FOI-R--0712--SE
December 2002
ISSN 1650-1942
Scientific report

Amund Hunstad, Jonas Hallberg

Modeling of Distributed Systems Focusing on IT Security Aspects

Command and Control Systems
SE-581 11 Linköping

SWEDISH DEFENCE RESEARCH AGENCY

Command and Control Systems
P.O. Box 1165
SE-581 11 Linköping

FOI-R--0712--SE

December 2002

ISSN 1650-1942

Scientific report

Amund Hunstad, Jonas Hallberg

Modeling of Distributed Systems Focusing on IT Security Aspects

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 Linköping	FOI-R--0712--SE	Scientific report
	Research area code 4. C4ISR	
	Month year December 2002	**Project no.** E7046
	Customers code 5. Commissioned Research	
	Sub area code 41 C4I	

Author/s (editor/s)	Project manager
Amund Hunstad Jonas Hallberg	Jonas Hallberg
	Approved by
	Sponsoring agency Swedish Armed Forces
	Scientifically and technically responsible Jonas Hallberg

Report title

Modeling of Distributed Systems Focusing on IT Security Aspects

Abstract (not more than 200 words)

The appearance of widely distributed systems providing services and information critical to both organizations and individuals results in new challenges for systems and security engineers. While adequate solutions to solve the unavoidable security issues have to be designed and implemented, the systems are increasingly difficult to be comprehended and assessed. Thus efficient design frameworks and modeling techniques are crucial for the development of future systems.

Since no distributed information system can be designed secure, but can include the necessary prerequisites to be secured during operation; the aim is *design for securability*. To achieve design for securability, three steps have to be supported in the design of distributed systems. Firstly, the interactions and relations between the system and its environment have to be captured. Secondly, a set of security requirements on the system has to be formulated. Thirdly, the set of requirements has to be implemented in the system. This report is focused on the third step, which requires system models and design methods and tools. Especially, efforts regarding the identification of security-relevant characteristics and the formulation of adequate modeling techniques are presented. The long-term goal is to build an environment suitable for modeling, simulation, and assessment of security architectures.

Keywords

IT security, distributed systems, modeling techniques, UML

Further bibliographic information	Language English
ISSN 1650-1942	**Pages** 52 p.
	Price acc. to pricelist

Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets Forskningsinstitut - FOI Ledningssystem Box 1165 581 11 Linköping	FOI-R--0712--SE	Vetenskaplig rapport
	Forskningsområde 4. Spaning och ledning	
	Månad, år December 2002	**Projektnummer** E7046
	Verksamhetsgren 5. Uppdragsfinansierad verksamhet	
	Delområde 41 Ledning med samband och telekom och IT-system	

Författare/redaktör	Projektledare
Amund Hunstad Jonas Hallberg	Jonas Hallberg
	Godkänd av
	Uppdragsgivare/kundbeteckning Försvarsmakten
	Tekniskt och/eller vetenskapligt ansvarig Jonas Hallberg

Rapportens titel (i översättning)

Modellering av distribuerade system med fokus på IT-säkerhetsaspekter

Sammanfattning (högst 200 ord)

Införandet av vitt distribuerade system som tillhandahåller tjänster och information vilka är kritiska för både organisationer och individer leder till nya utmaningar för dem som ska konstruera dessa system. Samtidigt som säkerhetskraven blir allt hårdare blir också systemen allt svårare att överblicka och värdera. Därmed blir effektiva designmetoder och modelleringstekniker centrala för utvecklingen av framtida informationssystem.

En effektiv design av säkerhetsarkitekturer innebär att systemen erbjuder adekvata mekanismer för att kunna säkras under drift. För att uppnå detta måste interaktioner och relationer mellan systemet och dess omgivning beskrivas. Vidare måste de för systemet relevanta säkerhetskraven specificeras. Slutligen måste de beskrivna säkerhetskraven implementeras i systemet. Denna rapport fokuserar på implementationen av säkerhetskraven, vilken kräver systemmodeller samt designmetoder och designverktyg. Speciellt studeras problemen gällande identifiering av säkerhetsrelevanta systemegenskaper och formuleringen av modelleringstekniker. Det långsiktiga målet är att konstruera en omgivning för modellering, simulering och värdering av säkerhetsarkitekturer.

Nyckelord

IT-säkerhet, distribuerade system, modelleringstekniker, UML

Övriga bibliografiska uppgifter	Språk Engelska
ISSN 1650-1942	**Antal sidor:** 52 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Contents

1. Introduction

In a networked information society, IT security becomes a crucial ability for individuals, organizations, and the society as a whole. The concept of a networked society by default results in widely distributed information systems that are difficult to control or even comprehend and, consequently, trust cannot be assumed. At the same time, the foundation of the society depends on the users' trust in distributed information systems. Thus, the ability to evaluate and validate the IT security ability becomes central and crucial.

Network centric warfare, NCW, (Alberts, Garstka & Stein, 1999) and critical infrastructure protection, CIP, (ETH, 2001) are two phenomena that emphasize the need for adequate IT security ability and, hence, the need to be able to evaluate the IT security ability. Basically, NCW means that all intelligence resources are connected to and available through a network common for all allied units. This results in dramatic increases in efficiency for several critical abilities, such as communication, situation awareness, and chain of command. Thus, NCW assumes an underlying information infrastructure, which has to be efficiently protected from information operations performed by adversaries. Naturally, the ability to withstand attacks has to be evaluated. CIP on the other hand will become increasingly dependent on IT security since all infrastructures will include, or depend on, distributed information systems.

1.1 Motivation

Dependence on information and IT results in dependence on IT security. However, there are many problems inherent with the current practice of a launch-and-patch approach to IT security. Also the more careful approach of using red teams/tiger teams, rather than waiting for things to break, has its limitations and drawbacks (Gula, 1999; Schudel & Wood, 2000). The use of vastly distributed information systems will drastically increase the risks associated with security breaches (Schneier, 2000). Thus, the need to get it right from the start should be a cornerstone in all future system development. Unfortunately, systems will always contain flaws. Thus, risk management becomes a necessity and "to get it right from the start" means building systems able to handle failing components and unexpected events. This creates new demands on the ability to comprehend vastly distributed systems and to understand the effects of events in and design decisions affecting these systems.

1.2 Problem Formulation

Design for securability, our approach to applying engineering principles to system security design, can be described as a process integrating knowledge of the system and its environment. The approach is based on:

- the importance of mutual trust between system owners and operators and in systems and organizations,
- requirements engineering,
- systems modeling,
- methods and tools for design support, and finally
- how risk management may be eased by such a design approach.

To contribute to the realization of the design for securability approach, this report targets the vital tasks of systems modeling and systems securability (IT security ability) assessment. Regarding the issue of a networked information society, this is performed in the context of widely distributed information systems.

Efficient modeling techniques are needed to be able to comprehend distributed information systems. The main issue is to find methods to model systems so that security aspects can be reasoned on. The security of a system is affected by a number of factors, such as the implementation of adequate security mechanisms, human system interaction, and organizational aspects. However, the scope of this work is limited to technical aspects of distributed information systems. The reason for this limitation is to be able to produce viable results in the selected area. However, the awareness of the limitations results in the demand that proposed methods and techniques should be possible to extend in order to incorporate additional aspects of relevance for IT security.

The evaluation of the securability of systems requires efficiency measures for the enabling IT-security mechanisms. If the securability of a system represented by a model can be evaluated beyond the currently used ad hoc and subjective measures, then the model can be used to reason about the system and its securability. Furthermore, an efficient evaluation ability will enable novel design methods and design support tools. The first step towards realizing securability assessment is to find security-relevant characteristics for distributed information systems.

1.3 Contributions

The main results produced by the effort described in this report are:

- a survey of contemporary security-related modeling techniques for distributed system,

- a method to identify security-relevant system characteristics,
- a set of IT security-relevant characteristics for distributed information systems and a structuring of the identified characteristics, and
- a modeling technique for distributed systems models focusing on IT security.

1.4 Report Layout

In chapter 2, background to the work presented in this report is discussed. In chapter 3, the concept of design for securability is introduced. In chapter 4, the method for extraction of security-relevant system characteristics and the resulting structuring of characteristics are described. In chapter 5, a modeling technique for distributed systems is introduced. Finally, in chapter 6, conclusions are drawn.

2. Background

In this chapter, the framework of the topic of the paper is set by introducing the concepts of IT security, IT defense, distributed systems, and security architecture and discussing related work.

2.1 IT Security

When defining IT security, there are several different dimensions that can be used. Firstly, IT security has often been divided into aspired characteristics; the most commonly used are confidentiality, integrity, and availability. Secondly, IT security can be defined using high-level functions such as the triplet protect, detect, and react, which states that adequate IT security requires the ability to protect a system, detect attacks against the system, and react to these attacks. Lately, the ability to survive attacks has been added, thereby forming the quartet protect, detect, react, and survive. Thirdly, IT security can be defined by stating what has to be secured. In this case, IT security can be defines as the part of information security relating to the use of IT. Which of these definitions that is most appropriate depends on the current perspective on IT security. To illustrate the vastness of the IT security area, the three aspects System environment, User, organization & system, and Technology & system of IT security can be used. These aspects are further discussed in the section on IT defense below.

In this report, IT security is considered to be the process of upholding the confidentiality, integrity, and availability of information and services provided by IT-based information systems. An important aspect of this definition is that IT security is a process, not a product (Schneier, 2000). This is why we introduce *securability* as the main feature of a system supporting the process of securing a system in operation.

2.2 IT Defense

Considering the widely distributed information systems that will form the information infrastructure of a networked society, traditional IT security will not suffice to handle the risks posed by IT-based attacks against these systems. In this case, traditional IT security is viewed as the protect, detect, and react paradigm. Abilities extending beyond the realm of control have to be at least considered, if not utilized. Moreover, other abilities completing the IT defense have to be acknowledged. Hallberg, Hunstad, Eriksson & Palmgren (2002) describes a set of five abilities: IT attack, IT security, IT reconnaissance, IT defense evaluation, and IT defense command and control. These five abilities together constitute an IT defense.

The division into several abilities is used to differentiate between the abilities concerning own systems (IT security) and the abilities aimed towards other systems (IT attack and IT reconnaissance). Moreover, the two abilities IT defense evaluation and IT defense command and control are required to build an effective IT defense that is efficient and capable to adapt to current needs. All five abilities of an IT defense can be viewed from three different aspects. These aspects are *System environment*, i.e. the interaction between the system (including users and organization) and the surrounding society; *Users, organization & system*, i.e. the interrelationship between man and machine and between organization and information system; and *Technology & system*, i.e. the technical issues of the system. All abilities and aspects are inter-dependent, resulting in a complex pattern of interaction. Figure 1 describes a simplified view of the relationships between the concepts and the abilities.

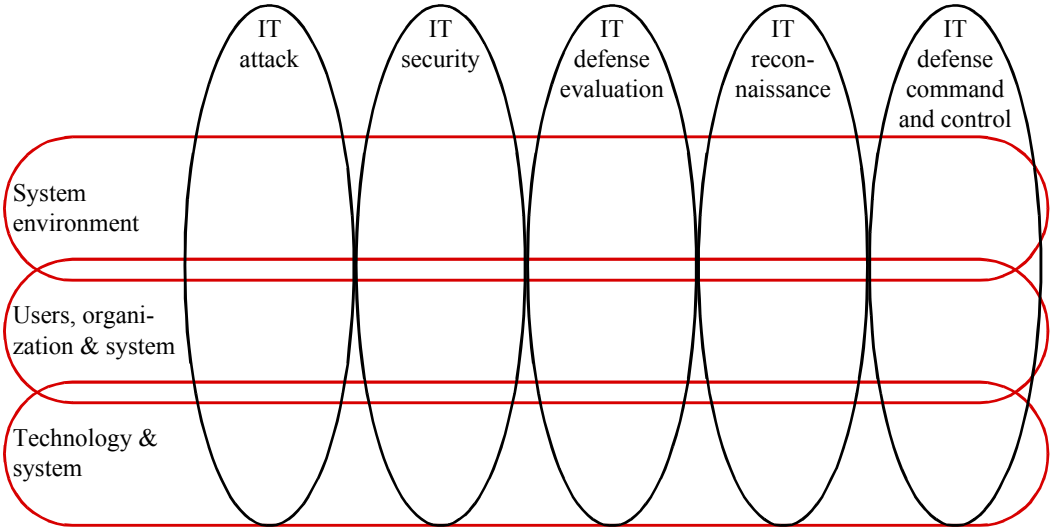

Figure 1: The abilities and aspects of IT defense (Hallberg, Hunstad, Eriksson & Palmgren, 2002).

There are three main purposes with the structuring of IT defense area depicted in Figure 1. Firstly, it illustrates the vastness and complexity of the area. Second, it emphasizes the need to take inter-disciplinary approaches to the research problems related to IT defense. Thirdly, it can be used to understand in which part of the area the results produced by a research project will apply and how it relates to the results of other projects. As specified in the Problem formulation, Section 1.2, the subject of this work is located in both the IT security and the IT defense evaluation abilities, but concentrated to the aspect Technology & system.

2.3 Distributed Systems

Since the purpose of this report is to discuss modeling of distributed systems, a definition of distributed systems is necessary. Like IT security, distributed systems can be defined in different dimensions, such as physical location, processing power, information, and services. Leslie Lamport intuitively captured the nature of distributed systems with the statement "You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done".

Coulouris, Dollimore, & Kindberg (2001) define distributed system as "one in which components located at networked computers communicate and coordinate their actions only by passing messages." This is a general definition comprising most contemporary systems, such as, the Internet, intranets, and mobile and ubiquitous computing.

Client-server and peer-to-peer are the two main principal architectures of distributed systems. The *client-server model* has been, and still is, the most used. Special solutions have been proposed to avoid the drawbacks of single-point-of-failure and performance bottlenecks. For example, by partitioning the objects used to build a service or replicating them, services can be provided by multiple servers. To increase performance and availability, caches and proxy servers are used. For example, web browsers maintain a cache on single-client basis, while a web proxy server provides a shared cache for several clients. Proxy servers can take on other roles as well, such as, the one of acting on behalf of a client when web pages or other Internet resources are accessed at remote servers. In this way, the clients are invisible to the rest of the Internet. On the other hand, *peer-to-peer architectures* can be used to build distributed systems without any distinction between clients and servers. Special methods are required to maintain consistency of resources and synchronize events when necessary.

Transparency is the term used to describe the goal of hiding the fact that a system is divided into parts, enabling the user (and application engineers) to view the system as a single component. Coulouris, Dollimore, & Kindberg (2001) describe eight forms of transparency based on: access, location, concurrency, replication, failure, mobility, performance, and scaling. For example, access transparency means that identical operations are used to access local as well as remote resources.

In this report, a distributed system is a computing system where the resources are spatially distributed and connected by some kind of network. This is in contrast with a centralized system where the resources are gathered in a single location. In a *truly* distributed system, the distribution of the resources is invisible (transparent) to the user. The definition of a distributed system clearly includes both hardware and software.

Sometimes only truly distributed systems are accepted as distributed systems and non-transparent systems are merely considered to be resources connected by a network. However, in such a case applications transparently distributed over the resources would be considered a distributed system. Thus, distributed and non-distributed systems would co-exist on the same network and resources.

An important aspect of distributed systems design is that in practice new systems are not self-contained but rather has to incorporate or share infrastructure and legacy systems. This is probably one reason why most development is targeted towards the software parts of these systems (Akehurst & Waters, 1999). One approach to handle the complexity and diversity of widely distributed information systems is to employ evolutionary system development.

The important and combined problem area of distributed systems and security is discussed in Chapter 3.

2.4 Security Architecture

Security is a continuous process and the security architecture of a system should support this process by describing (1) the security mechanisms and components included in the system and (2) the interaction between the security components. The security architecture is the part of the system architecture that is being used to enforce the security policy. What parts of the system architecture to be regarded is determined by (1) which security functions the system has or should have and (2) the current system architecture. These two factors together decide the extent of the security architecture. Consequently, two systems can have vastly different security architectures, although they implement the same security requirements.

Although the description of security architectures above may seem to imply technical issues, this is not necessarily the case. Like the IT defense abilities described in Section 2.2, there are several aspects of security architectures. Most notable are those relating to organizations, users, and technology. Since the technical aspect of IT security is the topic of this report, a strictly technical architecture is addressed. Thus, there are other aspects that have to be considered in order to cover all the issues related to the securability of systems.

2.5 Related Work

Because of the broad scope of the problem, there exists a vast amount of relevant literature. We have chosen to focus on

- two often cited general textbooks on computer security regarding the concept of security in distributed systems,

- literature on evaluation, information assurance and related problems
- a number of recent scientific articles regarding different system modeling approaches.

2.5.1 Distributed Systems and Security

Regarding distributed systems and security, we swiftly present the main views of Gollmann (1999) and Anderson (2001). These views are not directly addressed in the rest of the report, but being general and important observations they still influence choices and analysis in the report.

Gollmann's view

Gollmann (1999) states that a fundamental dilemma of computer security is that "security-unaware users have specific security requirements but usually no security-expertise". According to this argument the issue of security evaluation makes this dilemma clearly visible. To be able to evaluate whether a product delivers specified security services, security functions have to be stated, understood and assured. This is no easy task for the security-unaware user, and unfortunately there are no easy solutions to resolve the fundamental dilemma.

The dilemma is in now way any easier in a distributed environment. A security problem fairly understood in a smaller environment often becomes blurred in a distributed environment. An example stated by Gollmann, is authentication by passwords. In a situation with terminals connected by a fixed link to a host, you may have reasonable control of eavesdropping on passwords and other security problems. Thereby passwords may be a reasonable solution. In a distributed system this assumption will though hardly hold.

This raises, as discussed by Gollmann (1999), questions of security policies. Since users are not registered at the node where they are accessing an object, we have to decide how to authenticate users and what to base access control on (for example user identity, network address the user operates from, or the distributed service invoked). After having sorted out policy issues, issues of where to enforce authentication and access control emerges. We cannot expect an agreement between different operating systems on a common standard for security enforcement. Therefore any uniform security interface to users must be above the operating system layer. The described situation regarding operating systems also indicates the need of an interface to the operating system layer for application writers. Therefore, in the view of Gollmann and as illustrated in Figure 2, the service layer, between the operating system and the applications layer, is the most appropriate location for security enforcement in a distributed system.

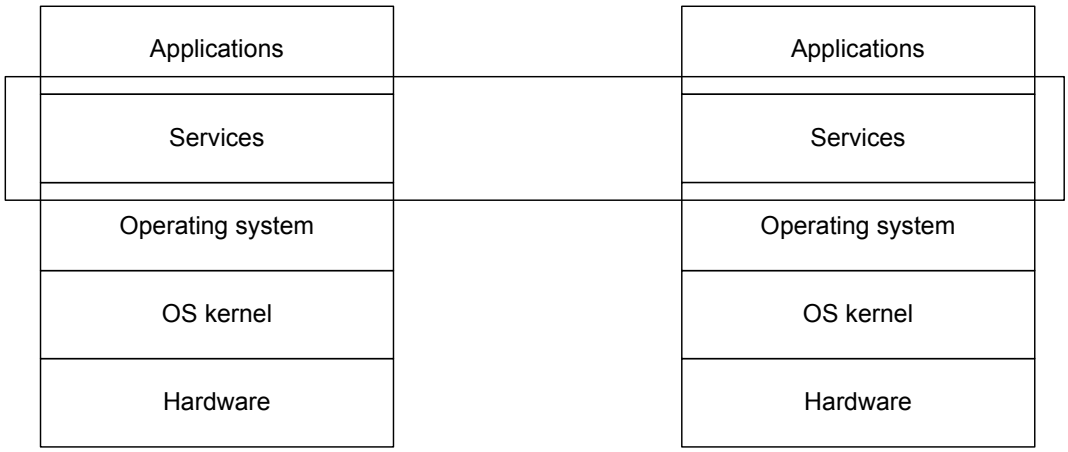

Figure 2: Appropriate location for security enforcement in a distributed system.

Anderson's view

Of special interest to security in distributed systems, according to Anderson (2001), are questions regarding concurrency, failure recovery and naming. A common observation to be made is that trivial problems in local networks may become major issues in large distributed systems. The scale-up problems may not be linear when the systems become larger. Regarding concurrency, failure recovery and naming, these growing problems are of crucial interest.

Processes running at the same time are said to be *concurrent*. The problems emerging from this are often discussed in literature on concurrent processes and concurrent programming in operating systems, but concurrency is also an issue of security. Concurrency control should prevent users from interfering with each other, accidentally or on purpose. This may occur for example by processes using old data, inconsistent updates, deadlocked processes (or system) or data in different systems not converging to consistent values. Timing issues are also vital when handling concurrent processes.

Availability issues tend to be neglected in computer security research according to Anderson (2001). Confidentiality and integrity issues have been given far more focus. Typical expenditures are though the other way round, with often higher costs for availability and recovery mechanisms, than for integrity and confidentiality mechanisms. Fault detection, error recovery and *failure recovery* have to be built into distributed systems to make them resilient to availability related problems, such as denial of service attacks. Introducing redundancy is one common strategy of achieving resilient systems.

With the emerging use of public key certificates the problem of *naming* becomes central. Anderson (2001) makes for example the observation that names used outside information systems are normally not sufficient within

them. For each name there are normally a number of people with the same name. Within systems one individual might be known by several names. As Anderson states: “Names exist in contexts, and naming the principals in secure systems is becoming ever more important and difficult”.

2.5.2 The Common Criteria

To be able to perform evaluations of various IT products and systems and to make the comparison easier, the Common Criteria (CC) for Information Technology Security Evaluation (CC, 1999) has been established. These evaluations are meant to aid companies and single consumers to choose the product or system best suited for their purposes.

CC security requirements are ordered into a hierarchy where classes contain families and families contain components. There are two sets of classes: functionality and assurance classes.

Kruger & Eloff (1997) used CC as a framework in order to find functional security requirements of IT systems. They introduce an evaluation process with three steps. The steps include determining the applicable security functions, choosing some of the functions, and comparing the chosen functions with those already existing. Thus, the result is a set of security functions specified in the CC.

2.5.3 System Modeling Approaches

To obtain an impression of the state of art regarding system modeling a set of recent scientific articles were evaluated. For the evaluation the following criteria were used:

- What kind of system is targeted?
- What is the basis of the analysis (method)?
- Which system characteristics are considered to be important for the security?
- Which steps of the design process are included in the analysis (method)?
- How are the systems modeled?

During the literature study a number of articles were divided into five broad categories of relevance to IT security and especially to design of security architectures. Those categories are: policy modeling, attack modeling, structural modeling, layer-based modeling, and security-indicators modeling. In general, the need of sound design of security architectures has been observed and even what interests ought to influence the design (policy interests, understanding of the attackers' way of reasoning and acting etc).

The vital need of knowledge of what security relevant parameters (or characteristics) to adjust and study is far more restricted. Thus, there is a lack of focus on precisely which security characteristics are needed in the forthcoming design of security architectures.

The shortcomings, considering the effort to model systems and to find tangible security characteristics, of the five categories are:

- Policy modeling. Attempts at implementing policy models mostly stops at a conceptual level far more abstract than the modeling techniques and security characteristics needed in this work.
- Attack modeling. Attack modeling tends to focus on categorizing types of attacks and to some degree on what level in a system the attack might be stopped. This results in the attacks being modeled and analyzed, but mostly not the information systems. In some approaches, the systems are captured, but at an abstract level.
- Structural modeling. By mainly focusing on modeling of systems and their structure as a whole, both the models and the security characteristics are described on a high level of abstraction.
- Layer-based modeling. This category of studies seems promising, with its more component-based “plug-in”-approach. So far though, security characteristics are usually described only indirect. There is also a need to improve the modeling of the whole system, that is, to include all the layers.
- Security indicators modeling. The complexity of designing and choosing security characteristics is reflected by the past attempts in this category only being partially successful.

Studies of special interest

Below some previous studies of interest are described. This includes studies categorized according to the five categories mentioned above, but also a few other more general approaches are discussed.

Policy modeling

An example of a policy modeling study is Burns, Cheng, Gurung, Rajagopalan, Rao, Rosenbluth, Surendran & Martin (2001) that aims at automatic network management, which upholds the stated policy while the system changes. Addressed challenges are methods for evaluation and supervision of security properties, tools for management of system nodes (firewalls, servers, etc), and automatic reconfiguration of systems. However, the level of abstraction in this study makes it complicated to sufficiently address the problem area of choosing security characteristics.

Attack modeling

Jonsson & Olovsson (1997) study attacker behavior and how this may be reflected in a quantitative model of the intrusion process. Apostal, Foote-Lennox, Markham, Down, Lu & O'Brien (2001) models information systems and simulates attacks or defenses. It “analyzes the vulnerability of a network based on its current configuration”. Adversary characteristics are collected in a table.

Both these studies, based on attack modeling, represent an approach which by further studies and development may yield valuable information regarding choices of security characteristics of importance to the design of security architectures.

Structural modeling

In structural modeling studies, attempts are made to model information systems and their structure as a whole.

Eschelbeck (2000) implements a security infrastructure where multiple components including intrusion detection systems, vulnerability assessment scanners, firewalls and other security devices are all to communicate and respond to changing security threats. Although, Eschelbeck (2000) describes efficient methods to structure security systems, the input on how to choose security characteristics is scarce.

Whitmore (2001) models networked information systems as a structure of security subsystems and their means of interaction. Important system characteristics are described on a rather high level of abstraction.

Layer-based modeling

In layer-based modeling studies, information systems are built of separately designed layers as “plug-in”-components.

Olivier (2001) describes an architecture for security systems that allows security components to be developed separately from the system to be protected and then integrated into the system at a later stage. The model supports customization. The problem area of choosing security characteristics is mainly addressed indirectly.

Security indicators modeling

ACSA (2002) addresses directly the problem area of choosing security characteristics, but immediately observes the vast depth and width of the area. Among the conclusions of ACSA (2002) some interesting observations are:

- No single measure will successfully quantify the assurance present in a system. Multiple measures will be needed and they will need to be refreshed frequently.
- Quality of software, architectures and designs chosen, tools used to build systems and requirements specified are of great importance.
- Past attempts, among them the Common Criteria, have to a large degree not been successful.
- Processes, procedures, tools and people all interact to produce assurance. Security measures incorporating these aspects remain critical.

3. Design for Securability

Realizing the fact that no distributed information system can be designed secure, but can include the necessary prerequisites to be secured during operation; the aim is *design for securability*. The characteristics aspired of a system vary from case to case and are influenced by factors such as time, cost, throughput and risks. In this section, the design process is divided into the two parts requirements extraction and requirements implementation. The topic of the report results in an emphasis on the latter part.

3.1 Requirements Extraction

As indicated above, it is vital to be able to identify the aspired characteristics of a system. Thus, requirements engineering, as described in (Sommerville & Sawyer, 1997), becomes an important tool for the security engineer. There are other reasons to use requirements engineering in the system development process. The main benefit is the ability to decide which system functions are required and, thus, decrease the total number of functions and number of flaws in the system.

To begin with, the interactions and relations between the system and its environment have to be captured. This usually results in a complex structure, as illustrated by Figure 3, in which trust is a central component. Trust relies on operations performed by operators, by information systems and on operations performed within an organizational context.

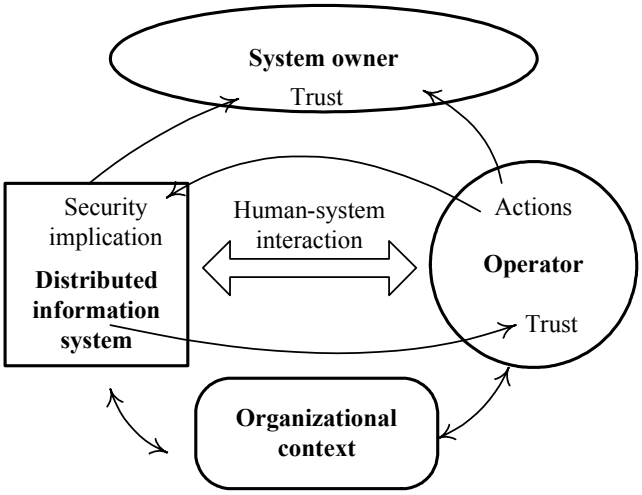

Figure 3: The relations between a system, the organization, operators and the system owner.

At the top level, the system owner's trust in the system relies on the performance of the information system and on different actions taken by opera-

tors. The operator's trust is more directly related to the performance of the information system and especially the way the system's performance is experienced through the human-system interaction. Actions taken by an operator has security implications within the information system and the way this makes the system perform influences the operator's trust or possibly lack of trust.

The actions taken by the operator and the functions of the information system is set within an organizational context, which also has an impact on trust. As an example, a policy regarding backup of data is worthless, if you have no routines to implement the policy.

When the interactions and relations between the system and its environment have been captured, a set of requirements on the system has to be formulated. From this set security-relevant requirements can be extracted, as illustrated by Figure 4. Starting with a textual description of the system requirements, the general system requirements are refined into statements concerning security. These are thereafter validated and checked for consistency. Through this process of requirements engineering, security related issues are integrated at an early stage of the system development. This is in contrast to what often happens with add-on security at a late stage, perhaps even after the rest of the system development is over.

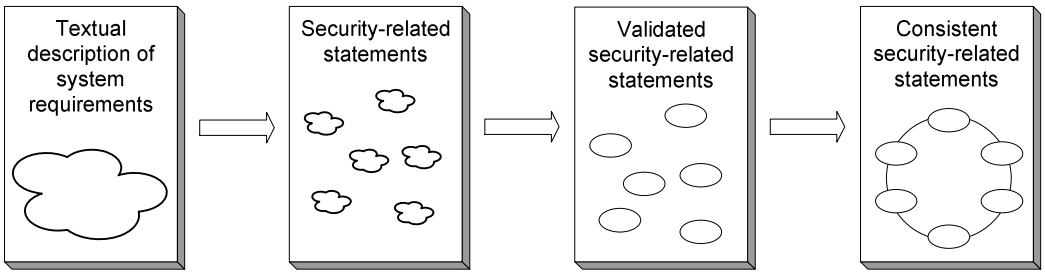

Figure 4: The process of formulating a set of consistent security related statements from a textual description of system requirements.

3.2 Requirements Implementation

When captured, the set of requirements has to be implemented in the system. This is a complex process that has to extend throughout the lifetime of the system. Ideally, there would be a well-formulated process extending from the set of requirements to the implemented system, and also facilitating and enhancing risk management of the implemented system. This leads to the idea of an implementation of such a process, where automatic design tools support the execution. However, this demands, on top of the task to design an efficient security architecture, the solution of all traditional system development issues. Therefore, at this point, the formulation of a framework for design and evaluation of security architectures, based on a system implemented at some level of abstraction, would be a great step

forward. Such a framework has to be based on the ability to efficiently model the studied systems.

Figure 5 illustrates the concept of a framework for assessment and modification of system descriptions. System requirements, high-level descriptions, and component descriptions are used to build system models. The resulting system models are analyzed and modified using design methods and tools. Finally, the result is fed back to the system descriptions and requirements. The number of ways this process can be performed with a mix of manual work and automatic tools is infinite. However, even assuming all analysis, modifications, and feedback to be manual, a systematic design process facilitated by system models would enable security engineers to validate the requirements specified for the system.

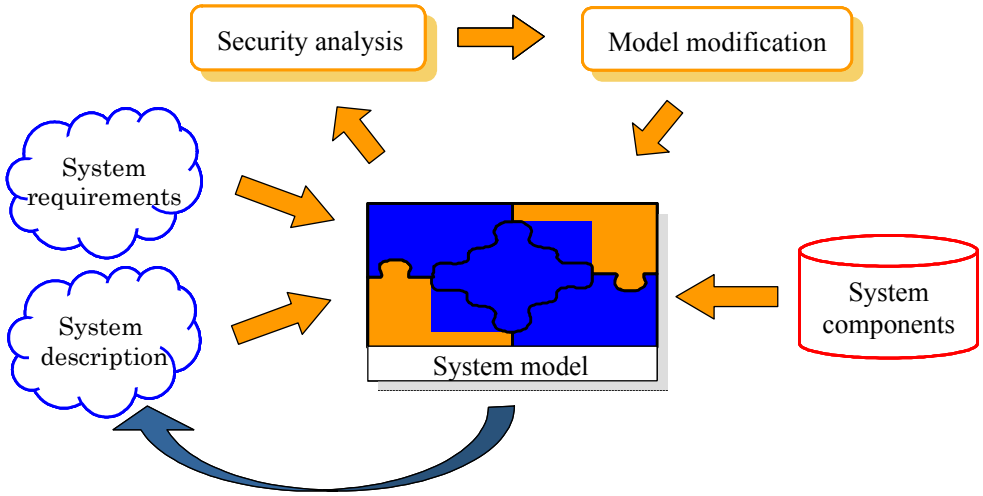

Figure 5: A framework supporting design for securability.

In essence, the proposed framework for security architecture design enables:

- the gap between specified security requirements and the system implementation to be closed,
- high-quality input to risk analysis methods,
- the application of engineering principles to security architecture design,
- increased comprehensibility of the design and design issues, and
- the possibility to explore and assess different possible solutions.

A concrete goal for the research in the area of security architecture design is to develop methods and tools aiding the system designer with simulations etc in the design loop. Such tools and methods would enable the designer to explore a larger part of the design solution space, and, thus increase the chance of finding feasible solutions. This is extremely relevant in the framework of evolutionary system development. A first step towards such design methods and tools is the formulation of adequate modeling techniques. Such

modeling techniques are fundamental for all methods and tools that can be developed to support the process of designing the security architecture.

The models have to capture all the information required for the designer, aided by tools, to be able to comprehend and analyze the current design problem. The choice of modeling techniques to represent system structure is crucial. A number of general requirements on such modeling techniques are:

- The modeling technique must be able to capture appropriate descriptions of system characteristics to facilitate further system development.
- The complexity, which may result from the requirement above, must be balanced by the modeling technique to facilitate simplicity and understanding. Thereby development of more secure systems will be supported.
- To support efficient automatic tools modifying the models, quantitative measures of computer security are necessary. The interaction with the concrete system level relies heavily on such measures.

A proposed modeling technique adequate for the security architecture design framework is formulated and described in Chapter 5.

4. Security-Related Systems Characteristics

The purpose of a set of security-relevant characteristics is to support the assessment of the security level of a system, the formulation of adequate modeling techniques, and the requirements engineering process. To formulate a set of security-relevant systems characteristics, an initial study was performed, as described in Section 4.1 and Appendix A. The result of the study was improved by an additional cross-check, also described in Section 4.1 and Appendix B. The resulting set of characteristics is presented in Section 4.2.

4.1 Initial Study and Cross-Check

To find a set of quantitative security measures a literature study, a structured analysis, a brainstorm session, and a cross-check were performed. The results of the literature study formed the initial input to the structured analysis, although it was continued through the whole study to find additional input. The brainstorm session was performed to gather more input to the structured analysis. Finally, the cross-check was performed as an extension of the structured analysis, which basically lasted during the whole study. The four tasks of the study are described in detail in Appendix B.

The master thesis described in (Stjerneby, 2002) details the main part of the initial study. The cross-check was mainly performed parallel to and after the master thesis work.

4.2 A Set of Security-Related Characteristics and Its Structure

The result of the study and the cross-check is a structuring of security-related characteristics with 60 distinct characteristics. The found characteristics are listed alphabetically below and the structure is depicted in Figure 6. In section 4.2.1 to 4.2.8, the most important sub-trees of the structure are discussed. A short description of during which task the different security characteristics were detected can be found in Appendix B. Individual characteristics are not defined in this report. These can though be found in an appendix to (Stjerneby, 2002).

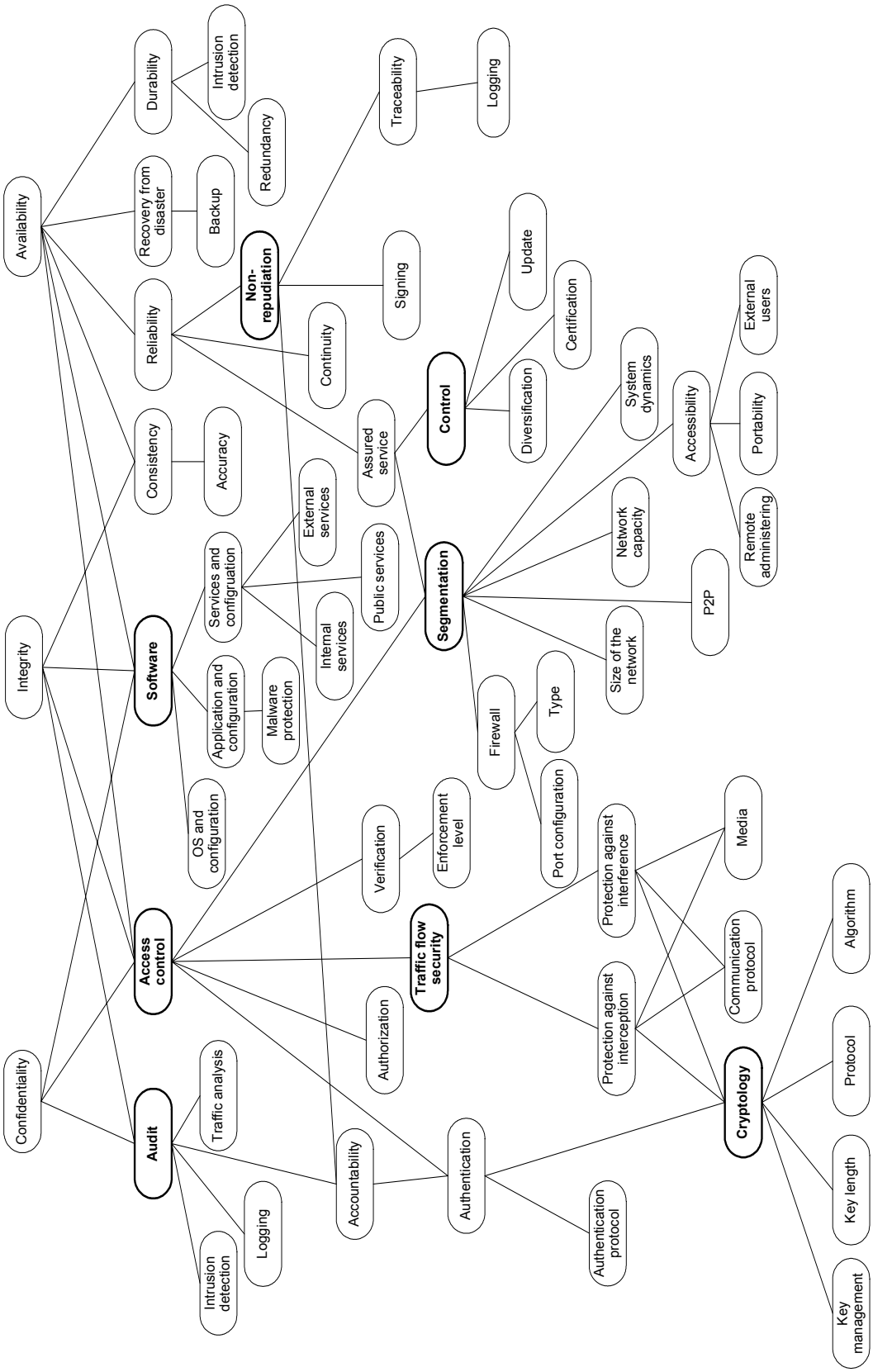

Figure 6: The characteristics formed as a tree structure. Highlighted nodes are roots of the sub-trees discussed in sections 4.2.1 to 4.2.8.

4.2.1 Cryptology

Cryptology is a central tool in IT security and emerges in several places of the structure. The level of security enforced by cryptology algorithms should be quantifiable. Unfortunately, the main factors for the security level results from the quality of the implementation, the protocol, and other peripheral factors. For example, public key infrastructure and key exchange are important issues to be addressed. Main areas of cryptology are; key management, key length, protocol, and algorithm.

4.2.2 Audit

To be able to trace activities that have occurred in the system, audit is needed. Audit includes: intrusion detection, logging, analysis of the traffic in the network, and accountability. Accountability in turn is closely related to authentication. Authentication relies on cryptology and authentication protocols, for example biometric user authentication and zero-knowledge protocols for device authentication.

4.2.3 Access control

Access control is fundamental to all three roots of the tree and can be achieved in a number of ways. Traffic flow security and segmentation is explained separately below. Authorization, that is to be able to regulate who can access what, is a fundamental mechanism in access control. Authentication, relying on cryptology and protocol, is also a building block. Verification that the authenticated entity is allowed to perform the operation is necessary and its efficiency depends on the enforcement level, that is, at what system level, e.g. OS or application, the verification is performed.

4.2.4 Traffic flow security

The characteristic traffic flow security is made up of two other characteristics. Protection against interference can be said to be another way of describing availability and integrity. Meanwhile, protection against interception can in a similar way be regarded to be the same as confidentiality. Important security characteristics to obtain traffic flow security are cryptology, choice/design of communication protocol and choice of media.

4.2.5 Software

Software is one of the two second-level characteristics, besides access control, that is important to all the three roots of the tree. The choice of software, e.g. OS, and the effort to configure and continually update it is extremely important to uphold the security level of a system. Applications

featuring protection against malware is an example of software that is crucial to update. Software providing services is another vital area since they possibly offer external and public as well as internal services. Internal services are reachable only for users inside the realm of control; public services are offered to entities outside the realm of control, e.g. web sites; and external services are offered to users both inside and outside the realm of control, e.g. web mail.

4.2.6 Segmentation

Segmentation of a system is important in order to decrease the exposure to potential threats. Thus, the size and capacity of the network become an issue. Peer-to-peer connections are important to scrutinize since the separation of applications and services is blurred. Mobile units, integration and collaboration between systems etc. result in increasingly dynamic systems. This causes problems in keeping track of who was connected at what time and did what. There is an increasing demand of accessibility, for example to be able to remotely administrate and externally access services. This has a large impact on the security of systems since units outside the realm of control are effectively becoming part of the systems, e.g. home PC used to read corporate emails. Firewalls are used to scan and restrict the connections to and from the outside. The type of firewall and the port configuration have a large impact on the security of the firewall.

4.2.7 Control

The control sub-tree considers the control system administrators and users have over the system. Knowledge and ability to handle the system and its weaknesses results in control and is necessary to be able to maintain a specified level of security. Important issues are updates, certification of system components, and system diversification (e.g. if there are any strengths or weaknesses resulting from use of different versions and kinds of software).

4.2.8 Non-repudiation

Non-repudiation concerns not being able to deny having received or sent messages. To enforce this, important characteristics are: signing (e.g. of messages), accountability, and traceability. Accountability relies on authentication, which is made up of cryptology and protocol. Traceability depends on the logging of relevant events.

4.3 Discussion

The structuring of the characteristics starts from a set of three top characteristics, the well known CIA, that is confidentiality, integrity, and availability, which are the three main characteristics of IT security. These are divided into a second layer that consists of high-level mechanisms[1] (e.g. access control and audit) and characteristics more specific than the three at the top (e.g. consistency and reliability). Generally the refinement continues with the descendant in the tree, via tools to attributes. In some cases, the refinement of a mechanism or tool results in a general characteristic (e.g. accountability under audit). This characteristic is then further refined.

Regarding the sub-trees of the three roots, confidentiality and integrity have no sub-trees specific to that characteristic. This illustrates the close relationship between these two roots and the fact that system characteristics required to achieve one of them generally are advantageous for both. On the other hand, availability has three specific second-level characteristics, which illustrates the difference between this and the other two roots.

Since the characteristics become, in general, more refined in deeper levels of the tree, the graph could be divided into different levels of abstraction. Thus, different levels of the tree can be extracted depending on the topic. For example, discussing the meaning of the three top characteristics, that is CIA, the first and second level of the tree may be sufficient. On the other hand, regarding more specific topics, such as authentication, only sub-trees need to be studied.

In a fully developed tree, all the leaves should be attributes that can be assigned a value. However, this has not been fully accomplished within the scope of the study. Thus, further refinement of characteristics is necessary to reach the goal of quantifiable system attributes. Moreover, the tree is not a final product considering the high and middle levels either. There are certainly more characteristics to be identified and introduced in the tree. Actually, the development of the tree should be an open process where some branches grow while others are cut off to accommodate the dynamics of system development and IT security.

Although the current set of characteristics cannot be used for quantitative evaluations of systems, it yields valuable support for qualitative reasoning about systems and their security architectures. Furthermore, the identified characteristics are essential for the effort of formulating modeling techniques enabling the modeling of systems focusing on security aspects. Finally, an elaborated set of security-related characteristics can be used to identify important aspects of systems to be considered and evaluated when

[1] Whenever mechanisms or tools are mentioned as nodes in the tree, the node is actually representing a characteristic being the quality of the mechanism or tool.

formulating the security requirements of systems and transforming them into a policy.

5. Modeling of Distributed System

The ability to model distributed information systems is essential for designers and design tools to be able to comprehend and assess the corresponding systems and design decisions. Thus, an efficient modeling technique is a prerequisite for the design of securable distributed information systems, as discussed in Chapter 3. As illustrated by Figure 7, a system model has to capture both the requirements specified for a system and its characteristics (from high-level descriptions and component models).

The overall purpose of the models is to enable improvements of the security in distributed information systems. There are several dimensions in the space of possible modeling techniques that can be used to describe the purpose of the resulting models and, thus, indicate aspired characteristics of the modeling technique. Such dimensions (issues) are:

- **Model user**. There are basically two categories of model users: humans and machines. Whether the models will be consulted by designers, design tools, or both has a large impact on the appropriateness of different modeling techniques.
- **Model use**. Different contexts of use result in different demands on the models. For example, the models can be used for general reasoning about the system, to collect statistical data, or to calculate detailed measures for assessment of system properties.
- **System development status**. The modeled system can be an existing system or a system at various states of development.
- **System scope**. An important issue in modeling of distributed systems is to define the range of the system. Thus, there is a need to define what is incorporated in the system, and what is external to the system.
- **System descriptions**. There are numerous ways to extract the information of a system necessary to build a system model. The number of available alternatives to a large extent depends on to which extent the corresponding system is currently described and implemented.
- **Completeness of model**. The system models can describe the corresponding systems at different levels of accuracy. Accordingly the modeling technique can demand completeness or allow the modeler to judge which parts of the system to be modeled.

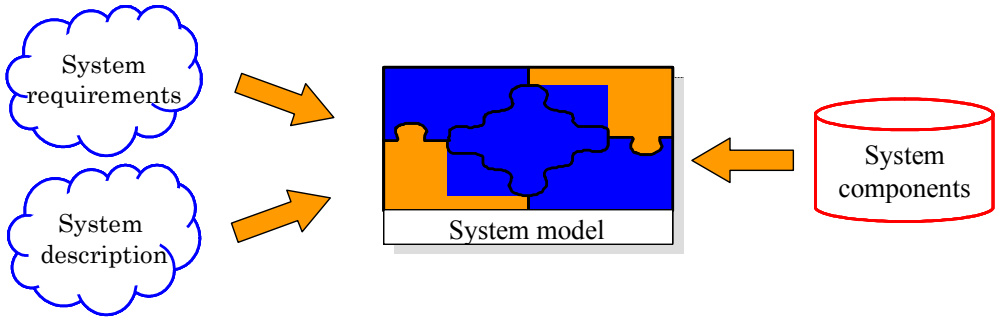

Figure 7: A system model has to capture both the requirements put on a system and its characteristics (from high-level descriptions and component models).

5.1 Characteristics of the Modeling Technique

Considering the essential characteristics of a model listed above this section contains discussions and design decisions regarding the modeling technique.

5.1.1 Model User

The models should be both human and machine readable. Essential characteristics of any model are the ability to capture all (or a sufficient amount of) relevant characteristics and to be comprehensible. Naturally, the amount of details captured by a model decides its usefulness in different contexts. For example, the ability of security engineers to study models and draw conclusion about the properties of the corresponding system is hampered by an overflow of detailed and irrelevant information. On the other hand, automatic analysis tools performing advanced calculations to deliver quantitative measures of the security level of a system require the model to capture all the necessary input data. This assumes that the comprehensibility of a model decreases with the amount of information captured. An obvious approach to solve this problem of context dependency is to introduce hierarchy in the models. In this way, the models can be used throughout the system development process absorbing more information when available.

Design implication: The modeling technique should support hierarchy in order to be both human and machine readable. Moreover, mechanisms for separation of information into several diagrams, in order to increase the simplicity of each diagram, should be supported.

5.1.2 Model Use

The modeling technique should support the design of a framework enabling design for securability. Initially, the framework will be based on reasoning based on the system models and, thus, will not require extensively detailed models. However, as the framework evolves automatic design tools will be

incorporated and consequently the level of details demanded from the models will increase. This is essential in order to be able to efficiently validate or assess system requirements and alternative implementations.

Another important aspect is whether the resulting models are end-products or if the information extracted from the models will in turn have a direct influence on the actual system. Here, end-product means that the information extracted from the models will not directly influence the system implementation. This is the case when the models are used for system assessment in order to assure that certain requirements are met by the current design or implementation. Hopefully, the models and the reasoning based on the models eventually will influence the system development or operation; otherwise their use will be limited. On the other hand, if the models are used in a framework enabling the system specification to be altered, as illustrated by Figure 5, traceability from the models back to the system specifications (and from the updated system specification to the models) has to be supported.

Design implication: The modeling technique should allow dynamic incorporation of additional system characteristics. Moreover, traceability between the models and corresponding system specification has to be supported.

5.1.3 System Development Status

The purpose of system models is to create a notion where system requirements and system characteristics meet, either before the system actually has been implemented or afterwards for a present system. Thus, depending on the purpose of the models they can be referred to as design models (for a future system) and analysis models (for a present system) respectively. Naturally, the status of the modeled system greatly influences the amount of information available when building a model, the mechanisms that can be used to extract the information, and the purpose of the models. Thus, this issue is closely related to both *model use* and *system specifications*. Handling different system development statuses requires a flexible modeling technique able to capture systems at different stages during the development process (from initial sketches to final implementation). Further, the ability to model a system will not depend only on the adequateness of the modeling technique, but also on the specifications and tools available to extract relevant information.

Design implication: The modeling technique should be able to capture systems at different levels of abstraction (development stages). The usefulness of the modeling technique depends on the availability of efficient methods and tools to extract the information needed to build an adequate system model.

5.1.4 System Scope

All systems have interfaces. These interfaces have to be well specified and their location will determine the size and range of the system. The extent of the system in the dimension of technical components, users, and organization has a great influence on the required modeling techniques. For example, if users are included in the model, the aspects of human-system interaction and user cognition and behavior have to be captured. These aspects influence the security of the system as a whole.

Considering physical extent an important issue is whether the modeled system is defined by the domain of control (of an organization) or by provided services. Consider the structure depicted by Figure 8. If organization A wants to assess the IT security of their information system, the model should capture relevant information considering system A. On the other hand, if the purpose is to assess the IT security of a distributed information system providing a set of services, the model should be based on the system components utilizing and providing these services. In this case, the model may span multiple organizations (A and B) and use the Internet for intra-system communication.

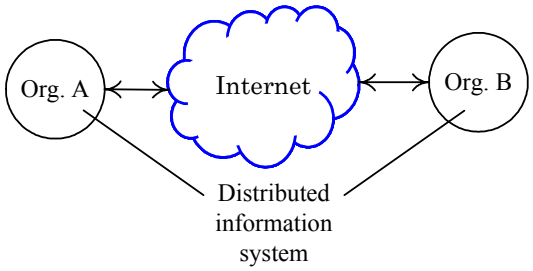

Figure 8: Systems controlled by different organizations can be interconnected via the Internet.

Design implication: The modeling technique should include mechanisms to capture human-system interaction and user cognition and behavior. This requires the ability to describe multiple system aspects, in order to capture all the security-relevant system characteristics. The modeling technique should handle system whose physical extent is decided by either domain of control or by provided services.

5.1.5 System Descriptions

The amount and character of available system descriptions varies greatly from system to system (and with time, as discussed above regarding *System development status*). Consequently, methods and tools for extraction of system characteristics need to be formulated for high-level descriptions of the system (at various abstraction levels) and models of system components, as illustrated by Figure 6, and from actual systems. Acknowledging this

problem, a general issue is how this affects (or should affect) the modeling technique. An important observation is that the use of a wide-spread and well-known modeling language will increase the possibility and decrease the effort of developing the needed tools.

Design implication: The modeling technique should be based on a standard modeling language.

5.1.6 Completeness of Model

An important aspect of system models is their completeness according to specified criteria. For example, a model used to assess the correct location (considering security, not function) of all computers in distributed systems has to capture the location of all computers to be useful. Naturally, many completeness criteria are impossible to enforce within the modeling technique. However, the modeling technique can demand the explicit representation of various system characteristics, such as the number and nature of external interfaces or the type and minimal number of network links needed to connect all the devices in the system.

However, there is a trade-off between flexibility and the ability to enforce completeness. Therefore, completeness enforcing mechanisms are not considered to be central for the modeling technique but are instead to be handled by design tools.

Design implication: The completeness of system models depends on the modeler, who can be supported by design tools to validate the completeness of a system model.

5.1.7 Discussion

In section 5.1.1 to 5.1.6, the following aspired characteristics of a modeling technique were discussed. In short, the modeling technique should

- support hierarchy, in order to be both human and machine readable,
- support mechanisms for separation of information into several diagrams, in order to increase the simplicity of each diagram,
- allow dynamic incorporation of additional system characteristics,
- support traceability between the models and corresponding system specification,
- be able to capture systems at different levels of abstraction (development stages),
- include mechanisms to capture human-system interaction and user cognition and behavior,

- handle system whose physical extent is decided by either domain of control or provided services,
- be based on a standard modeling language, and
- not include mechanisms to enforce the completeness of system models.

Moreover, the usefulness of the modeling technique depends on the availability of efficient methods and tools to extract the information needed to build adequate system models.

The scope of this work is limited to modeling of technical aspects of distributed information systems and, therefore, the requirement on user modeling is not met. Furthermore, the design of methods and tools to extract the information needed to build system models is not included.

5.2 A Modeling Technique

In addition to the general requirements on an adequate modeling technique discussed in the previous section, there are special requirements on models for design of security architectures. The characteristics of an adequate modeling technique are shaped by the security-relevant system characteristics as specified in Section 4.3. Thus, the modeling technique should enable models fulfilling design process requirements and capturing security-relevant system characteristics. Since distributed information systems are complex and involve both soft and hard artifacts, the modeling technique has to capture both the physical and the logical structure of systems.

Naturally, it is difficult to completely model the information needed to assess the securability of a system. Nevertheless, a number of system aspects judged to provide vital information for the assessment have been identified. Consequently, mechanism to capture these system aspects should be provided by an adequate modeling technique. The system aspects are:

- the system *components*, both hardware and software, used to implement the system,
- the system *structure*, that is how the components are connected,
- *services* provided by the system,
- how *information flow* through the system,
- *use cases*, that is how the system is used, and
- the *organization* operating the system.

In Figure 9, an abstraction of the structure of security-related system characteristics, illustrated by Figure 6, is depicted. Each leaf is labeled according to which system aspects are influencing the assessment of the corresponding system characteristic. To simplify the graph, only the roots of

sub-tress with identical labels are shown, hence the abstraction of the original graph.

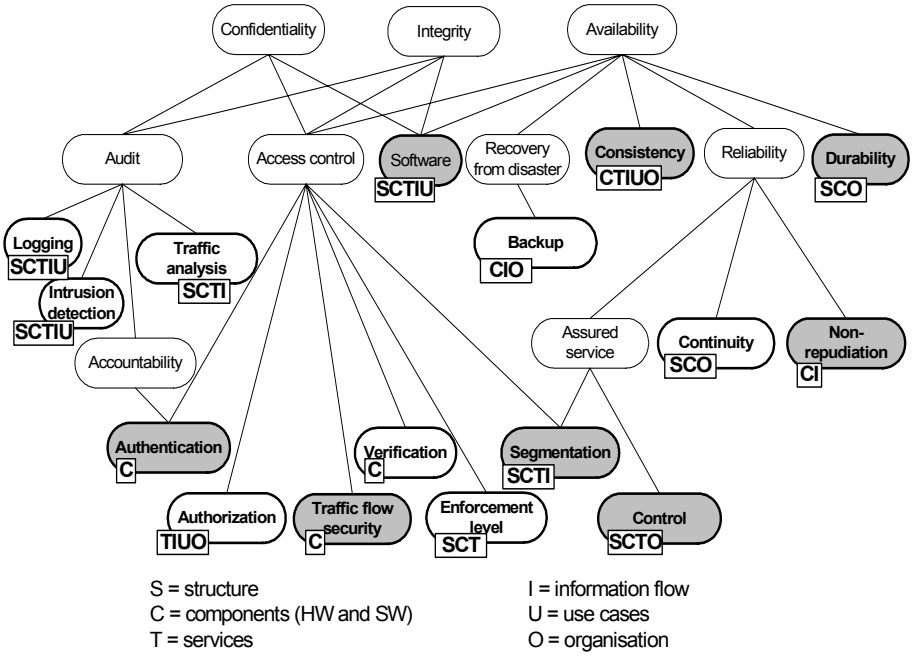

Figure 9: An abstraction of the structure of security-related system characteristics (grey nodes correspond to sub-trees in the full graph) with all leaves labeled according to the corresponding requirements on system models.

From the labeling in Figure 9, it is clear that all six of the system aspects have to be considered in order to fully characterize the securability of a system. However, since the scope of this work is limited to the technical aspects of distributed information system, the system aspects *information flow*, *use cases*, and *organization* are not considered. This should not be interpreted as an attempt to devalue the importance of these system aspects but merely a limitation of the presented work. Accordingly, the proposed modeling technique should be extendable to be able to incorporate these aspects in the future. However, at this point, the task is to formulate a modeling technique for system models capturing the system aspects *structure*, *components*, and *services*.

To capture the system aspects a structural system model is built using the specified system components available in a component library. The modeled system is assessed using the security-related characteristics captured in the structure. The designer and design tools influence the system model thorough the component library, the characteristics structure, and by directly altering the model. This results in a model framework as illustrated by Figure 10. To assess the system the model is interpreted.

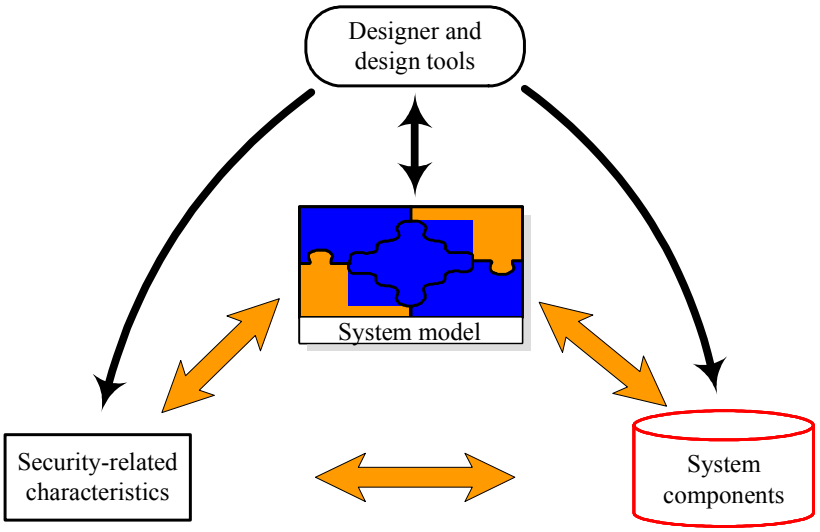

Figure 10: A model framework.

To implement the model framework depicted in Figure 10, a UML-based modeling technique is proposed. UML (Object Management Group, 2001) includes several types of diagrams allowing multiple system aspects to be modeled, including the three aspects identified but not further considered in this work. Moreover, UML supports hierarchy and separation of models into several diagrams. Additional benefits results from the wide use and, thus, knowledge of UML.

Apart from the system aspects the models have to capture the security-related characteristics of a distributed information system. For this purpose, two of the diagrams specified within the UML are used.

Class diagrams are used to capture the different types of components in the system, that is, to build a component library. In fact, the class diagram specifying the module library defines the building blocks of a distributed information system. In this way, the used module library will set the context of the modeling effort and, since it can be altered, allows additional concepts to be included in the models when needed. For example, to introduce hierarchy compound modules, consisting of several other modules, and recursion can be introduced. Moreover, to assess the securability of each system component the structure of security-related characteristics from Section 4.2 is used. Thus, this structure has been captured with a class diagram.

Deployment diagrams are used to capture the structure and configuration of the current system design. Deployment diagrams have been designed to capture the architecture of a system considering physical devices and logical components. In UML, component diagrams are used to model software components of a system. However, since the modeling technique is supposed to capture the system as currently specified, the software components are modeled with the use of components inside the nodes of deployment diagrams. Thus, the services and applications used in the system can be

modeled. Moreover, the security-related characteristics of a component are captured by objects in the corresponding deployment diagram node. These objects are instantiated using the classes capturing the relevant parts of the structure of security-related characteristics.

This approach results in a mechanism to validate that the modeling technique captures the relevant aspects of systems since only a modeling technique resulting in all the parts of the structure of security-related characteristics being instantiated as objects can be adequate. Moreover, the software influence on the securability of a system can be modeled by the instantiation of the corresponding objects in the nodes of the deployment diagram.

5.3 Examples

To illustrate the use of the UML-based modeling technique discussed above, this section contains examples of the class diagrams capturing the component library and the structure of security-related characteristics and a deployment diagram illustrating the modeling of a system.

A component library is illustrated by Figure 11. In this component library, a distributed information system is defined to consist of one or more computers, networks, input/output devices, and distributed information systems. A computer can be a server or an application engine, which executes application software. An input/output device can be a terminal, an output device, or an input device. A (user) terminal mediates interaction between a user and the system via a screen, keyboard, mouse, speakers etc. Combining an application engine with a user terminal results in a PC or workstation. This and other combinations of the server, application engine, and terminal classes have, for clarity, not been introduced in the class diagram depicted by Figure 11, although they are required by the deployment diagrams. A network consists of public networks, network devices, and network links, where public networks are outside the domain of control and network links are the point to point connections of computers, network devices, input/output devices, public networks, and distributed information systems. A number of different network devices are defined, such as wireless access point which mediates traffic between wireless and wired networks. The construction allowing a distributed information system to include distributed information systems enables hierarchies.

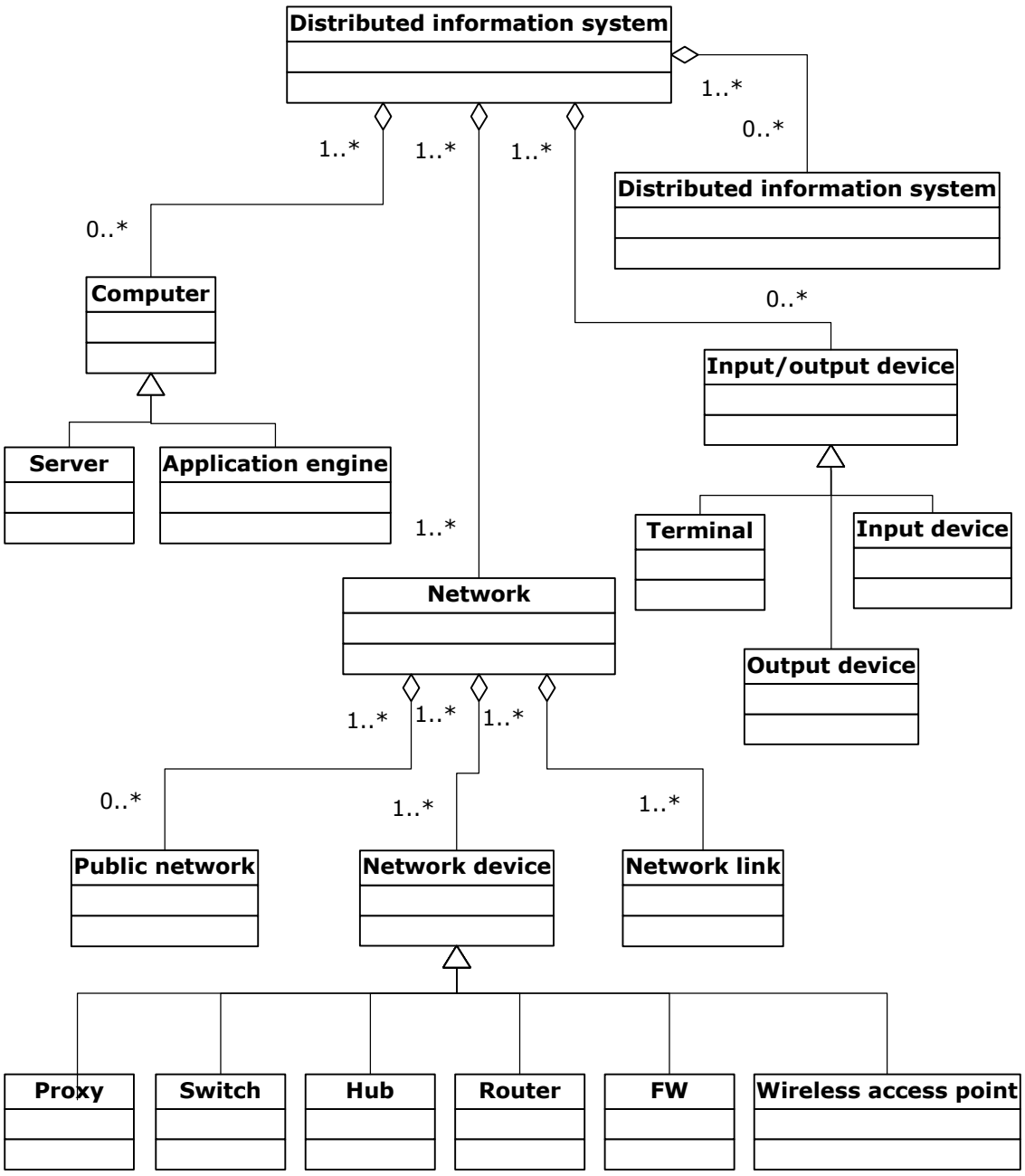

Figure 11: A class diagram capturing a component library.

The software sub-tree of the structure of security-related characteristics is depicted in Figure 12. All three roots of the structure depend on the software characteristic. Here, the term software is used as a short hand for the security characteristics of the software of the modeled system. This holds for the other classes as well. The software consists of three other classes which in turn can be further refined.

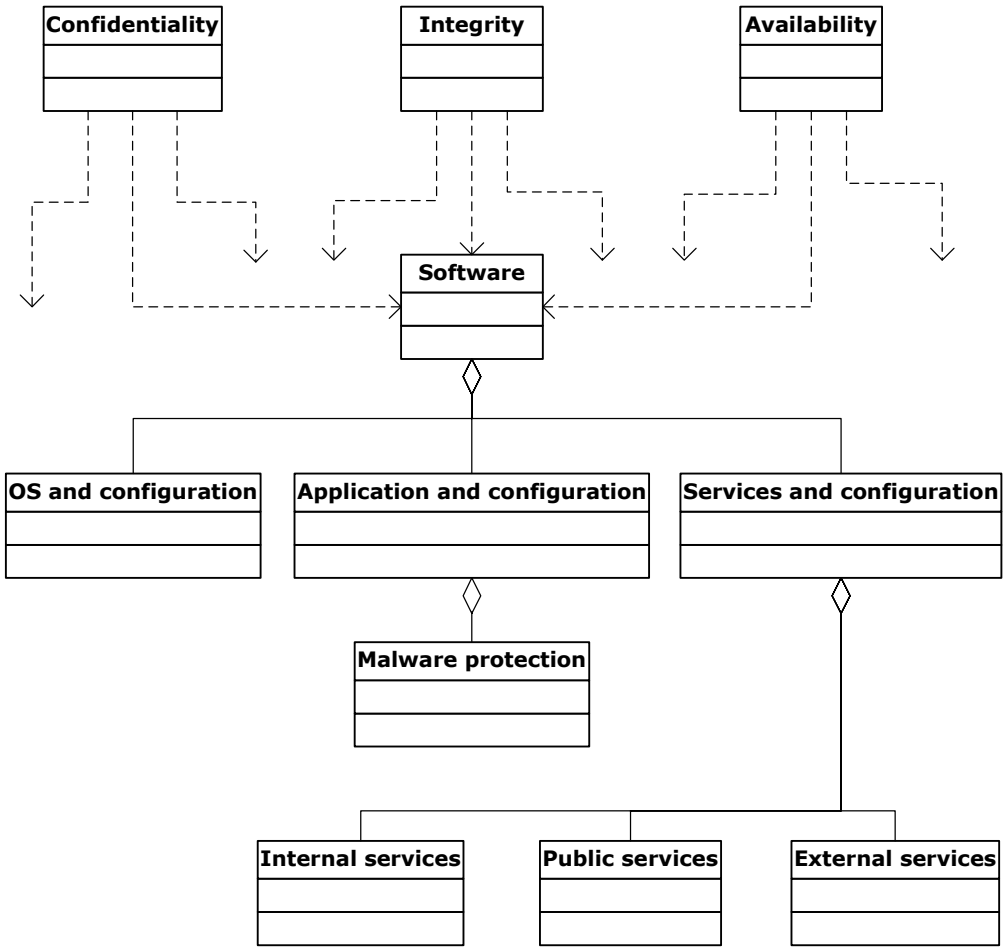

Figure12: Sub-tree of the structure of security-related characteristics.

In Figure 13, a deployment diagram modeling a system is depicted. The network links are modeled by the associations between nodes. Stereotypes are used to specify the type of the network links. Each node can include both components, representing software, and objects representing the security-related characteristics relevant for the node. Thus, the influence of software on the securability of system components and the system as a whole can be captured using either UML components or objects. In general, distributed software is better captured with components, while software more confined to the current node can be captured by instantiation of the corresponding classes in the structure of security-related characteristics. In Figure 13, the PC 1 and DB server node instances have the relevant parts of the characteristics structure instantiated as objects. However, a database system has been modeled using software components resulting in one component residing in each of the two nodes. These components also have the relevant parts of the characteristics structure instantiated as objects, although this is not shown in Figure 13.

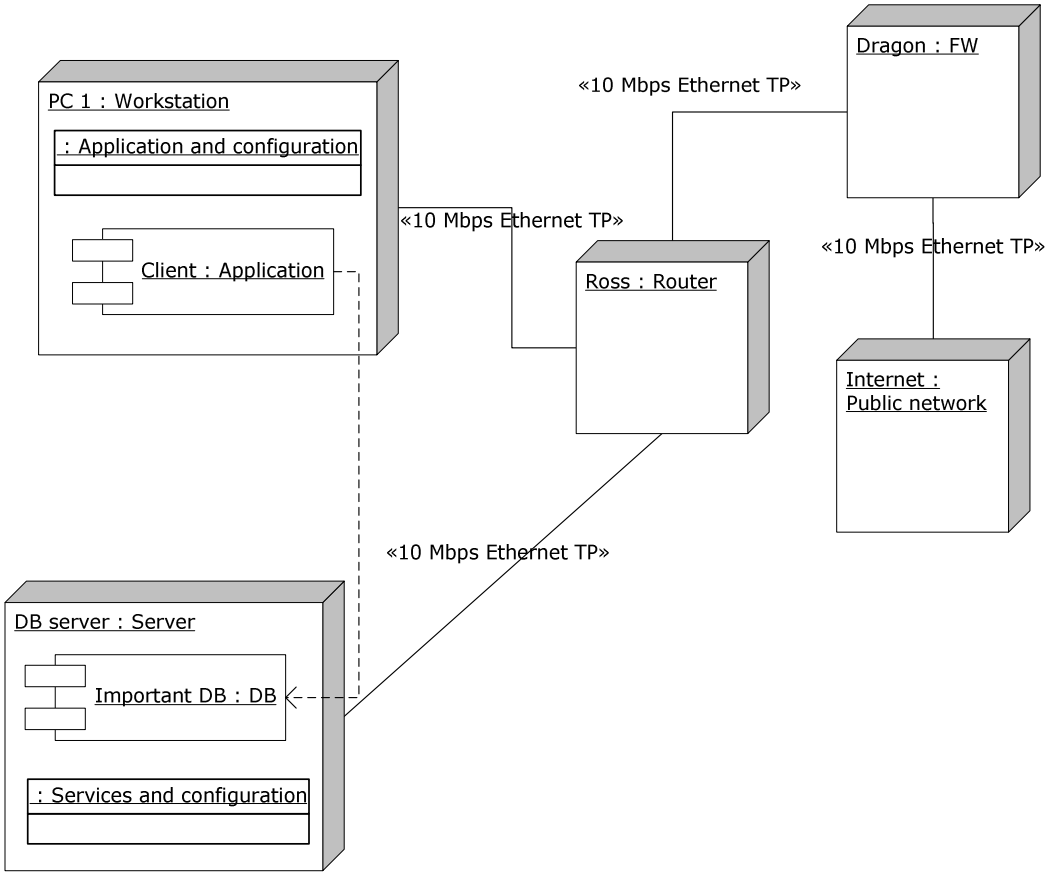

Figure 13: Deployment diagram modeling a system and illustrating the connection to the component library and the characteristics structure.

6. Conclusions

To be able to fulfill the demands on trustworthiness and security in future widely distributed systems; there is a need to formulate methods covering the chain of development steps from the requirements engineering process, via the design of security architectures, to system implementation and operation. Systematic design of security architectures requires powerful modeling techniques and design methods and tools. Since a system cannot be designed secure, security architectures should be designed to support the security process while the system is in operation. Thus, the process is referred to as design for securability.

To support the design for securability process, a design framework, illustrated by Figure 5, is proposed. Within the design framework, different design methods and choices may be reasoned about. The long-term goal is to build a design environment suitable for modeling, simulation and assessment of security architectures. Such a design environment ought to facilitate the construction of trustworthy systems, by realizing system requirements and implementing them into the system.

In essence, the proposed framework for security architecture design enables:

- the gap between specified security requirements and the system implementation to be closed,
- high-quality input to risk analysis methods,
- the application of engineering principles to security architecture design,
- increased comprehensibility of the design and design issues, and
- the possibility to explore and assess different possible solutions.

The framework would incorporate the use of several design methods executed by security engineers with a varying support by automatic design tools. The realization of such a framework requires the formulation of modeling techniques able to capture the security-relevant information needed by the design methods. In turn, this requires the identification of security-relevant system characteristics. These two tasks have been undertaken within the work described in this report.

The purpose of a set of security-relevant characteristics is to support the assessment of the security level of a system, the formulation of adequate modeling techniques, and the requirements engineering process. The set of characteristics, and their structuring, presented in this report contributes to all of these tasks. Still, the presented set of characteristics in no way aspires to be a complete listing of the relevant characteristics in distributed information systems. Issues to be discussed, among others, are the following:

- The set of characteristics could be regarded as too extensive for some purposes.
- There could be reasons to add characteristics that this work has chosen to ignore.
- All of the characteristics mentioned could be perceived and interpreted differently according to the experience and situation of the people involved in the work.
- It may be argued if the best approach was to form the characteristics into a tree structure.
- The brainstorming session does not assure that all aspects and characteristics have been found or discussed, but it assures that more opinions and views emerge.

To capture the necessary information, a modeling framework consisting of three parts is used. The parts are security-relevant characteristics, component library, and system model. The interactions between these three parts results in the ability to capture the dynamics of different aspects of the system without the need to alter the other parts. To implement the modeling framework, a UML-based modeling technique is used. Class diagrams are used to capture the characteristics and the components, while deployment diagrams are used to capture the structure and configuration of the current system design. There are several benefits of using UML.

- The ability to separate models into several diagrams allows the direct implementation of the proposed modeling framework.
- The support of hierarchies is crucial in order for the resulting models to be used by both humans and automatic tools.
- The wide use of UML results in greater availability of tools and more straightforward integration of the design for securability framework with existing design methods and tools.

We believe that system models enabling designers and design tools to assess and modify current and future systems are crucial. Even though the feasibility of creating a set of security-relevant system characteristics supporting quantitative measures is an open question, the set of characteristics identified in this report is a start and supports the formulations of modeling techniques. The formulation of adequate modeling techniques is central for the implementation of a design for securability framework.

6.1 Future Work

In this section, three main directions for future work are discussed. The directions are relating to the set of security-relevant characteristics, the modeling techniques, and the formulation of compound values of securability.

6.1.1 Security-Relevant Characteristics

The set of characteristics, or a future version of it, can result in improved security architectures. To enable this more research and development is needed. Some possible activities are:

- A study where the characteristics are applied to real systems to evaluate the usefulness of the approach.
- An exploration of different functions that solve the problems associated with the characteristics. In doing this, a list of functions to incorporate in systems is built.
- Other methods to gather characteristics, such as brainstorming sessions with different participants and based on case studies, scenarios, or parts of the proposed structure, could be used to improve the set of characteristics.
- A refinement of the set of characteristics to reach levels where quantifiable characteristics appear.

6.1.2 Modeling Techniques

Additional work is required to experiment with the proposed UML-based modeling technique and validate its usefulness. Moreover, the modeling technique can be extended in numerous ways to incorporate other system aspect and, thus, additional security-relevant information about the modeled system.

6.1.3 Compound Values of Securability

To create compound values of securability, the security-relevant characteristics of a system that can be quantified are sampled, as illustrated by Figure 14 (the dots and edges leading to the corresponding scales). The sampled values are then combined into a compound value, possibly a vector, that constitutes an indication of the security of the design or system considering the sampled characteristics. However, it is not likely that compound values presented out of context will provide any useful information about the corresponding systems. Therefore, it is essential to present compound values together with the sampled characteristics and the mechanism used to combine them into compound values.

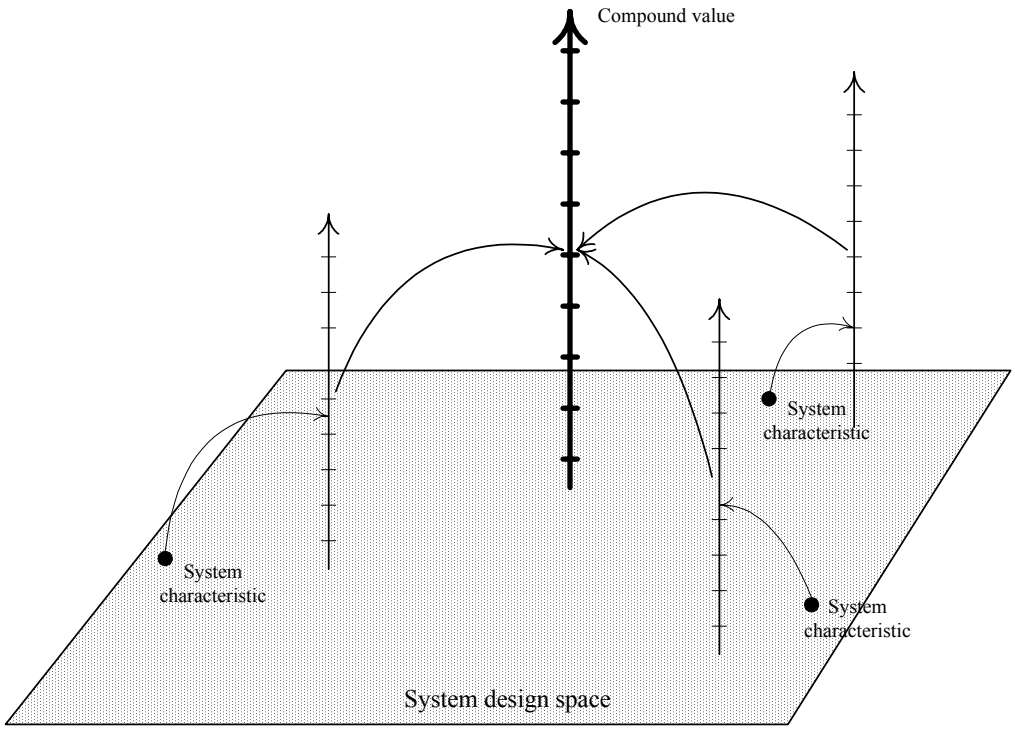

Figure 14: The security of system characteristics is measured on scales. These scales are combined into a scale for compound values.

Acknowledgements

The authors would like to acknowledge the valuable input on UML provided by Maria Andersson and Per-Ola Lindell both at the Swedish Defence Research Agency in Linköping.

The authors would also like to acknowledge input at various stages of the work from Anna Stjerneby who finished her master thesis within this project, especially regarding the sections about security-related characteristics and related work.

Bibliography

ACSA (2002), *Proc. Workshop on Information Security System Scoring and Ranking*. Applied Computer Security Associates, http://www.acsac.org/measurement/proceedings/wisssr1-proceedings.pdf

Akehurst, D. & Waters, A. (1999). *UML specification of distributed system environments.* Technical Report : 18-99**.** Computing Laboratory, University of Kent at Canterbury. UK.

Alberts, D. S., Garstka, J. J. & Stein, F. P. (1999). Network Centric Warfare: Developing and Leveraging Information Superiority. Second edition (Revised). CCRP Publication Series.

Anderson, R. (2001). *Security engineering – A guide to building dependable distributed systems*. John Wiley & Sons.

Apostal, D., Foote-Lennox, T., Markham, T., Down, A., Lu, R., O'Brien, D. (2001) Checkmate network security modelling. *Proceedings of DARPA Information Survivability Conference & Exposition II, 2001*. DISCEX '01. Vol. 1, pp. 214-26

Burns, J., Cheng, A., Gurung, P., Rajagopalan, S., Rao, P., Rosenbluth, D., Surendran, A., Martin, D. (2001). *Automatic Management of Network Security Policy*. www.cs.uml.edu/~dm/pubs/discex.pdf.

CC (1999). *Common Criteria for Information Technology Security Evaluation. Part 1: Introduction and general model, Part 2: Security functional requirements, Part 3: Security assurance requirements. Version 2.1*, August 1999. Visited 2002-05-31 at http://www.commoncriteria.org/cc/cc.html

Coulouris, G., Dollimore, J., & Kindberg, T. (2001). *Distributed Systems– Concepts and Design, third ed.*, Addison-Wesley.

Eschelbeck, G. (2000). Active Security – A proactive approach for computer security systems. *Journal of Network and Computer Applications*, Vol. 23, No.2, April 2000, pp. 109-130.

ETH (2001). Center for Security Studies and Conflict Research, Swiss Federal Institute of Technology. The International Critical Infrastructure Protection Handbook. Ernst Basler + Partners Ltd, Draft Version, November 8, 2001.

Gollmann, D. (1999). *Computer Security*. John Wiley & Sons.

Gula, R. (1999). *Broadening the scope pf penetration-testing techniques - The Top 14 Things Your Ethical Hackers-for-Hire Didn't Test.*, http://www.enterasys.com/products/whitepapers/security/9012542.pdf

Hallberg, J., Hunstad, A., Eriksson, E. A. & Palmgren, S. (2002). (In Swedish) *Områdesanalys: IT-försvar. User report, FOI-R-0469-SE*, Swedish Defence Research Agency.

Jonsson, E. & Olovsson, T. (1997). A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior. *IEEE Transactions on Software Engineering*. vol. 23. no. 4.

Kruger, R. & Eloff, J. (1997). A Common Criteria framework for the evaluation of Information Technology systems security. *IFIP TC11 13 international conference on Information Security (SEC '97)*. Copenhagen, Denmark. Chapman & Hall.

Object Management Group (2001). *OMG – Unified Modeling Language, v1.4*. September 2001.

Olivier, M. (2001). Towards a configurable security architecture. *Data & Knowledge Engineering*, Volume 38, Issue 2, August 2001, pp. 121-145.

Schudel, G. & Wood, B. (2000). Adversary Work Factor as a Metric for Information Assurance. *Proceedings of the New Security Paradigms Workshop*. Cork, Ireland, Sep. 18-22, 2000.

Schneier, B. (2000). Secrets & Lies – Digital Security in a Networked World. John Wiley & Sons.

Sommerville, I. & Sawyer, P. (1997). Requirements engineering: a good practice guide. Chichester: Wiley.

Stjerneby, A. (2002). Identification of security relevant characteristics in distributed information systems. Master's Thesis. LiTH-ISY-EX-3278-2002. Linköpings universitet.

Whitmore, J.J. (2001). A method for designing secure solutions. IBM Systems Journal, Armonk 2001, Volume 40, Issue 3.

Appendix A

In this appendix, the tasks of the study described in (Stjerneby, 2002) are detailed. The study consisted of the four tasks literature study, structured analysis, brainstorm session, and cross checking. The structured analysis is the core of the method while the other tasks were used to generate input to the structured analysis. The result of the study was a structure of security-related characteristics with 60 distinct characteristics as described in Chapter 4.

Literature Study

The purpose of the literature study was to survey the state of the art and gather security-relevant system characteristics. Several sources were used to collect relevant literature. Naturally, the result of the study is limited by the amount of material publicly available, the used search tools and other factors. Still, the conclusion of the literature study was that the amount of basic security-relevant system characteristics to be extracted directly from published work is limited. Instead, the result of the literature study consisted of high-level characteristics that were used as a starting point for the structured analysis.

Several sources were used to collect relevant literature. Naturally, the result of the study is limited by the amount of material publicly available, the used search tools and other factors. The authors screened the found literature and the outcome was compiled and evaluated. For the evaluation the following criteria were used:

- What kind of system is targeted?
- What is the basis of the analysis (method)?
- Which system characteristics are considered to be important for the security?
- Which steps of the design process are included in the analysis (method)?
- How are the systems modeled?

Structured analysis

To obtain a *suitable structure for further analysis* of the output from the literature study and the brainstorm session, a basic structured analysis was performed. This analysis in itself resulted in a number of security characteristics.

To be able to enumerate important characteristics, a tree structure with three roots was built. The roots of the tree structure are confidentiality, integrity, and availability (CIA) since these concepts are usually regarded to be the cornerstones of computer security. The tree structure was built by detecting which security characteristics are descendants of C, I, and A respectively. At this point, the literature study was valuable, primarily as a source of inspiration, when creating a first limited set of security-related characteristics, as illustrated in Figure 15.

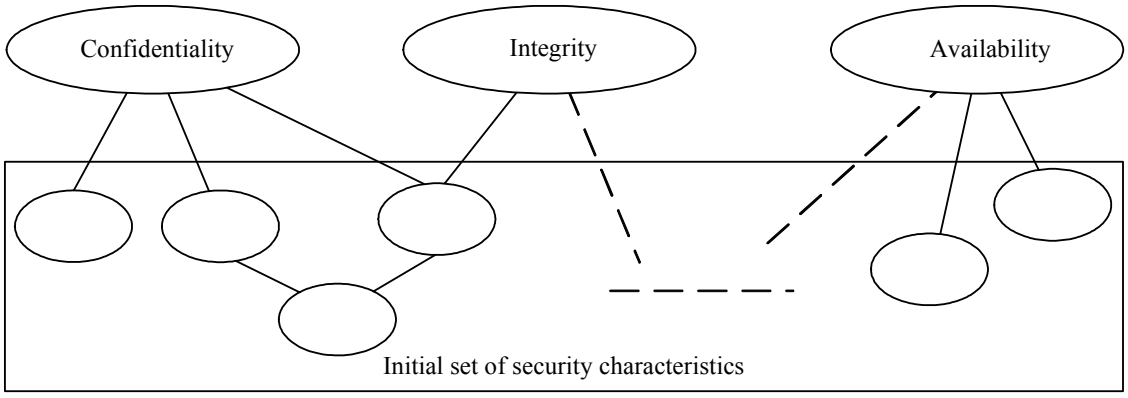

Figure 15: Security characteristics in a tree structure.

To obtain a reasonably comprehensive set of security characteristics, further input was needed. Thus, four case studies focused on in a brainstorm session were formulated. As illustrated in Figure 16, the security characteristics found during the brainstorm were at first not linked to the tree structure. The appropriate position in the tree structure for each characteristic was found through a continued structured analysis. The analysis was performed as several iterations, one iteration for each characteristic found during the brainstorm, consisting of the following steps:

- Evaluate characteristic and, if considered relevant, submit it for inclusion in the tree.
- If submitted, check for overlaps with characteristics in the existing structure.
- If overlaps are found, repartition the corresponding characteristics to eliminate the overlaps. This can result in a clustering of characteristics.
- Consider partitioning of characteristics on a high level of abstraction into more concrete characteristics.
- Insert the resulting set of characteristics into the structure by back-tracking to higher levels of abstraction to find the relations to the existing structure.

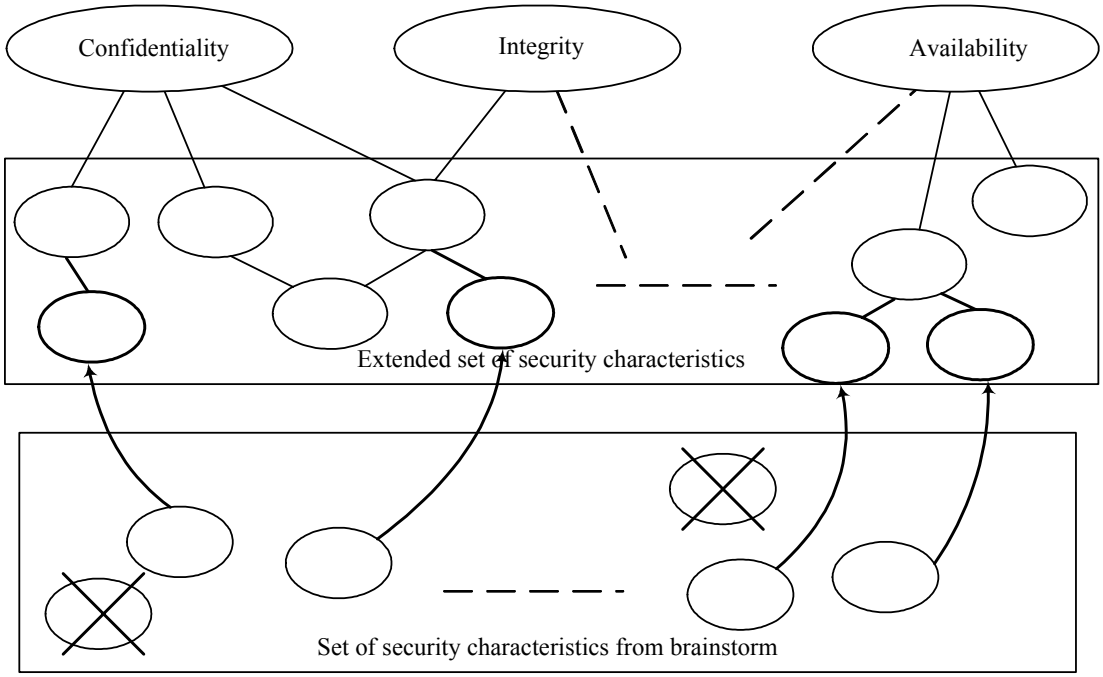

Figure 16: The tree structure was extended through structured analysis, represented by the edges and the crosses, with input from the brainstorm session.

Brainstorm session

To complement the characteristics found through the literature study and the structured analysis, a brainstorm session was arranged. Besides the authors, three IT security experts participated in the brainstorm. A large number of characteristics emerged from this session, e.g. those related to segmentation and software.

Four basic system cases were used to stimulate the discussions. They were deliberately made simple in order to be comprehensible. The following components are used in the case studies.

User terminal (UT)	Access point
Server	Public network
Application engine (AE)	Router
Firewall	Switch
Input/output devices	Proxy
Network link	

A *user terminal* mediates interaction between a user and the system via a screen, keyboard, mouse, speakers etc. *Application engines* execute application software. Combining an application engine with a user terminal

results in a PC or workstation. An *access point* mediates traffic between wireless and wired networks.

The first case, depicted in Figure 17, is called Firewall since it was anticipated to stimulate discussion relating to firewalls, access to external services from inside the firewall, and outside access to internal services. The second case, depicted in Figure 18, is based on the first case, but emphasizes the presence of external users that are authorized to use internal services of the system, and is called External. The third case, depicted in Figure 19, represents the use of peer-to-peer (P2P) solutions and is, consequently, called P2P. Finally, the fourth case, depicted in Figure 20, was created to emphasize the use of wireless network and is called Wireless.

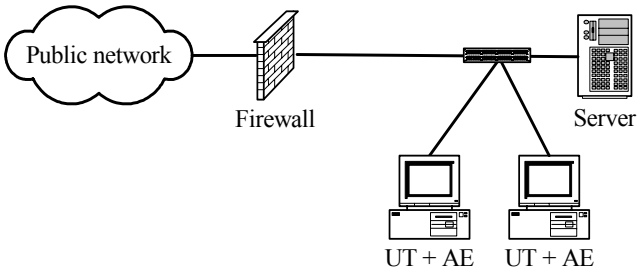

Figure 17: The Firewall case.

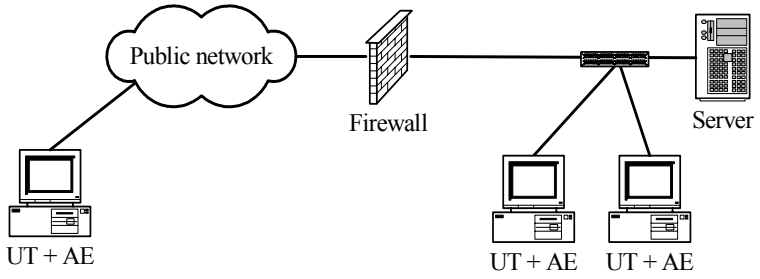

Figure 18: The External user case.

Figure 19: The P2P case.

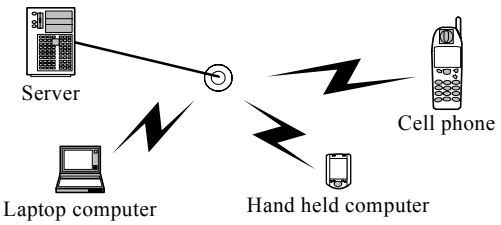

Figure 20: The Wireless case.

During the brainstorm session each case was handled in four steps. Firstly, approximately five minutes was used to individually write notes with thoughts and characteristics. Secondly, all notes were placed on a white-board, within a projection of the case, by the corresponding authors. While placing the notes, the authors commented their meaning. Thirdly, when all notes had been placed on the whiteboard, they were all discussed and structured into groups. Some notes were removed after being found redundant or irrelevant. The resulting structure was immediately documented on a laptop. Thus, fourthly, the participants were shown the resulting documentation and could provide further comments.

Cross-checking

Being based on the preparations and outcome of a brainstorming session, the found IT security characteristics represented in the tree structure are one of several possible sets of characteristics. There is no one and only truth concerning which IT security characteristics to study and apply.

To check whether there are any inconsistencies in the tree structure, a straightforward manual checking algorithm has been applied. The algorithm consists of two steps:

1) For every leaf or sub-tree of the tree, check whether
   a) it ought to be a node in any of the other sub-trees of the whole tree structure or
   b) it ought to be deleted.
2) If yes
   a) is the answer to question 1.a, add a copy of the leaf or sub-tree at appropriate position or
   b) is the answer to question 1.b, delete the leaf or sub-tree

This check does not capture all kinds of possible inconsistencies and errors, but it is a simple way to find the most obvious ones.

Acknowledgements

The security experts participating in the brainstorm and, thus, to a high degree contributed to the results presented in this paper are Lars Westerdahl, Arne Vidström, and Mats Persson at the Department of Systems Analysis and IT Security at the Swedish Defence Research Agency in Linköping.

Appendix B

As described in Chapter 4, the different security characteristics were detected during the execution of four tasks or phases. These tasks, which as methods, are described more in detail in Appendix A, are:

- a literature study (to detect what the state of the art is),
- a structured analysis (to obtain a suitable structure for further analysis and to detect some basic security characteristics),
- a brainstorm session (to obtain a larger set of security characteristics), and
- cross checking (to verify the suitability of the whole structure and its contents).

The tasks were not executed in sequence. The structured analysis was performed with input from both the literature study and the brainstorm and basically lasted during the whole study.

Figure 21 indicates during which task, prior to cross-checking, each security characteristic was detected. Characteristics detected prior to the brainstorm session are above the curved line in the figure. Characteristics detected during the brainstorm session and in later iterations of the four different tasks, are below the curved line. A set of characteristics found during the initial literature study phase is shaded in black. A set of characteristics found during the initial structured analysis is shaded in grey.

Observe that the majority of the characteristics were detected during the brainstorm session. Some parts of the tree structure detected during the brainstorm session are mainly elaborations of structures found prior to the brainstorm session. There are though some parts, such as the software and segmentation sub-trees, which as a whole were detected as a result of the brainstorming. This indicates the high value of brainstorming as a method of work.

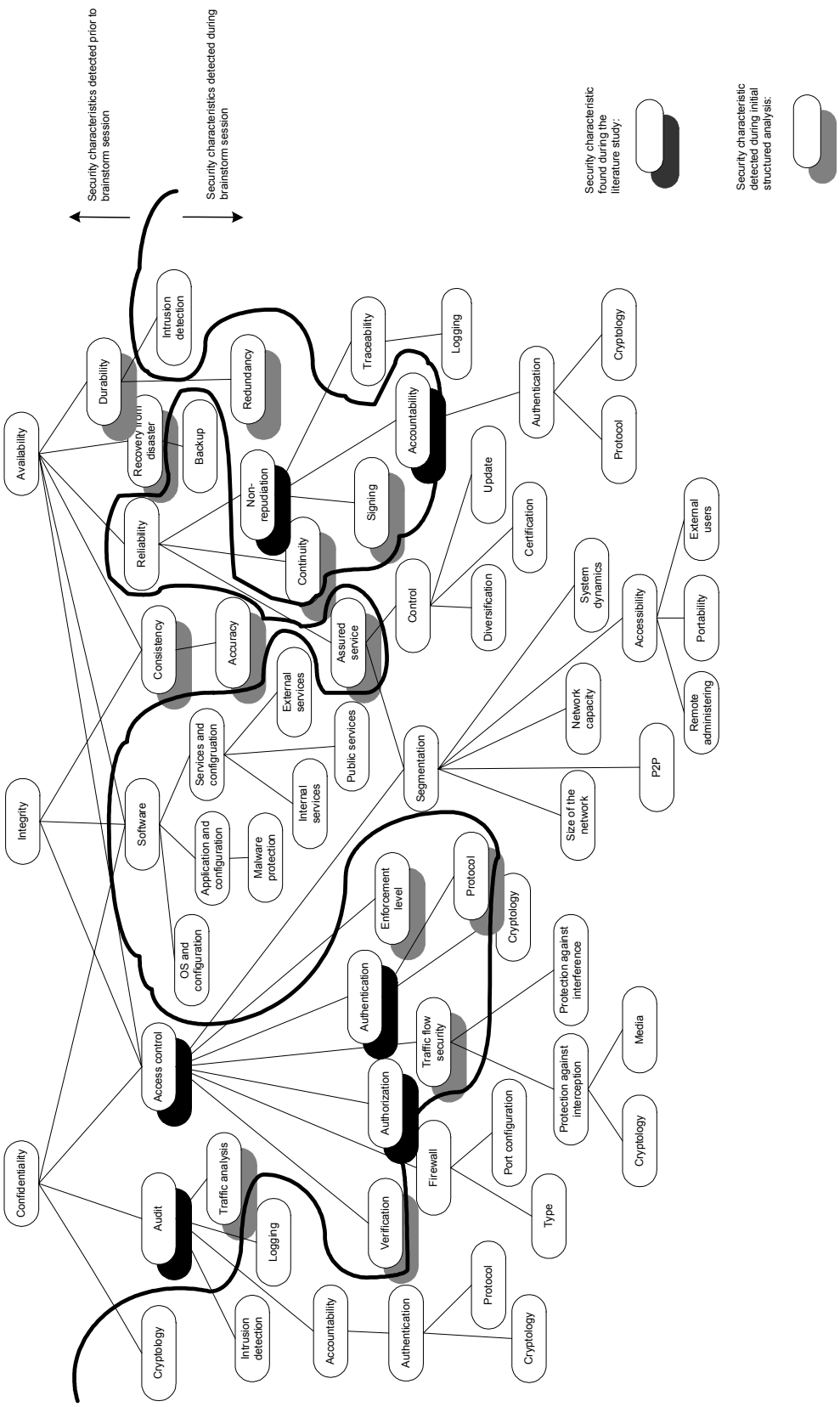

Figure 21: The characteristics formed as a tree structure, with indications of during which task each characteristic were detected.

Figure 22 indicates especially those characteristics which were detected during the cross-checking procedure or which names were adjusted. Names were adjusted mainly to make them more precise. Observe that cryptology forms a sub-tree together with four new characteristics (key management, key length, cryptographic protocol and cryptographic algorithm) introduced during cross-checking

Some further adjustments were also performed on the tree structure. The sub-tree Firewall is now a part of the segmentation sub-tree. Duplicates of the characteristic Cryptology have been removed, but none of the connections to this characteristic. The duplicate of the sub-tree Authentication has been removed, but the connection remains. Characteristics found under Protection against interception are also found under Protection against interference.

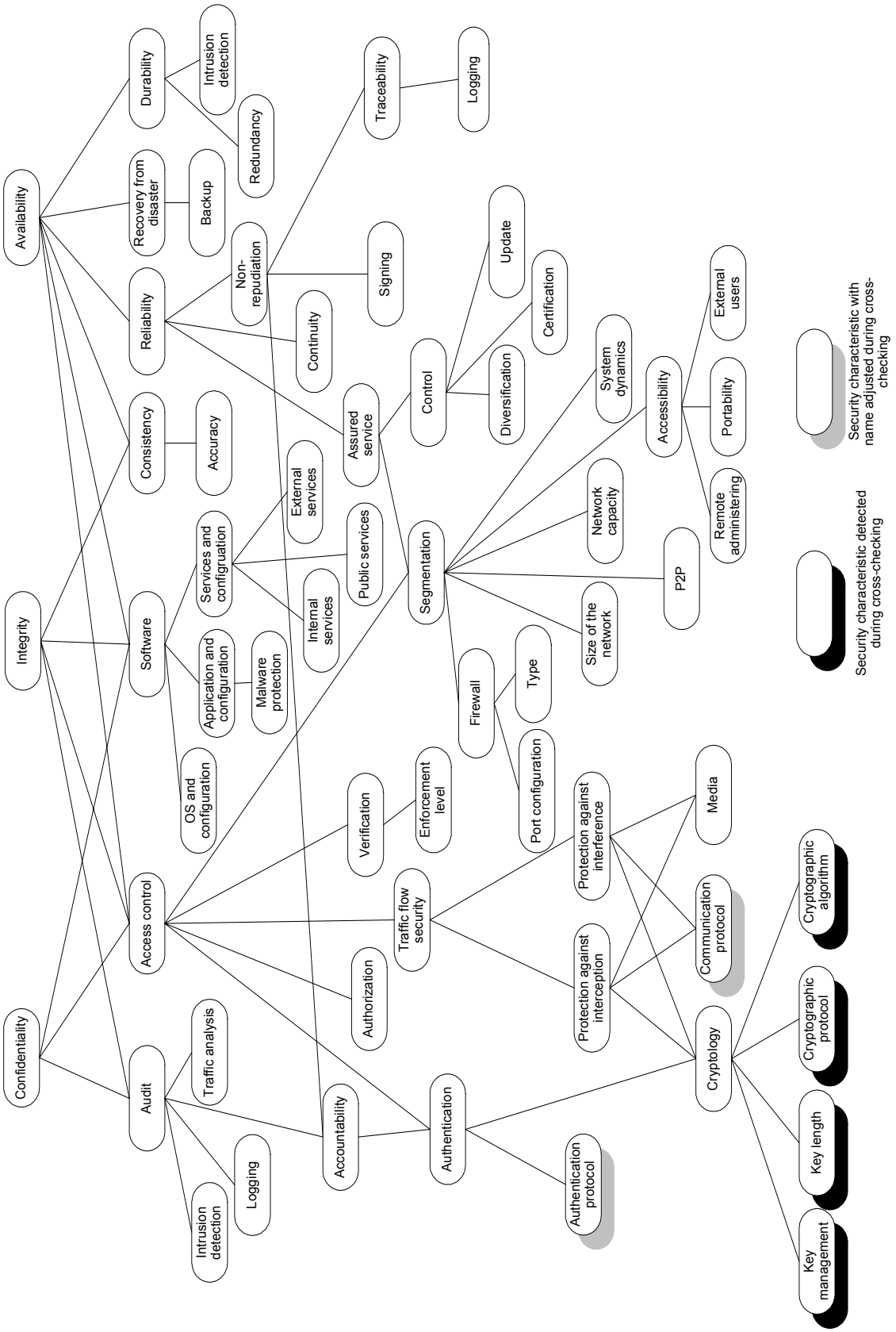

Figure 22: Tree structure revised during cross-checking, with indications of characteristics detected or with their name changed.