# Chapter 2

# Trustworthiness Assessment Framework for Net-Centric Systems

**Raymond Paul**, Jing Dong*, I-Ling Yen*, and Farokh Bastani*,**

*University of Texas at Dallas, Richardson, Texas, USA

**Department of Defense, USA

**Abstract** Modern applications are becoming increasingly large-scale and network-centric, involving a variety of different types of system entities. Also, the assurance requirements for these systems are evolving due to the continuing emergence of new threats from new operational environments. To assure the trustworthiness of these systems to a sufficiently high degree of confidence is a challenging task. Most existing methods require different specialized assessment techniques for not only different types of system entities but also different trustworthiness aspects. Also, most existing techniques lack consideration of the overall system trustworthiness assessment from an integrated system perspective or fail to provide a holistic view. To address these problems, we develop an ontology-based approach to provide systematic guidelines for net-centric system assessment. The ontology-based approach captures evolving system trustworthiness aspects and effectively models their relationships and correlations. It can also organize system entities and associate appropriate assessment techniques for each class of system entities and their integrations.

## 1. Introduction

Due to the advances of computer and networking technologies, many applications are becoming large-scale and network-centric. A net-centric system (NCS) typically involves a distributed set of sensors, actuators, processors, software, along with a variety of other resources interconnected together by a network and interacting with and controlled by end users. Operational scenarios range from telecontrol and tele-monitoring systems to distributed coordination and communication systems, command and control systems, emergency response, and other areas.

J. Dong et al. (eds.), *High Assurance Services Computing*,
DOI 10.1007/978-0-387-87658-0_2, 

All these domains are mission- and/or safety-critical since these systems interact with the physical world and failures could potentially have catastrophic consequences. Hence, it is imperative to be able to build ultra dependable and trustworthy NCS and to be able to certify the trustworthiness of these systems to a high degree of confidence before deploying them in the field.

Many techniques have been developed to achieve high assurance and trustworthiness. But almost all of these techniques focus on one or a few trustworthiness aspects. When designing a high assurance system, it does not have to be the invention of new techniques for every part of the system. Rather, it is mostly a decision process to determine which technique to use to achieve certain desired properties in a subsystem. There are many existing techniques that can be considered and adopted. However, how to know which technique is the best to use for a part of the system. The general solution is to use analysis techniques to determine whether a certain combination of techniques does result in a system that satisfies high assurance requirements. Thus, assessment techniques play an important role for the design as well as the assessment phases of mission- and safety- critical systems.

There are significant challenges in trustworthiness assessment for NCS [19]. In general, it is very expensive to assess the trustworthiness of software systems to a high degree of confidence. Considering just the reliability aspect, it has been shown that it would take hundreds of years of testing to achieve adequate confidence in the reliability of a safety-critical system. Net-centric systems face numerous other challenges, including security, usability, and performance issues that require even more time and effort for high-confidence assessment. Compounding these challenges is the fact that these systems are mission-specific and likely to be dynamically composed from existing COTS (commercial off the shelf) and GOTS (government off the shelf) hardware and software components and services [17,18]. Due to the potential lack of complete information regarding the development history of COTS components and their exact implementation details, they can pose severe but difficult-to-detect security and reliability threats, including the potential for embedded “Trojan horses” and other malicious logic designed to trigger rare failures during critical periods. These make it difficult to achieve high confidence in the assurance levels of COTS hardware and software components. In addition, while compositional assessment methods are widely used for certifying hardware systems by, for example, calculating the reliability of a complex system from the reliability of its constituent components, this type of assessment technique has in general proven to be difficult to achieve for software. The reason is that hardware assessment typically focuses on problems due to wear-and-tear and other degradations that impact components independently of each other. This is not the case for software where reliability problems are predominantly due to specification, design, development, and implementation faults. The reliability of the system depends on the way one component uses another component. Because of this, it is possible to build a system where some components are faulty but yet the system is highly reliable since those faults are not triggered due to the way the

components are used in the system. Likewise, it is possible to build a system using components that are individually highly reliable but that collectively lead to poor reliability due to unexpected interactions between the components [3].

Besides the complexity in trustworthiness assessment techniques, assessment time and cost are also major concerns. Mission-specific systems must typically be built and deployed rapidly since the mission requirements may change dynamically. Thus, there is a need to be able to rapidly and dynamically certify the trustworthiness of the system systems to a high degree of confidence. This is difficult to do using solely testing or verification methods.

In this chapter, we consider the challenges of assessing highly critical NCS systems and develop technical solutions to address the numerous and interdependent issues involved. We use ontology to capture the evolving trustworthiness metrics and increasing varieties of NCS system entities and their correlations. Based on the ontology, we develop systematic steps to guide trustworthiness assessment. Various assessment techniques are associated with the ontology nodes to facilitate systematic or even semi-automated assessment data collection, integration, and analysis.

A large number of techniques have been developed over the past years for trustworthiness and dependability assessment and most of these methods can be associated with the ontology to assist with NCS assessment. However, there are still many missing links in such techniques. For example, security assessment domain is still in its infancy. A major area in trustworthiness assessment that is missing is that of holistic evaluation. Almost all of the assessment techniques focus on a single aspect, such as software reliability, data security, system performance, etc.; however, this is not always adequate as can be seen by considering some scenarios. For example, when a commander in a battlefield needs to compose a plan to accomplish one or multiple missions, it is desirable to know whether the plan is good enough in terms of accomplishing the missions. It would also be interesting if there are multiple plans and the goal is to assess them to determine which plan is the best for the given missions. In this case, it is desirable to offer a single score for each plan, i.e., the probability that a given plan can successfully accomplish the specified missions. This requires the integration of the evaluations of various trustworthiness aspects of the system and provides a holistic measurement. Thus, in this paper, we also develop techniques for integration in an attempt to provide holistic assessments.

The rest of the chapter is organized as follows. In Section 2, the ontology for trustworthiness assessment, including the system entities dimension and the trustworthiness aspects dimension, is presented. Section 3 introduces an integrated assessment framework that provides systematic assessment procedures based on the trustworthiness assessment ontology. A holistic assessment technique is introduced in Section 4. Section 5 concludes the chapter and identifies some future research directions.

## 2. Ontology for Trustworthiness Assessment

To deal with trustworthiness assessment of high assurance NCS, we need to deal with two dimensions of complexity. First, the NCS is highly complex, consisting of systems of systems. Each subsystem and the constituent system entities and components can be of very different characteristics. Techniques for assurance and assessment of different system entities and components can be very different. For example, methods for hardware and software reliability assurance and assessment are significantly different. Similarly, assurance and assessment of security for data and for software components involve the use of different techniques. Also, the techniques for compositional assessment of different types of components may also be different. To facilitate the management of techniques for dealing with different entities and their integrations, we construct an evolving ontology of system entities and integrations and associate various high-confidence assurance and assessment techniques with the ontology nodes.

Another complex dimension in high assurance and trustworthiness assessment is the set of metrics to be used. The requirements for achieving "high assurance, ultra dependability, and trustworthiness" for critical applications have been evolving along with the continuing advances in computer and communication environments. In the early era, hardware and software reliability, system availability, and real-time concerns were the major focus in high-assurance systems engineering. In [3], the definition of dependability is clearly elaborated. However, with the growth of computing environments, some new requirements for high-assurance systems have emerged. For example, with advances in data and knowledge mining, the concept of "privacy-preserving" capabilities has been introduced and is an increasingly essential property for high-assurance information systems. Also, Internet applications are moving toward open environments and "trust" is increasingly becoming another measure that is important in dependable computing. To cope with this problem, we develop an ontology to capture the evolving requirements in high-assurance systems. Ontology facilitates easy evolution. To differentiate from conventional dependability definitions, we use *trustworthiness* to include dependability as well as other high-assurance attributes.

Most of the trustworthiness aspects are directly or indirectly related to each other to some extent. Frequently, techniques that improve one aspect may impact some other aspects. Thus, when building the ontology of trustworthiness aspects, it is necessary to express the interdependencies and correlations among the aspects. However, existing works on categorizing dependability/trustworthiness aspects do not consider such correlations. Consider an example of the correlations between trustworthiness aspects. Redundancy is always required for achieving reliability, availability, and survivability. A higher degree of redundancy implies a higher level of reliability, availability, and survivability. On the other hand, a higher degree of redundancy can lead to more points in the system that may be vulnerable to security attacks and a higher probability that one weak point be-

comes compromised and, hence, results in a system having weaker security. However, this only indicates the correlations among reliability, availability, survivability, and security, but they are not directly dependent upon each other. Instead, all these aspects are dependent on redundancy. To provide a clear view of the correlations of the trustworthiness aspects in the ontology, we further define an ontology of trustworthiness evidences. The trustworthiness evidences are quantitatively or qualitatively measurable properties of the system or system entities and they are orthogonal to each other. For example, the level of redundancy and software logical correctness can be trustworthiness evidences of the system. The trustworthiness evidence ontology facilitates the expression of the correlations of trustworthiness aspects and can help optimally balance various conflicting trustworthiness aspects in the design of high-assurance systems.

Overall, we consider an integrated ontology that spans the dimension of system entities and integrations and the dimension of trustworthiness aspects with a sub-ontology of trustworthiness evidences. The two dimensions can evolve independently and can be used together to provide a fine-grained guidance for trustworthy assessment. Trustworthiness assessment and assurance techniques can be associated with the corresponding nodes in the ontology. Current assessment techniques focus on individual types of components, such as reliability assessment for software versus reliability assessment for hardware, software aging models versus hardware degradation models, assessment of the efficacy of hardware redundancy methods versus those for software redundancy, etc. The merged ontology with associated assessment techniques can provide an organized view to link existing techniques together. It can also reveal the missing links in assessment techniques. Based on the ontology, a systematic and well guided trustworthiness assessment and verification process can be developed for large-scale NCS.

In the following subsections, the two dimensions of the ontology are discussed in detail.

## *2.1 Ontology of the Trustworthiness Aspects Dimension*

### 2.1.1 Trustworthiness Aspects

A variety of trustworthiness aspects have been proposed in the literature for high-assurance systems. The fundamental requirement of any high assurance system should include reliability and availability [3].

- **Reliability:** The reliability of a system for a time interval (0,t) is the probability that the system continuously operates correctly for the entire time interval given that it is available at the start of the time interval [4, 14].

- **Availability:** The availability of a system is the probability that the system is ready for correct service when needed.

The increasing use of computing systems in automation and control applications where failures can potentially have catastrophic consequences has led to the formulation of additional trustworthiness requirements for safety-critical systems. While reliability and availability measures are concerned with the "good" or "desirable" things that the system should do, safety concerns address the "bad" things that should not happen during the operation of the system. Safety analysis techniques were first used in Inter-Continental Ballistic Missile (ICBM)-based weapon systems to ensure the absence of scenarios that could potentially lead to disastrous failures [12]. System safety is defined as follows:

- **Safety:** The safety of a system is the probability that it does not result in any catastrophic consequences for the user(s) or the environment.

With the advent of networked systems and the growing concern about cyber attacks, concerns about other "bad" things that should not happen during the operation of a system have been investigated in the context of system security. Unlike reliability, availability, and safety issues, security is an umbrella term that covers several more specific trustworthiness issues, including system integrity, confidentiality, privacy, trust, authenticity, nonrepudiability, and credibility:

- **Security:** The security of a system is the probability that it can operate correctly in spite of intentional efforts to cause it to do otherwise. It consists of several additional aspects:
  - **Integrity:** The integrity of a system is the probability that it does not have any unauthorized system alterations.
  - **Confidentiality:** The confidentiality of the system is the probability that it does not allow unauthorized disclosure of information.
  - **Privacy:** The privacy of the system is the probability that private information will not be disclosed in spite of potential inferences from multiple information sources [1, 7].
  - **Authenticity:** The authenticity of a system is the probability with which it can assure the integrity of specified data items, including the integrity of the actual content of the information as well as auxiliary associated information such as the creator of the information or its creation time.
  - **Nonrepudiability:** The nonrepudiability characteristic of a system is the probability with which it can assure the availability and integrity of information regarding the creator of a data item as well as the availability and integrity of information regarding those who access that item [8].
  - **Credibility:** The credibility of a computer system is the probability that its operation is trustworthy (i.e., well-intentioned, truthful, unbiased) and reflects adequate expertise (i.e., knowledgeable, experienced, competent) [9].

Another umbrella term in trustworthiness is system maintainability. In its original hardware context, system maintainability was a measure of the ease with which the system can be repaired in the event of a failure. This was captured by the repairability measure of the system. With software playing an increasingly important role in computer systems, maintainability now also includes other factors as described below.

- **Maintainability:** The maintainability of a system is the probability that it has the ability to undergo repairs and modifications. Maintainability can be decomposed into the following attributes:
  - **Modifiability:** The modifiability of a system is the probability that its design and implementation can be updated to add new capabilities or alter existing capabilities.
  - **Repairability:** The repairability of a system is the probability that detected faults in the system, whether due to latent development defects or due to failures caused by physical wear and tear, can be successfully corrected to restore the system to its correct operational state.
  - **Configurability:** The configurability of the system is the probability that it has adjustable parameters that can be set during its operation to enable it to function correctly under different operational situations.
  - **Adaptability:** The adaptability of a system is the probability that its design and/or implementation can be rapidly altered to enable it to function correctly under different operating conditions.
  - **Autonomy:** The autonomy of a system is the probability that the system can correctly adapt to different operating conditions by itself.

Another set of quality factors is the performance of the system, including temporal and spatial measures. These are defined as follows:

- **Performance:** There is usually a range of acceptable values for each performance attribute. The specification of the acceptable range of values for an attribute can sometimes be a fuzzy quantity [10]. For example, for hard real-time systems, such as missile control systems, the system fails if it cannot meet complete its task within a specified deadline. For soft real-time applications, however, such as net-centric conferencing systems, some missed deadlines can be tolerated [15]. In the latter case, the range is a fuzzy value.
  - **Timeliness:** This is a measure of the time taken by the system to complete its task. This is especially critical for real-time systems [5, 11].
  - **Precision:** This is a measure of the quantity of data present in the output of the system, e.g., the number of bits in a numerical value [10].
  - **Accuracy:** This is a measure of the deviation of the output of the system from the correct output [10].

Though reliability, availability, and security address several major aspects of trustworthiness, the design issues concerning these aspects generally do not scale

up to catastrophic failures or attacks. With some specific types of redundancy, survivability can be an additional aspect that specifically addresses catastrophic failures or attacks.

- **Survivability:** This is defined as the probability that the system can complete its mission in a timely manner in spite of attacks, failures, and catastrophic natural disasters. It integrates security assurance techniques with risk management strategies to protect the core capabilities, such as essential services, of a net-centric system even under adverse conditions [13, 16].

A higher level grouping of these aspects includes dependability, resilience, and trustworthiness.

- **Dependability:** The dependability of a system is the probability that it delivers service that can be justifiably depended on, i.e., the probability that it will perform correctly under the specified operational and environmental conditions over a specified time period [3]. Dependability includes availability, reliability, safety, integrity, confidentiality, and maintainability.
- **Resilience:** The resilience of a system is the probability with which it can bring itself back to a correct state from an incorrect or failed state and then resume normal operation [2]. It is related to conventional fault-tolerant computing methods. Resilience aspects include maintainability and survivability.
- **Trustworthiness:** The trustworthiness of a system is the degree to which one can justifiably accept or rely upon the operation of the system [3]. Trustworthiness is a comprehensive system quality measure that includes all the dependability and resilience as well as security and performance attributes. Based on the individual trustworthiness aspects and group of aspects listed above, the ontology along the trustworthiness aspect dimension can be described above and illustrated as shown in Fig. 1.

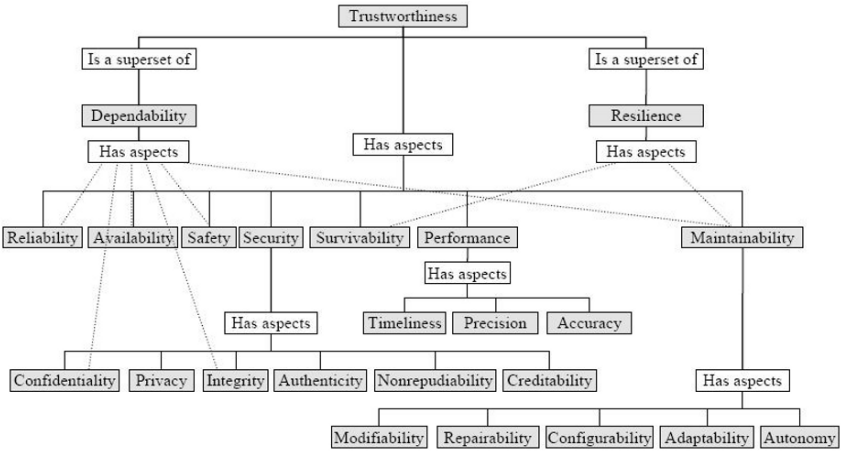


**Fig. 1.** High level ontology of trustworthiness aspects.

### 2.1.2 Trustworthiness Evidences

Many of the trustworthiness aspects are correlated. However, it is difficult to describe such correlations since it is not the case that one aspect is directly dependent on another; rather, the system trustworthiness evidences that these aspects are dependent on define the correlations among the trustworthiness aspects. We propose a novel and effective way to observe the correlations among trustworthiness aspects by defining a trustworthiness evidences ontology.

Each trustworthiness evidence defines a set of observable as well as quantitatively or qualitatively measurable properties of the system or system entities and the trustworthiness evidences are orthogonal to each other. We build different categories of trustworthiness evidences. At the top level, the trustworthiness evidence is partitioned into:

- Positive trustworthiness evidences. Positive trustworthiness evidences can be classified into many categories. Each trustworthiness evidence may be further decomposed into finer-grained trustworthiness evidences. Trustworthiness evidence can be collected to facilitate high assurance, dependability, trustworthiness assessment.
- Negative trustworthiness evidences. Negative trustworthiness evidences describe external evidences that are not within the system but may impact the system assurance. For example, faults and threats are negative trustworthiness evidences. In [3], a thorough taxonomy of faults and threats has been constructed, which can be used as the negative evidences.

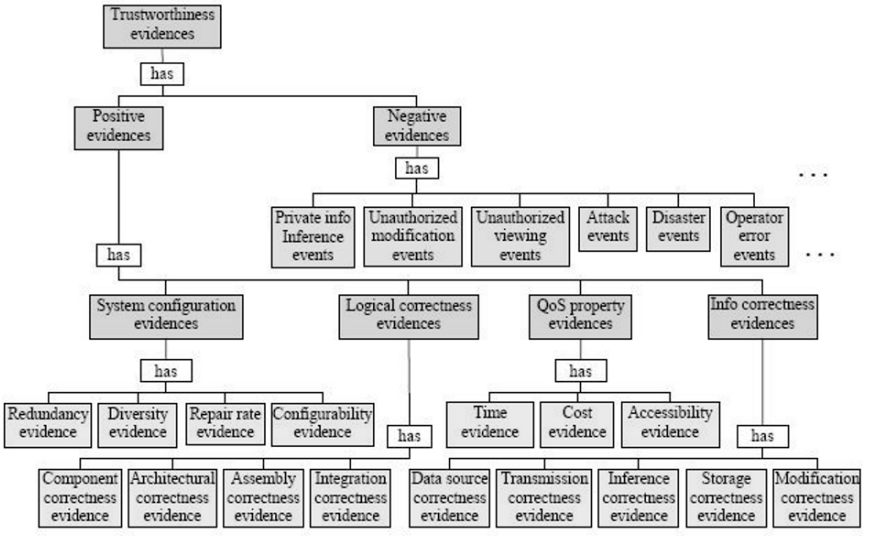


**Fig. 2.** Ontology of trustworthiness evidences.

The ontology of trustworthiness evidences can be quite extensive. The granularity of the evidences is determined based on whether it is possible for evidence data collection at the leaf nodes. A partial ontology is shown in Fig. 2. In this figure, some major faulty and attack evidences are included in the negative trustworthiness evidences. The positive evidences are divided into the system configuration, software logical correctness, information correctness, and QoS properties evidence sets. These positive evidences are further divided into finer grained evidences.

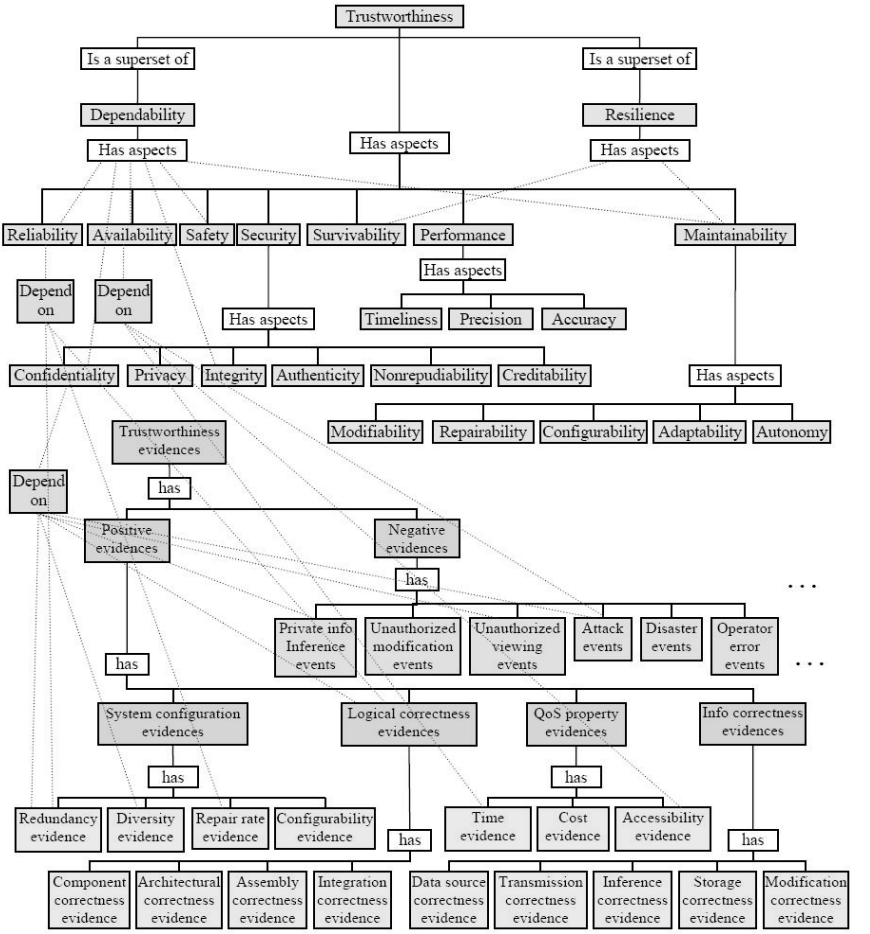


**Fig. 3.** Integrated trustworthiness ontology defined on trustworthiness evidence ontology.

### 2.1.3 Trustworthiness Ontology

The dependencies of the trustworthiness aspects on the trustworthiness evidences can be constructed by merging the high level trustworthiness aspects ontology given in Fig. 1 and the ontology of trustworthiness evidences given in Fig. 2 and drawing the dependency links from each trustworthiness aspect to the related trustworthiness evidences. The merged ontology is shown in Fig. 3.

The dependency definitions for the trustworthiness aspects given in Fig. 3 are partial and are shown only for two trustworthiness aspects, namely, availability and confidentiality. The relationship of the two trustworthiness aspects can be observed from the ontology. Some examples are as follows:

- Redundancy evidence contributes to the assessment of the availability, reliability, and confidentiality aspects. In other words, these aspects are correlated in terms of the redundancy evidence. In reality, the higher the level of redundancy, the higher will be the likelihood of a subsystem or a system entity being available. But the higher the redundancy level, the higher is the probability that one weak point in the system may be compromised.
- Attack evidences can result in unauthorized viewing evidences and unauthorized modification evidences. Thus, confidentiality and availability are both impacted due to attacks.
- Confidentiality and availability do not appear to share other trustworthiness evidences.

## *2.2 Ontology of the System Entities Dimension*

With the rapid advances in computer and communication technologies, many application systems are shifting into the network-centric paradigm. A network-centric system typically involves a distributed set of sensors, actuators, processors, software, along with a variety of other resources interconnected together by a network and interacting with and controlled by end users. The system entities in a network-centric application can have a significant impact on the types of faults and threats and on the trustworthiness analysis. Here we define the ontology for the system entities and the relationships of their trustworthiness evidences (as shown in Fig. 4). A subsystem consists of multiple system entities and their interactions. Similarly, a system consists of subsystems and system entities and their interactions. System entities can be categorized into:

- **Computer platforms**. Each computer platform consists of the hardware and many systems software components, such as operating systems and system utilities. In this chapter, we assume that the computer platforms are connected through public or private networks.

- **Devices**. Physical devices are of many different varieties, such as various sensors and actuators, robots, unmanned or crew controlled vehicles, etc. Some devices are equipped with software control units and/or communication capabilities.
- **Communication channels**. Communication channels provide the connectivity among computer platforms, devices, and human operators and users. They can be wired, wireless, or operate across some other medium.
- **Application software and policies**. Generally, in a large-scale system, there may be a lot of application software for achieving various tasks. They may run on a single computer platform or across multiple computer platforms and devices. Also, with the network-centric nature of many modern applications, the systems are becoming multi-institutional or even multi-national. Different policies must be defined in the system to govern the system operations and resource accesses.
- **Information**. Information category can be further decomposed into raw data, metadata, semantic information, inferred knowledge, etc.
- **Human**. Humans always play an important role in large-scale systems. Most of the system interactions involve operators and users.

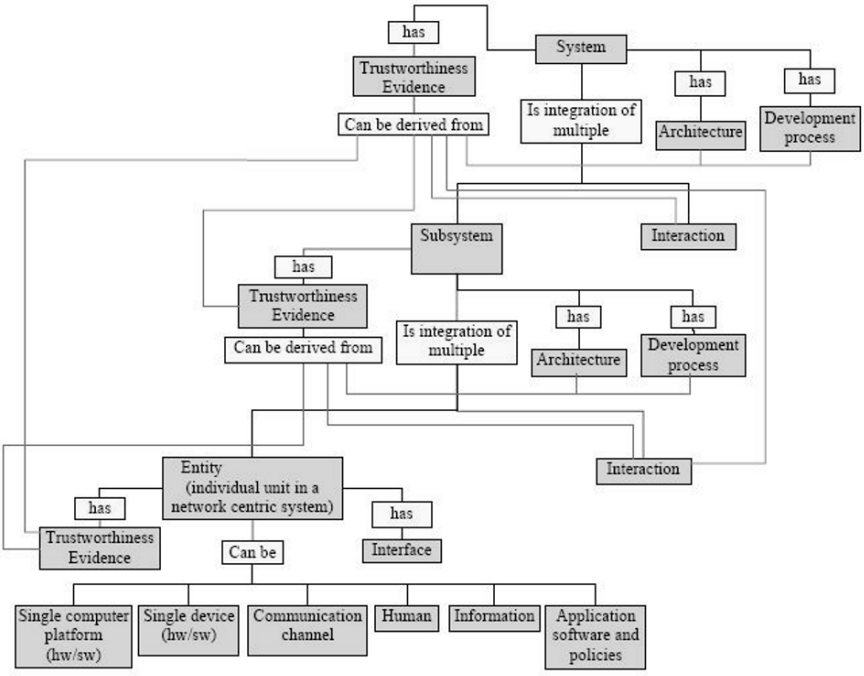


**Fig. 4.** Ontology of system, subsystem, and system entities.

We separate the hardware components into computer platforms and devices, though it is difficult to draw a clear line between these entities. Generally computer platforms have higher computation and storage power and have a common system structure, including the hardware, operating systems, etc., and are capable of hosting a variety of application software. On the other hand, devices are mostly specialized for specific purposes, are frequently mobile, and can vary greatly in their power. Software and policies are highly important in modern systems and they can have substantial variations. They are being placed in the same category since most policies are realized through software.

The interactions among system entities can have significant impact on the overall system trustworthiness analysis and assessment. In the literature, the analysis techniques for interactions among multiple system components have not been widely studied. This is especially true for different types of system entities. Thus, it is important to understand the possible interactions to facilitate systematic analysis and to ensure that all parts of an integrated system are covered in the assessment process. Each system entity can have interactions with another system entity. For example, software and hardware components may have close interactions. Successful completion of critical tasks requires both software and hardware to have correct behavior. Software techniques are frequently used to mask hardware failures. Hardware techniques can be used to detect and isolate software faults. Standalone subsystems interact with each other through communication channels. When delivering information via communication channels, the subsystem needs to process the information to make it suitable for delivery. Human interaction with other system entities is also a critical issue in high-assurance systems. Many system failures can be traced back to human errors [6]. Thus, it is important to investigate the human entity in high assurance systems.

System level trustworthiness evidences are defined based on the trustworthiness evidences of its entities. Methods for such derivations can be associated with the corresponding nodes in the ontology. Some of these methods can be very difficult to derive. For example, the reliability of the system depends on the way one system entity uses another. A system can be highly reliable even if some entities are faulty as long as those faults are not triggered under the system interaction patterns. Likewise, even if individual entities are highly reliable, collectively the system may lead to poor reliability due to unexpected interactions between the entities [3].

Consider an example information subsystem in a net-centric application. The system offers data, metadata, and semantics of the data and knowledge. An information system also needs to manage the access rights and host information processing software and environment. Thus, the information subsystem can consist of the following system entities.

- Devices:
  - Sensor networks that serve as one type of information sources.

- Platforms:
  - Server platforms that interface with the operators for entry of information from various sources.
  - Storage platforms that host raw data and metadata.
  - Platforms for access control management and authentication, such as certification authorities.
  - Platforms hosting data processing and knowledge inference.

- Policies:
  - Access control policies.
  - Data management and interoperation policies.

- Software:
  - Access control and authentication software.
  - Data management software.
  - Data processing software.

- Human:
  - Users who own the viewing and/or modification privileges for all or a subset of the data sets.
  - System administrators who manage the platforms, file systems, or databases.
  - System operators.

- Communication channels
  - Wireless and wired networks and communication software that link all platforms and devices together.

Besides the system entities, interactions among the system entities can also be defined. Based on these subsystems and system entities in each of the categories and interactions among the system entities, the ontology of system entities can be expanded. Some partial expansion for the example information subsystem is shown in Fig. 5.

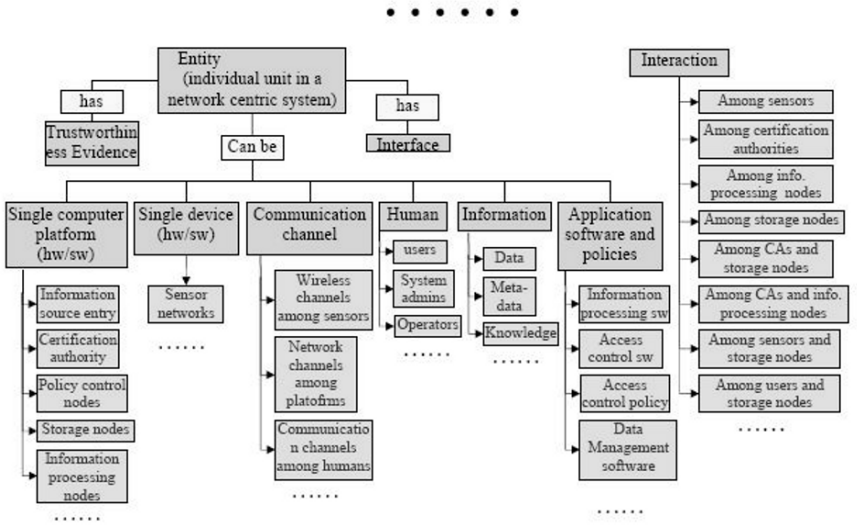


**Fig. 5.** Expanded ontology along the system entities dimension.

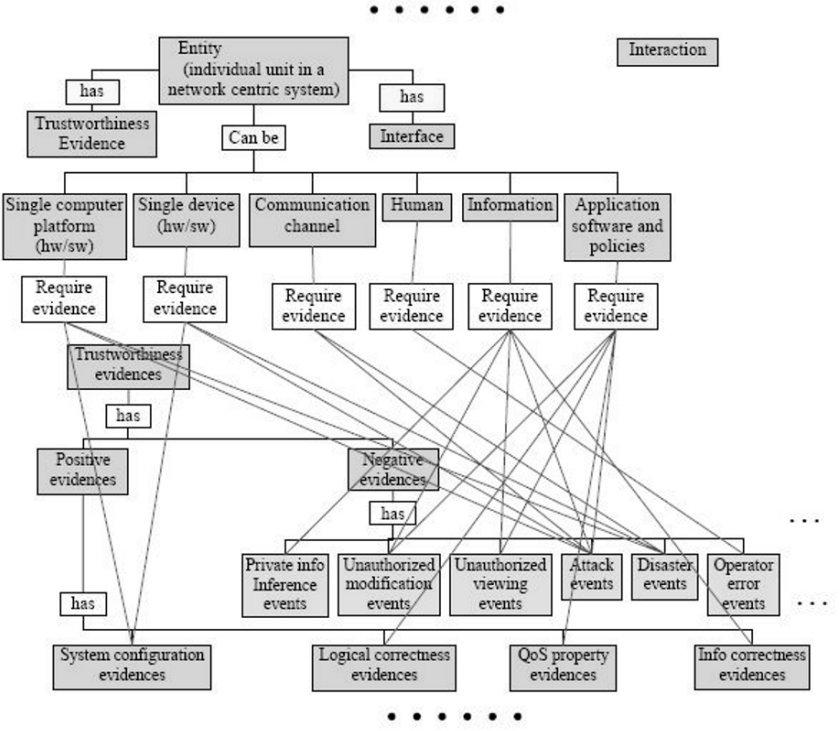


**Fig. 6.** Merging trustworthiness ontology and system entities ontology.

## 3. An Ontology-based Integrated Assessment Framework

We have developed ontologies along the trustworthiness aspects and system entities dimensions. These ontologies can be merged together to facilitate rigorous trustworthiness analysis. Merging ontologies requires the expansion of each trustworthiness evidences at the leaves of the ontology based on the system entity ontology to include the relevant system entities.

In this section, we illustrate the ontology merging process in several steps. First, a partial expansion at lower levels (system entities and trustworthiness evidences) is shown in Fig. 6. For example, the attack evidence can be applied to computer platforms, devices, software, and communication channels. An unauthorized viewing evidence can be applied to the information entity. The operator error evidence can only be applied to the human entity. The logical correctness evidence can be applied to software and policies. The information correctness evidence requires the verification of the information sources, such as from sensor networks (devices), existing information (information), or human operators and users.

The ontology can provide a clear categorization of negative evidences (faults and threats) based on the categories of system entities and evidences themselves. Also, it further indicates the necessary evidences required to achieve assurance of various system entities. Techniques for collecting the evidences should be associated with the merged ontology to facilitate overall system assessment. For example, there are many techniques for collecting the logical correctness evidences for software and hardware components, including testing and formal verification. Based on the testing or verification results, the reliability of the reliability of the corresponding component can be derived. The logical correctness evidences can be used for security assessment as well. The data collection for some of the evidences and events given in Fig. 6 cannot be collected directly and further decomposition is needed. For example, consider the undesired viewing event for the information components. This event can be further decomposed into node "compromisation" event, policy inconsistency event, etc. The probability of occurrence of these events in the system can be used for system confidentiality assessment.

The merged ontology can provide a clear categorization of negative events (faults and threats) based on the categories of system entities and events themselves. Also, it further indicates the necessary evidences required to achieve assurance of various system entities. To further illustrate the merged ontology, we expand the confidentiality aspect for the example information subsystem described in Fig. 5. The expanded ontology is shown in Fig. 7. In this example, the expansion is done partially, only considering the attack events and unauthorized viewing events. Each of the trustworthiness evidences is expanded based on the involved system entities. Some examples of the merged view are discussed in the following.

- The attack event may be applicable to platforms, devices, and communication channels. The platforms could be storage platforms, certification authorities, and nodes for data entries. Thus, for assessing confidentiality of the subsystem, trustworthiness evidence, the attack probability, for the storage platforms, the certification authority platforms, and the communication channels among them are to be considered.
- The unauthorized viewing event can be due to a compromised platform, a compromised device subsystem, a compromised communication channel, or an untrustworthy human. Also, incorrect software and policies can cause information breaches as well.

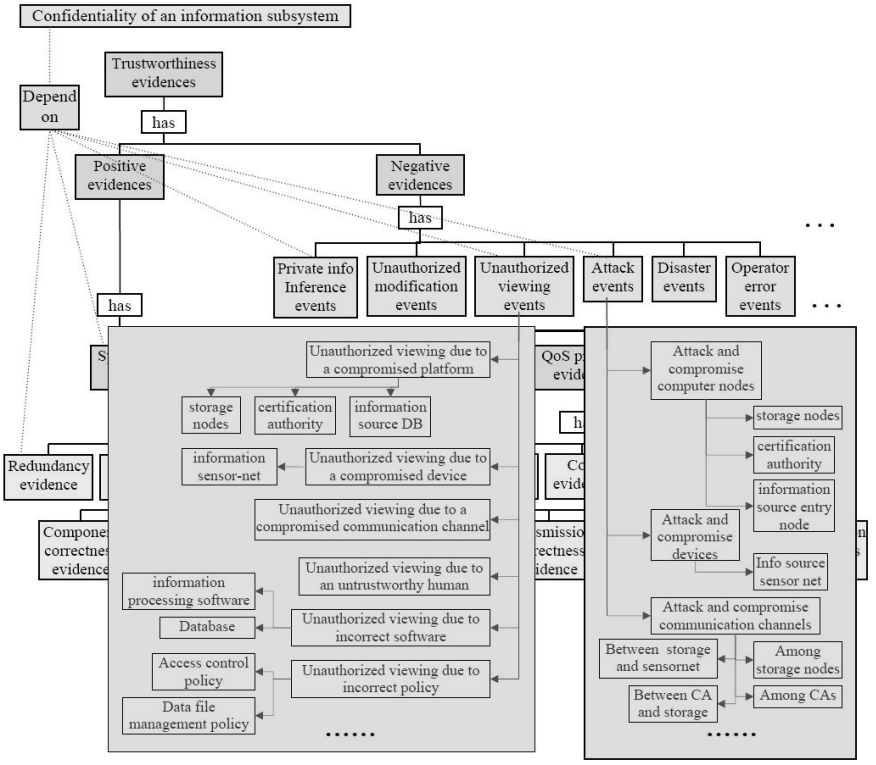


**Fig. 7.** Expanding trustworthiness evidences based on system entity ontology.

The merged ontology clearly indicates the evidence data to be collected for each system entity. Based on the merged ontology, the system analysis can be done in a well guided manner.

The discussion above (including Fig. 6 and 7) focuses on ontology merge of the system entities and the trustworthiness evidences. Consider the upper levels in the merged ontology. Each trustworthiness aspect of the system depends on the trustworthiness evidences of the system. The trustworthiness evidences of the system

can be derived from the trustworthiness evidences of the subsystems and individual system entities. The trustworthiness of the subsystem and individual system entities can also be derived from the trustworthiness evidences of the subsystems and individual system entities, respectively. Such derivations form the basis of the ontology-driven trustworthiness evidence based integrated trustworthiness assessment technology. In Fig. 8, the derivation of trustworthiness at various levels is illustrated.

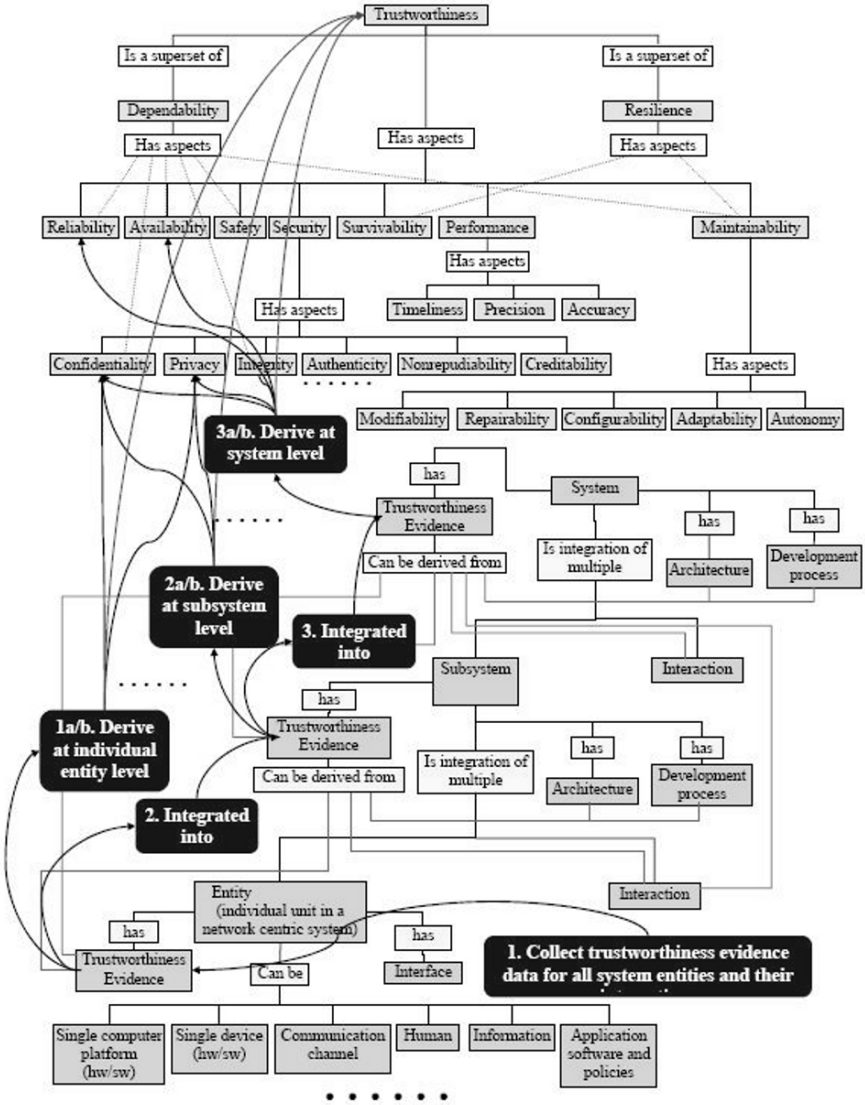


**Fig. 8.** Trustworthiness evidence based assessment procedure.

Based on the ontology, a systematic procedure can be used to guide system assessment and it is illustrated in the following.

- System entity level trustworthiness assessment.
  - **Step 1**: Collect trustworthiness evidence data. The first step for all trustworthiness assessment is to collect trustworthiness evidence data for each system entities. Note that earlier discussions (Fig. 6 and 7) offer more detailed guidelines for data collection for various system entities and various trustworthiness evidences.
    - If the goal is to assess the system entity level trustworthiness, then go to Step 1a.
    - If the goal is to assess the individual trustworthiness aspects at the system entity level, then go to Step 1b.
    - If the goal is to assess trustworthiness at a higher level, then go to Step 2.
  - **Step 1a**: The single trustworthiness measurement. This measurement can be derived from trustworthiness evidence collected for the system entity.
  - **Step 1b**: Measurements of each trustworthiness aspect.
- Integrated assessment for a subsystem.
  - **Step 2**: Collect or derive trustworthiness evidence data. Trustworthiness evidence data of the subsystem can be derived from the trustworthiness evidence data of the constituting system entities and the architecture that specifies the interactions among the entities. For some trustworthiness evidences, the data can be collected directly. The derivation algorithm is evidence set dependent.
    - If the goal is to assess the subsystem level trustworthiness, then go to Step 2a.
    - If the goal is to assess the individual trustworthiness aspects at the subsystem level, then go to Step 2b.
    - If the goal is to assess trustworthiness only at the overall system level, then go to Step 3.
  - **Step 2a**: The single trustworthiness measurement. This measurement can be derived from trustworthiness evidence of the subsystem.
  - **Step 2b**: Measurements of each trustworthiness aspect. The derivation formula for the measurements is aspect dependent.
- Integrated assessment for the overall system.
  - **Step 3**: Collect or derive trustworthiness evidence data. Trustworthiness evidence data of the system can be derived from the trustworthiness evidence data of the constituting subsystems and the architecture that specifies the interactions among the subsystems. The derivation algorithm is evidence type dependent.

  - If the goal is to assess the system level trustworthiness, then go to Step 3a.
  - If the goal is to assess the individual trustworthiness aspects at the system level, then go to Step 3b.
- **Step 3a**: The single trustworthiness measurement. This measurement can be derived from trustworthiness evidence of the subsystem.
- **Step 3b**: Measurements of each trustworthiness aspect.

The steps discussed above, including assessment data collection for individual system entities regarding various trustworthiness evidences, integration of the evidence data from system entities level to subsystem level and to system level, and derivation of trustworthiness aspect assessment results from the evidence data, involve various assessment techniques. To complete the framework, the merged ontology should be further expanded to include the assessment techniques. In Step 1, the techniques for evidence collection can be associated with the corresponding nodes in the merged ontology. In Steps 2 and 3, assessment of a subsystem or the overall system can be done directly at the system level. For example, testing can be conducted at the overall system level to collect evidences of its behavior and subsequently assess its trustworthiness properties. In some situations, such subsystem or system level testing and verification is infeasible. For example, in a large-scale system that is widely distributed, it may be difficult to simulate the realistic environment for testing. Also, such testing could be too costly. Further, at design time, it may often be difficult to understand the impact of selecting a certain technique or component in the overall system behavior. Since many different compositions may have to be considered, the testing of each composition is simply not possible. Thus, it is necessary to derive the system level properties from subsystem level and component level evidences. Such derivation techniques are highly challenging. Techniques for many different types of integrations are still to be investigated.

An example ontology with the assessment techniques is given in Fig. 9. The assessment techniques shown in the figure are for the reliability aspect. We consider the techniques for the integration of the trustworthiness evidences at the system entity level into the evidences at the subsystem level and direct assessment techniques at the subsystem level. For the integration of multiple system entities of the same type, we need to consider integration of software entities, integration of hardware entities, integration of communication channels, and integration of information sources (though some are not shown in the figure). In cyber world, we disregard human-human interactions and only consider humans interacting with the cyber world (mainly with software). We also consider integration of system entities of different types, such as integrating hardware and software, software and human, software and information, human and information, etc. Most of the existing techniques in reliability assurance and assessment are based on integrated testing.

The integrated assessment framework is flexible and expandable. Each dimension has its own ontology which can evolve independently. Expansion from the nodes in the merged ontology can be linked to the nodes in the individual ontologies. The ontologies can be customized to fit the needs of the special applications.

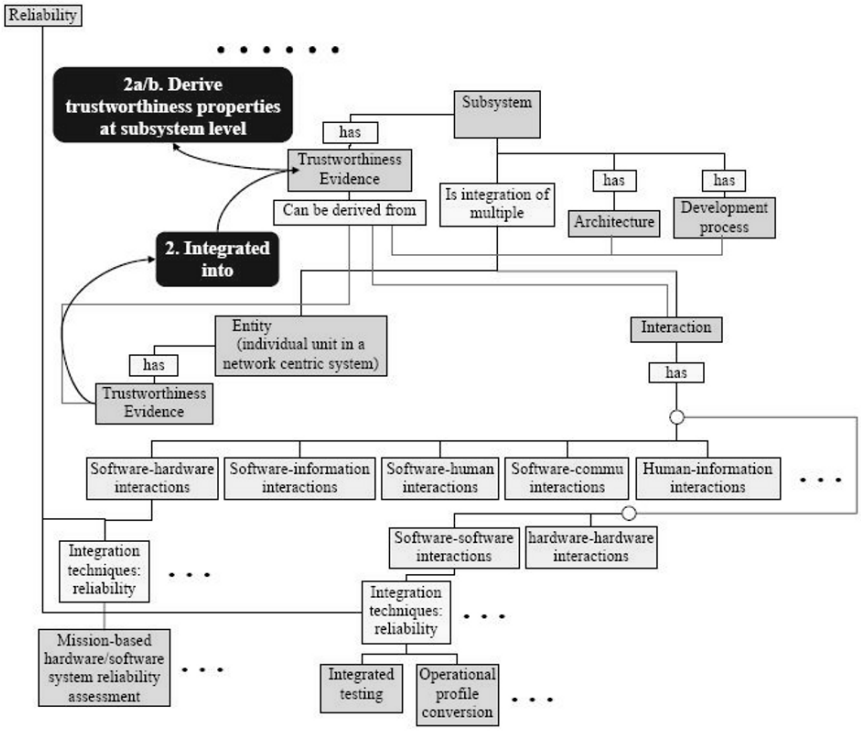


**Fig. 9.** Ontology with integrated assessment techniques.

## 4. Holistic Assessment Techniques

The goal of the framework discussed in Section 3 is to provide a comprehensive guidance toward systematic assessment of integrated systems considering various trustworthiness aspects. One important assessment that is frequently demanded is a holistic view of the overall system or subsystem. For example, a commander may demand a single "score" to represent the trustworthiness of a system. When a third party delivers a product, the manager may want to know the level of assurance in a holistic view, instead individual aspects. It is difficult to give such a single "score" in a rigorous way. In this section, we define a mission-driven integration method to integrate multiple aspects into one single measure of

the trustworthiness of the NCS, which indicates the probability of success of the given missions. The assessment is based on a collection of events that can impact the degree to which one can use the system to successfully accomplish all the specified mission objectives. These events are classified into two categories, namely, essential events, **E**, and adverse events, **A**. The set of essential events consists of all those events that must occur in order for the system to complete the mission successfully. These include the following types of events:

- **The system is available when needed:** This is the first step in using the system and requires the system to be operational when the user needs it. This corresponds to the classical availability measure among the set of dependability aspects. It is also affected by the reliability, maintainability, adaptability, and reconfigurability qualities of the system.
- **An authorized user can use the system when needed:** This event ensures that the system will not make it difficult for an authorized user to use it. It factors in the possibility that security measures to prevent unauthorized accesses could pose obstacles for legitimate users. Examples include the possibility of forgotten passwords and failures of biometric authentication systems.
- **The computations of the system are logically correct:** This is related to some aspects of reliability and safety dependability aspects. It requires the system to generate correct outputs when presented with inputs that satisfy the preconditions of the system.
- **The system timing and performance qualities are acceptable:** This is related to performance issues in addition to reliability and safety issues. It is a critical requirement for real-time systems that must generate outputs in a timely manner. It is also important in other situations and encompasses classical termination requirements, i.e., the requirement that the system must not have any infinite loops or be susceptible to deadlocks, livelocks, etc. In terms of real-time performance, it may be possible to specify the tolerance of missed deadlines, as well as the tolerance of the quality of a result to meet deadlines.
- **The cost and resource requirements of the system are acceptable:** This corresponds to the practicality of the system. For example, if the system requires too many processors in order to complete its computations on time, then it may be reliable but not practical.

The second class of events is the set of adverse events, i.e., events that should probably not occur if the system is to be able to complete its mission successfully. The occurrence of an adverse event does not automatically mean that the system will not be able to complete its mission successfully for the specified mission. Instead, it decreases the probability that the system will be successful. The potential adverse events are as follows:

- **Unauthorized users can access the system:** This is a part of the security requirements of the system. For safety-critical system, it can also lead to safety assurance issues since a malicious unauthorized user could deliberately lead the

system to an unsafe state. In practice, the authentication problem is more complicated since an authorized user for some capabilities of the system may be an unauthorized user for other features. For example, an authorized user of the system may be able to view and update some confidential information in the system but may not be allowed to reconfigure the system while another authorized user (such as a system administrator) may be able to reconfigure the system but may not be allowed to access any confidential information in the system.

- **The system triggers operator or user errors:** This is related to the usability aspects of the system. Human errors are often significant causes of failures of systems. These can be prevented by better human factors design as well as the use of sanity checks and other methods of detecting potential user errors. Increasing the system autonomy, as in the design of autonomic or self-stabilizing systems, can help alleviate the stress on the users and, hence, reduce human errors, especially with regard to system adaptation and configuration changes.
- **The system or the environment enters an unsafe state:** Malfunctions in systems that control the physical world via actuators can potentially lead to catastrophic losses of lives and/or property. Such systems are called safety-critical systems and must be designed and certified to be highly safe. Safety is independent of reliability. A classical example is that of a stalled car parked in an area that is away from other traffic. It is fully unreliable but it is safe. Likewise, a car being controlled by a small child can be very reliable but can also be very unsafe.
- **The system information regarding the state, input, output, or code can be viewed by others:** This is a security related attribute and corresponds to the confidentiality and privacy dependability assurance properties of the system. Methods such as data partitioning, code obfuscation, data encryption, etc., can be used to prevent retrieval of confidential or private information by hackers and other adversaries.
- **System information regarding the state, input, output, or code can be changed by others:** This is a security and resilience related aspect corresponding to integrity aspects of the system. Depending on the potential threats, as well as the sources of these threats, various mechanisms can be used to protect the integrity of the system. These include the use of redundancies, error detection codes, write-once memory devices, continuous monitoring, proof carrying codes, etc.
- **The system provides additional functionalities:** This is also a security related issue and corresponds to embedded malicious logic, "Trojan horses", and other extra functions, i.e., functions that are in addition to the ones specified in the requirements specification document. These are difficult to detect, especially if embedded by insiders during the development process. The system can be verified to be logically correct and shown to meet all non-functional requirements, but it may contain additional capabilities that could be exploited to subvert the system. An example is an extra functionality in the system that causes it to

transmit a lot of redundant data at critical occasions, thereby overloading the network and other computers.

The integrated assessment of a system for a specific mission requires the following information:

- For each essential event, methods have to be used to determine the probability of occurrence of that event. For example, consider the event, "The system is available when needed." In this case, the corresponding probability that must be determined is the probability that the system is available when needed. For some of the events, formal methods, including verification and analysis techniques, can be used to fully guarantee the occurrence of that event, in which case the corresponding probability is 1.0.
- For each adverse event, various methods have to be used to determine the probability of occurrence of that event. The probability of occurrence of an adverse event depends not only on the intrinsic capabilities of the system and the platform but also the likelihood of the sources of the corresponding threats. For example, the probability that an unauthorized user will be able to access the system is 0 if it can be guaranteed that there are no unauthorized users in the environment. A specific example would be a system deployed in a highly secure building that is protected by guards and locked doors.
- For each adverse event, determine the "criticality" of the event. The criticality of an adverse event ranges from 0 to 1. It is 0 if the event is fully acceptable, i.e., if the occurrence of the event has no consequence on the successful completion of the mission. It is 1 if the event is fully unacceptable, i.e., if the occurrence of the event will definitely lead to a failure of the system. The criticality of each event is a fuzzy quantity and must be specified as part of the requirements for the mission.

The overall assurance level of the system for the specified mission is given by the probability that the mission will be completed successfully after factoring in all the possible essential and adverse events:

P{the specified mission will be completed successfully} =

$$\prod_{i=1}^{n} P(e_i \mid e_i \in \mathbf{E}) * \prod_{j=1}^{m} \{1 - \sigma_j * P(e_j \mid e_j \in \mathbf{A})\},$$

where $\sigma_j$ is the criticality of adverse event $e_j$, for $1 \leq j \leq m$. $\sigma_j$ ranges from 0 to 1 with 0 indicating that the occurrence of event $e_j$ will not have any adverse consequences for the specified mission and 1 indicating that the occurrence of the event will definitely lead to failure of the mission. The overall result is one number that characterizes the overall effectiveness or assurance level of the system for accomplishing a given mission. This can be used to rank the potential candidates for implementing the system to enable the selection of the best candidate, i.e., the one that has the highest chance of success.

Often, it is necessary to be able to rank a collection of assets that are predeployed with the goal of supporting a range of potential missions that may arise in the future rather than any given specific mission. In this case, the integrated as-

sessment is based on the expected (average) value of the capability of the asset to support the specified set of possible missions. This yields,

P(the asset can support the set of specified missions) =

$$\Sigma_{k=1}^{n} \text{P(mission k can be completed successfully using the asset)|(mission k occurs).}$$

The overall result can then be used to select between different candidate set of assets, i.e., the one that is the most capable in supporting the specified set of missions.

## 5. Summary and Future Research Directions

We have introduced the concept of trustworthiness to include dependability and a comprehensive set of other high assurance attributes. A trustworthiness ontology is developed to capture the trustworthiness aspects and their correlations as well as to model various classes of system entities and their integrations. The ontology provides information to guide the trustworthiness analysis and data collection. Based on the ontology, a trustworthiness assessment framework is developed. In the framework, systematic steps are formulated to achieve trustworthiness assessments. Techniques and tools to perform the assessments in each step are incorporated in the ontology to allow the actual analysis and derivation of assessment results.

We have also identified some missing links in assessments techniques and developed a holistic assessment technique to provide a single overall measure of the trustworthiness of a system or a subsystem.

Future research includes two major directions. First, we plan to analyze the current techniques and tools for each step of the trustworthiness assessment. Based on the ontology, we will identify areas that require further research for new or better analysis techniques. Second, we plan to develop integration based assessment techniques to facilitate assessment of large-scale systems from trustworthiness attributes of their individual subsystems with known trustworthiness assessment. We also plan to develop assessment techniques with holistic views for different levels of NCS systems.

## References

1. Mark S. Ackerman, Lorrie Faith Cranor, Joseph Reagle, “Privacy in e-commerce: examining user scenarios and privacy preferences,” Proceedings of the 1st ACM conference on Electronic commerce, Denver, Colorado, 1999, pp. 1-8.
2. T. Anderson, *Resilient Computing Systems*, John-Wiley, New York, 1985.

3. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. on Dependable and Secure Computing*, Vol. 1, No. 1, Jan.-Mar. 2004, pp. 11-33.
4. F. B. Bastani and A. Pasquini, "Assessment of a sampling method for measuring safety-critical software reliability," *Proceedings of 5th International Symposium on Software Reliability Engineering*, November 1994, pp. 93-102.
5. A.M.K. Cheng, Real-Time Systems: Scheduling, Analysis, and Verification, Wiley Interscience, 2002.
6. Mike Chen, Emre Kıcıman, Eugene Fratkin, Eric Brewer, and Armando Fox, "Pinpoint: Problem determination in large, dynamic Internet services," Dependable Systems and Networks, 2002.
7. Julie E. Cohen, "DRM and privacy," Communications of the ACM (Special issue on digital rights management and fair use by design), Vol. 46, No. 4, April 2003, pp. 46-49
8. Riccardo Focardi, Fabio Martinelli, "A uniform approach for the definition of security properties," World Congress on Formal Methods, 1999.
9. B.J. Fogg and H. Tseng, "The elements of computer credibility," Proc. 1999 SIGCHI Conf. on Human Factors in Computing Systems, Pittsburgh, PA, 1999, pp. 80-87.
10. T.F. Lawrence, "The quality of service model and high assurance," Proc. 1997 IEEE High-Assurance Systems Engineering Workshop, Washington, DC, Aug. 1997, pp. 38-39.
11. E. A. Lee and S. Edwards., "Precision Timed (PRET) Computation in Cyber-Physical System", National Workshop on High Confidence Software Platforms for Cyber-Physical Systems: Research Needs and Roadmap, November, 2006.
12. N. Leveson, Software: System Safety and Computers, Addison Wesley, New York, 1995.
13. H.F. Lipson and D.A. Fisher, "Survivability - A new technical and business perspective on security," Proc. 1999 workshop on New security Paradigms, Caledon Hills, Ontario, Canada, 1999, pp. 33-39.
14. B. Littlewood and L. Strigini, "Software reliability and dependability: A roadmap," *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, A. Finkelstein (ed), June 2000, pp. 177-188.
15. J.W.S. Liu, Real-Time Systems, Prentice Hall, 2000.
16. J. McDermott, "Attack-potential-based survivability modeling for high-consequence systems," 2005. Proc. 3rd IEEE Intl. Work. on Information Assurance (IWIA'05), March 2005, pp. 119-130.
17. R. A. Paul, "DoD towards software services," *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, February 2005, pp. 3-6.
18. G. Vecellio and W. M. Thomas, "Issues in the assurance of component-based software," Proc. 2000 IEEE Intl. Work.on Component-Based Software Engineering, Limerick, Ireland, Jun. 2000.
19. J. Voas, "Certifying software for high-assurance environments," IEEE Software, Vol. 16, No. 4, Jul./Aug. 1999, pp. 48-54.

http://www.springer.com/978-0-387-87657-3

High Assurance Services Computing
Dong, J.; Paul, R.; Zhang, L.-J. (Eds.)
2009, XII, 324 p. 130 illus., Hardcover
ISBN: 978-0-387-87657-3