CO2 concentration in the expiratory limb is not at least twice ambient concentration (i.e. at least 0.08% by volume)

The system shall deliver a gas flow with the commanded O2 concentration, accurate to 2%

The system shall monitor the breathing circuit continuously during operation for CO2 concentration in the expiratory linb.

The system shall show pressure in both inspiratory and expiratory limbs of the breathing circuit.

The following measured values shall be displayed at all times: SpO2, inspiratory gas flow, inspiratory O2 concentration, drug type, and inspiratory drug concentation. All other measured values shall be available but need not be constantly displayed.

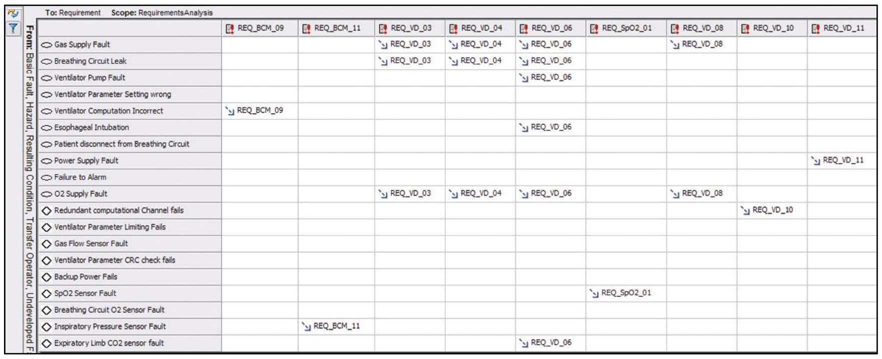
The system shall monitor the breathing circuit continuously during operation for gas flow in both inspiratory and expiratory limbs.

The system shall display the CO2 concentration in the expiratory limb of the

The system shall raise a critical alarm if the CO2 concentration in the expiratory limb is not at least twice ambient concentration (i.e. at least 0.08% by volume)

The system shall raise a serious alarm if the pressure in the inspiratory limb does not rise in concorrdance with the inspiratory phase of ventilation.

Figure 18: Faults linked to requirements



To: Requirement **Scope:** RequirementsAnalysis

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

	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

Highlights

UML models can become more elaborate during subsequent analysis and design.

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

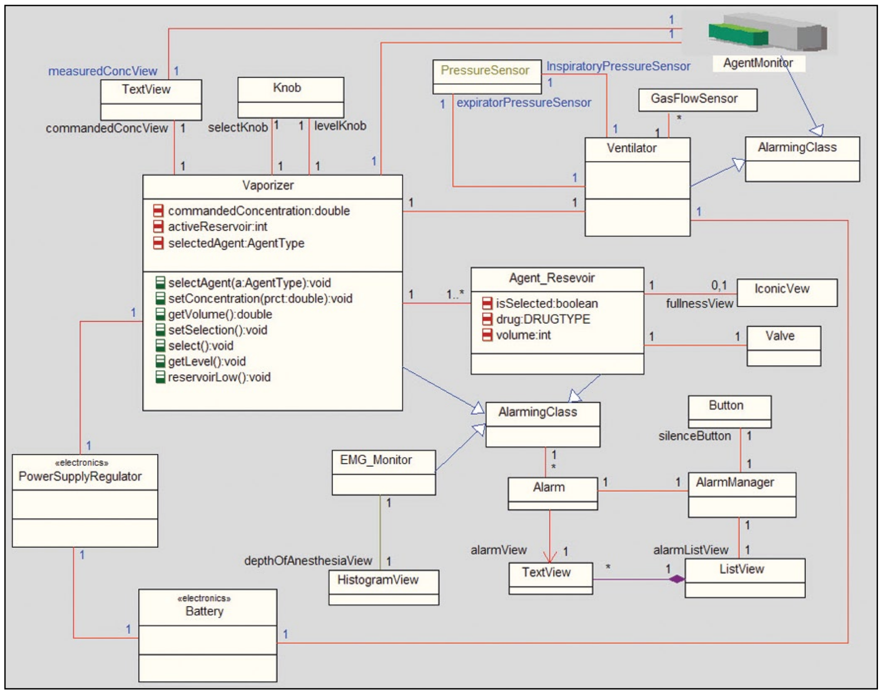


Figure 20: SleepyTime subsystems

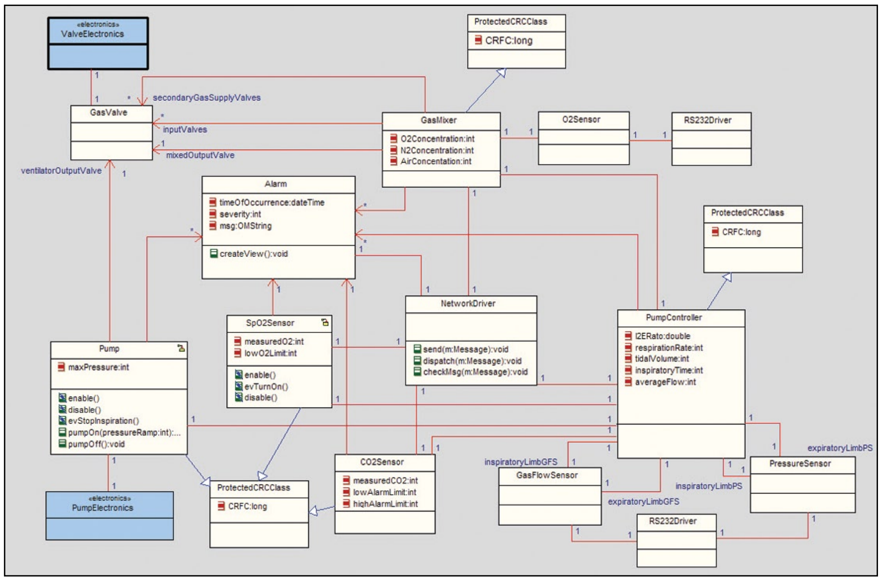


Figure 21: Ventilator design model

Highlights

As you add new hazards through design and development work, you can elaborate the FTA model with the new information.

The safety analysis profile helps enable you to link the analysis and design elements to the faults using the manifests, detects and extenuates relations.

Figure 21 shows some electronic components with the stereotype electronics colored in blue. The other elements are all software classes that collaborate 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—and questions about whether the design appropriately addresses the safety concerns may arise. At this point, you can elaborate the FTA model with that information and 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 (page 35) show examples of such elaborated FTA diagrams.

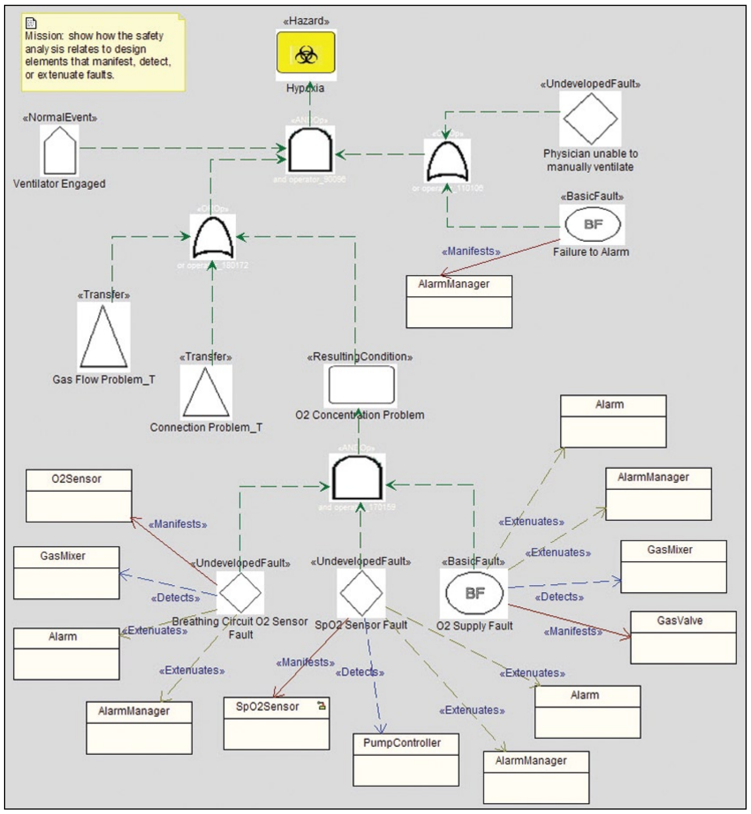


Figure 22: FTA with design element links

Highlights

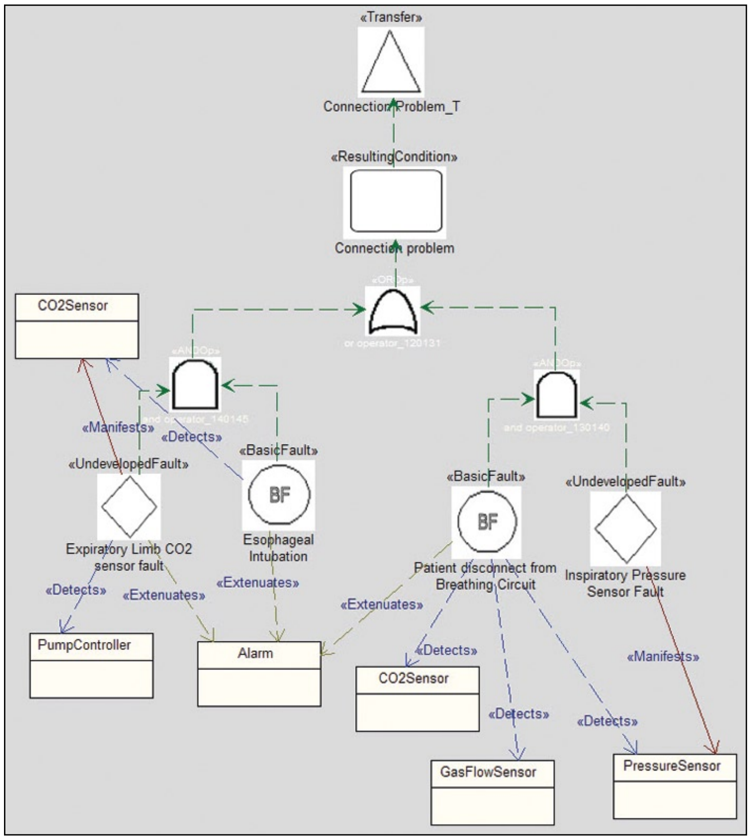
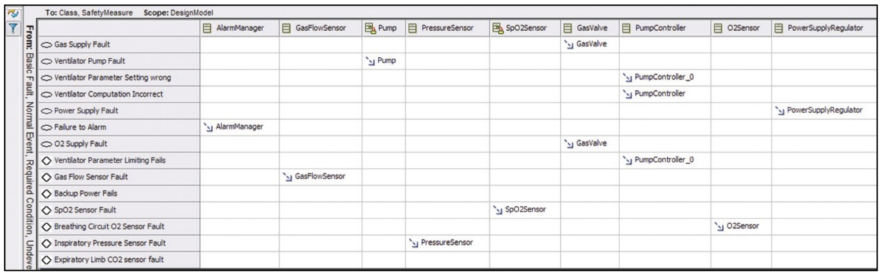


Figure 23: FTA with additional design elements

Using the FTA, information is both visually apparent and represented in the associated matrixes.

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 (page 36) shows a portion of the matrix of the faults and the design elements that can manifest them, while figure 25 (page 36) shows the same view for the faults and the design elements that detect them.

Highlights

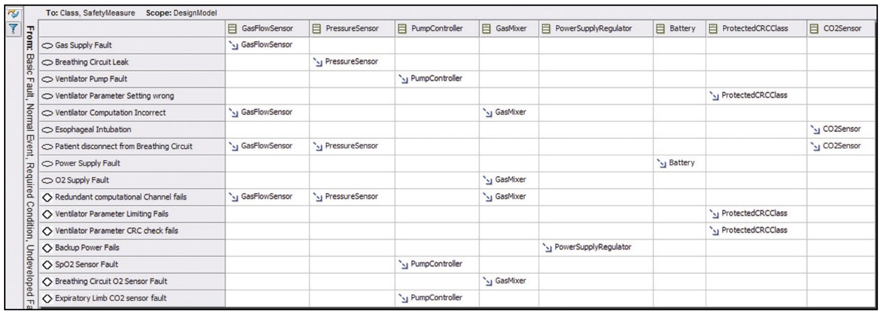


To: Class, SafetyMeasure **Scope:** DesignModel

From: Basic Fault, Normal Event, Required Condition, Undeve

	AlarmManager	GasFlowSensor	Pump	PressureSensor	SpO2Sensor	GasValve	PumpController	O2Sensor	PowerSupplyRegulator
Gas Supply Fault						GasValve			
Ventilator Pump Fault			Pump						
Ventilator Parameter Setting wrong							PumpController_0		
Ventilator Computation Incorrect							PumpController		
Power Supply Fault									PowerSupplyRegulator
Failure to Alarm	AlarmManager								
O2 Supply Fault						GasValve			
Ventilator Parameter Limiting Fails							PumpController_0		
Gas Flow Sensor Fault		GasFlowSensor							
Backup Power Fails									
SpO2 Sensor Fault					SpO2Sensor				
Breathing Circuit O2 Sensor Fault								O2Sensor	
Inspiratory Pressure Sensor Fault				PressureSensor					
Expiratory Limb CO2 sensor fault									

Figure 24: Fault source matrix



To: Class, SafetyMeasure **Scope:** DesignModel

From: Basic Fault, Normal Event, Required Condition, Undeveloped Fa

	GasFlowSensor	PressureSensor	PumpController	GasMixer	PowerSupplyRegulator	Battery	ProtectedCRCClass	CO2Sensor
Gas Supply Fault	GasFlowSensor							
Breathing Circuit Leak		PressureSensor						
Ventilator Pump Fault			PumpController					
Ventilator Parameter Setting wrong							ProtectedCRCClass	
Ventilator Computation Incorrect	GasFlowSensor			GasMixer				
Esophageal Intubation								CO2Sensor
Patient disconnect from Breathing Circuit	GasFlowSensor	PressureSensor						CO2Sensor
Power Supply Fault						Battery		
O2 Supply Fault				GasMixer				
Redundant computational Channel fails	GasFlowSensor	PressureSensor		GasMixer				
Ventilator Parameter Limiting Fails							ProtectedCRCClass	
Ventilator Parameter CRC check fails							ProtectedCRCClass	
Backup Power Fails					PowerSupplyRegulator			
SpO2 Sensor Fault			PumpController					
Breathing Circuit O2 Sensor Fault				GasMixer				
Expiratory Limb CO2 sensor fault			PumpController					

Figure 25: Fault detection matrix

A comprehensive hazard analysis can be generated as a .tsv file from the annotated fault model.

In addition, a comprehensive 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.

Table 3: Hazard analysis part 1—hazard metadata

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 percent O_2 environment, death or irreversible injury occurs after five minutes of no oxygen. If the patient is breathing 100 percent O_2 for a significant period of time, this time is about 10 minutes.	5	minutes	1.00E-02	8	8.00E-02	3
Overpressure	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	Inadequate 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
Overanesthesia	Overanesthesia can lead to death.	3	minutes	1.00E+03	4	4.00E+03	4
Anesthesia leak into ER	Anesthesia leak can lead to short-term, in smaller doses, or to long-term poisoning of medical staff.	10	minutes	1.00E+05	5	4.00E+05	5

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.

Table 4: Hazard analysis part 2—hazard fault matrix

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 is unavailable. This may be due to any number of root causes, such as a stuck or closed valve, running out of gas, or a leak.	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 the breathing circuit.	1.00E+06		1.00E-06
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 (such as memory) results in an incorrect computation that in turn results in incorrect delivery of ventilation.	1.00E+05		1.00E-05

Hazard	Fault or event	Fault type	Fault description	MTBF	MTBF time units	Probability
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. For example, allowing 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 (such as 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 CO_2 sensor on the expiratory limb of the breathing circuit.	1.00E+05		1.00E-04

Hazard	Fault or event	Fault type	Fault description	MTBF	MTBF time units	Probability
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 exists solely for safety reasons. Therefore, it need not be extenuated 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	SpO_2 sensor fault	UndevelopedFault	The SpO_2 sensor is a fingercuff O_2 sensor. This fault occurs if the sensor does not accurately determine the blood concentration of O_2 or if the sensor is unable to communicate its readings to the system.	1.00E+07		1.00E-07

Hazard	Fault or event	Fault type	Fault description	MTBF	MTBF time units	Probability
Hypoxia	Breathing circuit O_2 sensor fault	UndevelopedFault	The breathing circuit O_2 sensor is provided to ensure that the O_2 delivered from the system matches expectations. This fault means that it is unable to either determine the O_2 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 minimum and maximum 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 CO_2 sensor fault	UndevelopedFault	The expiratory limb CO_2 sensor exists to ensure that the breathing circuit is properly connected to the patient. If there is inadequate CO_2 in the expiratory limb, then either the patient isn't generating CO_2 or the expiratory limb is disconnected from the patient. This fault means that the sensor is either unable to accurately determine the CO_2 concentration or is unable to communicate those values to the system.	1.00E+07		1.00E-07
Hypoxia	O_2 supply fault	BasicFault	The O_2 supply fault can occur because of an 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

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.

Table 5: Hazard analysis part 3—fault to model element relations

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			

Fault or event	Requirements	Manifestors	Detectors	Extenuators
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		CO_2Sensor	Alarm
Esophageal intubation	REQ_BCM_07			
Esophageal intubation	REQ_VD_06			

Fault or event	Requirements	Manifestors	Detectors	Extenuators
Patient disconnect from breathing circuit	REQ_Display_Pressures		CO_2Sensor	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			

Fault or event	Requirements	Manifestors	Detectors	Extenuators
SpO_2 sensor fault	REQ_SpO2_01	SpO_2Sensor	PumpController	AlarmManager
SpO_2 sensor fault				Alarm
Breathing circuit O_2 sensor fault	REQ_BCM_01	O_2Sensor	GasMixer	Alarm
Breathing circuit O_2 sensor fault	REQ_BCM_05			AlarmManager
Breathing circuit O_2 sensor fault	REQ_BCM_06			
Inspiratory pressure sensor fault	REQ_BCM_11	PressureSensor		
Expiratory limb CO_2 sensor fault	REQ_BCM_02	CO_2Sensor	PumpController	Alarm
Expiratory limb CO_2 sensor fault	REQ_BCM_07			
Expiratory limb CO_2 sensor fault	REQ_VD_06			
O_2 supply fault	REQ_VD_03	GasValve	GasMixer	AlarmManager
O_2 supply fault	REQ_VD_04			Alarm
O_2 supply fault	REQ_VD_08			
O_2 supply fault	REQ_VD_06			
O_2 supply fault	REQ_BCM_01			

Highlights

FTA is a useful method for understanding how normal events, conditions and faults combine to create hazardous conditions.

The safety profile discussed in this paper supports the safety approach identified in the IBM Rational Harmony for Embedded RealTime Development process.

Summary

This paper has shown how to use the UML to aid in the requirements analysis, safety analysis and design of safety-critical systems. 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 in 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 IBM Rational Harmony for Embedded RealTime Development process. Using this profile, developers and safety analysts can use a common language and tool environment, improving their collaboration and quality of work.

For more information

To learn more about how you can use the UML to perform safety analysis, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/rational

Endnotes

1 Nancy Leveson, *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, 6, 7 Bruce Powel Douglass, *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*, Reading, MA: Addison-Wesley, 1999.

5, 8 Bruce Powel Douglass, *Real-Time Agility*, Reading, MA: Addison-Wesley, 2009.

RAW14124-USEN-01