JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2004**

Vol. 3, No. 3, March-April 2004

Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements

Donald Firesmith, Software Engineering Institute, U.S.A.

Abstract

As software-intensive systems become more pervasive, more and more safety-critical systems are being developed. In this column, I will use the concept of a quality model to define safety as a quality factor. Thus, safety (like security and survivability) is a kind of defensibility, which is a kind of dependability, which is a kind of quality. Next, I discuss the structure of quality requirements and show how safety requirements can be engineered based on safety's numerous quality subfactors. Then, I define and discuss safety constraints (i.e., mandated safeguards) and safety-critical requirements (i.e., functional, data, and interface requirements that can cause accidents if not implemented correctly). Finally, I pose a set of questions regarding the engineering of these three kinds of safety-related requirements for future research and experience to answer.

1 INTRODUCTION

As software-intensive systems become more pervasive, a large number of safety-critical systems are being developed and fielded. To improve productivity, performance, and cost-effectiveness, we are developing more and more safety-critical systems that are under computer control. And centralized computer control is enabling many safety-critical systems (e.g., chemical and pesticide factories) to grow in size, complexity, and potential for catastrophic failure.

We use software to control our factories and refineries as well as power generation and distribution. We also use software in our transportation systems including airplanes, trains, ships, subways, and even in our family automobiles. Software is also a major component of many medical systems in which safe functioning is critical to the safety of patients and operators alike. Even when the software does not directly control safety-critical hardware, software can provide operators and users with safety-critical data with which they must make safety-critical decisions (e.g., air traffic control or medical information such as blood bank records, organ donor information, and patient medical

Cite this column as follows: Donald G. Firesmith: "Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements", in *Journal of Object Technology*, vol. 3, no. 3, March-April 2004, pp. 27-42. http://www.jot.fm/issues/issue_2004_03/column3

records). As we have come to rely more on software-intensive systems, we have come to rely more on those systems functioning safely.

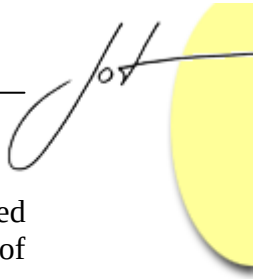
Many accidents are caused by problems with system and software requirements, and "empirical evidence seems to validate the commonly stated hypothesis that the majority of safety problems arise from software requirements and not coding errors" [Leveson1995]. Major accidents often result from rare hazards, whereby a hazard is a combination of conditions that increases the likelihood of accidents causing harm to valuable assets (e.g., people, property, and/or the environment). Most requirements specifications are incomplete in that they do not specify requirements to eliminate these rare hazards or mitigate their consequences. Requirements specifications are also typically incomplete in that they do not specify what needs to happen in exceptional "rainy day" situations or as a response to each possible event in each possible system state although accidents are often caused by the incorrect handling of rare combinations of events and states that were considered to be either impossible or too unlikely to worry about, and were therefore never specified. Even when requirements have been specified for such rare combinations of events and conditions, they may well be ambiguous (an unfortunately common characteristic of requirements in practice), partially incomplete (missing assumptions obvious only to subject matter experts), or incorrect, or inconsistently implemented. Thus, the associated hazards are not eliminated or the resulting harm is not properly mitigated when the associated accidents occur. Ultimately, safety related requirements are important requirements that need to be better engineered.

The goal of this column is to define safety requirements and clarify how they differ from safety constraints and from functional, data, and interface requirements that happen to be safety critical. I start by defining safety in terms of a powerful quality model and show how quality requirements (including safety requirements) can be specified in terms of the components of this quality model. I will then show how to use the quality model to specify safety requirements. Then, I will define and discuss safety constraints and safety-critical requirements. Finally, I will pose a set of questions regarding the engineering of these three kinds of safety-related requirements for future research and experience to answer.

2 SAFETY AS A QUALITY FACTOR

Quality Model

The term 'quality' is quite complex and often means different things to different people. To avoid this ambiguity, requirements engineers must use a quality model that makes the term quality specific and useful for specifying requirements [Firesmith2003a]. A quality model does this by decomposing the term 'quality' into its component concepts and their relationships to one another. A quality model first splits quality into its component quality factors (aspects, attributes, or characteristics) and subfactors (i.e., parts of quality factors). It then uses system-specific quality criteria (descriptions) and metrics (means of



measurement) to turn these general high-level quality factors and subfactors into detailed and system-specific measurable descriptions that can be used to specify an aspect of quality (and to determine if that aspect of quality actually exists at a level equal to or greater than the minimum amount specified in a requirements specification).

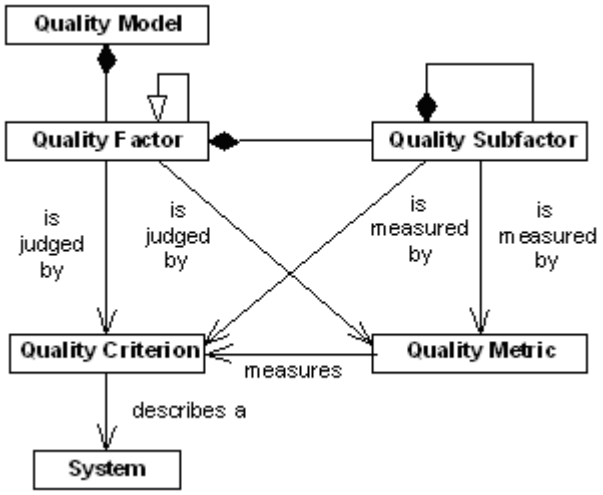


Fig. 1: Components of a Quality Model

Thus as illustrated in figure 1, a quality model is typically composed of the following four types of components:

1. **Quality factors** (also known as quality attributes or quality characteristics) are high-level characteristics or attributes of something that captures an aspect of its quality. Quality has to do with the degree to which something possesses a combination of characteristics, attributes, aspects, or traits that are desirable to its stakeholders. There are many different quality factors such as availability, extensibility, performance, reliability, reusability, safety, security, and usability [Firesmith2003b]. These factors determine whether or not something is of sufficiently high quality. Because many of the quality factors end in the letters "ility", they are often collectively referred to as the "ilities". Quality factors can be subclassed into more specific kinds of quality factors (e.g., safety is a *kind* of defensibility, which is a kind of dependability, which is a kind of quality).
2. **Quality subfactors** are major components (aggregation) of quality factors or other quality subfactors. Thus, throughput and response time are subfactors of performance, whereas integrity and privacy are subfactors of security.
3. **Quality criteria** are system-specific descriptions that provide evidence either for or against the existence of a given quality factor or subfactor. Quality criteria significantly contribute toward making the high-level quality factors and subfactors detailed enough to be unambiguous and verifiable. When quality criteria are adequately specific, they lack only the addition of quality metrics to make them sufficiently complete and detailed to form the basis for detailed quality requirements. If quality is the trunk of the tree and the quality factors and subfactors are the branches and twigs, then quality criteria are the leaves. There

are many more quality criteria than quality factors and subfactors because there are typically numerous criteria per factor and subfactor. Quality criteria are also more domain-specific and less reusable than quality factors and subfactors because they are specific descriptions of specific applications, components, centers, or business units. To deal with the large number of criteria and to make them reusable, quality criteria can often be parameterized in the quality models, and specific instances of the parameterized classes of criteria can then be used to produce quality requirements [Firesmith 03a]

4. **Quality metrics** provide a way to measure and quantify a quality criterion and thus makes it objective and unambiguous. A quality metric is a way of measuring that quantifies a quality criterion. Quality metrics thus provide numerical values specifying or estimating the quality of a work product or process by measuring the degree to which it possesses a specific quality factor or subfactor.

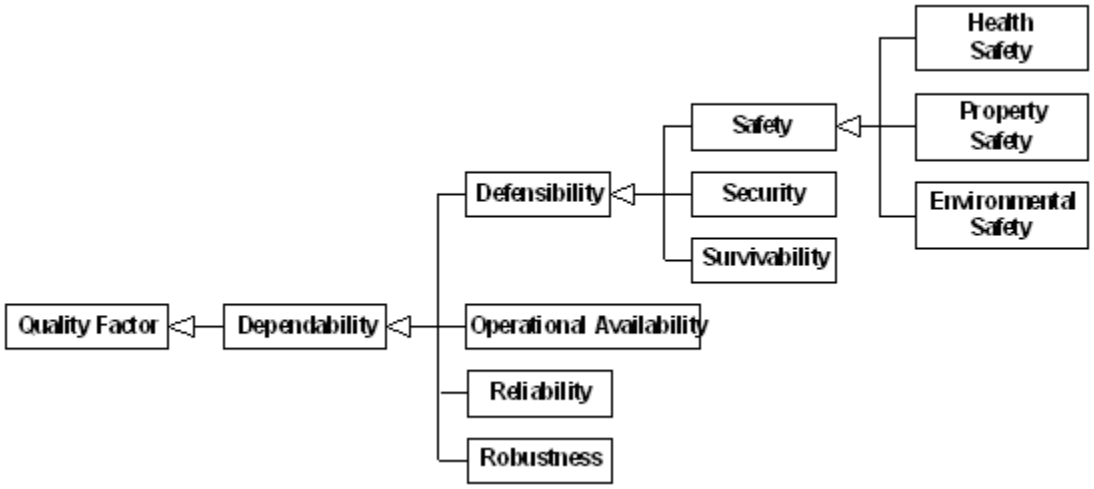
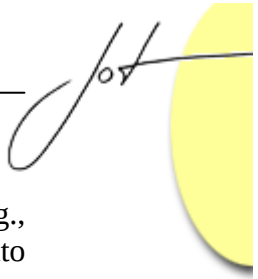


Fig. 2: Safety as a Quality Factor

As illustrated in figure 2, safety is part of a classification hierarchy of quality factors, whereby safety is a kind of defensibility, which is a kind of dependability, which in turn is a kind of quality factor. Safety can also be classified into health safety, property safety, and environmental safety. Because of the mapping from quality factors to associated requirements, safety requirements are a kind of quality (and dependability and defensibility) requirements. Safety requirements can also be classified as general safety requirements or else as health, property, or environmental safety requirements. To clarify these different kinds of quality factors, they are defined in the following list:

- **Dependability** is the degree to which various kinds of users can depend on a work product. Dependability is classified into the following quality factors:
  - **Defensibility** is the degree to which a system or component defends itself from accidents and attacks (e.g., defends itself from hazards and threats) Defensibility is classified into the following quality factors:



- **Safety** is the degree to which *accidental* harm is properly addressed (e.g., prevented, identified, reacted to, and adapted to). Safety is classified into the following quality factors:
  - **Health safety** is the degree to which illness, injury, and death are prevented, detected, and properly reacted to. Health safety involves all of the people who may reasonably be expected to be harmed by the system during an accident. For example, health safety for an automotive control system may include the driver, passengers, pedestrians, and mechanics. For a chemical plant control system, health safety may include operators, maintenance engineers, other staff at the plant, and nearby residents.
  - **Property safety** is the degree to which accidental damage and destruction of property is prevented, detected, and properly reacted to. This can include property that is part of the system, property that is owned by system stakeholders, or third party property such as property belonging to innocent bystanders or various branches of government.
  - **Environmental safety** is the degree to which accidental damage to (and destruction of parts of) the environment is prevented, detected, and properly reacted to.
- **Security** is the degree to which *malicious* harm is properly addressed (e.g., prevented, identified, reacted to, and adapted to).
- **Survivability** is the degree to which essential, mission-critical services continue to be provided in spite of either accidental or malicious harm.

— **Operational availability** is the degree to which a work product is operational and available for use by the user(s).

— **Reliability** is the degree to which a work product operates without failure under given normal conditions during a given time period.

— **Robustness** is the degree to which an executable work product continues to function properly under given abnormal conditions or circumstances (e.g., improper input and failures).

Safety as a Quality Factor in a Quality Model

The previous subsection introduced the fundamental concepts of a quality model and illustrated safety's position within that quality model. This information will be useful for defining the structure of quality requirements and for showing how safety requirements are related to other quality requirements, especially to other defensibility requirements (i.e., to security and survivability requirements).

Now, we will take a closer look at safety by decomposing it into its quality subfactors. The resulting aggregation hierarchy of safety subfactors will be used to identify a corresponding hierarchy of safety requirements built upon safety metrics and system-specific safety criteria for these safety subfactors. Because many requirements engineers know about only a few of the most obvious safety subfactors, they can also use the hierarchy of safety subfactors to help identify potentially missing types of safety

requirements. The hierarchy of subfactors can also be used to organize the resulting safety requirements.

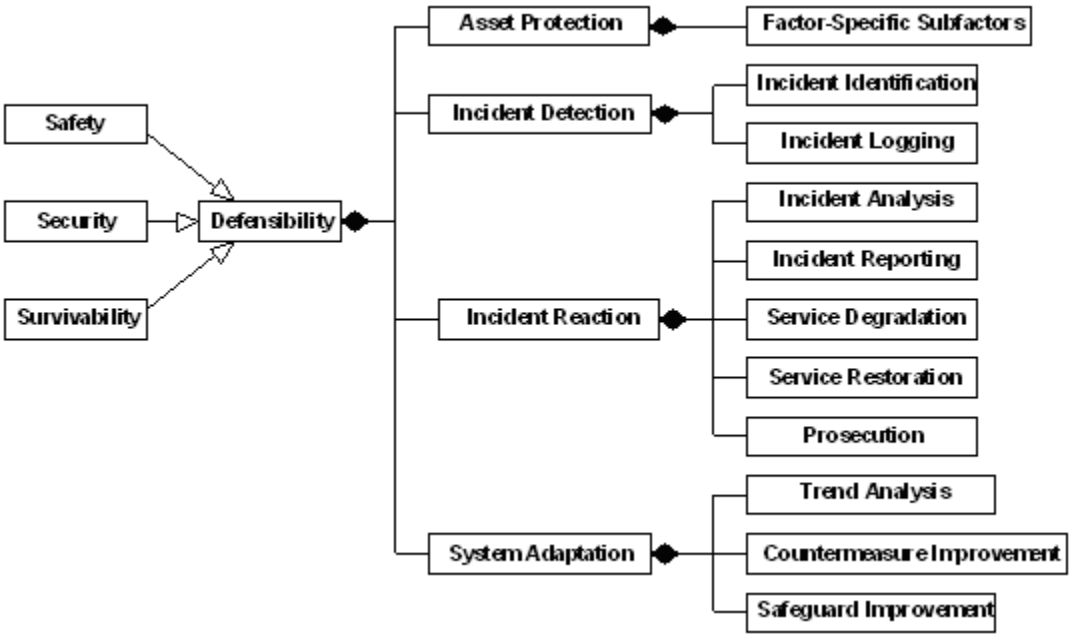


Fig. 3: Common Quality Subfactors of Defensibility (and therefore Safety)

As illustrated in figure 3, defensibility (and therefore safety, security, and survivability) is an aggregate that can be decomposed into a set of common quality subfactors.

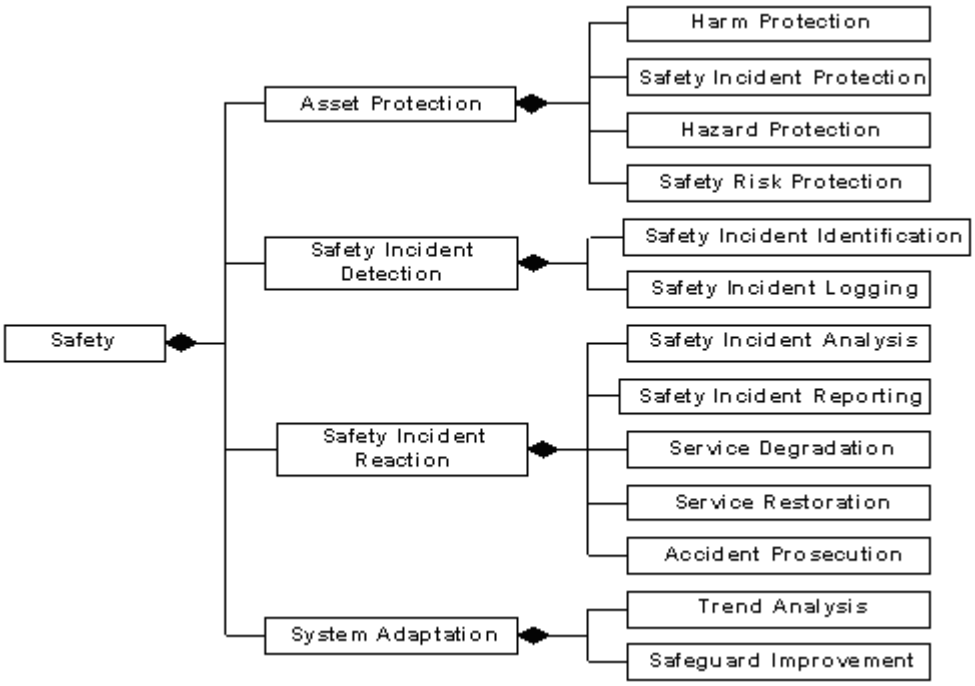


Fig. 4: Quality Subfactors of Safety

Figure 4 expands upon figure 3, illustrating both the quality subfactors that are common to all defensibility quality factors, as well as those that are specific to safety. Whereas all of these safety subfactors may not be relevant with regard to each system (and its hardware and software components), these safety subfactors should be considered during the engineering of safety requirements. To clarify these quality subfactors, they are defined in the following list:

- **Asset protection** (also known as prevention and resistance) is the degree to which valuable assets are protected. Asset protection is classified into the following quality subfactors:
  - **Harm protection** is the degree to which the likelihood or amount of harm to assets is eliminated or decreased.
  - **Safety incident protection** is the degree to which the likelihood of safety incidents is eliminated or decreased. Safety incident protection is classified into the following quality subfactors:
    - **Accident protection** is the degree to which the likelihood of accidents is eliminated or decreased.
    - **Near accident protection** is the degree to which the likelihood of near accidents (also known as near misses) is eliminated or decreased. Note that a near accident is the occurrence of an unplanned event during the occurrence of a hazard that does not result in significant harm.
  - **Hazard protection** is the degree to which the likelihood of hazards (i.e., sets of hazardous conditions that can cause an accident) is eliminated or decreased [DSCG1998].
  - **Safety risk protection** is the degree to which the likelihood of safety risks (typically the maximum amount of harm multiplied by the likelihood of associated hazards) is eliminated or decreased.
- **Safety incident detection** (also known as recognition) is the degree to which relevant safety incidents (or the harm that accidents cause) are recognized as they occur so that the system can react accordingly (e.g., to notify operators or safety personnel, to maintain essential services, to degrade gracefully). Safety incident detection is classified into the following quality subfactors:
  - **Safety incident identification** is the degree to which safety incidents (both accidents and near accidents) are identified as they occur.
  - **Safety incident logging** is the degree to which relevant information about safety incidents is logged as the safety incidents occur.
- **Safety incident reaction** (also known as recovery) is the degree to which the system responds to a safety incident (e.g., recovers from an accident). Safety incident reaction is classified into the following quality subfactors:
  - **Safety incident analysis** is the degree to which safety incidents are properly analyzed in a timely manner.
  - **Safety incident reporting** is the degree to which logged (and possibly analyzed) safety incidents are properly reported in a timely manner.

  - — **Service degradation** is the degree to which system services are properly degraded as a result of an accident (e.g., where practical, non-essential services are lost before essential services).
  - — **Service restoration** is the degree to which system services are promptly restored after being lost due to an accident.
  - — **Prosecution** is the degree to which the prosecution of malfeasance causing an accident is supported. Although this is more commonly a subfactor of security than of safety, it may be appropriate if gross negligence causes serious harm.
- **System adaptation** is the degree to which the system adapts itself (based on previous safety incidents) so that in the future it may better protect its assets, detect safety incidents, and react to them. System adaptation is classified into the following quality subfactors:
  - — **Trend analysis** is the degree to which the system tracks trends regarding the occurrence and impact of safety incidents.
  - — **Safeguard improvement** is the degree to which the system improves its safeguards as a consequence of previous safety incidents and the result of trend analysis. Although few systems today are smart enough to automatically improve their safeguards, this safety subfactor may become more practical in the future. This safety subfactor is also more important when accidents are common, which is why countermeasure improvement (security) is more prevalent (i.e., security attacks are unfortunately common whereas major accidents are thankfully rare).

Because the preceding definitions are relative in nature (i.e., they include the phrase “the degree to which”), they are consistent with the quality model relationship showing a quality subfactor being measured using a quality metric scale. As documented in the following section, this will support the engineering of associated safety requirements that specify a mandatory minimum acceptable measurement of some quality criterion using that metric.

3 SAFETY REQUIREMENTS

Quality Requirements

Building upon figure 1 which illustrated the structure of a quality model in terms of its quality factors, quality subfactors, quality criteria, and quality metrics, we now define quality requirements (such as safety requirements) and show how they relate to the components of the quality model.

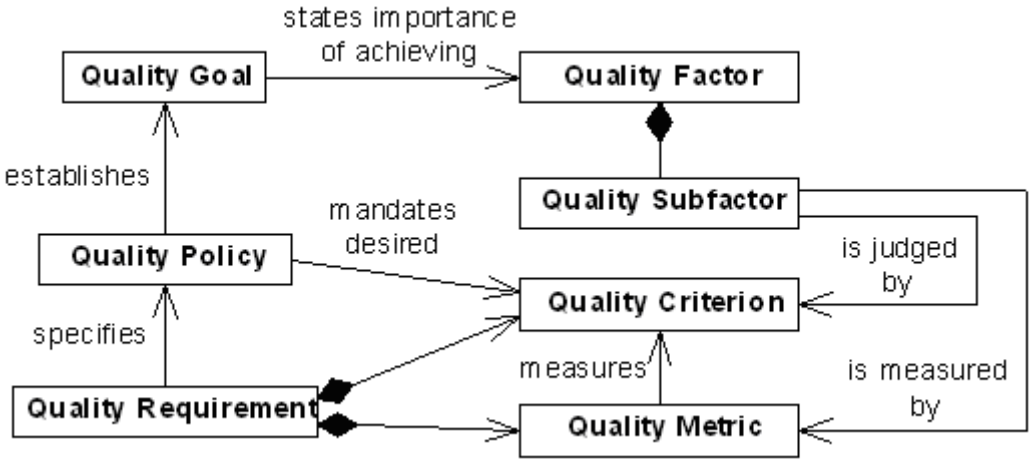


Fig. 5: Relationships from the Requirements Model to the Quality Model

As illustrated in figure 5, a quality requirement consists of a system-specific quality criterion that provides evidence for the existence of a quality subfactor together with a minimum required amount of a corresponding quality metric [Firesmith2003a]. Thus a safety requirement consists of a system-specific safety criterion providing evidence for the existence of one of the safety subfactors defined in the preceding section together with a minimum required amount of an associated quality metric (e.g., accidents per unit time, damage amount, and percentage of safety incidents logged).

Safety Requirements

To clarify the meaning of the concepts on the left side of figure 5, they are defined in the following list [Firesmith2003c]:

- A **goal** is a statement of the importance of achieving a desired target regarding some behavior, datum, characteristic, interface, or constraint. It is above the level of a policy and not sufficiently formalized to be verifiable.
  - A **quality goal** is a goal stating the importance of achieving a desired target regarding some quality factor or subfactor.
    - A **safety goal** is a quality goal stating the importance of achieving a desired target regarding some safety factor (e.g., Health Safety) or safety subfactor (e.g., Hazard Protection"). Examples of such safety goals would be "The system must not harm its users" or "The petroleum refinery control system must eliminate the hazards involving chemical explosions".
- A **policy** is any strategic decision that establishes a desired goal.
  - A **quality policy** is a policy mandating a desired criterion (or type of criteria) of a quality factor or subfactor.
    - A **safety policy** is a quality policy mandating a safety criterion (or type of safety criterion). An example of a safety policy would be "The petroleum

refinery control system must keep the pressures within reactant tanks significantly below their maximum pressure ratings".

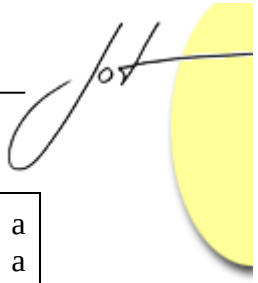
- A **requirement** is any mandatory, externally observable, verifiable (e.g., testable), and validatable behavior, datum, characteristic, or interface.
  - A **quality requirement** is any requirement that specifies a minimum amount of a mandatory quality subfactor in terms of a quality criterion and quality metric. A quality requirement is a kind of non-functional requirement like a data requirement, an interface requirement, or a constraint.
    - A **safety requirement** is any quality requirement that specifies a minimum, mandatory amount of safety (subfactor) in terms of a system-specific quality criterion and a minimum level of an associated metric. An example of a safety requirement would be "The petroleum refinery control system shall keep the pressures within reactant tanks at least 30% below their maximum pressure ratings at all times". Sometimes, safety requirements can be specified in an equivalent inverse of the normal way (e.g., in terms of the maximum acceptable amount of harm rather than the minimum acceptable amount of harm protection).

Example Safety Requirements

As illustrated in figure 4, there are a great many safety subfactors from which to choose. Thus, there are a great many types of safety requirements to be considered. Too often, true safety requirements are not engineered. Even when they are, the requirements engineer or safety engineer most often think in terms of specifying one of the four types of asset protection requirements. After all, it is better to prevent accidents (and near accidents) than to have to detect and react to them after they occur. Still because some safety incidents will occur no matter how much we try to engineer them out of our systems, it remains important to engineering safety incident detection and reaction requirements. Only in relatively intelligent systems are engineers beginning to think about specifying system adaptation requirements.

With the preceding in mind, I will present a few safety subfactors and representative examples of their corresponding safety requirements:

Safety Subfactor	Example Safety Requirement
Harm Protection	The automated airport subway system shall not injure passengers sufficiently to require hospitalization at an average rate greater than 0.000,000,1 passengers per passenger trip. (Note: this is estimated to be no more than approximately 5 injured passengers per year.)
Hazard Protection	The automated airport subway system train shall not start moving when its doors are still open more than once per year.



Safety Incident Identification	The automated airport subway system shall identify a combination of a train moving with its doors open with a probability of at least 99.99%.
Safety Incident Reporting	The automated airport subway system shall report to the safety officer occurrences of identified safety incidents at least 99.999% of the time.

Engineering Safety Requirements

Now that the different kinds of safety requirements have been identified, defined, and some representative examples have been given, a few words are in order concerning how they can be engineered:

- **Requirements elicitation.**
  - The list of safety subfactors can be used as a checklist to ensure that all kinds of *safety requirements* have been considered when eliciting safety requirements from the various stakeholders (e.g., customers, users, operators, maintenance personnel, etc.).
  - The identification of *safety constraints* (and some safety requirements) is supported by the existence of numerous industry-specific standards [Herrmann1999].
  - Although most current emphasis has been on *safety-critical requirements* rather than safety requirements, reuse of safety requirements and the experience of domain experts, safety engineers, and requirements engineers can still be valuable.
  - Safety policies provide high-level safety goals and policies that must be formally specified by the safety requirements (as well as safety constraints and safety-critical requirements).
- **Requirements analysis.**
  - Safety requirements deal with preventing accidental harm to valuable assets due to hazards that cause safety risks. Thus, an asset-based hazard-oriented risk analysis approach may well be the most appropriate approach to analyzing (and identifying) these requirements.
  - Although safety and reliability are not the same quality factors (e.g., you can have safe systems that are unreliable and reliable systems that are not safe), techniques for analyzing reliability may well provide some guidance. Some of these reliability analysis techniques include event tree analysis (ETA), fault tree analysis (FTA), and failure modes and effects analysis (FMEA).
  - Because reliability analysis techniques and most safety techniques such as hazard and operability studies (HAZOP) typically assume the existence of an architecture that will have parts that can fail [Leveson1995], these analysis techniques will typically be more useful for analyzing software requirements

rather than system requirements because the basic system architecture should exist by the time that software requirements analysis occurs.

Many systems are variants of existing systems. Thus, requirements engineers and safety analysts will have the architectures of previous systems on which to base their hazard analyses. Thus, the preceding architecture-based analysis techniques can be performed during system requirements analysis if they use the probable system architecture based on the architecture of previous systems. Besides, the systems requirements and system architecture may well be done iteratively, incrementally, and *in parallel with each other* so that partial architectures are another source of system architecture information for architecture-based analyses techniques.

Analysts should also remember that safety is an emergent property at the system level. Accidents occur when the system does the wrong thing or does the right thing at the wrong time or in the wrong sequence. Software by itself cannot cause these accidents. Software only causes accidents when interfacing with hardware (e.g., controlling actuators, communicating with other systems) or when providing misinformation (e.g., incorrect or obsolete information) to humans.

— Many quality factors are related to other quality factors (e.g., safety tends to decrease when performance is increased whereas safety tends to increase when reliability and robustness increase). Thus, a management and engineering tradeoff needs to be made when deciding on the appropriate level of safety to specify.

— Because the normal approach is to determine a minimum acceptable level of safety (and thus a maximum acceptable level of harm such as death, injury, and damage), this forces people to agree that some level of risk (and therefore harm) is acceptable and that some higher level of risk may not be acceptable. To bypass this unpleasant admission of fact, other approaches are sometimes used. For example, assuming that the current state of affairs is acceptable, the requirements team can specify at least as much safety as existing similar systems. Another approach is to specify as much safety as can be achieved given cost and schedule constraints [Gabb2004]. Another approach is to set a policy of as safe as reasonably practical (ASARP).

- **Requirements specification.**

— Because safety requirements are a combination of system-specific safety criteria and a mandatory minimum level of an associated metric, individual safety requirements can have a relatively standard structure based on a safety criterion combined with a minimum metric level.

— Because safety requirements are a subtype of quality requirements (specifically defensibility and dependability requirements) that can also be organized by safety subfactor, safety requirements can be organized into the same hierarchical structure.

- **Requirements management.**
  Safety requirements are derived from safety goals and safety policies (as well as from hazard analyses). Safety requirements also drive the selection of safeguards at the architecture, design, and implementation levels. Thus, the management of safety requirements must include traceability between these different kinds of work products.

4 SAFETY CONSTRAINTS

Rather than safety requirements, many industry and governmental standards and regulations typically concentrate on the specification of safety constraints. As defined in the following hierarchical list, safety constraints are clearly another way of specifying safety-related requirements:

- A **requirement** is any mandatory, externally observable, verifiable (e.g., testable), and validatable behavior, datum, characteristic, or interface.
  - A **constraint** is any engineering decision (e.g., architectural mechanism, design decision, implementation technique) that has been selected to be imposed as a requirement.
    - A **safety constraint** is any constraint that specifies a specific safeguard (e.g., architectural safety mechanism, safety design feature, safety implementation technique).

Safety constraints typically include things like requiring interlocks and physical barriers around moving parts, safeguards concerning electricity and the handling of toxic chemicals, and the mandatory placement of warning signs. A potential danger in the mandating of specific safeguards is that it may well be possible to architect a better system in which the associated hazards cannot occur and thus the mandated safeguards become unnecessary or inappropriate. In fact, the new system without the safeguards may be both cheaper and safer. For example, using magnets to keep refrigerator doors closed eliminated the need for installing safeguards to allow trapped children to open the previous locks from the inside.

5 SAFETY-CRITICAL REQUIREMENTS

The most common approach to dealing with safety during requirements engineering is to concentrate on the identification of safety-critical requirements. Unlike safety requirements which are a type of quality requirement, safety-critical requirements are typically functional, data, or interface requirements that must be properly implemented if hazards and their associated accidents are to be avoided or minimized. The completeness of safety-critical requirements (a potential source of accidents) can be addressed by considering all events in all modes (i.e., performing state and event completeness analysis) as well as performing I/O variable completeness analysis (e.g., all sensor input

should be used somewhere in the specification, and all legal output should be specified somewhere in the specification).

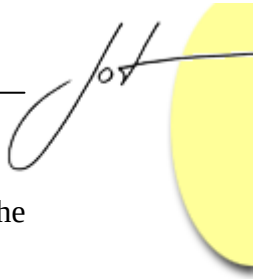
Consider for example an automatic subway system connecting terminals within an airport. There will be functional software requirements for starting and stopping the subway train, for accelerating and decelerating the trains between terminals, and for opening and closing the subway car doors. These requirements may be analyzed using use cases or using state transition diagrams. Clearly, while it is important to be able to travel between terminals and important to open the subway car doors at stops, it is also important to not open the doors when the subway is moving because people and their luggage can fall out, resulting in injury, death, and property damage. Similarly, accelerating or decelerating too fast can also cause people to fall and be injured. Thus, the software controlling subway train starting, stopping, accelerating, and decelerating as well as the software controlling the opening and closing of the doors is safety critical. Thus, the associated functional requirements are safety-critical, whereas the functional requirements concerning the announcement of arrival at a terminal are not safety critical.

Actually, identifying functional, data, and interface requirements as either safety-critical and non-safety-critical is probably too gross of a categorization. When using hazard analysis to categorize safety risks into safety assessment levels (a.k.a., safety integrity levels), one often obtains a larger number of safety risk categories such as: very high risk, high risk, medium risk, low risk, and no risk. Requirements having no safety risks can be viewed as non-safety-critical, but requirements having higher categories of safety risks should probably not be grouped together and categorized as safety-critical. Instead, such requirements should be categorized by safety risk level, possibly as follows: very high risk safety can be referred to as safety-critical requirements, high risk safety requirements can be referred to as safety-important, medium risk requirements can be referred to as safety-significant, and low risk requirements can be referred to as safety-relevant. This allows different safety evidence assurance levels for the requirements. For example, functional, data, and interface requirements that are safety-critical may need to be specified using a formal specification language, whereas the lower levels may need successively less evidence to support safety certifications.

With so many types and levels of safety-related requirements, finding appropriate names for each type of requirements is difficult. A comprehensive name is needed for safety requirements, safety constraints, and the non-lowest preceding safety assessment levels of functional, data, and interface requirements; these can all be referred to as safety-related requirements.

6 CONCLUSION

Hopefully, this column has opened your eyes to the existence of safety-related requirements and clarified the difference between safety requirements, safety constraints, and safety-critical requirements. All three types of requirements are important when



engineering the requirements for a safety-critical system, while as noted in the introduction, more and more systems are becoming safety critical.

However, the answers to many questions relating to the engineering of safety-related requirements either remain unanswered or deserve better answers. Hopefully, future research and practical experience will better address the following questions:

- How can we determine the optimum mix of the different kinds of safety-related requirements (i.e., safety requirements, safety constraints, and safety-critical requirements)? How much effort should be expended on each? Which are most important or are they all critical?
- What are the best, better, or most appropriate ways to elicit, analyze, specify, and manage these three different kinds of safety-related requirements? For example, safety requirements may need an asset-based hazard-oriented risk management approach, whereas safety-critical functional requirements may be better analyzed using a use case approach.
- What are the best, better, or most appropriate techniques for analyzing safety requirements? Many of the current techniques were designed for reliability and robustness rather than safety. Because many of them assume the existence of a system architecture, what techniques should be used during system requirements analysis prior to the existence of the system architecture?
- How do we keep the different kinds of safety-related requirements consistent? How can we best perform management and engineering trade-offs between them and the other types of requirements?
- Safety-critical software may well cost 3 times as much to develop as non-safety critical software, especially when safety certification is involved. What is the optimum way to perform trade-offs between safety and cost/schedule?
- What is the best approach for involving the requirements team, the safety team, and the stakeholders when engineering safety-related requirements?
- Many safety policies contain safety requirements and safety constraints which more properly belong in the requirements specifications as well as safeguards which belong in the architecture documents, design documents, and conventions such as coding standards. How do we ensure that the correct content ends up in the correct documents?
- How do we ensure that safety-related requirements are properly maintained (e.g., iterated or extended) based on the result of hazard analyses and the selection of safeguards?
- How do we convince requirements engineers that they are responsible for safety-related requirements and that safety is not just something that the safety team and the architecture team must deal with?

REFERENCES

[DSCG1998] Defence Standardization Coordination Group: *Procurement of Safety-Critical Systems Standard*, DEF(AUST)5679, Australia, August 1998. Available at http://www.dsto.defence.gov.au/isl/defaust5679.pdf

[Firesmith2003a] Firesmith, Donald G.: "Using Quality Models to Engineer Quality Requirements", in *Journal of Object Technology (JOT)* vol. 2, no. 5, September-October 2003, pp. 67-75. http://www.jot.fm/issues/issue_2003_09/column6

[Firesmith2003b] *OPEN Process Framework Website*. Available at http://www.donald-firesmith.com/

[Firesmith2003c] Firesmith, Donald G.: *Common Concepts Underlying Safety, Security, and Survivability*, CMU/SEI-2003-TN-033, Software Engineering Institute, December 2003. Available at http://www.sei.cmu.edu/publications/documents/03.reports/03tn033/03tn033.html

[Gabb2004] Gabb, Andrew: private communication, January 2004.

[Herrmann1999] Herrmann, Debra S.: *Software Safety and Reliability*, IEEE Computer Society, 1999.

[Leveson1995] Leveson, Nancy G.: *Software: System Safety and Computers*, Addison-Wesley, 1995.

About the author

Donald Firesmith is a senior member of the technical staff at the Software Engineering Institute. He has worked exclusively with object technology since 1984 and has written 5 books on the subject. He is currently writing a book on the engineering of safety, security, and survivability requirements. Most recently, he has developed a 1000+ page informational website on the OPEN Process Framework at http://www.donald-firesmith.com. He can be reached at dgf@sei.cmu.edu.