Carnegie Mellon
Software Engineering Institute

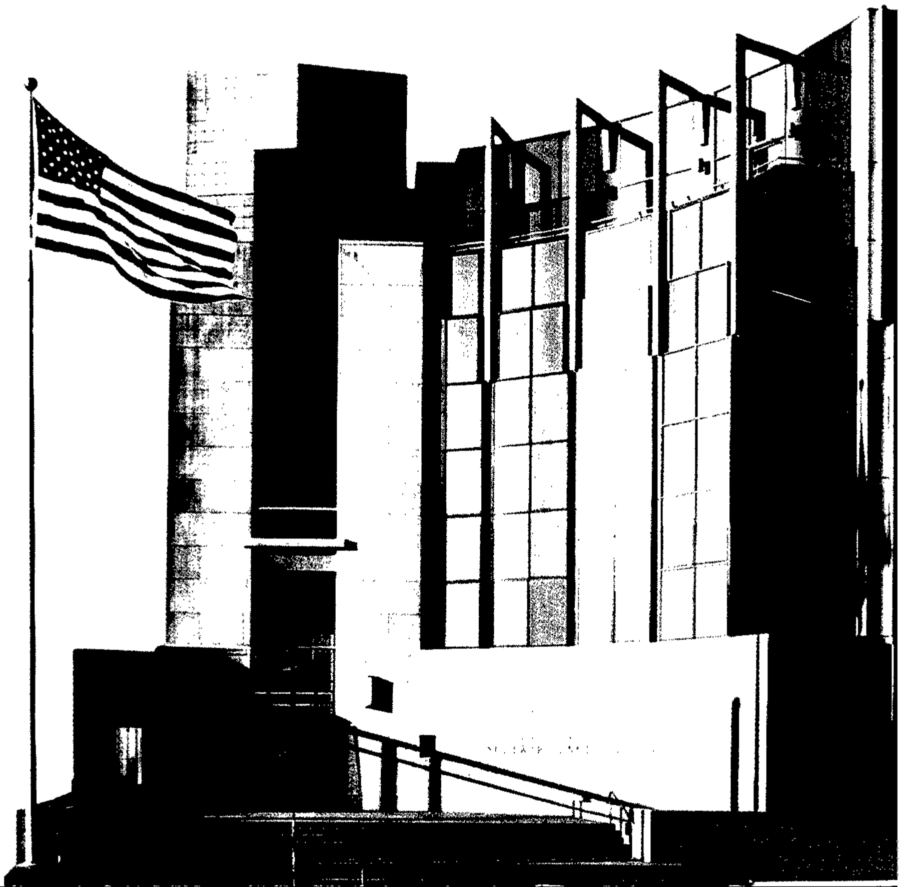
Common Concepts Underlying Safety, Security, and Survivability Engineering

Donald G. Firesmith

December 2003

Acquisition Support Program

Technical Note
CMU/SEI-2003-TN-033



Common Concepts Underlying Safety, Security, and Survivability Engineering

Donald G. Firesmith

December 2003

Acquisition Support Program

Technical Note
CMU/SEI-2003-TN-033

Contents

List of Figures

Acknowledgments

No non-trivial work springs *de novo* from its author like Athena from the head of Zeus, though as Zeus learned, creation is not without both hard work and headaches. Thus, this technical note is based on the hard work of many others, many of whose papers and books are listed in the reference section at the end. I would especially like to thank Jonathan Moffett and Bashar Nuseibeh for their cited paper and Carol Woody for her insightful discussions, both of which helped trigger the production of this technical note. Thank you for helping me to get these diagrams off my whiteboard and into print. And for the valuable constructive criticism that they have provided, I would also like to thank the following people who have reviewed this technical note:

- Ian F. Alexander (private consultant in requirements engineering, UK)
- Ulf Lundberg Andersson (private consultant, Sweden)
- Peter Capell (Carnegie Mellon® Software Engineering Institute [SEI[SM]])
- Gary Chastek (SEI)
- Suzanne Couturiaux (SEI)
- Kirk Dameron (Chaparral Network Storage, Colorado)
- Brian Henderson-Sellers (University of Technology Sydney and COTAR - the Centre of Object Technology and Research, Australia)
- Raimundas Matulevicius (University of Norway)
- Nancy Mead (SEI)
- Tim Morrow (SEI)
- Bashar Nuseibeh (The Open University, UK)
- Andreas L. Opdahl (University of Norway)
- Bhavani Palyagar (Macquarie University, Australia)
- Mike Phillips (SEI)
- Mary Popeck (SEI)
- Guttorm Sindre (University of Norway)
- Paul Tiplady (TRW Automotive, UK)
- Carol Woody (SEI)

[SM] SEI is a service mark of Carnegie Mellon University.

Executive Summary

Safety, security, and survivability engineering are three very closely related disciplines that could greatly benefit from a widespread recognition of their similarities and differences. Yet currently, members of these disciplines have inadequate interaction, either with each other or with members of other engineering disciplines.

This inadequate interaction among disciplines is especially true with regard to the engineering of safety, security, and survivability requirements upon which the associated architectural mechanisms (e.g., safeguards and countermeasures) should be based. Safety, security, survivability, and requirements engineers typically use different terminologies and processes that emphasize their differences and obscure their similarities. Far too often, the result is requirements specifications that are very incomplete with regard to safety, security, and survivability requirements. The "requirements" that are specified typically lack necessary characteristics such as verifiability and lack of ambiguity.

Most books and articles about safety, security, and survivability do not adequately describe the requirements for specifying minimum, mandatory amounts of these quality factors. That is, they do not address the elicitation, analysis, and specification of such requirements, nor do they address what such requirements should look like. Instead of discussing how to specify the level of safety, security, and survivability that is needed, they discuss accidents and attacks, hazards and threats, risks, vulnerabilities, and the architectural mechanisms that are used to prevent, detect, or react to these hazards and threats.

Although safety, security, and survivability requirements are beginning to attract interest, they typically lack any kind of theoretical foundation that defines what they *are* and how they relate to other important concepts. Until recently, the emphasis has clearly been on architectural mechanisms and the evaluation of the safety, security, and survivability of existing systems and architectures.

This technical note presents a set of related information models that provides the theoretical foundation underlying safety, security, and survivability engineering. It starts by summarizing the concept of a quality model and its component parts, three of which are the quality factors: safety, security, and survivability. Next, the different types of requirements are described, and a framework is provided showing how quality goals, policies, requirements, and architectural mechanisms are related to the quality factors and subfactors of the quality model. After the general model is summarized, three specific models are provided that relate safety, security, and survivability concepts, and the close relationships between the three engineering

disciplines are described. In addition to the graphical information models, precise definitions are provided for each concept. Finally, this technical note summarizes the similarities and differences between the models underlying these three types of engineering disciplines and suggests ways to take advantage of this commonality.

Ultimately, this technical note is about showing (and taking advantage of) the great similarities between

- *safety* (the degree to which *accidental harm* is prevented, detected, and reacted to)
- *security* (the degree to which *malicious harm* is prevented, detected, and reacted to)
- *survivability* (the degree to which *both accidental and malicious harm* to essential services is prevented, detected, and reacted to)

Abstract

This technical note presents a consistent set of information models that identify and define the foundational concepts underlying safety, security, and survivability engineering. In addition, it shows how quality requirements are related to quality factors, subfactors, criteria, and metrics, and it emphasizes the similarities between the concepts that underlie safety, security, and survivability engineering. The information models presented in this technical note provide a standard terminology and set of concepts that explain the similarities between the asset-based, risk-driven methods for identifying and analyzing safety, security, and survivability requirements as well as a rationale for the similarity in architectural mechanisms that are commonly used to fulfill these requirements.

1 Introduction

1.1 Challenges

Software-intensive systems are commonplace, and society relies heavily upon them. Software is found in automobiles, airplanes, chemical factories, power stations, and numerous other systems that are business and mission critical. We trust our lives, our property, and even our environment to the successful operation of these technology-based systems.

However, software-intensive systems are neither perfect nor invulnerable. They commonly fail due to software defects, hardware breakdowns, accidental misuse, and deliberate abuse. They are also the target of malicious attacks by hackers, disgruntled employees, criminals, industrial spies, terrorists, and even agents of foreign governments and their militaries. Yet, failure is becoming less and less of an option as we depend on these systems more and more. Thus, safety, security, and survivability engineering are becoming essential components of systems engineering.

Safety, security, and survivability engineering are three very closely related disciplines that could greatly benefit from a widespread recognition of their similarities and differences. Yet currently, members of these disciplines have inadequate interaction, either with each other or with members of other engineering disciplines.

This inadequate interaction among disciplines is especially true with regard to the engineering of safety, security, and survivability requirements upon which the associated architectural mechanisms (e.g., security practices such as training and procedures, security countermeasures such as encryption and firewalls, and safety mechanisms such as redundancy and safety components) should be based. Safety, security, survivability, and requirements engineers typically use different terminologies [CNSS 03, van der Meulen 00] and processes that emphasize their differences and obscure their similarities. Far too often, the result is requirements specifications that are very incomplete with regard to safety, security, and survivability requirements. The requirements that are specified typically lack necessary characteristics such as verifiability and lack of ambiguity.

However, most books and articles about safety, security, and survivability do not adequately describe the requirements for specifying minimum, mandatory amounts of these quality factors or adequately address the stakeholders who care about them. Instead, they typically concentrate on accidents and attacks, hazards and threats, risks, vulnerabilities, and especially the architectural mechanisms that are used to prevent, detect, or react to these hazards and threats. In fact, much of the material published in this area does not even mention the engineering of the associated requirements[1] [Alberts 03, Herrmann 99, Hughes 95, McNamara 03, Peltier 01, Power 00, Schneier 00, Shema 03, Tulloch 03]. The books that do mention requirements typically provide only the briefest and highest level overview without clearly saying what such requirements are [Anderson 01, Leveson 95, McDermid 91]. But without adequate goal- and policy-based requirements, how can we be sure that the safety, security, and survivability mechanisms that we choose to protect us will be adequate or appropriate?

Although safety, security, and survivability requirements are beginning to attract interest, they typically lack any kind of theoretical foundation that defines what they *are* and how they relate to other important concepts of safety, security, and survivability engineering. Until recently, the emphasis has clearly been on architectural mechanisms (such as security countermeasures) and the evaluation of the safety, security, and survivability of existing systems and architectures.

The analysis and specification of safety, security, and survivability requirements is inherently difficult. Unlike other requirements that specify a required (and desired) capability, these requirements specify what is to be prevented (e.g., accidents and attacks due to safety hazards and security threats). These requirements deal with assets that must be protected and with the risks of harm to these assets that must be managed. These requirements should be appropriate and cost effective; there is no value in specifying a requirement that will cost far more to implement than the value of the damage to the asset (and any downstream assets that might subsequently be harmed). And yet, there is an inherent level of uncertainty because what these requirements seek to prevent may or may not ever happen. This situation is especially true of safety requirements because some systems (e.g., nuclear power plants, chemical factories) are so critical that even a single, rare accident may render the system a complete failure. Although other systems (e.g., e-commerce Web sites) are essentially under constant attack, harm due to security threats often tends to be less mission critical, and a successful attack will not render the system a complete failure.

Another problem is that the hazards and threats associated with software-intensive systems are also constantly changing, making the risks very difficult to quantify. Estimates of risks are often actually "guesstimates," and thus the risks are typically forced to be qualitative rather than quantitative.

This technical note addresses these problems by showing (and recommending that engineers take advantage of) the great similarities between

- *safety* (the degree to which *accidental harm* is prevented, detected, and reacted to)
- *security* (the degree to which *malicious harm* is prevented, detected, and reacted to)

[1] This seems to be especially true for security books, perhaps because security engineers tend to think in terms of security policies rather than security requirements.

- *survivability* (the degree to which *both accidental and malicious harm* to essential services is prevented, detected, and reacted to)

1.2 Goals

A major goal of this technical note is to provide the reader with a solid foundation for safety, security, and survivability engineering. This technical note presents an overlapping set of information models that define the concepts of safety, security, and survivability engineering and clarify the relationships between them. When used, the foundation provided by these models will also enable stakeholders in safety, security, and survivability engineering to better communicate with each other.

A second goal of this technical note is to use these information models to clarify the similarities and differences between the foundational concepts of these three disciplines. The similarities between these three disciplines greatly outweigh their differences. These similarities are important because they allow engineers to develop relatively uniform development processes. These uniform processes can then be used across the traditionally separate disciplines of safety, security, and survivability engineering.

1.3 Motivation

Often, safety, security, and survivability engineering are not adequately recognized as highly related disciplines. Safety is largely about protecting valuable assets (especially people) from harm due to accidents. Security is largely about protecting valuable assets (especially sensitive data) from harm due to attacks. Survivability is largely about protecting valuable assets (essential services) from both accidents and attacks. In all three cases, a primary focus is in dangers (hazards and threats) and their associated risks and the system's vulnerabilities to them. All three disciplines often require a risk-driven approach in determining the appropriate policies, requirements, and architectural mechanisms. In fact, the similarities between these three disciplines have prompted the production of this technical note.

However, these similarities are seldom recognized, and their relevance to requirements engineering is largely unknown in actual practice. One problem is that requirements engineers are rarely taught safety engineering, security engineering, or anything about survivability, and they rarely have any significant experience in these disciplines. Yet, they are often responsible for developing safety, security, and survivability requirements. A good place for them to start is to learn the fundamental concepts underlying safety, security, and survivability. Similarly, safety and security engineers are rarely taught requirements engineering and typically have no experience engineering requirements. They tend to think in terms of safety program plans and security policies rather than requirements and requirements specifications. In addition, the policies they produce are at a higher level of abstraction than requirements and typically lack the characteristics of good requirements such as completeness, lack of ambiguity, and verifiability.

1.4 Contents

This technical note presents a set of related information models that provide a theoretical foundation underlying safety, security, and survivability engineering. Section 2 summarizes the concept of a quality model and its component parts, three of which are the quality factors of safety, security, and survivability. Next, Section 3 describes the different types of requirements and provides a framework showing how quality goals, policies, requirements, and architectural mechanisms are related to the quality factors and associated subfactors of the quality model. Section 4 provides three specific models that relate the underlying concepts of safety, security, and survivability engineering. In addition to the graphical information models, precise definitions are provided for each concept. Section 5 summarizes the similarities and differences between the models underlying the three disciplines, and Section 6 recommends ways to take advantage of this commonality.

2 Quality Model

Quality means much more than merely meeting functional requirements. Even if an application provides all of its required features and fulfills each of its use cases, it can still be totally unacceptable if it has insufficient quality attributes (e.g., it has inadequate availability, its capacity is too low, its performance is too slow, it is not interoperable with other systems, it is not safe to use, it has numerous security vulnerabilities, or it is not considered to be user friendly by its end users). Thus, the term "quality" is an abstract term that can mean very different things to different stakeholders (including customers, users, management, marketing, developers, testers, quality engineers, maintainers, and support personnel). In this technical note, the term "quality" is used in its widest sense. Thus, quality includes all quality factors and not just the few that are mentioned above.

Similarly, it is not adequate to specify required quality by merely stating that an application shall have high capacity and performance, be safe and secure, and be usable by its end users. Such "quality requirements" are typically specified at such a high level of abstraction that they are virtually useless because they are incomplete, vague, ambiguous, and impossible to verify. Thus, they are high-level goals rather than specific requirements.

Therefore, we need to decompose the term "quality" into its relevant component factors and subfactors. We also need to provide operational definitions for these components of quality so that we can create clear, unambiguous, and verifiable requirements. One purpose of this technical note is to provide such operational definitions for safety, security, and survivability and their requirements.

To specify quality requirements, we need a way to organize, clarify, and standardize the relevant meanings of the term quality when applied to software-intensive systems. Doing this will form a proper foundation for identifying, analyzing, and specifying the large number of quality requirements that are needed on any significant endeavor.

A quality model is a model (i.e., a collection of related abstractions or simplifications) that models the quality of something [Firesmith 03a]. The quality model does for the concept of quality what a map does for a city, state, or country; it captures all of the important generalities about quality while ignoring all of the diversionary details. This then is the role of a quality model: to make the general term "quality" specific and useful by decomposing it into its component concepts and their relationships to one another. A quality model first decomposes quality into its component quality factors (aspects, attributes, or characteristics) and subfactors (i.e., parts). It then provides specific quality criteria (descriptions) and metrics (means of measurement) that can be used to turn these general high-level quality factors into detailed and specific measurable descriptions that can be used to specify an aspect of quality or to determine if that aspect of quality actually exists at a level equal or above the minimum amount specified in a requirements specification. By mandating a combination of both a quality criterion and metric, the likelihood of obtaining a clear, unambiguous, and verifiable statement of a quality requirement increases.

There are many quality models of varying degrees of completeness and usability. Some are international standards [ISO 00], some are de facto industry standards [Firesmith 03d], some are organization specific [Barbacci 00], and some are published in software engineering books [Boehm 76, Chung 93, Chung 00, Davis 93, Keller 90, Loucopoulos 95, Mylopoulos 92, Roman 85, Sommerville 92, Thayer 90]. This technical note uses the Object Process, Environment, and Notation (OPEN) quality model [Firesmith 03d] due to its completeness, especially with regard to safety, security, and survivability.

2.1 Information Model for a Quality Model

As illustrated in Figure 1, a quality model is a hierarchical model consisting of quality factors (also known as quality attributes) containing quality subfactors. The model formalizes the concept of quality, and it decomposes quality into a taxonomy of relevant quality factors and subfactors. For each of these, it defines associated quality criteria and metrics that provide specific measurable descriptions of the quality that is being analyzed or specified.

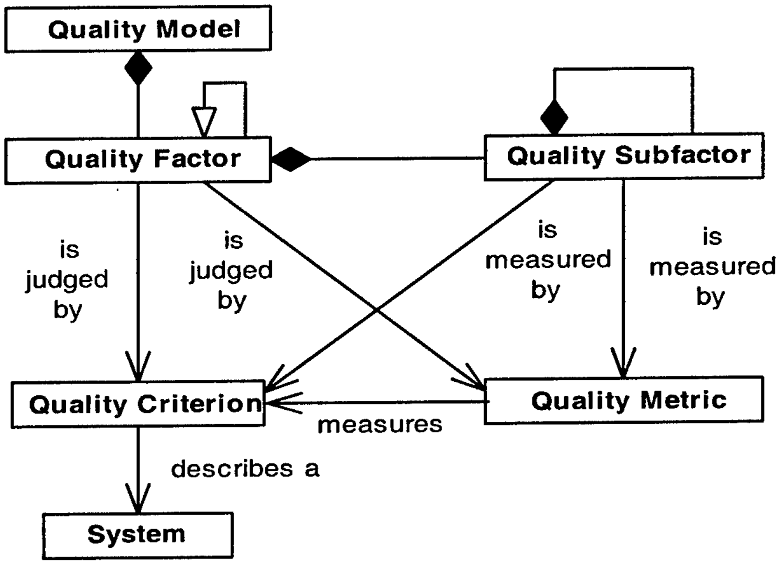


Figure 1: Information Metamodel for Quality Models

Figure 1 shows that quality factors and subfactors are the primary ways in which the concept of quality is decomposed. However, quality is made specific with regard to systems when it is judged by application-specific quality criteria and measured in terms of specific quality metrics.

Figure 1 is a Unified Modeling Language (UML) class diagram[2] that documents a metamodel for a quality model (where a metamodel is actually a model of a model). The *metamodel* for the quality model defines the *kinds of things* that make up any quality model, whereas a quality *model* contains the *actual quality factors, subfactors, criteria,* and *metrics.* Thus, for example, the metamodel contains the concept "quality factor," whereas the quality model contains specific quality factors such as safety, security, and survivability.

The concepts documented in Figure 1 can be defined as follows:

- **Quality model** is a hierarchical model for formalizing the concept of quality in terms of its component quality factors and subfactors.
  As such, the quality model forms a taxonomy of the characteristics that make up quality. The components of any taxonomy should be disjoint and cover the entire subject area of the taxonomy. Thus, the quality factors of a quality model should be disjoint and cover all of quality. Unfortunately, safety, security, and survivability engineering have developed and evolved relatively independently of one another; therefore, safety, security, and survivability have tended to overlap. Potential commonalities between these three may even show that they could benefit from adopting concepts from the others. Thus, some definitions of security include accidental harm that more correctly lies within safety. Similarly, some definitions of safety are incomplete because they include harm only to people and do not include harm to either property or the environment.
- **Quality factor** (also known as quality attribute or quality characteristic) is a high-level characteristic or attribute 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. 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 referred to as the "ilities." Quality factors can be subclassed into more specific kinds of quality factors (e.g., reliability is a *kind* of dependability). Quality factors can also be decomposed into their component parts (e.g., privacy is a *part* of security).
  - **Quality subfactor** is a major component (aggregation) of a quality factor or another quality subfactor.

[2] This technical note uses simplified UML class diagrams as entity relationship (ER) diagrams to capture the information models that define key concepts and the relationships between them. Concepts (entities) are signified by rectangles. Aggregation (i.e., whole-part, consists of) relationships are signified by arcs with a black diamond on their "whole" ends and nothing on their "part" ends. Inheritance (i.e., is-a-kind-of) relationships are signified by arcs with white arrowheads on their "superclass" ends and nothing on their "subclass" ends. All other relationships (associations) are signified by arcs that are labeled with verb phrases and that have arrowheads on their "dependent" ends. By reading in the directions of the arrows, associations can be read as sentences by reading the starting-box label, the arc label, followed by the ending-box label.

- **Quality criterion** is a specific description of something that provides evidence either for or against the existence of a specific quality factor or subfactor. Quality criteria significantly contribute toward making the high-level quality factors 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. 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 03b].

- **Quality metric** is a metric that quantifies a quality criterion and thus makes it measurable, 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.

- **System** (also known as system-level application) is an integrated collection of data components, hardware components, software components, human-role components (also known as wetware or personnel), and document components (also known as paperware) that collaborate to provide some cohesive set of functionality with specific levels of quality.
  Hardware components include firmware components, whereas software components include both application-specific software components as well as commercial off-the-shelf (COTS) components such as operating systems, database systems, and infrastructure components. Using a bank as an example, as far as the bank customer is concerned, the bank system includes the tellers, the procedures that tellers follow, and the training that they need. This inclusion of people and procedures is critical because safety and security are both chains that are only as good as their weakest links. These weakest links are often not in the hardware and software, but rather in the way they are used and supported.

2.2 A Taxonomy of Quality Factors and Subfactors

As pointed out in the previous list, a quality factor (also known as quality attribute or quality characteristic) is a high-level characteristic or attribute of something that captures an aspect of its quality. Note that quality factors and subfactors merely describe desired or existing capabilities. As such, they provide a foundation and organization for discussing quality goals, policies, requirements, and the architectural mechanisms for fulfilling these requirements. However, they are not themselves goals, policies, requirements, or architectural mechanisms.

Different books and articles decompose quality factors differently (e.g., qualitative quality factors versus quantitative quality factors). However, although it may be difficult to make all quality requirements quantitative, it is definitely both possible and useful to do so (e.g., for testability). Therefore, this technical note decomposes them into development-oriented and usage-oriented quality factors. Such a taxonomy seems to be very intuitive because it separates concerns according to their audiences (development oriented for developers and maintainers, usage oriented for customers and users).

The following taxonomy is also relatively complete. Although few projects need to develop requirements for all of the quality factors in the following two lists, requirements engineers should definitely determine the applicability of all of the different types of quality factors. Having a large hierarchical taxonomy of quality factors and subfactors also helps ensure that no relevant quality factors are ignored.

The following taxonomy of quality factors [Firesmith 03d] serves two primary functions:

- The lists provide a context for safety, security, and survivability. It thus provides a hierarchical taxonomy into which safety, security, and survivability must logically fit.
- By documenting the most important quality factors, the lists make it clear that a complete quality model can have a great number of quality factors and quality subfactors.

2.2.1 Development-Oriented Quality Factors

Development-oriented quality factors are quality factors that are primarily important during development and maintenance rather than usage. Examples of development-oriented quality factors and subfactors include the following:

- **Maintainability** is the ease with which an application or component can be maintained between major releases. Maintainability includes the following quality subfactors:
  - **Correctability** is the ease with which minor defects can be corrected between major releases while the application or component is in use by its users.
  - **Extensibility** is the ease with which an application or component can be enhanced in the future to meet changing requirements or goals.
- **Portability** is the ease with which an application or component can be moved from one environment to another.
- **Reusability** is the ease with which an existing application or component can be reused.
- **Scalability** is the ease with which an application or component can be modified to expand its existing capacities.

- **Verifiability** is the ease with which an application or component can be verified to meet its associated requirements and standards. Verifiability includes the following subfactor:
  - **Testability** is the ease with which an application or component facilitates the creation and execution of successful tests (i.e., tests that would cause failures due to any underlying defects).

2.2.2 Usage-Oriented Quality Factors

Usage-oriented quality factors are quality factors that are primarily important after deployment and during actual usage of an application or component. Examples of usage-oriented quality factors and subfactors include the following:

- **Auditability** is the degree to which sufficient records are kept to support a financial audit.
- **Branding** is the degree to which a work product (e.g., application, component, or document) successfully incorporates the brand of the customer organization's business enterprise.
- **Capacity** is the minimum number of things (e.g., transactions, storage) that can be successfully handled.
- **Configurability** is the degree to which something can be configured into multiple forms (i.e., configurations). Configurability includes the following quality subfactors:
  - **Internationalization** (also known as globalization and localization) is the degree to which something can be or is appropriately configured for use in a global environment.
  - **Personalization** is the degree to which each individual user can be presented with a unique user-specific experience.
  - **Subsetability** is the degree to which something can be released in multiple variants, each of which implements a different subset of the functional requirements and associated quality requirements.
  - **Variability** is the degree to which something exists in multiple variants, each having the appropriate capabilities.
- **Correctness** is the degree to which a work product and its outputs are free from defects once the work product is delivered. Correctness includes the following quality subfactors:
  - **Accuracy** is the magnitude of defects (i.e., the deviation of the actual or average measurements from their true value) in quantitative data.
  - **Currency** is the degree to which data remain current (i.e., up to date, not obsolete).
  - **Precision** is the dispersion of quantitative data, regardless of its accuracy.
- **Dependability** is the degree to which various kinds of users can depend on a work product. Dependability includes the following quality factors:

— **Availability** is the degree to which a work product is operational and available for use.

  The issue of availability is a difficult one for any quality model that seeks to minimize the overlap of quality factors in its taxonomy. When systems and software engineers think of availability requirements, they think in terms of non-malicious causes of lack of availability. However, a denial-of-service (DoS) attack is clearly a security problem. To maintain the disjointed nature of the taxonomy of quality factors, the scope of availability will remain non-malicious and DoS will be dealt with as a violation of a type of security requirements.

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

— **Robustness** is the degree to which an executable work product continues to function properly under abnormal conditions or circumstances. Robustness includes the following quality subfactors:

  - **Environmental tolerance** is the degree to which an executable work product continues to function properly despite existing in an abnormal environment.
  - **Error tolerance** is the degree to which an executable work product continues to function properly despite the presence of erroneous input.
  - **Failure tolerance** is the degree to which an executable work product continues to function properly despite the occurrence of failures, where
    - A failure is the execution of a defect that causes an inconsistency between an executable work product's actual (observed) and expected (specified) behavior.
    - A defect may or may not cause a failure depending on whether the defect is executed and whether exception handling prevents the failure from occurring.
    - A fault (also known as defect, bug) is an underlying flaw in a work product (i.e., a work product that is inconsistent with its requirements, policies, goals, or the reasonable expectations of its customers or users). Defects are typically caused by human errors, and defects have no impact until they cause one or more failures.

    Failure tolerance includes the following quality subfactor:

    - **Fault tolerance** is the degree to which an executable work product continues to function properly despite the presence or execution of defects.

— **Safety** is the degree to which *accidental* harm is prevented, reduced, and properly reacted to.

— **Security** is the degree to which *malicious* harm is prevented, reduced, and properly reacted to.

  The term "malicious" is used intentionally to clearly differentiate safety from security and thereby avoid an unnecessary overlap in the taxonomy of quality factors. Thus, safety deals with accidents, whereas security deals with attacks. However, accidents (safety) can result in security vulnerabilities that can be exploited by attacks, at which time their consequences fall within the realm of security. Similarly, attacks may cause safety hazards that in turn may cause accidents.

  - **Survivability** is the degree to which essential, mission-critical services continue to be provided in spite of either accidental or malicious harm.

- **Efficiency** is the degree to which something effectively uses (i.e., minimizes its consumption of) its resources. These resources may include all types of resources such as computing (hardware, software, and network), machinery, facilities, and personnel.
- **Interoperability** is the degree to which a system or one of its components is properly connected to and operates with something else.
- **Operational environment compatibility** is the degree to which a system or a component can be used and functions correctly under specified conditions of the physical environment(s) in which it is intended to operate.
- **Performance** is the degree to which timing characteristics are adequate. Performance includes the following quality subfactors:
  - **Jitter** is the precision (i.e., variability) of the time when one or more events occur.
  - **Latency** is the time it takes to provide a requested service or allow access to a resource.
  - **Response time** is the time it takes to *initially respond* to a request for a service or to access a resource.
  - **Scheduleability** is the degree to which events and behaviors can be scheduled and then occur at their scheduled times.
  - **Throughput** is the number of times that a service can be provided within a specified unit of time.
- **Utility** is the degree to which something can be accessed and used by its various types of users. Utility includes (but is not limited to) the following subfactors:
  - **Accessibility** is the degree to which the user interface of something enables users with common or specified (e.g., auditory, visual, physical, or cognitive) disabilities to perform their specified tasks.
  - **Installability** is the ease with which something can be successfully installed in its production environment(s).
  - **Operability** is the degree to which something enables its operators to perform their tasks in accordance with the operations manual.
  - **Transportability** is the ease with which something can be physically moved from one location to another.
  - **Usability** is the ease with which members of a specified set of users are able to use something effectively.

— **Withdrawability** is the ease with which an existing problematic version of the system or one of its components can be successfully withdrawn and replaced by a previously working version.

The preceding taxonomy makes it clear that safety, security, and survivability are kinds of dependability and are therefore usage-oriented quality factors. The preceding lists also emphasize the fact that safety, security, and survivability are only three of a great many potentially relevant quality factors.

2.3 Safety as a Quality Factor

As one of numerous quality factors, safety can be classified into the following subclasses of safety: health safety, property safety, and environmental safety. In fact, safety is often defined in terms of health, property, and environmental safety.

The information model illustrated by Figure 2 shows that safety is a kind of dependability and thus a kind of quality factor. It is also shows that safety has traditionally been classified into health, property, and environmental safety (also kinds of quality factors) based on the type of asset that will be harmed if an accident should occur.

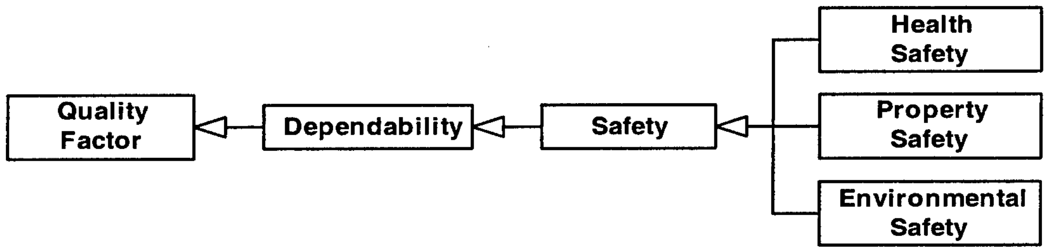


Figure 2: Safety as a Quality Factor

The concepts documented in Figure 2 can be defined as follows:

- **Safety** is the degree to which *accidental* harm is prevented, detected, and properly reacted to.
  In common English, people are not considered safe unless they are safe from both accidental and malicious harm. Also, security is often defined to include security from accidental harm. However, when dealing with systems, safety emphasizes accidental harm, and security emphasizes malicious harm. Thus, to have a taxonomy of quality with disjoint quality factors, safety will be restricted to accidental harm, and security will be restricted to malicious harm.
  The quality factor of safety can be classified into the following subclasses of safety, which themselves are also quality factors:

  — **Health safety** is the degree to which illness, injury, and death are prevented, detected, and properly reacted to.
  Health safety involves all 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.

— **Environmental safety** is the degree to which accidental damage to (and destruction of parts of) the environment is prevented, detected, and properly reacted to.

2.4 Security as a Quality Factor

The quality factors of safety and security can be viewed as two sides of the same coin. If safety can be defined as the degree to which *accidental* harm is properly managed, then security can be defined as the degree to which *malicious* harm is properly managed.

The information model illustrated by Figure 3 shows that security is also a kind of dependability and thus a kind of quality factor. The information model also shows that security has traditionally been classified into such subclasses as communications security, data security, emissions security (also known as TEMPEST), personnel security, and physical security based largely (as with safety) on the type of asset that will be harmed if an attack should occur.

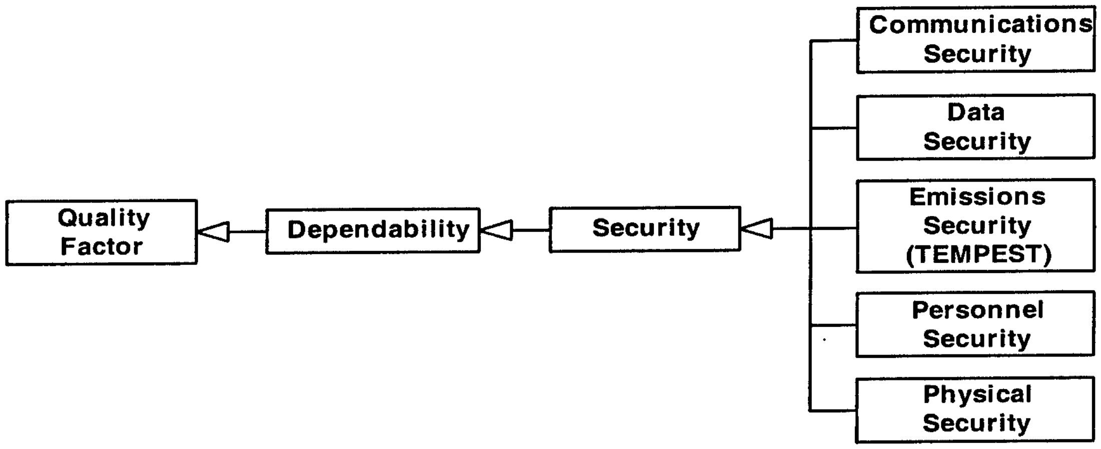


Figure 3: Security as a Quality Factor

The concepts documented in Figure 3 can be defined as follows:

- **Security** is the degree to which *malicious*[3] harm to a valuable asset is prevented, detected, and reacted to. Security is the quality factor that signifies the degree to which

[3] Some may argue that the term "malicious" is too strong. But what about hacktivists who vandalize the Web site of a company that pollutes the environment? What about someone who uses company computers to surf the Web in violation of company policy? The first example is a cybercrime, and the second is an unauthorized use of property. In both cases, the victims would be justified to consider these acts malicious. If the term "malicious" still seems too harsh, consider it to mean the combination of unauthorized and intentional harm.

valuable assets are protected from significant threats posed by malicious attackers. The quality factor of security can be classified into the following subclasses, which are also quality factors:

— **Communications security** is the degree to which communications are protected from attack.
— **Data security** is the degree to which stored and manipulated data are protected from attack.
— **Emissions security** is the degree to which systems do not emit radiation that is subject to attack.
— **Personal security** is the degree to which personnel are protected from attack.
— **Physical security** is the degree to which systems are protected from physical attack.

Figure 4 shows that security can also be decomposed (aggregation) into many different quality subfactors. In fact, security has historically been defined more often in terms of its most popular subfactors (typically availability, integrity, and privacy) than in terms of its subclasses. Note that security is a relatively complex concept and cannot be adequately addressed merely in terms of availability, integrity, and privacy. Unfortunately, there is no widely accepted, industry-standard decomposition of security into a taxonomy of its component quality subfactors, and these quality subfactors do not have industry-standard definitions. Perhaps this technical note will help to stimulate the development of a consensus concerning the optimal decomposition of security into its quality subfactors.

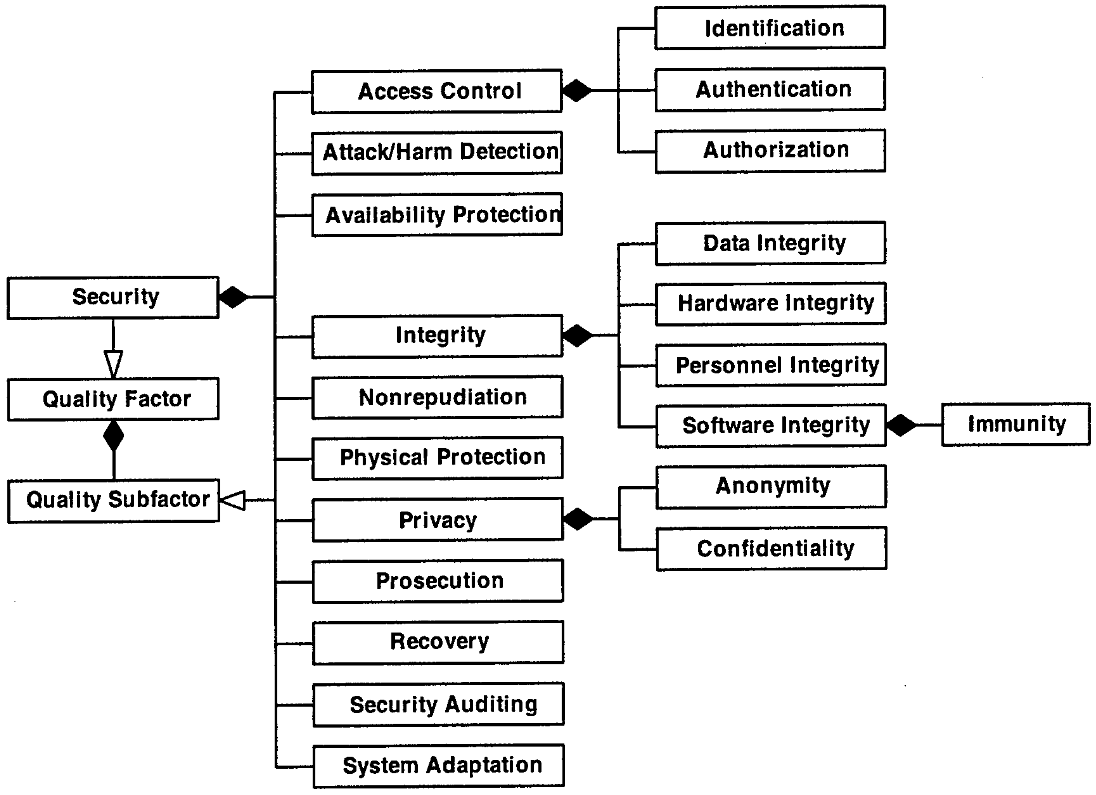


Figure 4: Decomposition of Security into Quality Subfactors

The security subfactors illustrated in Figure 4 can be defined as follows:

- **Access control** is the degree to which the system limits access to its resources only to its authorized externals (e.g., human users, programs, processes, devices, or other systems). The following are quality subfactors of the access-control quality subfactor:
  - **Identification** is the degree to which the system identifies (i.e., recognizes) its externals before interacting with them.
  - **Authentication** is the degree to which the system verifies the claimed identities of its externals before interacting with them. Thus, authentication verifies that the claimed identity is legitimate and belongs to the claimant.
  - **Authorization** is the degree to which access and usage privileges of authenticated externals are properly granted and enforced.
- **Attack/harm detection** is the degree to which attempted or successful attacks (or their resulting harm) are detected, recorded, and notified.
- **Availability protection** is the degree to which various types of DoS attacks are prevented from decreasing the operational availability of the system. This is quite different from the traditional availability quality factor, which deals with the operational availability of the system when it is not under attack.
- **Integrity** is the degree to which components are protected from intentional and unauthorized corruption. Integrity includes the following:
  - **Data integrity** is the degree to which data components (whether stored, processed, or transmitted) are protected from intentional corruption (e.g., via unauthorized creation, modification, deletion, or replay).
  - **Hardware integrity** is the degree to which hardware components are protected from intentional corruption (e.g., via unauthorized addition, modification, or theft).
  - **Personnel integrity** is the degree to which human components are protected from intentional corruption (e.g., via bribery or extortion).
  - **Software integrity** is the degree to which software components are protected from intentional corruption (e.g., via unauthorized addition, modification, deletion, or theft).
    - **Immunity** is the degree to which the system protects its software components from infection by unauthorized malicious programs (i.e., malware such as computer viruses, worms, Trojan horses, time bombs, malicious scripts, and spyware). Such protected software components include complete programs, partial programs, processes, tasks, and firmware.
- **Nonrepudiation** is the degree to which a party to an interaction (e.g., message, transaction, transmission of data) is prevented from successfully repudiating (i.e., denying) any aspect of the interaction.

  Nonrepudiation covers information such as the identities of the sender and recipient of the transaction; the send time, receive time, and dates of the transaction; and any data that flowed with the transaction. Nonrepudiation thus assumes data integrity so that a party cannot argue that the associated data were corrupted.

- **Physical protection** is the degree to which the system protects itself and its components from physical attack.
  Physical attack may mean something as violent as the use of a bomb or the kidnapping or blackmailing of personnel. It can also mean something as relatively minor as the prevention of the theft of a laptop by means of a cable and lock.
- **Privacy** is the degree to which unauthorized parties are prevented from obtaining sensitive information. Privacy includes the following subfactors:
  - **Anonymity** is the degree to which the users' identities are prevented from unauthorized storage or disclosure.
  - **Confidentiality** is the degree to which sensitive information is not disclosed to unauthorized parties (e.g., individuals, programs, processes, devices, or other systems).
- **Prosecution** is the degree to which the system supports the prosecution of attackers.
- **Recovery** is the degree to which the system recovers after a successful attack.
- **Security auditing** is the degree to which security personnel are enabled to audit the status and use of security mechanisms by analyzing security-related events.
- **System adaptation** is the degree to which the system learns from attacks in order to adapt its security countermeasures to protect itself from similar attacks in the future.

2.5 Survivability as a Quality Factor

Survivability is concerned with essential mission-critical services that must continue to be provided in spite of either *accidental* or *malicious* harm [Ellison 99, Ellison 03, Knight 00, Knight 03, Lipson 99]. Thus, in some sense, survivability can be seen as both a union of safety and security (accidental and malicious harm) as well as a subset of them (only interested in harm to essential services).

Survivability is typically not subclassed into lower level kinds of survivability based on the type of asset protected because it only deals with essential services; thus, there is only one kind of asset to protect. However, like security, survivability is typically decomposed into quality subfactors. Specifically, survivability consists of prevention, detection, and reaction as illustrated in the class diagram in Figure 5.[4]

[4] The use of a single yoke connected to both a black diamond (aggregation) and white arrowhead (inheritance) is not exactly valid UML, but drawing two overlapping yokes would significantly clutter the diagram. For example, the quality subfactors (inheritance arrow) prevention, detection, and reaction are components (aggregation diamond) of survivability.

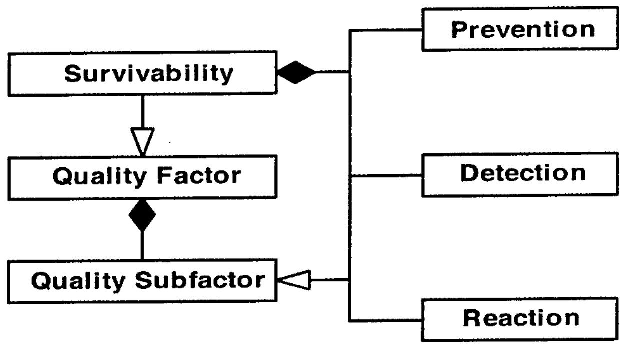


Figure 5: Decomposition of Survivability into Quality Subfactors

Survivability and its subfactors can be defined as follows:

- **Survivability** is the degree to which essential services continue to be provided in spite of either *accidental* or *malicious* harm. As noted by H. Lipson and D. Fisher [Lipson 99], "Ultimately, it is the mission [of the system] that must survive, not any particular component of the system or even the system itself." The following are the quality subfactors of the survivability quality subfactor:
  - **Prevention** (also known as resistance) is the degree to which hazards and threats are resisted so that essential services continue to be provided both during and after accidents and attacks. Prevention includes both the elimination of such hazards and threats as well as steps taken to minimize the negative outcome should an accident or successful attack occur.
  - **Detection** (also known as recognition) is the degree to which relevant accidents and attacks (or the harm they cause) are recognized as they occur so that the system can react accordingly to maintain essential services. It also typically involves the recording of attacks so that legal evidence exists with which to prosecute attackers. Detection may also include the recognition of conditions or events preceding accidents (e.g., hardware nearing failure) or detecting attackers collecting information during probes prior to attacks.
  - **Reaction** (also known as recovery) is the degree to which the system responds (e.g., recovers) after an accident or attack. This recovery includes the establishment of a priority-based recovery approach so that any essential services that may have been lost or degraded are recovered before the recovery of any non-essential services that were lost or terminated. Although some authors [Mead 03] have stated that recovery differentiates survivability from safety and security, reaction including recovery should apply not only to essential services but also to harm to any asset. For example, security has unfortunately concentrated on prevention and largely ignored how systems should detect and react to attacks.

2.6 Summary

The preceding section has documented the concept of a quality model, as well as its components and their interrelationships, and has led us to the definitions of the quality factors of safety, security, and survivability. With its figures and definitions, this section allows us to conclude the following:

- The quality of a software-intensive system is *not* a simple concept. To be useful, it must be decomposed into quality factors and subfactors, which allow us to speak about the different specific aspects of quality.
- There are many different kinds of quality factors, both development oriented and usage oriented. These quality factors will become part of the basis for organizing, identifying, and analyzing quality requirements.
- Safety, security, and survivability are quality factors that can be decomposed into standard quality subfactors that capture different fundamental aspects of what it means for a software-intensive system to be safe, secure, and survivable.
- Although safety, security, and survivability are different quality factors with different subfactors, they are actually related to each other in that all three involve the degree to which harm (whether accidental, malicious, or both) is prevented, detected, and properly reacted to.
- Safety, security, and survivability are different (although very similar) quality factors. A system can be safe and yet neither secure nor survivable as long as no person or property is harmed when a common attack successfully causes the loss of an essential service (e.g., a typical denial-of-service attack). Similarly, a system can be secure without being safe or survivable if an accident causes the loss of an essential service. Finally, a system can be survivable without being either safe or secure as long as frequent accidents and attacks do not cause the loss of an essential service.
- These quality factors and subfactors that make up the quality model are in turn judged by application-specific quality criteria and measured by associated quality metrics.
- Quality criteria are where the generally reusable quality factors and subfactors become very specific and application specific. There are often very many possible quality criteria that can be chosen for any given quality factor or subfactor, and these criteria can often be parameterized using a standard template with variable parts.
- Quality metrics describe the actual or required level of some quality factor or subfactor.

This section has provided a foundation for engineering safety, security, and survivability requirements because such requirements are typically a combination of quality criteria and metrics for the safety, security, and survivability quality factors.

3 Requirements Models

Proper requirements are critical for creating software-intensive systems that are safe, secure, and survivable. However, the related concepts of quality goals, policies, and architectural mechanisms are often confused with requirements. This section will clarify the differences between these concepts as well as their relationships to the quality models discussed in the previous section.

3.1 Information Model for Requirements

A complete requirements model must document the major, different kinds of requirements (e.g., functional requirements, quality requirements, data requirements, interface requirements, and constraints) [Firesmith 02]. It should also document the relationships between requirements and such other concepts as goals, policies, and architectural mechanisms.

Figure 6 shows that functional requirements, no matter how critical, are only one kind of requirement. Data, quality, and interface requirements as well as constraints must also be identified, analyzed, specified, and managed. The diagram also clarifies that safety, security, and survivability requirements are quality requirements rather than functional requirements. They are also architecturally significant requirements that have a much larger impact on the architecture (and application cost and development schedule) than most functional requirements.

Figure 6 is incomplete because it does not show all of the different kinds of quality requirements (i.e., those specifying mandatory amounts of other quality factors listed in the preceding taxonomy). As such, it is an oversimplification and therefore is not itself a strong argument for the similarity between safety, security, and survivability requirements. Note that different quality and requirements models may decompose things differently and thereby produce different diagrams.

Although requirements engineers have not traditionally recognized any intermediate model element between goals and requirements, the safety and security communities are very familiar with safety and security policies and have correctly understood that these critical policies are below goals but above requirements in this hierarchy. Therefore, policies are included between goals and requirements in our requirements information model.

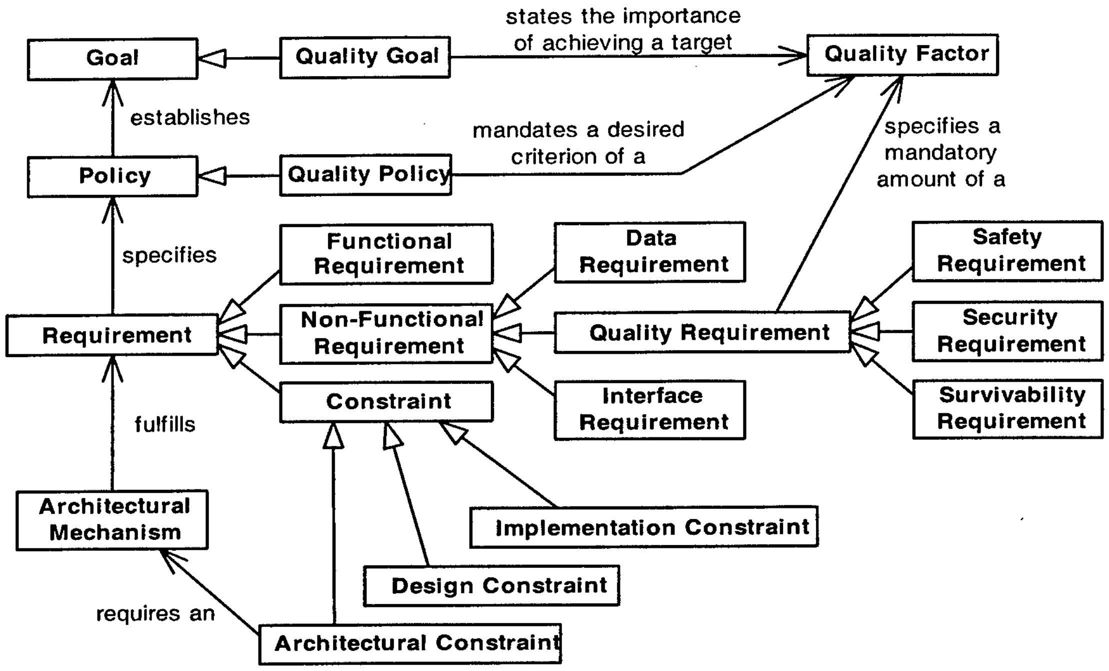


Figure 6: Requirements Information Model

Similarly, architecture mechanisms as well as architectural constraints are explicitly included in Figure 6 because many requirements engineers (as well as many safety and security engineers) mistakenly specify architectural mechanism (e.g., safeguards and countermeasures) as architectural constraints rather than specifying the true underlying safety and security requirements. Instead, they should leave the selection of safety and security architectural mechanisms to the architecture, safety, and security teams.

The concepts documented in Figure 6 are defined as follows:

- **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.
  - — **Quality goal** is a goal stating the importance of achieving a desired target regarding some quality factor or subfactor. (For example, "Sensitive information must be made secure," or "The confidentiality and integrity of sensitive information must be guaranteed.")
- **Policy** is any strategic decision that establishes a desired goal.
  - — **Quality policy** is a policy mandating a desired criterion (or type of criteria) of a quality factor or subfactor.
    (For example, "All information about customer credit cards that is entrusted to our organization shall be given a combination of technological and procedural security measures that together will ensure that all currently known types of attacks will be prevented from causing the unauthorized access, modification, or theft of this information.")

- **Requirement** is any mandatory, externally observable, verifiable (e.g., testable), and validatable behavior, datum, characteristic, or interface.
  Because we are concerned with the quality of a system or its components, the term "requirement" is used to refer to "product requirement." Thus, this document does not cover other types of requirements (e.g., process requirements such as development costs and development schedule) that are at the same level of abstraction.
  - **Functional requirement** is any requirement that specifies a mandatory behavior.
  - **Non-functional requirement** is any requirement that is not a functional requirement. Some authors mistakenly equate non-functional requirements with quality requirements. This taxonomy clearly shows that there are non-functional requirements that are not quality requirements. Types of non-functional requirements including the following:
    - **Quality requirement** is any requirement that specifies a minimum amount of a mandatory quality factor (i.e., characteristic, attribute). There are numerous types of quality requirements, including the following:
      - **Safety requirement** is any requirement that specifies a minimum, mandatory amount of safety.
      - **Security requirement** is any requirement that specifies a minimum, mandatory amount of security.
      - **Survivability requirement** is any requirement that specifies a minimum, mandatory amount of survivability.
    - **Data requirement** is any requirement that specifies a mandatory aspect of a datum or data type.
    - **Interface requirement** is any requirement that specifies a mandatory aspect of an external interface or protocol.
  - **Constraint** is any engineering decision that has been selected to be mandated as a requirement. There are several types of constraints, including the following:
    - **Architectural constraint** is any architectural decision that has been selected to be treated as a mandatory constraint (i.e., as a requirement).
    - **Design constraint** is any design decision that has been selected to be treated as a mandatory constraint (i.e., as a requirement).
    - **Implementation constraint** is any implementation decision that has been selected to be treated as a mandatory constraint (i.e., as a requirement). Examples include the selection of a programming language or a standard way of using the programming language (e.g., a "safe" and /or "secure" subset of the language constructs).
- **Architectural mechanism** is an architectural choice that provides a means for fulfilling one or more related requirements.
  This definition does *not* include only the software architecture. From a system architect's viewpoint, architectural mechanisms may be implemented by one or more of the system's components (including hardware, software, data, personnel, and documentation). Thus, architectural mechanisms may include training materials and operating procedures for both system-internal and system-external personnel. This systems-level viewpoint

recognizes that safety, security, and survivability cannot be achieved using only hardware, software, and data. All components of the system (including people, their procedures, and the training they receive) must be included to achieve a required level of safety, security, and survivability.

3.2 Quality Requirements and Quality Factors

At the highest level of abstraction, quality goals can be for either a quality factor or a quality subfactor. For example, there may be quality goals concerning safety, security, and survivability. Examples of safety goals might be, "The system must be safe" or "The system must not injure its users." Below this level, there may be more detailed quality policies that establish the associated quality goal by mandating a system-specific quality criterion (or criteria type) for the associated quality factor or subfactor. An example of a safety policy might be, "The automated tape library's moving parts shall not injure the tape technician when performing his or her duties. Finally, a specific and verifiable quality requirement consists of a combination of a quality criterion with a mandatory minimum level of an associated quality metric. Thus, the previous safety policy becomes a safety requirement when it is rewritten as follows: "At least 99.99% of the times that the tape technician performs the 'remove tape' use case, the automated tape library's moving parts shall not move (and thereby potentially cause an injury to the tape technician)."

Figure 7 relates the concepts of requirements to the concepts of the quality model.[5] It also provides a convenient way to partition work, with management setting quality goals and policy, while the requirements team (with input from the safety and security teams) specifies the associated quality requirements.

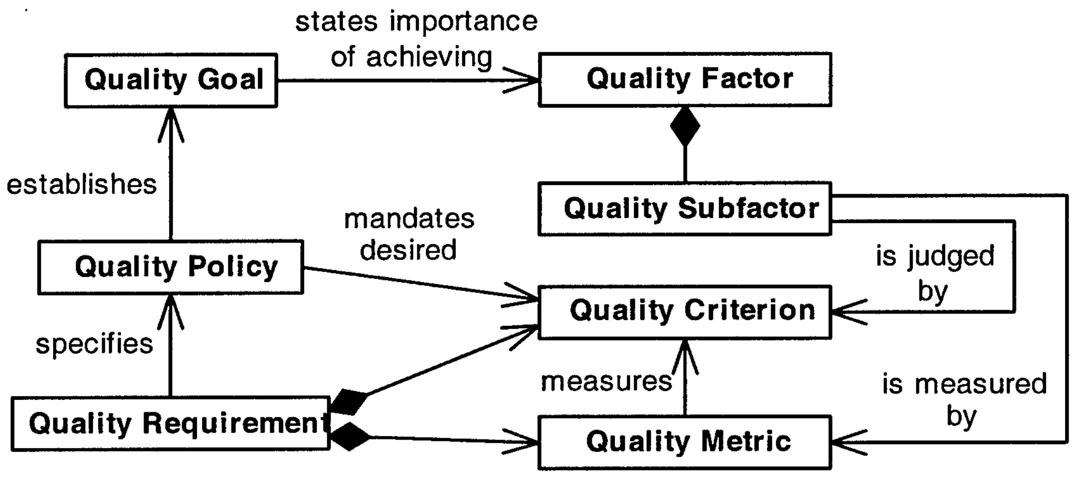


Figure 7: Relationships from Requirements Model to Quality Model

[5] The left side of Figure 7 comes from the left side of Figure 6, and the right side of Figure 7 comes from Figure 1.

As illustrated at the top of Figure 8, quality factors *characterize* different aspects of the system's quality and quality requirements *specify* different aspects of the system's quality by specifying levels of the associated quality factors and subfactors. Thus, safety requirements specify mandatory levels of safety, security requirements specify mandatory levels of security, and survivability requirements specify mandatory levels of survivability. Specifying these requirements is done by specifying an associated quality criterion and an associated minimum value for a quality metric. This information model thus pulls together parts of previous figures.

Figure 8 shows how the specific quality requirements of this technical note relate to their associated quality factors, quality criteria, and quality metrics that are described in the previous section.[6]

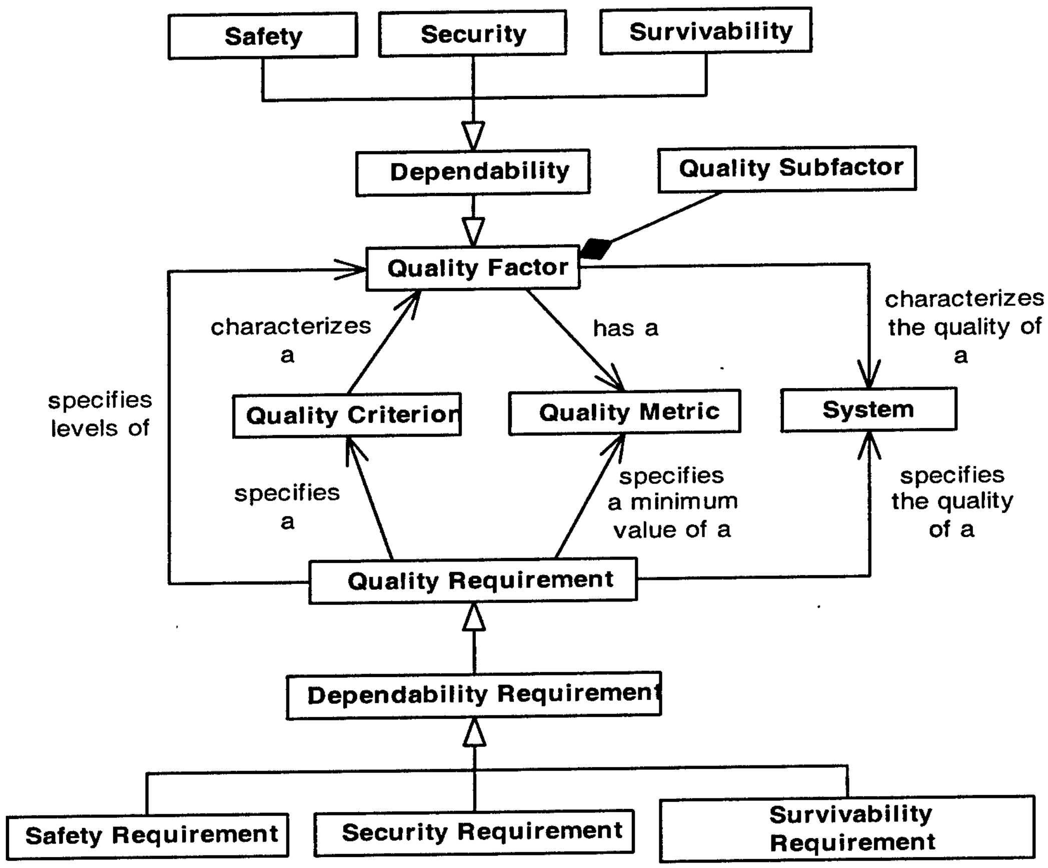


Figure 8: Relationships from Quality Requirements to Quality Model

[6] Figure 8 is composed of bits of Figure 1, Figure 5, and the list of usage-oriented quality factors.

3.3 Example Quality Requirements

To make the previous theoretical discussion more specific, consider those quality criteria associated with the integrity subfactor of the security quality factor. Quality criteria *types* describing integrity could include the following:

- Protect Transmissions from Corruption
- Detect Corruption of Transmitted Data
- React to Corruption of Transmitted Data
- Protect Online Stored Data from Corruption
- Detect Corruption of Online Stored Data
- React to Corruption of Online Stored Data
- Ensure Proper Restoration of Data and Software in Case of Corruption
- Protect Hardware Components from Corruption
- Protect Software Components from Corruption
- Protect Personnel Components from Corruption

An example of a parameterized version of the first quality criteria type above could be stated as follows:

- "The A protects B transmissions over C networks from D corruption by E attacks (or E attackers)," in which the preceding parameters can be replaced as follows:
  - A can be replaced with the following: business, center, application, or component.
  - B can be replaced with the following: a specific, personal, business confidential, classified, or all.
  - C can be replaced with the following: all, public, or internal.
  - D can be replaced with the following: creation, modification, deletion, replay, or all.
  - E can be replaced with the following: all, sophisticated, or unsophisticated.[7]

Thus, a specific integrity *quality criterion* for protecting *transmissions* from corruption could be written as follows: "The application protects all personal transmissions over all public networks from all types of corruption by unsophisticated attacks." An example of an integrity *quality criterion* for protecting data *stored online* from corruption might be, "The application protects stored customer information including account balances from unauthorized modification by sophisticated attacks."

[7] These terms (e.g., unsophisticated attack) must be officially defined in some sort of project glossary. There are also other possible decompositions besides the sophistication of the attack including known versus unknown attacks (e.g., viruses).

Whereas functional requirements tend to be binary and are specified as either all or nothing, quality requirements are specified using a scale of measurement. For example, the performance quality subfactor of throughput is typically specified in terms of number of transactions per unit time, while the quality subfactor of response time is typically specified in terms of seconds elapsed. Similarly, the scale of measurement for the integrity example in the previous section could be either of the following:

- average percent of transmissions protected per unit time under given conditions
- average number of transmissions corrupted per unit time under given conditions

If a quality criterion for data integrity is, "The application protects personal transmissions over all public networks from corruption by unsophisticated attacks," then the associated quality metric might be "average number of corruptions per hour." By providing a minimum acceptable level of this quality metric, we get the following data-integrity requirement: "At least 99.9% of the time, the application shall protect personal transmissions over all public networks from corruption by one hour of unsophisticated attacks."

To make the preceding requirement verifiable, it is necessary to require a percentage of the time that transmissions are successful (i.e., the security test fails to corrupt the transmission) as well as a specific attack load. This information is intended for testing (verification) purposes only and may not correspond to the application's actual future attack load, which most likely will be difficult or impossible to estimate accurately. This attack load needs to include the attacker's level of effort ("one hour") as well as an indication of the attacker's expertise and resources ("unsophisticated attack") that would be needed for the attack to be successful.

3.4 Summary

The preceding section has documented the different kinds of requirements and the way that quality requirements are related to their corresponding quality factors and subfactors. With its figures and definitions, this section leads to the following conclusions:

- Quality requirements are closely related to the corresponding components of a quality model.
- Quality requirements are made specific and verifiable by being based on a combination of quality criteria and quality metrics.

4 Engineering Models

This section documents the basic fundamental concepts underlying safety, security, and survivability engineering in terms of information models (UML class diagrams) and associated definitions. Building on the previous models, this section clarifies the similarities of and differences between these three disciplines. It also provides the basis for recommending their unification under the new umbrella discipline of defensibility engineering.

4.1 Information Model for Safety Engineering

Safety goals state the importance of achieving a target level of the quality factor safety. Safety policies establish safety goals by mandating desired safety criteria. Safety requirements (a kind of quality requirement) specify the safety policies by specifying mandatory amounts of safety in terms of specific safety criteria with an associated minimum acceptable measurement. Safety requirements thus require the elimination or reduction of safety risks. These safety risks are due to hazards, which provide an organizational framework to similar accidents that can cause harm to valuable system assets. Safety requirements in turn are fulfilled by safety architectural mechanisms (safeguards), which are intended to prevent or reduce vulnerabilities of the assets to accidental harm.

These basic concepts and their relationships are illustrated in Figure 9, an information model for safety engineering.

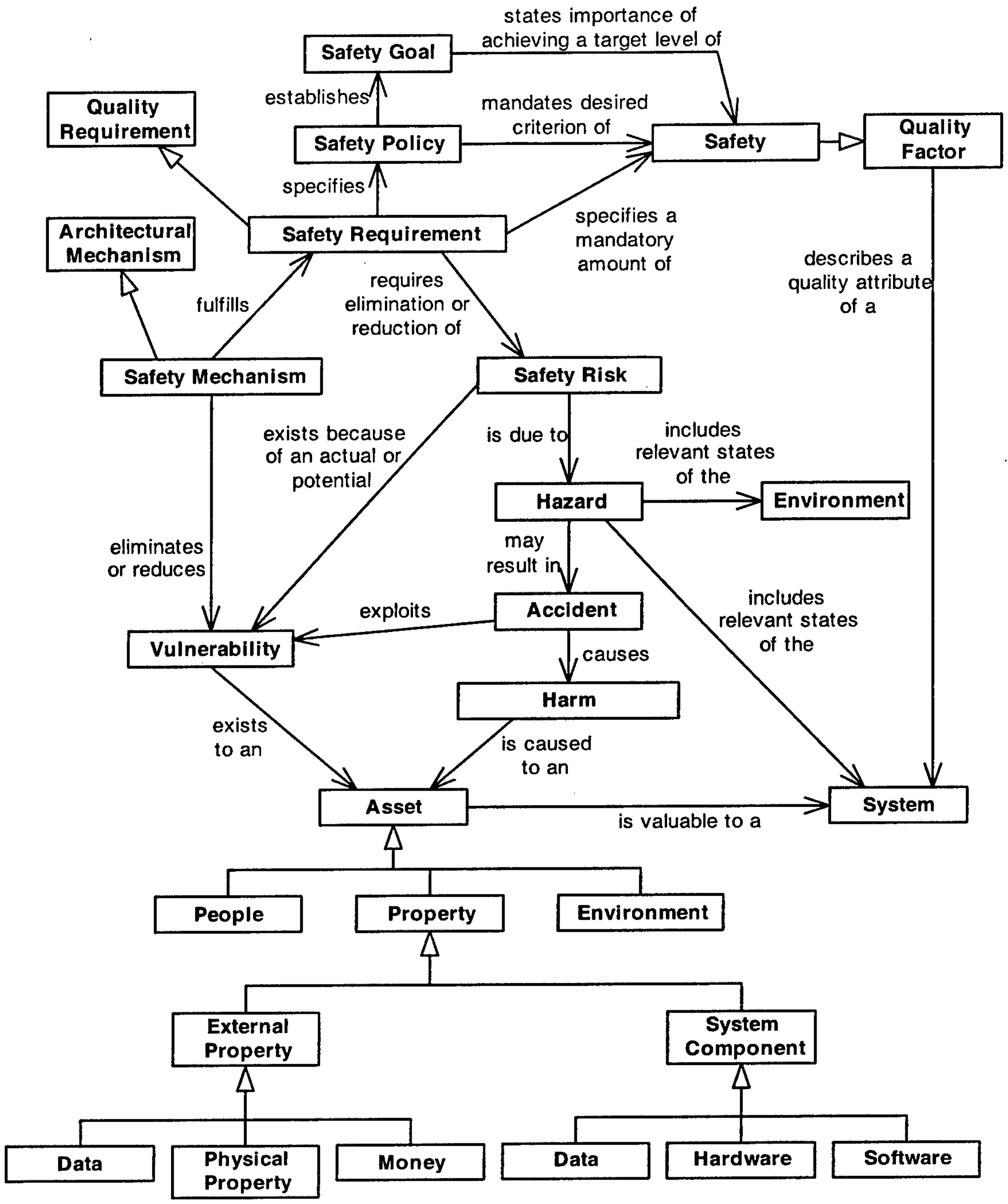


Figure 9: Information Model for Safety Engineering

Figure 9 shows how the important concepts from safety engineering (e.g., asset, accident, hazard, risk, vulnerability) relate to the important terms from requirements engineering (e.g., safety goal, policy, and requirement), quality engineering (e.g., safety), and architecture (e.g., safety mechanism). It also explains and justifies the common safety-analysis approach of analyzing risks in terms of vulnerabilities, hazards, accidents, and assets.

The concepts documented in Figure 9 can be defined as follows:

- **Accident** (also known as mishap) is an unplanned and unintended (but not necessarily unexpected) event or series of related events resulting in harm to an asset [ESA 97, IEEE 94, Leveson 95, NASA 97, UK 96].
  Accidents can be classified as follows:
  - **Health accidents** cause significant harm (e.g., illness, injury, or death) to people. Although safety engineering tends to emphasize the protection of people from accidental harm, its scope also includes property and the environment.
  - **Property accidents** result in damage to or destruction of properties.
    The emphasis is on external properties, but system components should not be ignored.
  - **Environment accidents** result in damage to the environment.
- **Asset** (with regard to safety engineering) is anything of value that should be protected from accidental harm.
  An asset requires protection because it is the potential subject of an accident. Assets can be any of the following:
  - **People** (also known as victims) are human beings who are harmed (i.e., develop occupational illnesses, become injured, or are killed) as a result of accidents. People can be classified as follows [Perrow 84]:
    - **First-party victims** are victims who are part of the system.
      Common examples include operators, managers, and maintenance personnel.
    - **Second-party victims** are victims who are external to the system but who intentionally interact with it.
      Common examples include users and suppliers.
    - **Third-party victims** are victims who are members of the general public who are innocent bystanders having no intentional involvement with the system.
    - **Fourth-party victims** are victims who are members of future generations (primarily victims of radiation, toxic chemicals, or pathogens).
      Common examples include unborn fetuses, would-be children that parents will not be able to conceive, deformed children, and future generations that must live in contaminated environments.
  - **Property** is any valuable property that may be damaged or destroyed if an accident occurs.
    In an automotive-control example, property could include the car itself and anything that it might damage while being driven (such as other vehicles, buildings, signage, telephone poles, traffic lights, and street lights).
    Property includes the following:
    - **External property** is personal, commercial, and civic property that exists outside the system.
      (Common examples include data, money, and physical property such as buildings and facilities.)

    - **System component** is any property that is a component of the system (including data, hardware, and software components).
  - **Environment** is the physical environment that may be damaged if an accident occurs. Continuing with the automotive-control example, the environment is the physical environment that can be harmed by accidents that release fluids (e.g., gasoline, coolant, hydraulic fluid) or start a roadside brush fire.

- **Harm** (when dealing with safety requirements) is significant damage to or a negative impact (i.e., negative outcome) associated with an asset due to an accident. Harm must be sufficiently significant to warrant relatively prompt remedial action to prevent such harm in the future.

  Harm is due to an *accident* when dealing with *safety* engineering, is due to an *attack* when dealing with *security* engineering, and may be due to both *accidents and attacks* when dealing with *survivability* engineering.

- **Hazard** is a situation that increases the likelihood of one or more related accidents [ESA 97, Leveson 95, NASA 97, UK 96].

  A hazard thus consists of hazardous states (i.e., a set of one or more incompatible system conditions or states, possibly including one or more conditions in the system's environment) together with the accident (type) they may cause.

  Potential hazards should be identified early during requirements engineering or architecting, while actual hazards may be identified in existing systems. The following are two examples of such potential and actual hazards with their various components identified:

  - **Potential hazard:** The subway doors are opening, open, or closing while the subway is moving (hazardous conditions), which may result in passengers and/or their property (assets) falling out (accident) and being injured, killed, or damaged (harm).
  - **Actual hazard:** Riders and/or their property within the doorway when the subway doors are closing (hazardous conditions) may result in the passengers and their property (assets) being crushed (accident) and thus injured, killed, or damaged (harm).

  Examples of primarily internal hazardous conditions include dangerous conditions involving hazardous chemicals, high voltages, and robotic-controlled moving machinery. A more specific example would be a moving elevator with open doors, two incompatible states of an elevator. Examples of primarily external hazardous conditions include fires and such natural disasters as earthquakes, floods, hurricanes, and tornadoes.[8]

- **Safety** is the quality factor signifying the degree to which *accidental* harm is prevented, detected, and properly reacted to.

- **Safety goal** is a quality goal that states the importance of achieving a target level of safety or one of its subfactors.

- **Safety policy** is a quality policy that mandates a system-specific quality criterion for safety or one of its subfactors.

[8] Safety engineering is clearly related to disaster management.

- **Safety mechanism** (also known as a safeguard or safety tactic) is an architectural mechanism (i.e., strategic decision) that helps fulfill one or more safety requirements and/or reduces one or more safety vulnerabilities.
- **Safety requirement** is a quality requirement that specifies a required amount of safety (typically a subfactor of safety) in terms of a system-specific criterion and a minimum mandatory level of an associated quality metric that is necessary to meet one or more safety policies.
  System-specific criteria can also involve the system's environment, the infrastructure in which it exists, and any assumptions about the system.
- **Safety risk** is the potential risk of harm to an asset due to accidents.
  Safety risk is defined as the sum (over all relevant hazards) of the products of the following two terms: (1) the largest negative impact of the harm to the asset (i.e., its criticality, severity, or damage) times (2) the likelihood that the hazard will result in an accident [Leveson 95].
  Using the basic theory of conditional probability, the likelihood that a hazard results in an accident causing harm can be calculated/estimated as the product of the following terms: (1) the likelihood that the hazard exists, (2) the likelihood that other necessary conditions also exist (also known as latency), and (3) the likelihood that the hazard will lead to an accident if it and the other necessary conditions exist (also known as danger).
  Potential safety risks should be identified early during requirements engineering or architecting, while actual safety risks may be identified in existing systems. The following are two examples of such potential and actual safety risks with their various components identified:
  - **Potential safety risk:** Unless one or more safety mechanisms are installed (potential vulnerability) to prevent the doors from opening when the subway is moving (hazardous conditions), there is an unacceptably high probability (likelihood) that passengers and/or their property (assets) will fall out (accident) and be injured, killed, or damaged (harm).
  - **Actual safety risk:** Due to a lack of sensors and associated software (vulnerability) to detect persons and their property within the doorway when the subway doors are closing (hazardous conditions), there is a significant probability (likelihood) that passengers and/or their property (assets) will be crushed (accident) and thus injured, killed, or damaged (harm).
- **Safety vulnerability** is a weakness in the system that increases the likelihood that an accident will occur and cause harm.
  This weakness may be in the architecture, design, implementation, integration, deployment, and configuration of the system. Examples of safety vulnerabilities include the lack of safety features, the lack of warning mechanisms, or defects that could cause failures.

4.2 Information Model for Security Engineering

Security goals state the importance of achieving a target level of the quality factor security. Security policies establish security goals by mandating desired security criteria. Security requirements (a kind of quality requirement) specify the security policies by specifying mandatory amounts of security in terms of specific security criteria with an associated minimum acceptable measurement. Security requirements thus require the elimination or reduction of security risks. These security risks are due to threat of attack by attackers, which are intended to cause harm to valuable system assets. Security requirements, in turn, are fulfilled by security architectural mechanisms (countermeasures), which are intended to prevent or reduce vulnerabilities of the assets to accidental harm.

These basic concepts and their relationships are illustrated in Figure 10, an information model for security engineering.

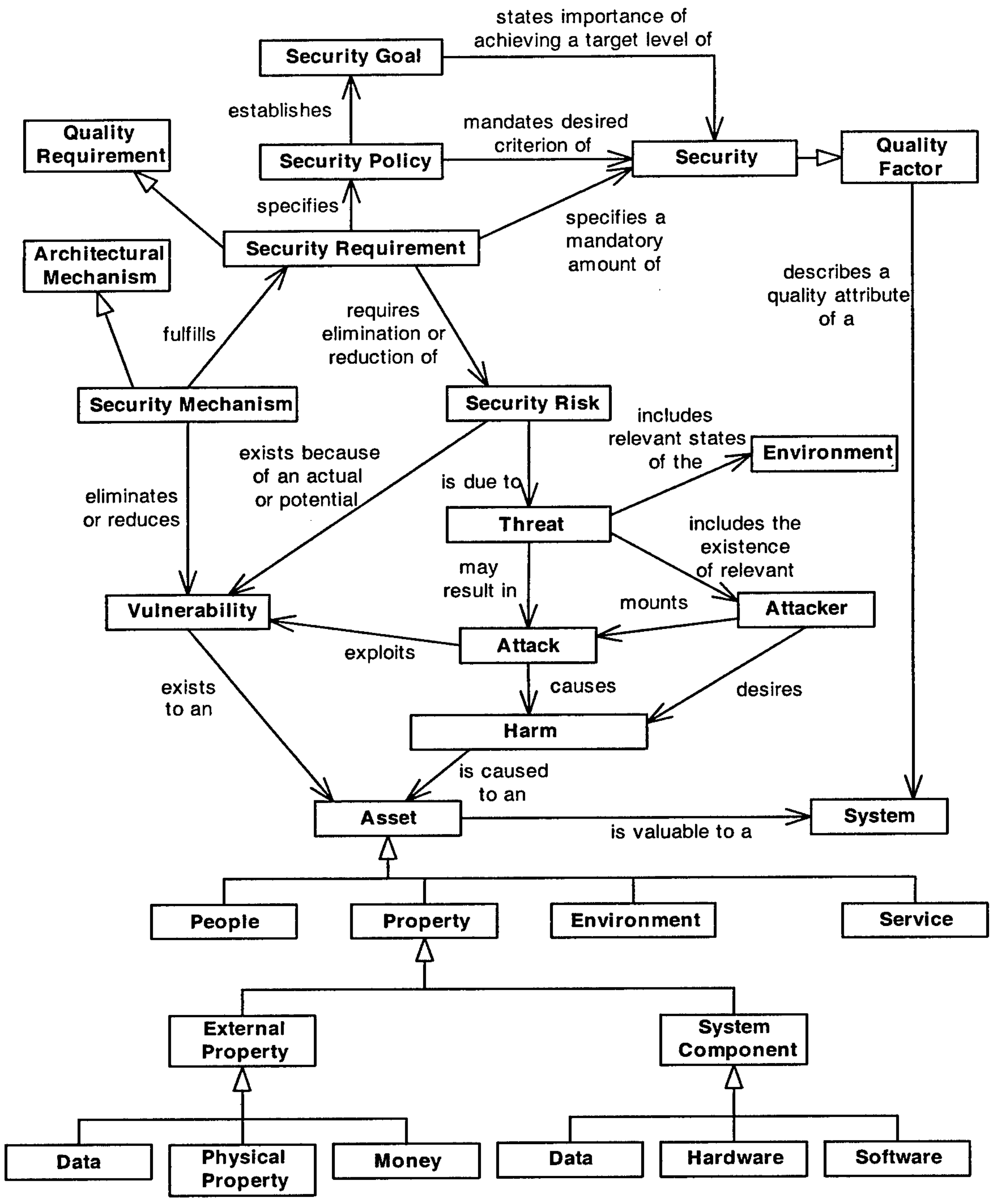


Figure 10: Information Model for Security Engineering

Figure 10 shows how the important concepts from security engineering (e.g., asset, attack, attacker, threat, risk, security policy, vulnerability) relate to the important terms from requirements engineering (e.g., security goal, policy, and requirement), quality engineering (e.g., security), and architecture (e.g., security mechanism). It also explains and justifies the common security-analysis approach of analyzing risks in terms of vulnerabilities, threats, attacks, and assets. Finally, its contents and topology show the clear relationship between safety and security engineering.

The concepts documented in Figure 10 can be defined as follows:

- **Asset** (with regard to security engineering) is anything of value that should be protected from malicious harm.
  With regard to security engineering, an asset requires protection because it is the potential target of attack. In security engineering, the emphasis tends to be on data assets (e.g., integrity and privacy), but security also includes software assets (e.g., integrity) and services (e.g., theft and denial of services). Physical security also deals with protecting people and other property including hardware and facilities.
- **Service** is any function or capability provided by the system.
- **Attack** (also known as security breach) is an attacker's unauthorized attempt to cause harm to an asset (i.e., violate the security of the system, bypass security mechanisms). An attack may be either successful or unsuccessful. Due to their malicious nature, most attacks are cybercrimes, which are crimes (e.g., theft of money or services, fraud, espionage, extortion, vandalism, terrorism, child pornography) carried out using computer resources. However, some unauthorized misuses of software-intensive systems are merely unethical or malfeasant rather than criminal.
- **Attacker** (also known as adversary) is an agent (e.g., person or program) that causes an attack due to the desire to cause harm to an asset.
- **Harm** (when dealing with security requirements) is a negative impact associated with an asset due to an attack. Harm is due to an *accident* when dealing with *safety* requirements, is due to an *attack* when dealing with *security* requirements, and may be due to both *accidents and attacks* when dealing with *survivability* requirements.
- **Threat** is a situation that increases the likelihood of one or more related attacks.
  The threat consists of the existence of one or more potential attackers together with a set of one or more system conditions or states that provide motivation to the attackers. Thus, the threat of theft may result in an actual theft (attack), and threats correspond to attacks that are typically classified by attacker motivation (e.g., theft) as opposed to technique (e.g., spoofing). In some books and articles, the different but highly related terms "attack" and "threat" are sometimes confounded by being used as synonyms [Tulloch 03].
- **Security** is the degree to which *malicious* harm to a valuable asset is prevented, detected, and properly reacted to. Security is thus the quality factor that signifies the degree to which valuable assets are protected from significant threats posed by malicious attackers.
- **Security goal** is a quality goal that states the importance of achieving a target level of security or one of its subfactors [Lamsweerde 00].
- **Security policy** is a quality policy that mandates a system-specific quality criterion for security or one of its subfactors. System-specific quality criteria can also involve the system's environment, the infrastructure in which it exists, and any assumptions about the system.

- **Security mechanism** (also known as countermeasure or security tactic) is an architecture mechanism (i.e., strategic decision) that helps fulfill one or more security requirements and/or reduces one or more security vulnerabilities.
  Security mechanisms can be implemented as some combination of hardware or software components, manual procedures, training, etc. It should also be noted that the same architectural mechanism (e.g., redundancy) can often be used as a safety, security, and survivability mechanism.
- **Security requirement** is a quality requirement that specifies a required amount of security (actually a quality subfactor of security) in terms of a system-specific criterion and a minimum level of an associated quality metric that is necessary to meet one or more security policies.
- **Security risk** is the potential risk of harm to an asset due to attacks.
  Security risk is the sum (over all relevant threats) of the negative impact of the harm to the asset (i.e., its criticality) multiplied by the likelihood of the harm occurring.
  Using the basic theory of conditional probability, the likelihood that harm results from an attack can be calculated/estimated as the product of the following terms: (1) the likelihood that the threat of attack exists, (2) the likelihood that other necessary conditions (e.g., vulnerabilities) also exist, and (3) the likelihood that the threat will lead to a successful attack if it and the other necessary conditions exist. The term "likelihood" is used rather than probability because the probability is typically not accurately or precisely known but rather only grossly estimated ("guesstimated").
- **Security vulnerability** is any weakness in the system that increases the likelihood that a successful attack (i.e., one causing harm) will occur.
  Security vulnerability is not restricted to only those vulnerabilities due to programming problems. It also includes vulnerabilities in the system's architecture and design, how the system is installed and configured, how its users are trained, etc. The vulnerabilities of a system may involve its data components, hardware components, software components, human-role components (i.e., wetware or personnel), and document components (i.e., paperware).

4.3 Information Model for Survivability Engineering

Survivability goals state the importance of achieving a target level of the quality factor survivability. Survivability policies establish survivability goals by mandating desired survivability criteria. Survivability requirements (a kind of quality requirement) specify the survivability policies by specifying mandatory amounts of survivability in terms of specific survivability criteria with an associated minimum acceptable measurement. Survivability requirements thus require the elimination or reduction of survivability risks. These survivability risks are due both to the hazard of accidents and the threat of attacks by attackers, which could cause harm to valuable system assets. Survivability requirements, in turn, are fulfilled by survivability architectural mechanisms, which are intended to prevent or reduce the vulnerabilities of the assets to accident and attack.

These basic concepts and their relationships are illustrated in Figure 11, an information model for survivability engineering.

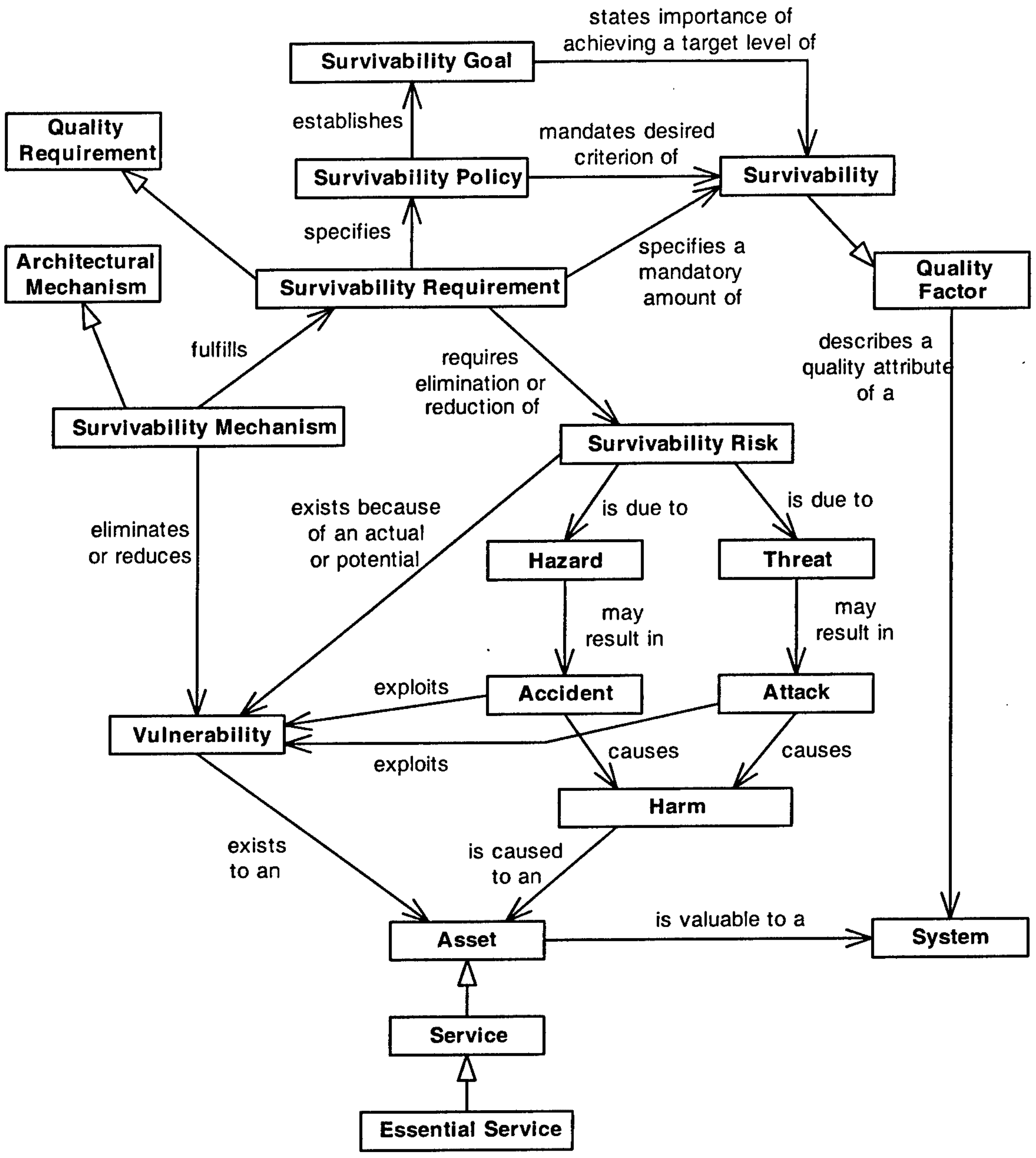


Figure 11: Information Model for Survivability Engineering

The similar content and topology of Figures 9, 10, and 11 show the clear and close relationships between survivability, safety, and security engineering. Figure 11 also justifies the common survivability-analysis approach of analyzing risks in terms of vulnerabilities, threats, hazards, and assets. In addition, it clearly shows how survivability is restricted to essential services rather than the other types of assets.

The additional concepts in Figure 11 that have not been defined in previous subsections include the following:

- **Essential service** is any mission-critical service that must continue to be provided in spite of either accident or attack.
- **Survivability** is the degree to which essential mission-critical services continue to be provided in spite of accidental and malicious harm.
- **Survivability goal** is a quality goal that states the importance of achieving a target level of survivability or one of its subfactors.
- **Survivability policy** is a quality policy that mandates a system-specific quality criterion for survivability or one of its subfactors.
- **Survivability mechanism** is an architecture mechanism (i.e., strategic decision) that helps fulfill one or more survivability requirements and/or reduces one or more survivability vulnerabilities.
- **Survivability requirement** is a quality requirement that specifies a required amount of survivability in terms of a system-specific criterion and a minimum level of an associated quality metric that is necessary to meet one or more survivability policies. Survivability requirements typically require the identification of essential mission-critical services (possibly as a function of system state and time) that must be provided without interruption, the identification of acceptable degraded modes of operation, the prioritization of the remaining alternative services, and the establishment of the time required for full service to be restored.
- **Survivability risk** is the potential risk of harm to an asset due to the sum (over all relevant hazards and threats) of the negative impact of the harm to the asset (i.e., its criticality) multiplied by the likelihood of the harm occurring.
- **Survivability vulnerability** is a weakness in the system that increases the likelihood that an accident or a successful attack will occur and stop an essential service from being provided.

4.4 Summary

The preceding section has documented the important concepts underlying safety, security, and survivability engineering. With its figures and definitions, this section leads to the following conclusions:

- The information models of safety, security, and survivability engineering are remarkably similar in both content and topology.
- Because of this consistency, safety, security, and survivability requirements can be elicited and analyzed in terms of a risk-oriented, asset-based approach that takes into account the associated hazards and threats from which these assets must be protected.

- Survivability engineering is nearly (but not quite) the combination of safety and security engineering. The primary difference lies in the type of asset to be protected (in this case, critical services that need to be maintained).

5 Similarities and Differences

As shown in the previous section, safety, security, and survivability engineering are very similar. This section summarizes these similarities as well as the differences and will provide a basis for the specific recommendations made in Section 6.

5.1 Similarities

As illustrated in Figures 9, 10, and 11, the information models for safety, security, and survivability have many similarities in contents and topology [Leveson 95]. The following subsections discuss these similarities in more detail.

5.1.1 Similarities Common to All Requirements

As pointed out in Figure 6, all quality requirements can be placed into the following hierarchical chain:

1. goals that drive policies
2. policies that drive requirements
3. requirements that drive architectural mechanisms
4. architectural mechanisms that fulfill requirements

This hierarchy applies to all quality factors and does not signify anything special about safety, security, and survivability engineering.

5.1.2 Similarities Common to All Quality Requirements

As pointed out in Figure 8, all quality requirements are related to quality factors. The relationships between requirements and quality factors also do not signify anything special about safety, security, and survivability.

5.1.3 Specific Similarities

Whereas the previous similarities were general, the following similarities are specific to safety, security, and survivability engineering. On Figures 9, 10, and 11, these similarities include the following:

All three disciplines

- — require the prevention or reduction of risks associated with hazards and/or threats
- — require the recognition and response to associated accidents and attacks

- exist to prevent, detect, or react to harm that may occur to some asset
- address potential vulnerabilities of assets to harm

5.2 Differences

The following subsections discuss the essential and accidental differences between the information models underlying safety, security, and survivability engineering.

5.2.1 Quality Factor Subclasses and Subfactors

Safety, security, and survivability are quality factors that differ in that they currently have quite different subclasses and quality subfactors:

- **Different subclasses**: Safety and security have different subclasses, whereas survivability does not seem to have subclasses. The subclasses of safety and security are both essentially based on the types of assets to which harm can occur.
  - **Safety subclasses**
    - health safety
    - property safety
    - environmental safety
  - **Security subclasses**
    - communications security
    - data security
    - emissions security (TEMPEST)
    - personnel security
    - physical security
- **Different subfactors**: Security and survivability have different subfactors, whereas safety does not seem to have subfactors. The subfactors of security and survivability do not seem closely related.
  - **Security subfactors**
    - access control
      - identification
      - authentication
      - authorization
    - attack/harm detection
    - availability protection
    - integrity
      - data integrity
      - hardware integrity

            - personnel integrity
            - software integrity
                - immunity
        - nonrepudiation
        - physical protection
        - privacy
            - anonymity
            - confidentiality
        - security auditing
        - system adaptability
    - **Survivability subfactors**
        - prevention
        - detection
        - reaction

Other than the historical accident that these three disciplines have been developed largely independently of one another and that all three have been largely driven by the architectural mechanisms that have been used to fulfill them, there seems to be little reason why the preceding three decompositions should be so different. Although prevention is preferable to cure, it is clear that prevention has been strongly emphasized in all three disciplines, while detection and especially response have been highly under-emphasized. Thus, most of the quality subfactors of security can be placed under the banner of prevention.

5.2.2 Assets

Safety, security, and survivability tend to emphasize the protection of different kinds of valuable assets from different kinds of harm.

- **Safety** emphasizes protecting people from harm, although it also can and should protect property and the environment from harm.
- **Security** emphasizes protecting property (data) and services (e.g., denial of service) from harm, although it also can and should protect people (e.g., physical protection) and other kinds of property (e.g., hardware theft, facility sabotage, and software integrity from viruses).
- **Survivability** is currently restricted to protecting essential services from harm. Yet, the meaning of "essential" is sometimes neither absolute nor constant. There may be multiple valid sets of services with a sequence of priorities that may change as external circumstances change [Knight 00, Knight 03]. Also, the safety and security of other assets may directly affect the system's ability to provide essential services and therefore fulfill its primary mission.

Nevertheless, these differences are somewhat minimized when safety and security properly address all types of valuable assets. However, protecting the environment from malicious harm does not seem to be within the current scope of security, although acts of eco-terrorism and international terrorism could well change that.

5.2.3 Accidents vs. Attacks

Because of the separation of scope of safety and security into accidental and malicious harm, safety and security deal with different (though related) types of incidents, and survivability deals with both kinds of incidents.

- **Safety requirements** protect assets from harm due to *accidents*.
- **Security requirements** protect assets from harm due to *attacks*.
- **Survivability requirements** protect assets (essential services) from harm due to both accidents and attacks.

As illustrated in Figure 12, the source of relevant harm differs with respect to the different kinds of requirements:

- **Safety engineering** deals with accidents that are due to human error (e.g., operator or user error), technological failures (e.g., failures resulting from defects), and natural causes such as science (e.g., the physics of electricity, the chemistry of explosives), engineering (e.g., normal "wear and tear"), or "acts of God" (e.g., natural disasters).
- **Security engineering** deals with attacks that are mounted by attackers (e.g., people, organizations, software tools).
- **Survivability engineering** deals with harm due to both accidents and attacks.

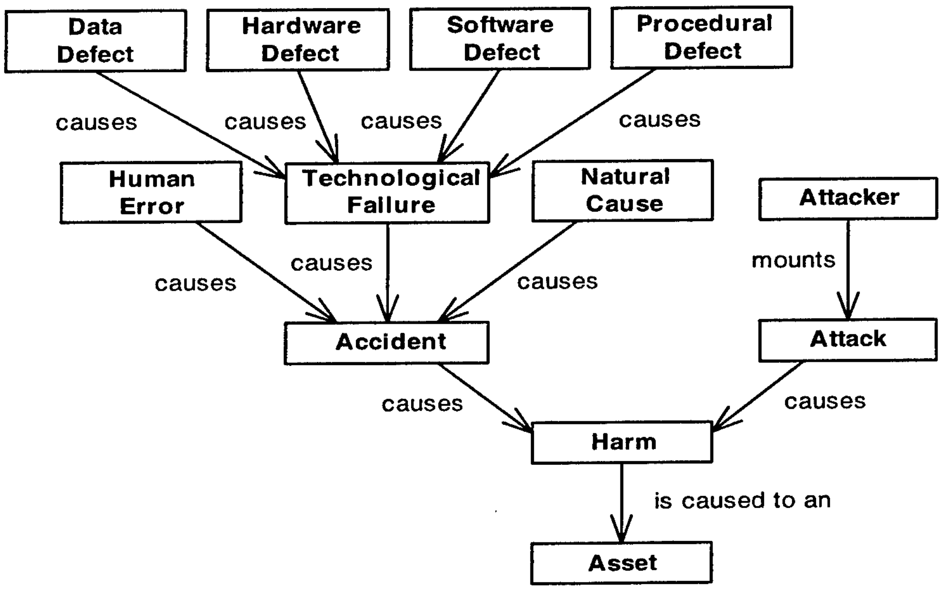


Figure 12: Accidents vs. Attacks

5.2.4 Hazards vs. Threats

Because of the separation of scope of safety and security into accidental and malicious harm, safety and security deal with different types of dangers, and survivability deals with both. Figure 13 illustrates the following relationships:

- **Safety requirements** protect assets from harm due to hazards.
- **Security requirements** protect assets from harm due to threats.
- **Survivability requirements** protect assets (essential services) from harm due to both hazards and threats.

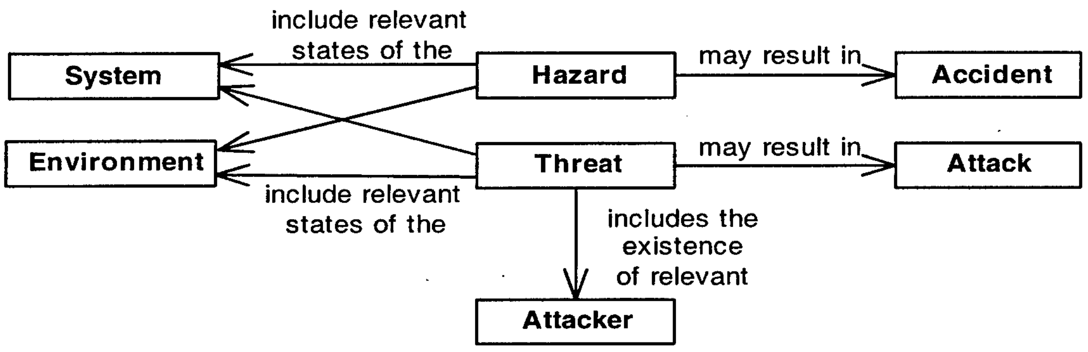

include relevant states of the
System
Hazard
may result in
Accident
Environment
Threat
may result in
Attack
include relevant states of the
includes the existence of relevant
Attacker

Figure 13: Hazards vs. Threats

6 Recommendations and Future Work

6.1 Specific Recommendations

The following recommendations are based on the preceding observations concerning similarities and differences among the fundamental concepts underlying safety, security, and survivability engineering.

6.1.1 Use Common Concepts and Terminology

Where appropriate and practical, use common concepts and terminology when performing or describing safety, security, and survivability engineering [Mead 03]. For example, this common terminology should include such common concepts as asset, harm, accident, attack, hazard, threat, risk, vulnerability, requirement, policy, and goal. Add new concepts to clarify the relationships between highly similar terms (such as accident, attack or hazard, and threat). Clearly define these concepts so that their similarities and differences are obvious. Then, codify these definitions in standards, books, and training materials.

6.1.2 Add Defensibility as a New Quality Factor

As illustrated in Figure 14, create a new abstract quality factor called defensibility that is a subclass of dependability and which, in turn, is subclassed into the three quality factors of safety, security, and survivability. Defensibility can act as a focal point for common concepts and related processes (e.g., asset-based risk analysis) that can be inherited by its three subclasses.

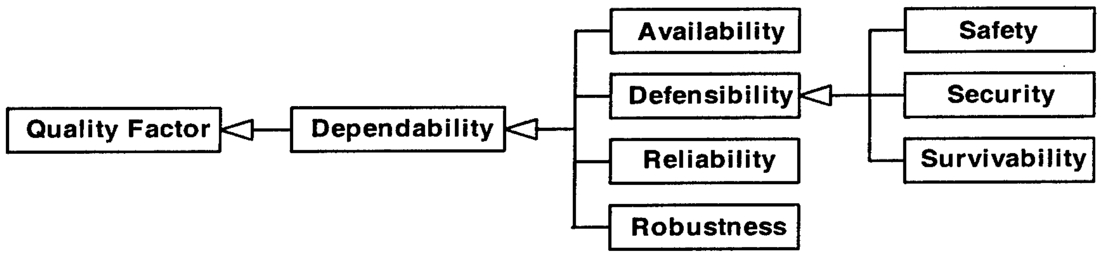


Figure 14: Defensibility as a Kind of Dependability

There is a trend to combine safety and security engineering into integrity assurance [ISO 96, Jarzombek 03]. However, integrity has a very specific meaning within security engineering (i.e., the integrity subfactor of security), so the term "integrity" is too specific and the more general term "defensibility" is a more appropriate term for the combination of safety, security, and survivability.

6.1.3 Decompose Defensibility

Decompose defensibility (and therefore safety, security, and survivability) into a standard set of quality subfactors: asset protection, incident detection, incident reaction, and system adaptation. These quality subfactors can then be decomposed into a standard set of lower level quality subfactors that are essentially common to safety, security, and survivability.

By explicitly decomposing safety, security, and survivability into asset protection, incident detection, incident reaction, and system adaptation as illustrated in Figure 15, we obtain the following benefits:

- This decomposition provides welcome standardization across the three kinds of defensibility.
- In light of the historic emphasis on asset protection, this decomposition helps to ensure that the other three subfactors are not forgotten and that a complete set of defensibility requirements are produced.
- This decomposition enables the different kinds of defensibility to have their own additional factor-specific subfactors (e.g., availability protection, integrity, privacy, etc., as subfactors of asset protection).
- This decomposition can be extended with new subfactors as they are identified.

Figure 15 illustrates this standard decomposition.

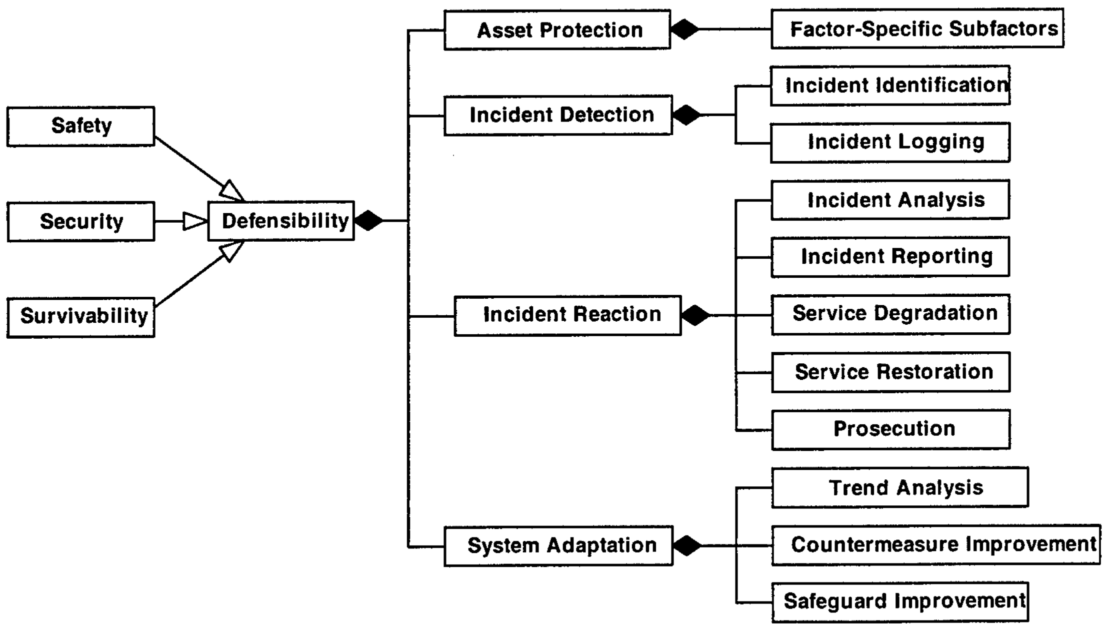


Figure 15: Standard Decomposition of Defensibility into Quality Subfactors

The subfactors illustrated in Figure 15 can be defined as follows:

- **Asset protection** (also known as prevention and resistance) is the degree to which assets are protected from accidents and attacks.
  Asset protection includes both the elimination of hazards and threats as well as steps to minimize the negative outcome should an accident or successful attack occur. Asset protection can be decomposed into various quality-factor-specific subfactors such as access control (identification, authentication, and authorization), availability protection, integrity, nonrepudiation, physical protection, and privacy.
- **Incident detection** (also known as recognition) is the degree to which relevant accidents and attacks (or the harm they cause) are recognized as they occur so that the system can react accordingly (e.g., to maintain essential services, to degrade gracefully).
  Incident detection typically involves incident identification and logging. For example, the recording of attacks can provide legal evidence with which to prosecute attackers. It may also include the recognition of conditions or events preceding accidents (e.g., hardware nearing failure) or the detection of attackers collecting information during probes prior to attacks.
- **Incident reaction** (also known as recovery) is the degree to which the system responds (e.g., recovers) after an accident or attack.
  Incident reaction can involve incident analysis and reporting, service degradation and restoration, as well as the prosecution of people causing accidents and attackers mounting attacks. Service restoration typically involves the establishment of priority-based recovery approaches so that any essential services that may have been lost or degraded are recovered before the recovery of any non-essential services that were lost or terminated.
- **System adaptation** is the degree to which the system adapts itself (based on current accidents and attacks) so that in the future it may better protect its assets, detect incidents, and react to them. System adaptation my involve trend analysis for incidents as well as the improvement of safeguards and countermeasures.

6.1.4 Include All Types of Valuable Assets

As illustrated in Figure 16, ensure that all relevant assets are considered when developing defensibility (e.g., safety, security, and survivability) requirements. Consider each kind of asset, not just the most popular type (e.g., people for safety and data for security). These concepts now have the following definitions:

- **Asset** is anything of value that should be protected from harm.
- **Harm** is significant damage to or negative impact (i.e., negative outcome) associated with a valuable asset that is due to an incident. Harm can be decomposed according to the type of asset harmed (e.g., harm to people includes such things as injury, illness, death, or victimization by a cybercrime) or the type of incident (e.g., harm due to attack may include exposure of sensitive information). Harm must be sufficiently significant to warrant relatively prompt remedial action to prevent such harm in the future.

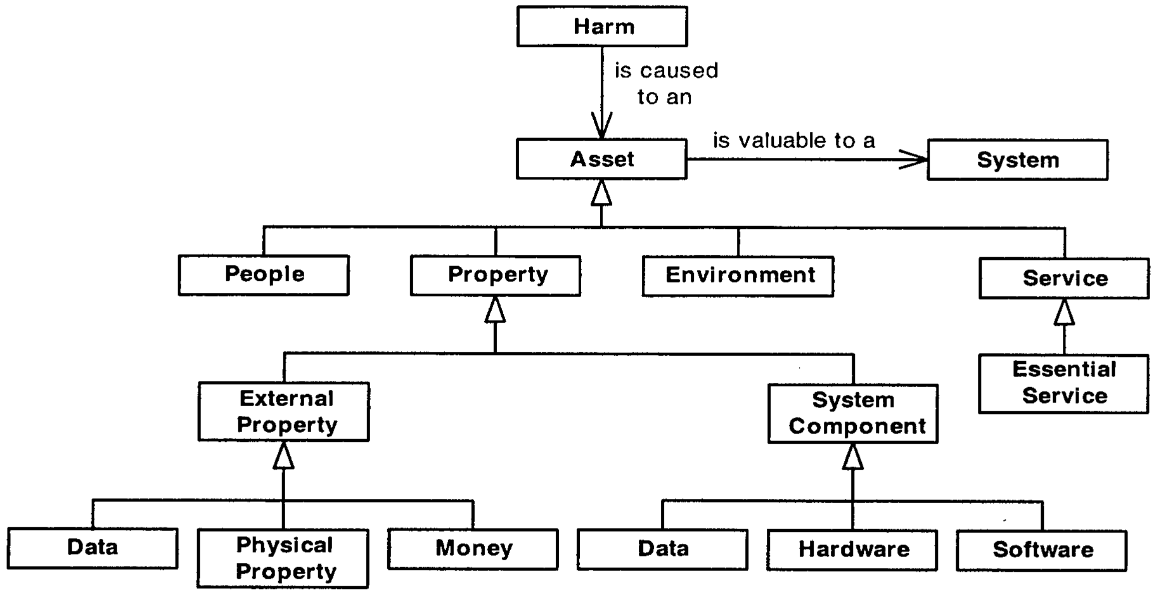


Figure 16: Assets and Harm

6.1.5 Incidents

Accidents and attacks are obviously related in that they both cause harm to valuable assets, which must be protected from them. As illustrated in Figure 17, create a new concept, "incident," to act as an abstract superclass for both accident and attack. The resulting concepts in this inheritance hierarchy now have the following definitions:

- **Incident** is any event or cohesive collection of related events that *may* cause harm to a valuable asset.
  - **Safety incident** is any incident that is unplanned and unintended, although not necessarily unexpected (i.e., accidental).
    - **Accident** (also known as mishap) is any safety incident that causes harm.
    - **Near miss** is any safety incident that does not cause harm.
  - **Attack** (also known as security breach) is any incident that is both intentional and unauthorized (i.e., malicious). Thus, an attack is a malicious attempt mounted by an attacker to violate the security of the system, bypass the system's security mechanisms, and cause harm to an asset.
    - **Successful attack** is any attack that causes harm.
    - **Unsuccessful attack** is any attack that does not cause harm.

As illustrated in Figure 17, safety and security have similar generalization subtrees with "safety incident" corresponding to "attack," "accident" corresponding to "successful attack," and "near miss" corresponding to "unsuccessful attack." Note also from the above definitions that all events associated with an incident are not necessarily part of that incident. For example, typically a series of events occurs that move a system into a hazardous state (more correctly a set of incompatible states) prior to the occurrence of an accident. Similarly, a series of exploratory or probing events often occur prior to and leading up to an actual attack [Allen 01].

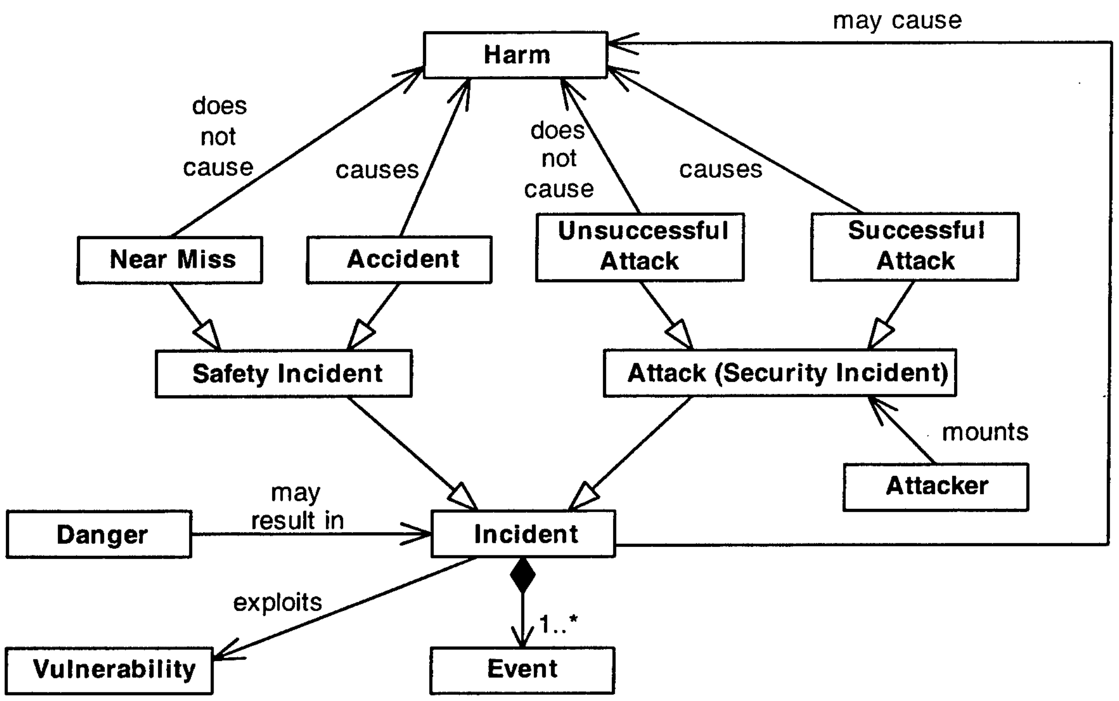


Figure 17: Incidents (Accidents and Attacks)

6.1.6 Dangers

As with accidents and attacks, hazards and threats are also obviously related in that both involve incidents that cause harm to valuable assets. Create a new concept, danger, to act as an abstract superclass for both hazard and threat. As illustrated in Figure 18, these concepts now have the following definitions:

- **Danger** (also known as obstacle) is a situation (i.e., a set of one or more incompatible conditions or states of the system, possibly including one or more conditions in the system's environment) that increases the likelihood of one or more related incidents. As such, dangers are ways of organizing related categories of incidents.
  - — **Hazard** is a danger that may result in one or more related accidents.
  - — **Threat** is a danger that may result in one or more related attacks.

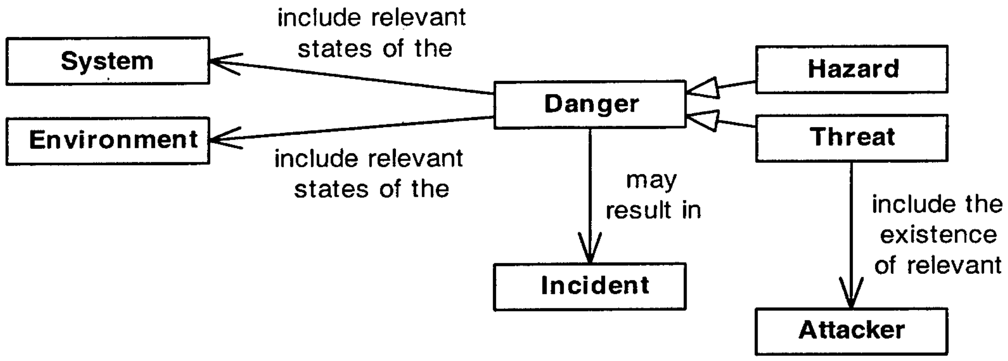

include relevant states of the
System
Environment
include relevant states of the
Danger
Hazard
Threat
may result in
Incident
include the existence of relevant
Attacker

Figure 18: Dangers (Hazards and Threats)

6.1.7 Exploit Commonality in the Information Models

As illustrated in Figure 19, commonality existing in the preceding diagrams can be combined to form a single information model for defensibility. Save effort by taking advantage of the commonality between the models underlying safety, security, and survivability when it comes to identifying and analyzing the

- assets to protect from harm
- harm that can come to these assets
- incidents (e.g., accidents and attacks) that can cause harm
- dangers (e.g., hazards and threats)
- risks to the assets
- vulnerabilities of the assets
- mechanisms that will protect the assets and fulfill the requirements

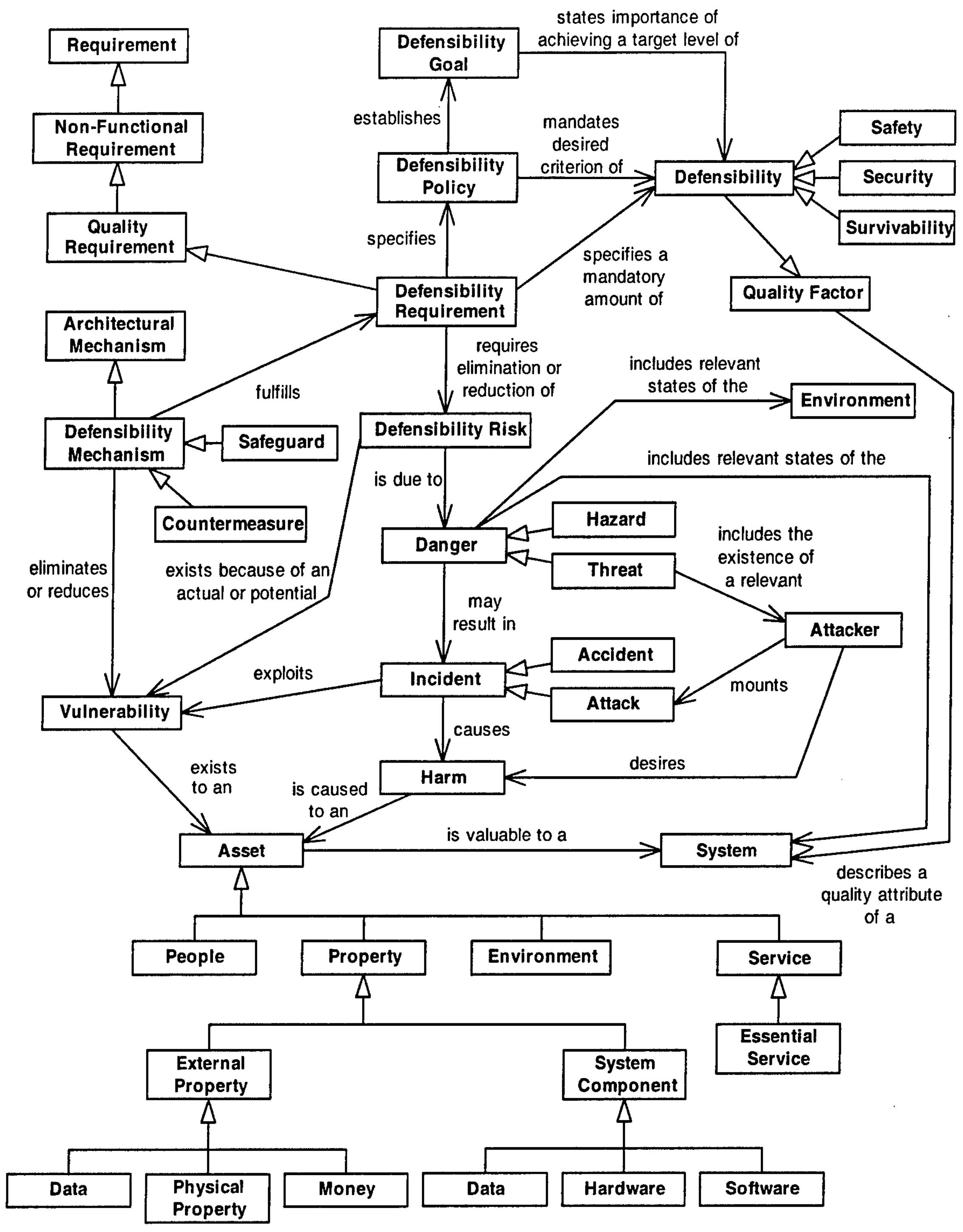


Figure 19: Information Model for Defensibility Engineering

6.1.8 Develop a Common Process

The similarities of the information models in Figures 9, 10, and 11 provide justification for the recommendation in many books and articles that safety and security requirements be engineered based on the associated risks of dangers to valuable assets [Alberts 03, Firesmith 03c, Herrmann 99, Moffett 03, NASA 97, Peltier 01, UK 96]. If the requirements are too strong for the risks, then excess money and time are wasted on architectural mechanisms that are more powerful than needed. If the requirements are too weak for the risks, then dangers will not be adequately prevented, detected, and/or reacted to. Also, because all three types of defensibility involve largely the same assets and their related dangers and risks, it makes sense to engineer these quality factors simultaneously as a group to avoid wasting redundant effort. Therefore, use the commonality of these information models as a foundation on which to build a single, common, asset-based, risk-driven danger (hazard/threat) analysis approach for safety, security, and survivability engineering. For example, the following steps form one such process when performed iteratively, incrementally, and in parallel with the other activities and tasks:

1. Integrate defensibility engineering with the rest of the engineering process.
2. Develop a defensibility program plan that includes safety, security, and survivability.
3. Identify and prioritize the valuable assets that are in danger and thus may be harmed.
4. Set the defensibility goals and policies to protect these assets.
5. Determine the negative impacts that could occur to these valuable assets if the dangers were to cause incidents (accidents and attacks).
6. Identify and profile potential attackers as well as system-external causes of accidents (fires, floods, etc.).
7. Identify, categorize, and prioritize the dangers (threats and hazards) that may harm these valuable assets. Identify and analyze their potential causes.
8. Estimate the associated risks to these valuable assets and prioritize them based on the extent of the negative impact that can occur and the likelihood of the danger's occurrence.
9. In priority risk order, select the relevant quality subfactors of the protection, detection, and reaction quality subfactors of the safety, security, and/or survivability quality factors. For example, select authentication or data integrity when dealing with security.
10. Determine one or more system-specific quality criteria to determine the existence of the associated quality subfactor.
11. Select the associated quality metric for each criterion and determine a minimum required level of that quality metric.

12. Identify, analyze, and specify defensibility requirements as combinations of the quality criterion with a minimum level of the associated quality metric. Perform tradeoffs between these requirements and other potentially conflicting requirements.
13. Architect mechanisms (safeguards and countermeasures) to fulfill these requirements.
14. Design and implement these architectural mechanisms.
15. Identify, analyze, and fix any remaining vulnerabilities.
16. Perform verification (e.g., security testing).
17. Obtain certification and/or accreditation.
18. Store evidence of defensibility actions.
19. Analyze and record incidents.

6.1.9 Collocate Requirements

Store the resulting defensibility requirements in the appropriate part of the quality requirements section of the system requirements specification. Do not store them in three separate documents that may or may not be used as input to the system architecture and design. For example, do not rely on the security policy document to store the security requirements; requirements are not policy, and collocating the security requirements with the security policies decreases the likelihood that they will drive the architecture and be tested for consistency with the other requirements.

6.1.10 Engineer Defensibility Requirements and Architecture Early

Do not wait until the rest of the architecture exists and components have been determined before engineering the safety, security, and survivability requirements and associated architectural mechanisms. By then, the other requirements will largely be completed, and the resulting defensibility requirements will not have driven the architecture and will not have been checked for consistency with the other requirements. It is difficult, costly, and time consuming to try to add safety, security, and survivability to an existing architecture.

6.2 Future Work

Clarifying terminology is important but by itself is insufficient. Clarifying the terminology regarding safety, security, and survivability is primarily important because it provides a solid foundation on which to engineer safety, security, and survivability. It also provides a solid foundation on which to perform research concerning these three engineering disciplines, especially if one is to try to take advantage of their commonality.

The contents of this technical note will form the foundation of an SEI internal research and development (IR&D) effort to identify ways of reusing safety, security, and survivability requirements as well as recommended processes for eliciting and analyzing such requirements.

Other future work could include the following:

- Better identify quality subfactors for safety and survivability to the same level of decomposition as has been done for security.
- Determine the best way that these quality subfactors can be allocated to prevention, detection, and reaction, given that some of the quality factors seem to fit under more than one of these three categories.
- Perform further research to determine the best way of dealing with availability (normal versus accidental loss versus malicious loss via DoS attacks) to minimize overlap between quality factors.
- Determine the overlap of defensibility architectural mechanisms.

7 Conclusion

As can be seen from the foundational models documented in this technical note, safety, security, and survivability are very closely related. Safety deals with *accidental* harm due to accidents that are typically caused by human errors from carelessness, hardware failures, or "acts of God," whereas security deals with *malicious* harm due to attacks by human or automated attackers. Survivability deals with both accidental and malicious harm if the harm is sufficient to adversely affect essential services. All three quality factors are based on the protection of valuable assets from harm, although each has historically emphasized different kinds of assets. Safety thus involves the prevention or reduction of hazards, security involves the prevention or reduction of threats, and survivability involves the prevention or reduction of both hazards and threats. All three involve the analysis and management of risks based on potential negative outcomes and the probability of their occurrence. All three also involve requirements to prevent, reduce, or mandate the proper response or adaptation of the system to the occurrence of these risks. These requirements are typically fulfilled by appropriate architectural mechanisms (e.g., safeguards, security practices and countermeasures, survivability mechanisms) that address potential and actual vulnerabilities.

Safety, security, and survivability requirements can be defined as quality requirements for the quality factors of safety, security, and survivability, respectively. As such, they can be organized into associated quality subfactors. They can also be decomposed into two parts: a quality criterion (a specific description that provides evidence either for or against the existence of a specific quality factor or subfactor) and a quality metric (a minimum level based on an associated scale of measurement). In all three cases, the quality criterion typically involves the asset being protected and the danger (e.g., hazard or threat) from which it is being protected.

This technical note has provided a relatively formal foundation for safety, security, and survivability engineering by defining an information model for

- a quality model and its component parts
- requirements with emphasis on quality requirements for safety, security, and survivability
- safety, security, and survivability

This technical note has identified the essential foundational concepts underlying safety, security, and survivability engineering, provided rigorous definitions for them, illustrated the important relationships between them, and provided a series of recommendations based on the commonality between safety, security, and survivability. In so doing, I hope that this technical note will clarify and standardize the terminology associated with safety, security, and survivability and will thereby simplify and improve the engineering of complex systems by improving the consideration of their safety, security, and survivability.

References

URLs are valid as of December 2003.

[Alberts 03]	Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The Octave™ Approach.* Boston, MA: Addison Wesley, 2003.
[Allen 01]	Allen, Julia H. *The CERT Guide to System and Network Security Practices.* Boston, M.A.: Addison Wesley, 2001.
[Anderson 01]	Anderson, Ross. *Security Engineering: A Guide to Building Dependable Distributed Systems.* New York, NY: Wiley Computer Publishing, 2001.
[Barbacci 00]	Barbacci, Mario R.; Ellison, Robert J.; Weinstock, Charles B.; & Wood, William G. *Quality Attribute Workshop Participant's Handbook* (CMU/SEI-2000-SR-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports /00sr001.html>.
[Boehm 76]	Boehm, Barry; Brown, J. R.: & Lipow, M. "Quantitative Evaluation of Software Quality," 592-605. *Proceedings of the 2nd International Conference on Software Engineering*, San Francisco, CA, Oct. 13-15, 1976. New York, NY: IEEE Computer Society, 1976.
[Chung 93]	Chung, Kyungwha Lawrence. *Representing and Using Non-Functional Requirements: A Process-Oriented Approach* (DKBS-TR-93-1, Ph D dissertation). Toronto, Canada: Department of Computer Science, University of Toronto, 1993.
[Chung 00]	Chung, Kyungwha Lawrence; Nixon, Brian A.; Yu, Eric; & Mylopoulos, John, *Non-Functional Requirements in Software Engineering.* Norwell, MA: Kluwer Academic Publishers, 2000.

[CNSS 03]	Committee on National Security Systems (CNSS). *National Information Assurance (IA) Glossary* (CNSS Instruction No. 4009). Fort Meade, Maryland: Committee on National Security Systems (CNSS), National Security Agency (NSA), May 2003.
[Davis 93]	Davis, Alan M. *Software Requirements: Objects, Functions, and States*, Englewood Cliffs, NJ: Prentice Hall, 1993.
[Ellison 99]	Ellison, R. J.; Fisher, D. A.; Linger, R. C.; Lipson, H. F.; Longstaff, T. A.; & Mead, N. R. "Survivable Systems: An Emerging Discipline," 93-116. *Proceedings of the 11th Canadian Information Technology Security Symposium (CITSS).* Ottawa, Ontario, Canada, May 10-14, 1999. Ottawa, Ontario, Canada: Communications Security Establishment, 1999.
[Ellison 03]	Ellison, Robert J. & Moore, Andrew P. *Trustworthy Refinement Through Intrusion-Aware Design (TRIAD)* (CMU/SEI-2003-TR-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, March 2003. <http://www.sei.cmu.edu/publications/documents/03.reports /03tr002.html>.
[ESA 97]	European Space Agency (ESA). *Glossary of Terms, Rev. 1* (ECSS-P-001A). The Netherlands: European Space Agency, 1997.
[Firesmith 02]	Firesmith, Donald & Henderson-Sellers, Brian. *The OPEN Process Framework.* London, England: Addison-Wesley, 2002.
[Firesmith 03a]	Firesmith, Donald G. "Using Quality Models to Engineer Quality Requirements." *Journal of Object Technology (JOT)* 2, 5 (September/October 2003): 67-75. Zurich, Switzerland: Swiss Federal Institute of Technology (ETH). <http://www.jot.fm/issues/issue_2003_09/column6>.
[Firesmith 03b]	Firesmith, Donald G. "Analyzing and Specifying Reusable Security Requirements," 7-11. *Requirements Engineering 2003 Requirements for High Assurance Systems (RHAS) Workshop Proceedings*, Monterey, CA, Sept. 9, 2003. Washington, D.C.: IEEE Computer Society, 2003.

[Firesmith 03c]	Firesmith, Donald G. "Specifying Reusable Security Requirements." *Journal of Object Technology (JOT) 3*, 1 (January/February 2004): 61-75. <http://www.jot.fm/issues/issue_2004_01/column6>.
[Firesmith 03d]	Firesmith, Donald. *Firesmith's OPEN Process Framework Website* <http://www.donald-firesmith.com> (2003).
[Herrmann 99]	Herrmann, Debra. *Software Safety and Reliability*, Los Alamitos, CA: IEEE Computer Society, 1999.
[Hughes 95]	Hughes, Larry. *Actually Useful Internet Security Techniques*, Indianapolis, Indiana: New Riders, 1995.
[IEEE 94]	Institute of Electrical and Electronics Engineers (IEEE). *IEEE Standard for Software Safety Plans*, IEEE-Std-1228. New York, NY: IEEE, 1994.
[ISO 96]	International Standards Organization (ISO). *System and Software Integrity Levels*, ISO/IEC 15026, Quebec, Canada: ISO, 1996.
[ISO 00]	International Standards Organization (ISO). *Software Engineering – Product Quality – Part 1: Quality Model*, ISO/IEC 9126-1, Quebec, Canada: ISO, 2000.
[Jarzombek 03]	Jarzombek, Joe. "A Framework for Applying Safety and Security Practices in Acquisition, Development, and Sustainment: Building upon Standards and Models to Guide Process Improvement and Appraisal," Software Safety Summit, Naval Ordinance Safety and Security Activity, 7 April 2003. <http://www.ih.navy.mil/summit/Presentations/FrameworkSafety_Security.ppt>.
[Keller 90]	Keller, Steven E.; Kahn, Laurence G.; & Panara, Roger B. "Specifying Software Quality Requirements with Metrics," 145-163. *Tutorial: System and Software Requirements Engineering*, Los Alamitos, CA: IEEE Computer Society Press, 1990.

[Knight 00] Knight, John C. & Sullivan, Kevin J. *On the Definition of Survivability* (Technical Report CS-TR-33-00). Charlottesville, VA: University of Virginia, Department of Computer Science, 2000.

[Knight 03] Knight, John C.; Strunk, Elisabeth A.; & Sullivan, Kevin J. "Towards a Rigorous Definition of Information System Survivability," 78. *Proceedings of DISCEX 2003.* Washington, D.C., April 2003.

[Lamsweerde 00] Lamsweerde, Axel van & Letier, Emmanuel. "Handling Obstacles in Goal-Oriented Requirements Engineering." *IEEE Transactions on Software Engineering 26,* 10 (October 2000): 978-1005.

[Leveson 95] Leveson, Nancy. *Safeware: System Safety and Computers.* Reading, MA: Addison-Wesley, 1995.

[Lipson 99] Lipson, Howard F. & Fisher, David A. "Survivability – A New Technical and Business Perspective on Security," 33-39. *Proceedings of the 1999 New Security Paradigms Workshop.* September 22-24, 1999, Caledon Hills, Ontario, Canada. New York, NY: ACM, 1999.

[Loucopoulos 95] Loucopoulos, P. & Karakostas, V. *System Requirements Engineering.* New York, NY: McGraw Hill, 1995.

[McDermid 91] McDermid, John. "Issues in Developing Software for Safety Critical Systems." *Reliability Engineering and Systems Safety 32,* 1-2 (1991): 1-24.

[McNamara 03] McNamara, Joel. *Secrets of Computer Espionage: Tactics and Countermeasures.* Indianapolis, IN: Wiley, 2003.

[Mead 03] Mead, Nancy R. *Requirements Engineering for Survivable Systems* (CMU/SEI-2003-TN-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tn013.html>.

[Moffett 03]	Moffet, Jonathan D. & Nuseibeh , Bashar A. *A Framework for Security Requirements Engineering* (Report YCS 368). United Kingdom: Department of Computer Science, University of York, 20 August 2003.
[Mylopoulos 92]	Mylopoulos, John; Chung, Lawrence; & Nixon, Brian. "Representing and Using Non-Functional Requirements: A Process-Oriented Approach." *IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Development* 18, 6 (June 1992): 483-497.
[NASA 97]	National Aeronautics and Space Administration (NASA). *Software Safety NASA Technical Standard* (NASA-STD-8719.13A). Washington, D.C.: NASA, 1997.
[Peltier 01]	Peltier, Thomas R. *Information Security Risk Analysis*. Boca Raton, FL: CRC Press, 2001.
[Perrow 84]	Perrow, Charles. *Normal Accidents: Living with High-Risk Technologies*. New York, NY: Basic Books, 1984.
[Power 00]	Power, Richard. *Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace*. Indianapolis, IN: QUE, 2000.
[Roman 85]	Roman, Gruia-Catalin. "A Taxonomy of Current Issues in Requirements Engineering." *IEEE Computer 18*, 4 (April 1985): 14-23.
[Schneier 00]	Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York, NY: Wiley, 2000.
[Shema 03]	Shema, Mike. *Hack Notes: Web Security Portable Reference*. Emeryville, CA: McGraw Hill, 2003.
[Sommerville 92]	Sommerville, Ian. *Software Engineering*, fourth edition. Reading, Ma: Addison-Wesley, 1992.
[Thayer 90]	Thayer, R. & Dorfman, M., eds. *System and Software Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1990.
[Tulloch 03]	Tulloch, Mitch. *Microsoft Encyclopedia of Security*. Redmond, WA: Microsoft, 2003.

[UK 96] Great Britain, Ministry of Defence. *Safety Management Requirements for Defence Systems* (UK MoD Def Stan 00-56). Glasgow, UK: United Kingdom Ministry of Defence, 1996.

[van der Meulen 00] van der Meulen, Meine. *Definitions for Hardware and Software Safety Engineers*. London, England: Springer, 2000.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE December 2003	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Common Concepts Underlying Safety, Security, and Survivability Engineering		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Donald G. Firesmith		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2003-TN-033
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note presents a consistent set of information models that identify and define the foundational concepts underlying safety, security, and survivability engineering. In addition, it shows how quality requirements are related to quality factors, subfactors, criteria, and metrics, and it emphasizes the similarities between the concepts that underlie safety, security, and survivability engineering. The information models presented in this technical note provide a standard terminology and set of concepts that explain the similarities between the asset-based, risk-driven methods for identifying and analyzing safety, security, and survivability requirements as well as a rationale for the similarity in architectural mechanisms that are commonly used to fulfill these requirements.

14. SUBJECT TERMS safety, safety engineering, safety requirements, security, security engineering, security requirements, survivability, survivability engineering, survivability requirements, requirements engineering, quality requirements, quality model, quality criteria	15. NUMBER OF PAGES 70
16. PRICE CODE	

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL