A SYSTEMATIC APPROACH TO THE MANAGEMENT OF SYSTEM SECURITY REENGINEERING PROCESS

A THESIS IN ESM

Engineering Systems Management

Presented to the faculty of the American University of Sharjah
School of Engineering
In partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

By
GHANEM IBRAHIM ELSHAHRY
B.S. 1993

Sharjah, UAE
May 2005

We approve the thesis of Ghanem Ibrahim Elshahry

Date of signature

Kassem Saleh Professor Thesis Advisor	
Ibrahim Y. Al Kattan Associate Professor Graduate Committee	
Rana Ahmad Associate Professor Graduate Committee	
Ibrahim Y. Al Kattan ESM Program Director	
Leland Blank Dean of School of Engineering	
Judith Killen Director, Graduate Studies & Research	

A SYSTEMATIC APPROACH TO THE MANAGEMENT OF SYSTEM SECURITY REENGINEERING PROCESS

Ghanem Ibrahim Elshahry, Candidate for the Master of Science in Engineering Systems Management

American University of Sharjah, 2005

ABSTRACT

With the increasing dependency on the electronic world for doing business using computers, palms, wireless devices, and the Internet, there is a need for revising the security measures and controls built into existing communication and computer systems. Several computer-based systems were originally built without considering the security in the system development phase. Consequently, a systematic approach to the management of the reengineering of system's security is recommended. The goal is to ensure that all critical services are well protected and less vulnerable to security threats. Ultimately, the system will be secured according to the organization's business security needs and business continuity plan.

The proposed approach uses formal and standard specification techniques for describing security requirements and developing security acceptance test cases. A security gap analysis is first performed, and the system is reengineered starting from the requirements analysis ending with the user acceptance testing. The benefits of the approach are twofold. First, security requirements, as expected by the system stakeholders, will be satisfied by the current implementation, hence enhancing the system security and improving the trust in it. Second, any additional and future security requirements or modifications to existing requirements will be dealt with in a formal way and not as security patches or fixes to the implementation.

The current security standards have been examined. International Organization for Standardization 17799:2000, security standard was chosen since it addresses all current types of security requirements, and for its international visibility. An overview of all known security requirements related to the four security goals, namely confidentiality, integrity, availability and accountability, were discussed. A comprehensive listing and mapping of all known types of security requirements are linked to the Secure Unified Modeling Language security stereotypes. As a result of this mapping, the author has extended the language stereotypes to address additional availability, accountability and immunity security requirements. Finally, security requirements are mapped to the security mechanism using corresponding serotypes.

This approach is a product-independent and mechanism-independent system security reengineering process to cope with the rapid changes in evolving technologies and the dynamic nature of the technology world.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to thank Allah all might for blessing me and giving me the opportunity, strength, and health to complete this work.

I would like to express my deepest appreciation to my thesis supervisor Professor Kassem A. Saleh for his support, encouragement and fruitful discussions through my thesis research. Many thanks also go to Dr. Ibrahim Al Kattan, and Dr. Rana Ahmad of my thesis advisory committee.

Special thank go to my father Ibrahim, mother, brothers, uncle Farrok, aunt, and the rest of my family for their prayers and encouragement through out my study. Special thanks to my wife for her support and patience that were valuable to finish this work, and to my sons Ibrahim, Mohammad and the little Yousuf who was generally comprehensive in not interrupting my short nights after long and tiring work days.

Also I would like to thank the ESM faculty, and my colleagues, the pioneers, for their help and support through out the ESM program.

CHAPTER 1

INTRODUCTION

1.1. Introduction and Background

With the increasing dependency on the electronic world for doing business using computers, palms, wireless devices, and the Internet, there is a need for revising the security measures and controls built in existing communication and computer systems. Several computer-based systems were originally built without considering the security in the system development phase. Consequently, a systematic approach to the management of the reengineering of system's security is recommended. The goal is to ensure that all critical services are well protected and less vulnerable to security threats. Ultimately, the system will be secured according to the organization's business security needs and business continuity plan.

In this research, the author addresses security requirement as an initial step in the management of a systematic security reengineering cycle. Requirements specifications languages such as the Goal-Oriented Requirement Language (GRL) [1] and UMLsec [2], an extension to the Unified Modeling Language (UML), are used to capture and specify the business security needs. Then the current security implementation in a system is evaluated, and examined whether it meets the requirements identified earlier. Recommendations and implementations of possibly additional or modified measures and controls will be reassessed to ensure that the original security requirements are met.

The benchmarking for managing the security reengineering process is based on the international standard ISO/IEC 17799:2000 [3], with some additional references to the Control Objectives for Information and related Technology (COBIT) [4, 5] according to the model needs.

The rapid changes in evolving technologies and the dynamic nature of the technology world necessitate the need for establishing product-independent and mechanism-independent system security reengineering management process. Standards and tools

selection are depending upon the deployment phase. For example, electing ISO/IEC 17799:2000 as security standard, and GRL with UMLsec as modeling tools was based on matching the current contemporary technologies. Also the security requirements vary from one system to the other according to the sensitivity and exposure of assets to risks. Risk is the possibility that any specific threat will exploit a specific vulnerability to harm the system. Vulnerability is characterized by the absence of security safeguards or countermeasures, and possibly the weakness of a security counter-measure.

In summary, security requirements, tools, and standards can be changed within the frame of the author systematic approach to the management of system security reengineering process.

1.2. Literature Review and Background

IT security issues are discussed from several perspectives for ultimately insuring the broad security goals and requirements, namely, confidentiality, integrity, and availability, commonly known as the CIA triad. Confidentiality requirements ensure that the objects are not disclosed to unauthorized party. Integrity requirements ensure that the objects retain their correctness, and are only intentionally modified by an authorized party. Availability requirements ensure that the authorized party is granted timely and uninterrupted access to objects [7].

Confidentiality and integrity are dependent upon each other. Without the integrity element, confidentiality cannot be maintained. Other concepts, conditions, and aspects of confidentiality include sensitivity, discretion, criticality, concealment, secrecy, privacy, seclusion, and isolation. Also without confidentiality, integrity cannot be maintained. Other concepts, conditions, and aspects of integrity include accuracy, truthfulness, authenticity, validity, non-repudiation, accountability, responsibility, completeness, and comprehensiveness. However Availability is dependent upon both integrity and confidentiality. Without integrity and confidentiality, availability cannot be maintained. Other concepts, conditions, and aspects of availability include usability, accessibility, and timeliness. In addition to the CIA triad, there are additional security related concepts, principles, and tenants that should be considered and addressed when designing a security policy and deploying a security solution. like

identification, authentication, authorization, accountability, non repudiation, and auditing [7].

The success of a system can be assessed based on meeting two complementary types of requirements. First, the user-centered requirements that concentrate on the functionalities provided to the system's user. Second, the context-centered requirements which can be system, process and human requirements. The context-centered requirements are referred to as Non-Functional Requirements (NFRs). NFRs are mainly system's constraints that may affect greatly the operational environment and design choices a developer may pursue during the system development process. NFRs include among other things, operational environment: hardware, software interfaces, accuracy, performance: timeliness and storage requirements, security, reliability, maintainability, portability, robustness, and usability. Consequently, system acceptance testing is based on both functional and non-functional system's requirements [8, 9].

International standards for security requirements specifications were developed to meet the needs for the modeling and specification of system's functional behaviors. The Unified Modeling Language (UML) for systems (SysUML) is a standard developed by the Object Management Group to deal specifically with systems [10]. To model security requirements as NFRs, an international standard is being developed by the International Telecommunication Union (ITU) [1]. This standard includes a description of a requirement specification language called GRL (Goal-oriented Requirements Language) [1]. Furthermore, a non-standard security extension to UML, called UMLsec, was proposed by Jurjens [2].

There are several engineering systems requirements for business, software application, or network infrastructure. Most of these requirements focus on the functional aspects of these systems. Quality, data, and interface requirements as well as the architectural, design, implementation, and testing constraints also need engineering. Whereas some non-functional requirements are elicited, analyzed, specified, and managed such quality requirements, interoperability, operational availability, performance, portability, reliability, usability, and security requirements are often overlooked. Most requirements engineers are not trained in security, and the few that have been trained have only been given an overview of security architectural

aspects such as passwords and user rights rather than in actual security requirements. Thus, the most common problem with security requirements, when they are specified, is that they tend to be replaced with security-specific architectural constraints that may unnecessarily constrain the security engineers from using the most appropriate security aspects for meeting the needed security requirements [11]. Several reference frameworks are considered for the evaluation and assurance of security systems such as the Common Criteria (CC) which defines specific assurance and functional security requirements in terms of Evaluation Assurance Levels (EAL). Security requirements can also be specified in Protection Profiles, and Targets of Evaluation (TOES). Protection Profiles define implementation independent sets of security requirements and objectives for categories or general classes of systems. Security Targets of Evaluation are used to define the IT security objectives and requirements for a specific system design implementation. The functional security services requirements, as well as the system assurance requirements are specified in Protection Profiles. The ability of a given implementation to address these requirements is captured in the evidence associated with a given Target of Evaluation. Whether defined in a TOE, Protection Profile, or an EAL, Common Criteria functional and assurance requirements form the primary source of requirements which must be addressed in the system reengineering process, and assist in the generation of internally consistent design packages. During the early stages of the security system engineering development, the appropriate functional and assurance requirements for a system should be defined in conjunction with the Information System Security Officer (ISSO). These requirements are based on the system mission requirements, and can be refined during the system design process. Selection of the functional and assurance requirements determines the amount of evidence generated during the security reengineering process [12, 13].

In this research, the author focuses on security as a non-functional requirement. Specifically, the author concentrates on the management of the re-engineering of security in legacy systems in which security was overlooked or was incorporated in the system in an ad hoc manner.

Systems are designed and built according to two differing categories, Closed systems are designed to work well with limited systems generally are from the same

manufacturer. The standards for closed systems are often proprietary and not disclosed. However open systems are designed using agreed-upon standards. These o pen systems are much easier to integrate with systems from different manufacturers that support the same standards. Closed systems are harder to integrate with unlike systems, but they are more secure. A closed system normally comprises proprietary hardware and software. This lack of integration means that attacks on many standard systems will not work. Attacking closed systems are harder than launching an attack on open systems. Many software and hardware components with known vulnerabilities may not exist on a closed system. In addition to the lack of known vulnerable components on a closed system, it is often necessary to possess more in-depth knowledge of the specific target system to launch a successful attack. On the other side when the security weakness are discovered in closed system it is hard to get fixes and deploy solutions from standard manufacturer, so this research recommends going for standardization [14].

Finally, model-driven testing methodologies have been used successfully to validate the conformance of software implementations to their requirements. Trong [15] introduces a systematic approach to testing UML design models, and Pilskalns et al. [16] propose a rigorous testing technique by merging structural and behavioral UML design representations. In this research, the author intends to examine these approaches and extend them in order to develop security assessment test plans based on UMLsec design models.

1.3. Problem Statement and Objectives

Surveys have shown that large numbers of systems were implemented starting from their elicited functional requirements without a clear and formal consideration of their non-functional counterparts such as the security requirements. Furthermore, system requirements engineers and analysts are not well-trained in capturing security requirements early in the system development process. Security assurances are often based on the traditional and ad hoc approach of conducting penetration tests followed by a patching process. This approach is very costly and endangers the fulfillment of the basic goals of system security, namely, confidentiality, integrity, availability and accountability. Recently, many researchers addressed security engineering and requirements as an integral and essential element of systems engineering and

requirements. Devanbu et al. [17] propose a roadmap for software engineering for security, and Linger et al. [18] consider life cycle models for survivable and secure systems.

To address the urgent need for incorporating security requirements in existing systems and legacy systems in general, a formal methodology for managing the system security reengineering process should be developed. This process includes the elicitation of the stakeholders security requirements, assessment of the implemented security measure, finding the security gap, implementation of recommendation for closing the gap, then finally, the verification that the implemented measures meet the requirements or not.

The objectives of this work are to: 1) specify system security needs using GRL and UMLsec, 2) develop assessment test plans from system security specifications, 3) perform security assessment plans leading to the identification of the security vulnerabilities and weaknesses to address in order to meet the current security requirements, and 4) incorporate the above steps in a continuous process for managing system security.

1.4. Research Plan and Methodology

The system reengineering process consists of the following phases. In the first phase, the system security requirements from the business stakeholders are elicited and mapped to the latest security standards such as ISO/IEC 17799:2000 [3]. The requirements are adjusted to match the standards while coordinating with the stakeholders. There are several benefits for incorporating the latest standards in the requirements elicitation phase. Examples of such benefits include 1) any institution electing to adopt this methodology will be certifiable from process perspective, and 2) avoiding reinventing the wheel since the best practices scenarios are embedded in the standards. The system security requirements are also depicted using the graphical Goal-Oriented Requirements Language (GRL) [3] and UMLsec, an extension to the Unified Modeling Language [4].

In the second phase, a generic assessment plan is developed to assess an existing system against the identified security requirements, and the author highlights the consequences of the assessment procedure and gets the stakeholders consent.

In the third phase, a gap analysis is performed by applying the developed assessment procedure on the existing system's security implementation, and a verdict on the satisfaction of each of the stakeholders' requirements is produced.

In the fourth phase, an action plan is developed including design modifications to the existing system security controls. In this step, these modifications must be inline with the formal requirement specifications obtained in the first phase. Once approved by the stakeholders, this plan will be executed.

In the fifth phase, the security design modifications are implemented in the system according to the action plan.

Finally, in the sixth phase, the re-engineered system security is reassessed against the initial security requirements, and a final security acceptance test is performed. Figure 1.1 shows the phases of the formal approach to the system security reengineering process.

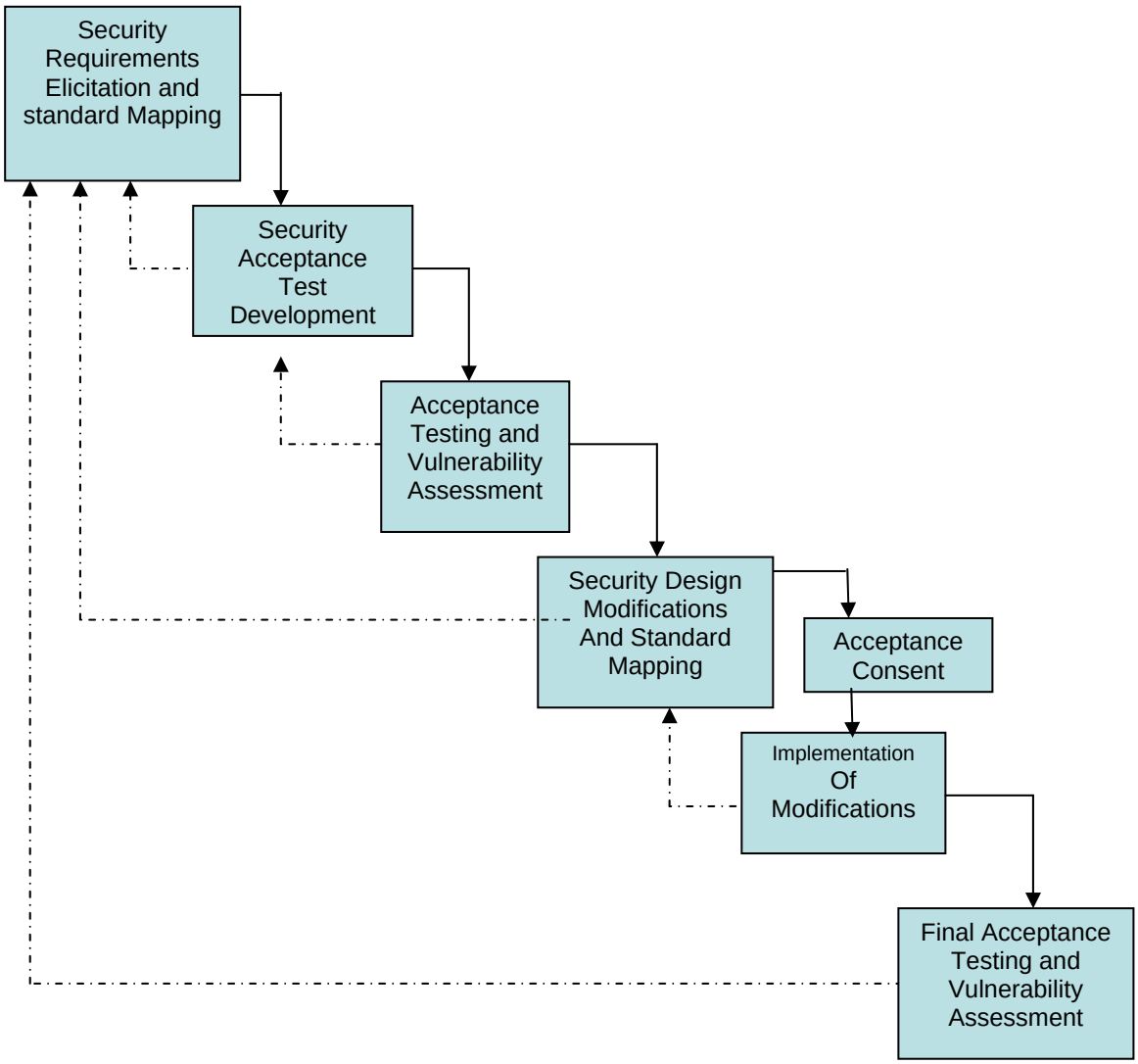


Figure 1.1 Formal system security reengineering process.

Finally, the reengineered system is handed to the operations department to maintain the security status, and update the system security batches according to its security requirements whenever new vulnerabilities are discovered. Standard baseline will be maintained and the gap will be closed by following the process described in Figure 1.2. Closing the gap must be an ongoing process, and must be semi-automated after putting the management tools and procedures in place.

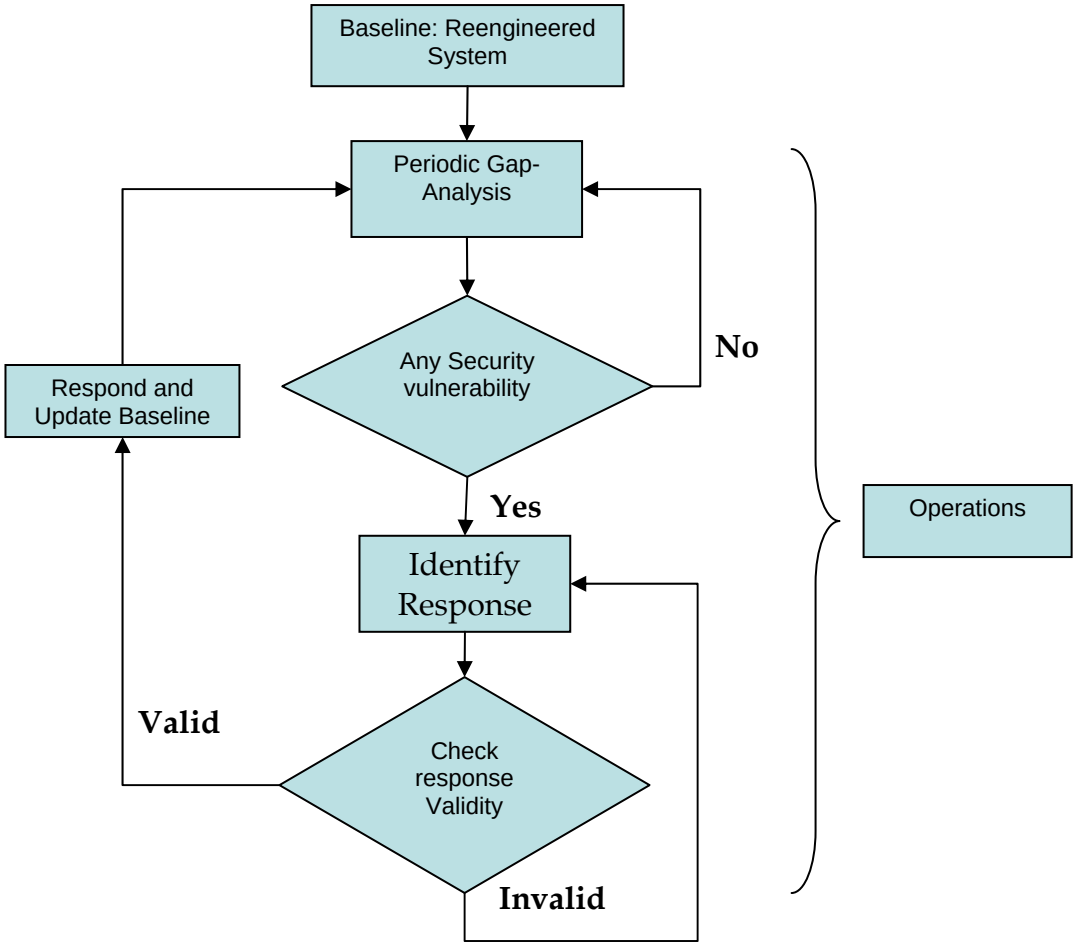


Figure 1.2 Continuous processes for addressing security weaknesses.

1.5. Contributions and their Significance

The main contributions of this thesis are:

1. Extension of the UMLsec language to improve its generality, and its expressive power for the modeling of all security requirements types. These extensions included stereotypes like <<*audit log*>>, <<*multiplicity*>> and <<*no misuse*>>. These stereotypes are used to address additional availability, accountability and immunity security requirements. These stereotypes can be used in UMLsec security use case, deployment and sequence diagrams.

2. Mapping the international standard on information systems security, ISO/IEC 17799:2000, to a classification of security requirements and UMLsec stereotypes.
3. Introducing an approach to develop security test cases starting from the formal security requirements specifications described using UMLsec and GRL specifications languages.
4. Incorporating the use of requirements models such as UMLsec and GRL, the mapping to ISO/IEC 17799:2000, and the security test case development process, into an overall security reengineering life cycle approach that can be applied any time security requirements change.
5. Establish Product-independent and mechanism-independent system security reengineering process to cope with the rapid changes in evolving technologies and the dynamic nature of the technology world.

This research shows the importance of building IT systems in conformance with the latest security standards starting from the system initiation phase. The author developed the steps to be followed by IT specialists in managing the system security reengineering process in cases where security was not considered earlier during the system development phases. A model-based approach to generating assessment plans for testing the conformance of the system implementation to the system security requirements is developed. Furthermore, the author proposes an ongoing process for continuously reducing the security gap and consequently enhancing the overall system security.

1.6. Outline of Thesis

The rest of the thesis is organized as follows.

Chapter 2 describes the requirements specification and standardization including functional and the non-functional requirements. Chapter 3 explains the need for security and highlights the different types of security requirements. Chapter 4 discusses the Goal-Oriented Language (GRL) and provides some examples. Chapter 5 describes the Unified Modeling Language (UML) diagrams, and introduces its security extension (UMLsec). Chapter 6 walks-through the UMLsec, highlights its

approach in enforcing security requirements in the security model design phase. Chapter 7 walk-through the latest security standards such as ISO/IEC 17799:2000, Control Objectives for Information and Related Technologies (COBIT) and the Common Criteria (CC). Chapter 9 describes different methodologies in testing system's security, highlighting UMLsec rationale with examples. Develop security test cases along with Webinterface UMLsec tools short tutorial. Chapter 10 explains in details the system security reengineering process using a case study. Applies the security concepts, standards and tools as per of the discussions in earlier chapters. Finally, the author summarizes, concludes, and presents future research and work in Chapter 11.

CHAPTER 2

REQUIREMENTS

2.1. The Importance of Requirements

A requirement is something that the system must do or a quality that the system must have. A requirement exists either because the type of system demands certain functions or qualities, or the client wants that requirement to be part of the delivered system.

The requirements must be specified before attempting to construct the product. If the correct requirements are not in place, the correct system can't be built, and consequently the system does not enable the users to do their work.

Sadly, this is not always the case. Steve McConnell reports that 60% of errors exist at design time [19]. Jerry Weinberg confirms this with statistics that show that up to 60 % of errors originate from the requirements and analysis activities [20]. Although system's builders have the opportunity to almost eliminate the entire largest category of errors, they choose, or worse their managers choose, to rush heading into constructing the wrong system. And thus they pay many times the price for the system than they would have to if the requirements and analysis have been done correctly in the first place.

The cost of good requirements gathering and systems analysis is minor compared to the cost of dealing with poor requirements [21].

2.2. Requirements in the System Development Life Cycle

The previous section shows the importance of specifying the requirements before building the system. Normally, the System Requirements definition is the first phase in the System Development Life Cycle (SDLC) models. Considering the Waterfall SDLC model as an example, among the seven phases, the first two phases are concerned with Requirements as shown in Figure 2.1

The waterfall model views the system development life cycle as a series of sequential and iterative activities. The traditional waterfall model has seven phases of development. As each stage is completed, the project moves into the next phase. As

illustrated by the backward arrows, in practice the model does allow development to return to the previous phase to correct defects discovered during the subsequent phase. This is often known as the feedback loop characteristic of the waterfall model [7].

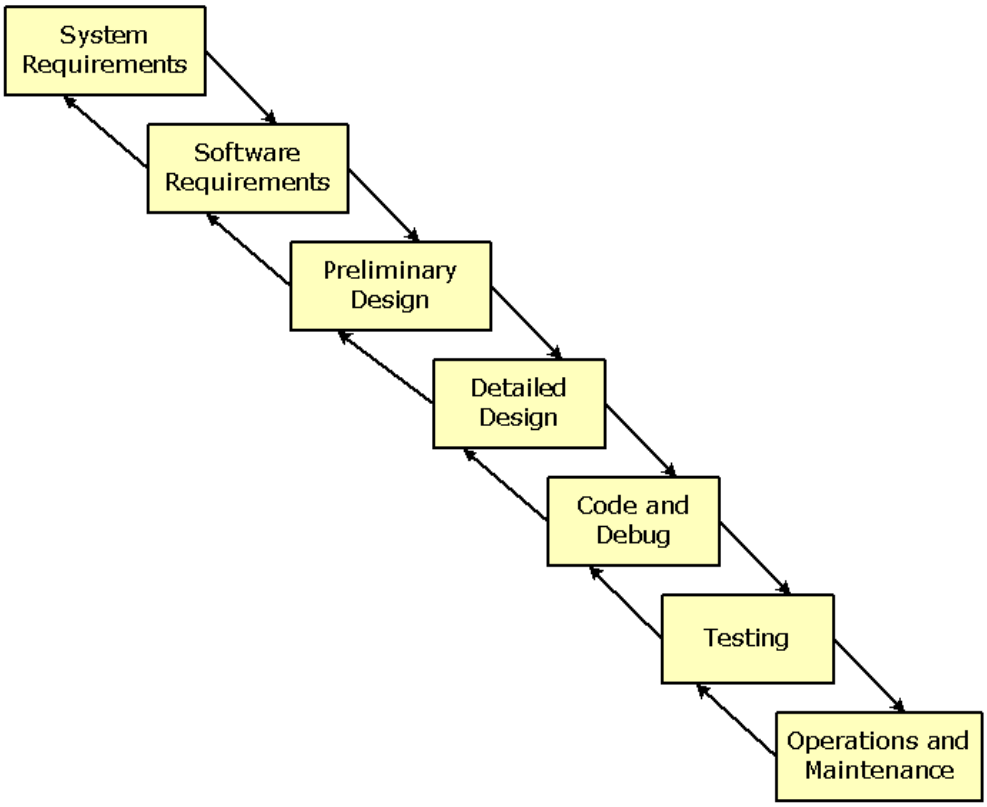


Figure 2.1 The waterfall life cycle model
(Reproduced from [7])

2.3. Requirements Standards

The success of any system depends on meeting two complementary categories of requirements. First the functional requirements which are the system's operations from the user's perspective. Second, the Non-Functional Requirements (NFRs) which can be system, process, and human requirements. NFRs are mainly system's constraints that may affect greatly the operational environment and influence the system's acceptance. NFRs include among other things, operational environment: hardware and software interfaces, accuracy, and performance: timeliness and storage requirements, security, reliability, maintainability, portability, robustness, and usability. Functional Requirements describe mainly the visible and external input and

output interactions with the system under consideration, whereas Non-Functional Requirements are those that impose special conditions and qualities on the system to construct. Consequently, system acceptance testing is based on both functional and non-functional system's requirements.

International standards were developed to meet the modeling and specification of system's functional requirements needs. The Unified Modeling Language (UML) is a standard developed by the Object Management Group, a consortium of software companies, to deal with systems in general [22]. Because of the need for satisfying NFRs, and developing standards for formalizing the description and capturing the requirements, the International Telecommunication Union (ITU) Group 17 has started to work towards the standardization of a User Requirements Notation (URN) targeting the description of requirements for future telecom systems and services. URN provides a notation for the representation of non-functional requirements (NFRs) such as performance, cost, security, and usability, in addition to a complementary scenario notation for functional requirements. The proposed NFR notation is the Goal-oriented Requirement Language (GRL). It is the first attempt for the explicit capture and representation of NFRs. In addition, Use Case Maps (UCMs) are used as scenarios that help describing and understanding complex functional behavior of software systems [24]. URN fits nicely in the very first stage of existing design and standardization methodologies. On top of traceability links between GRL and UCMs, the proposed GRL standard intends to provide additional insights on the derivation of other models, including Message Sequence Charts, performance models, and test cases. Links to the UML standard will also be investigated in order to unify common concepts and perhaps notations. UML is discussed in Chapter 5.

2.4. Non-Functional Requirements

The following is a brief description of the three categories within the non-functional requirements [8].

2.4.1. System-related NFRs

These types of requirements impose some criteria related to the internal qualities of the system under construction and the component types and features in which the system will operate.

Operational requirements: Operational requirements specify the environment in which the system will operate, including, hardware platforms, external interfaces, operating systems, and appliances.

Performance requirements: Performance requirements specify the upper and lower limits of speed, response time and storage characteristics of the system.

Maintainability requirements: These requirements specify the expected response time for dealing with the various maintenance activities, such as future release dates.

Integrity and portability requirements: Integrity and portability requirements specify future plans for integrate the system with others modules, or running the system in different operating environments. These requirements are linked to both operating and maintainability requirements and may impose certain design decisions and implementation choices.

Security requirements: Security requirements specify the levels and types of security mechanisms that need to be satisfied during system's life cycle. These requirements may include the compliance with specific security standards. Since security requirements are key components in this research, they will be discussed in more detailed in Chapter 3.

2.4.2. Process and Project-related NFRs

These types of NFRs impose criteria to be followed in the system life cycle.

Conformance to standards requirements: Also known as compliance with standard requirements specifies the internal, international and industry standards that have to be followed in the system life cycle. Security standard is discussed in Chapter 7.

Time and cost requirements: Time and cost requirements are set by the various stakeholders and can be refined in the project plan and modified as the project progresses.

System's structure requirements: System's structure requirements impose specific design model or framework, in addition, to work products, milestones and deliverables expected. These requirements will affect the project plan and the chosen architecture.

Assessment and Testing requirements: Assessment and testing requirements impose specific assessment and testing strategies or guidelines to be used in the system's structure. The types of test plans needed and the testing quality may be

affected by these requirements. It can also specify the accepted rate of passed test cases, or stipulate mandatory features to be assessed.

Installation and deployment requirements: Installation and development requirements impose specific installation restrictions and guidelines. These restrictions can be related to the network zone, traffic type, protocol features, and enabled features.

2.4.3. Human-related NFRs

Human-related non-functional requirements deal with constraints related to the stakeholders and the social and societal context in which the system is deployed.

Usability requirements: Usability requirements specify criteria for judging the degree of usability and user friendliness of the system. It may also specify guidelines or prescribe standards for developing the graphical user interfaces for the system, such as query and reporting formats.

Look and feel requirements: Look and Feel requirements specify criteria and standards the look and feel of the system may have to follow. These standards could be internal or international standards.

Legal requirements: Legal requirements specify the laws and regulations that the system has to satisfy to avoid any legal implications. These laws and regulations could be international, regional, and organizational policy.

Cultural and political requirements: Cultural and political requirements specify the social and political constraints defined by the context and environment in which the system stakeholders live. These requirements could affect other requirements like, legal, conformance to standards and usability.

2.5. Standards for Capturing NFRs

International standardization bodies are concerned with capturing and specifying NFRs. The International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) Study Group 10 (SG10) is focusing on languages and software aspects for telecommunication systems. SG10 addresses issues related to modeling languages and mechanisms including the Unified Modeling Language (UML), System Description Language (SDL), and Message Sequence Charts (MSC). Study Group 17 deals with the User Requirement Notation (URN) and is finalizing a standard for the URN [23]. A brief background on the URN will be discussed in this section [8].

2.5.1. The User Requirement Notation (URN)

URN is designed to deal with both functional and non-functional requirements [1].
The main objectives of URN design are [8]:

- Describe scenarios for functional requirements as high level reusable entities with no design details or sub-structuring provided for these entities.
- Facilitate the transition and mapping of requirements specification to a high level design and provide alternative architectural designs.
- Facilitate the elicitation and synthesis of additional requirements.
- Facilitate the early detection and avoidance of undesirable feature interactions and side-effects among them.
- Provide facilities and artifacts to describe, analyze and deal with non-functional requirements.
- Provide facilities to express the relationship between business objectives and goals to system requirements, expressed as scenarios for functional requirements, and global constraints over the system, its development, deployment, maintenance and evolution, and operational processes for nonfunctional requirements.
- Provide facilities to capture reusable analysis and design knowledge related to previous experiences dealing with non-functional requirements.
- Support traceability and mappings to other languages including the UML.

The above objectives are refined and developed in the following URN standard documents.

Z.150 URN: Z.150 URN standard includes the motivations, scope, and objectives for the URN, in addition to some basic terminology and requirements engineering concepts. Many specific requirements also refine the list of URN objectives into URN-NFR, and URN-FR (URN - Functional Requirements). FRs describes the external and observable input/output interactions with the system under development. Notations and methodologies for eliciting functional requirements and using them in the system development process are being used extensively in the software industry. However, URN-NFR is used to help standardize the modeling of business goals and product quality features [24].

Z.151 GRL: Z.151 GRL standard presents the Goal-oriented Requirement Language (GRL) as a notation for URN-NFR. Notation constructs are defined, together with their visual representation and concrete grammars (textual and XML) [1].
Chapter 4 demonstrates the use of GRL as a proposed tool for capturing the non-functional system requirements. Also includes detailed description and tutorial for GRL.

Z.152 UCM: Z.152 UCM standard presents the Use Case Map (UCMS) as a notation for URN-FR [23].

Z.153: Z.153 is a proposed standard document will focus on the relationships between GRL and UCM, and between URN and other languages such as UML.

CHAPTER 3

SECURITY REQUIREMENTS

The main system security goals are to ensure confidentiality, integrity, availability and accountability. Confidentiality ensures that only authorized users or processes are allowed to interact with the system. Integrity ensures that the critical data has not been changed in an improper way in the system. Availability ensures that the information and/or services are available to an authorized user on demand. Accountability ensures that once an authorized user accesses the system, they are accountable for all their actions [25].

Since the objective is to reengineer an existing system security, it is assumed that security requirements were not clear or known during the implementation phase. In this chapter, the need for security, the objectives of security requirements, and the different types of security requirements are discussed along with examples.

3.1. Why Security

Information systems security is usually defined as the protection of the systems flow of processes, and the data stored within against unauthorized access, modification, destruction, and against actions or situations that deny authorized access or use of the system for other authorized users. While accidental events that threaten security are included, the emphasis is on deliberate attempts to weaken security. The need for system security is evidenced by the increasing number of reported security incidents, the associated damage and the financial liabilities incurred.

In general, the following are the principal reasons for ensuring that the security goals in information systems and applications are met [26].

3.1.1. Protection of Resources

Computer systems, and more importantly, the data stored and processed within, tend to be critical and valuable assets to an organization's operation. Some data in the system, such as financial accounts, directly represent tangible assets and must be protected against unauthorized access and alterations.

3.1.2. Mandated by Law

Several countries have enacted laws that require protection of certain categories of information; other laws require that the integrity of financial data be ensured.

3.1.3. Maintain Management Control

A goal of any organization is to be in full control of its assets and the way it is used.

3.1.4. Ensure Safety and Integrity

Security is an essential prerequisite for the system safety and for the integrity of the databases and processes being used.

3.1.5. Operational Advantages or Economies

A secure system is important in organizations whose business depends on their customer's trust, such as financial institutions. It can increase the organization's competitive advantage over competitors with less secure systems. It can also reduce operating costs such as insurance premiums.

3.2. Objectives of Security Requirements

The recent reported incidents, including viruses' outbreaks, malicious crackers, industrial spies, governmental spies and threats of cyber crimes, lead to reiterate the following objectives of security requirements [11].

- Ensure the identification of client applications and users, and that their identities are properly verified.
- Ensure that users and client applications can only access data and services for which they have been properly authorized.
- Detect attempted intrusions by unauthorized persons and client applications.
- Ensure that unauthorized malicious programs (e.g. Trojans, spyware) do not infect the application or component.
- Ensure that communications and data are not intentionally corrupted.
- Ensure that no party interaction with the application or component can be later repudiated.
- Ensure that confidential communications and data are kept private.
- Enable security personnel to audit the status and usage of the security mechanisms.
- Ensure that applications and data centers survive attacks, possibly in a degraded mode.

- Ensure that data centers and their components and personnel are protected against destruction, damage, theft, or unplanned replacement (e.g., due to vandalism, sabotage, or terrorism).
- Ensure that system maintenance does not unintentionally disrupt the security mechanisms of the application, component, or data center.

3.3. Types of Security Requirements

Normally the security requirements should not be specified in terms of the types of security architecture mechanisms, and controls that are currently used for implementation.

To meet the security objectives, the following corresponding types of security requirements will be briefly defined. Detailed information can be found in [11].

3.3.1. Identification Requirements

The identification requirements specify the extent to which a business, application, component, or center shall identify its externals before interacting with them. The externals can be users or applications.

Example: The system shall ensure that the name of the personnel in the official human resource and payroll databases exactly matches the name printed on the passports.

3.3.2. Authentication Requirements

The authentication requirements specify that the business, application, component, or center shall verify the identity of its external user before any interaction. An external user can be a human or an application.

Example: The system shall verify the identity of all of its users before allowing them to update their user information.

3.3.3. Authorization Requirements

The authorization requirements specify the access and usage privileges of authenticated users and client applications. This will prevent users from accessing confidential data or performing inappropriate action.

Example: The system shall not allow one or more users to misuse the authorized services such as flooding the system with multiple legitimate requests.

3.3.4. Immunity Requirements

The immunity requirements specify the extent to which a system or component shall protect itself from infection by unauthorized undesirable programs, such as Trojan horses, viruses, or worms. The immunity requirement objectives are to prevent any undesirable programs from destroying or damaging the system.

Example: The system shall protect itself from infection by scanning all entered or downloaded data and software for known computer spyware, viruses, worms, Trojan horses, and other similar harmful programs.

3.3.5. Integrity Requirements

The integrity requirements specify the extent to which a system or component shall ensure that its data and communications are not intentionally tampered via unauthorized deletion, creation, or modification. The objective of an integrity requirement is to ensure trusting data and communications.

Example: The system shall prevent unauthorized corruption of all communications traffic through the networks that are external to the protected premises.

3.3.6. Intrusion Detection Requirements

The intrusion detection requirements specify the extent to which a system or component shall detect and record attempted access or modification by unauthorized individuals. The objectives of intrusion detection requirements are to detect, record, and notify security officers for any unauthorized attempts of accessing the system or component.

Example: The system shall notify the concerned security officer within five minutes of any repeated failed attempt to access the databases of employees and corporate financials.

3.3.7. Intrusion Prevention Requirements

The intrusion prevention requirements specify the extent to which a system prevents and records attempted accesses or modifications by unauthorized actors before reaching the core system. The objectives of intrusion prevention requirements are to prevent, record, and notify security officers for any unauthorized attempts to access the system or its components.

Example: The system shall notify daily the concerned security officer of all attempted attacks and failed attempted accesses during the last 24 hours.

3.3.8. Non-repudiation Requirements

The non-repudiation requirements specify the extent to which a system shall prevent a party to one of its interactions from denying participation in all or part of the interaction. Interaction can involve a financial transaction, or a message. The objectives of a non-repudiation requirement are to ensure that adequate tamper-proof records are kept to prevent parties to interactions from denying that they have taken place, and to minimize future potential legal liability problems.
Example: The system shall keep tamper-proof records of each ATM transaction done by the client including, the amount of transaction, the date and time of the transaction, and the client's identity.

3.3.9. Confidentiality/Privacy/Secrecy Requirements

The privacy requirements specify the extent to which a system shall keep its sensitive data and communications private from unauthorized programs and individuals. The objectives of a privacy requirement are to ensure that unauthorized programs and individuals do not gain access to sensitive data and communications. Access is provided based on need to know basis to minimize the loss of user confidence and legal liabilities.
Example: The system shall not allow unauthorized programs or individual accesses to any communications.

3.3.10. Security Auditing Requirements

The security auditing requirements specify the extent to which a system enables security personnel to audit the status and use of its security mechanisms. The objectives of a security auditing requirement are to ensure that the system collects, analyzes, and reports information about the status (i. e. enabled,. disabled, updated versions) of its security mechanisms.
Example: The system shall collect the logs related to identification, authentication, and authorization, software scans, and network access.

3.3.11. Survivability Requirements

The survivability requirements specify the extent to which the system shall survive the intentional loss or destruction of a component. The objective of a survivability requirement is to ensure that the system either fails gracefully or else continues to function, for example, in a degraded mode, even though certain components have been intentionally damaged or destroyed.
Example: The system shall continue to function in degraded mode even if a data center is destroyed.

3.3.12. Physical Protection Requirements

The physical protection requirements specify the extent to which a system shall protect itself from physical attack. The objectives of physical protection requirements are to ensure that a system is protected against the physical damage, destruction, theft, or replacement of hardware, software, or personnel components due to vandalism or sabotage.
Example: The system shall protect its hardware components from physical damage, unplanned replacement, destruction, or theft.

3.3.13. System Maintenance Security Requirements

The system maintenance security requirements specify the extent to which the system shall prevent authorized modifications such as: defect fixes, enhancements, updates and patches, from accidentally defeating its security mechanisms. The objective of a system maintenance security requirement is to maintain the levels of security specified in the security requirements during the operations phase.
Example: The system shall not violate its security requirements as a result of a change to its existing requirements and their implementations.

3.4. Mapping Security Requirements to CIAA Security Goals

The security goals Confidentiality, Integrity, Availability and Accountability (CIAA) are the benchmarking in which the security components use as a reference. Table 3.1, developed by the author, summarizes the relationship between the four security goals CIAA and the types of security requirements.

Table 3.1 Mapping security requirements to security goals

Security Requirement	Confidentiality	Integrity	Availability	Accountability
Identification Requirements	●	O		O
Authentication Requirements	●	O		O
Authorization Requirements	●	O		O
Immunity Requirements	O	●	O	
Integrity Requirements		●		
Intrusion Detection Requirements	●	O	O	
Intrusion Prevention Requirements	●	O	O	
Non-repudiation Requirements		O		●
Privacy/Secrecy Requirements	●			
Security Auditing Requirements		O		●
Survivability Requirements			●	
Physical Protection Requirements	●	●	●	O
System Maintenance Requirements	O	O	●	

● Heavily contributes O partially contributes

Some requirements completely contribute to the achievement of a CIAA goal. For example, Privacy/Secrecy Requirements completely contribute to the achievement of Confidentiality goal. However, the non-satisfaction of the survivability, physical protection, and system maintenance security requirements may compromise availability.

CHAPTER 4

GOAL-ORIENTED REQUIREMENT LANGUAGE (GRL)

GRL (Goal-oriented Requirement Language) [1] is a language for supporting goal-oriented modeling and reasoning of requirements, especially for dealing with non-functional requirements such as security and performance. GRL provides the elements to express several concepts appearing during the requirement elicitation phase. GRL is supported by a tool [27] which was used to develop all GRL diagrams in the figures of this thesis.

There are three main categories of concepts in GRL: intentional elements, links, and actors. The intentional elements in GRL are goals, tasks, softgoals, resources, and beliefs. They are intentional because they are used for models that allow answering questions such as why particular behaviors, informational and structural aspects were chosen to be included in the system requirement, what alternatives were considered, what criteria were used to deliberate among alternative options, and what the reasons were for choosing one alternative over the other.

Intentional elements are concerned with exposing why certain selections of behavior or structure were decided. At this point the modeler is not yet interested in the operational details of processes, system requirements, or component interactions. Omitting these kind of details during early phases of analysis and design allows taking a higher-level decision which is sometimes called a strategic stance towards modeling the current or the future system and its embedding environment.

GRL supports the reasoning about scenarios by establishing correspondences between intentional GRL elements and non-intentional elements in scenario models of the User Requirements Notation – Functional Requirements URN-FR.

Modeling goals and scenarios are complementary and may aid to identify further goals, scenarios and scenario steps important to stakeholders, thus contributing to the completeness and accuracy of elicited requirements.

4.1. GRL Characteristics

GRL is characterized by its support to the goal-oriented and agent-oriented models, as well as dealing with non-functional requirements issues.

The GRL model comprises of the three categories: Intentional elements, Intentional links, and actors. Intentional elements include goals, tasks, resources, softgoals, and

beliefs. Intentional links include dependency, means-ends, and decomposition. Actors which are the systems' external stakeholders.

4.2. Why Goal Oriented

The importance of goal oriented model came from its systematic approach of deriving requirements from goals. Its support to “why” “how” and “How Else” intuitive questions, provides rationales for requirements, as well as it provides structure for the requirements document.

4.3. GRL Models

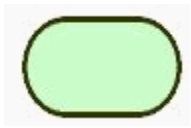
A Goal-Oriented Requirement Model can be either composed of a global goal model, or a series of goal models distributed on several actors. If the model includes more than one actor, then there might exist a series of dependencies which represent the intentional dependency relationships between agents.

4.3.1. Intentional Elements

Goal

A goal is a condition that the stakeholders must achieve. How the goal is to be achieved should not be specified, allowing implementation alternatives to be considered. A goal can be either a business goal or a system goal. A business goal expresses goals regarding the business or state of the business affairs the individual or organization wishes to achieve. System goals express goals the target system should achieve.

A graphical notation of a goal is shown as a rounded rectangle.



Task

A task specifies a particular way of doing something. When a task is specified as a sub-component of a higher-level task, this restricts the higher-level task to that particular course of action. Tasks can also be seen as the solutions in the target system, which will satisfy the softgoals (called operationalizations in NFR). These solutions provide operations, processes, data representations, structuring, constraints and agents in the target system to meet the needs stated in the goals and softgoals

A graphical notion of a task is shown as a hexagon.

Example

Create Router Access List is one particular way of controlling the access.

Resource

Resources are drawn as rectangles, and represent physical or informational entities with which the main concern is whether it is available.

Example: In the security architecture, Firewall is a resource that must be available.

Softgoal

A softgoal is a condition or state of affairs in the world that the actor would like to achieve. However, unlike in the concept of hard goal, there are no clear-cut criteria for whether the condition is achieved, and it is up to subjective judgment and interpretation of the developer to judge whether a particular state of affairs in fact achieves sufficiently the stated softgoal.

Example: A softgoal, which is "soft" in nature, is shown as an irregular curvilinear shape. For instance, reliability of firewall is a softgoal to be achieved during the design of a security system.

Belief

Beliefs are ellipse shapes used to represent design rationale. Beliefs make it possible for domain characteristics to be considered and properly reflected into the decision making process, hence facilitating later review, justification and change of the system, as well as enhancing traceability.

Example: The following is an argument supporting that the task of Create Router Access List is Improving Security.

4.3.2. Intentional Relationships

Each model structure is a link connecting two elements. These structures together constitute the overall goal model. So they are seen as the basic building blocks of models.

Means-ends Relationship

Means-ends relation is used by GRL to describe how goals are achieved. Each task provided is an alternative means for achieving the goal. Normally, each task would have different types of impacts on softgoals, which would serve as criteria for evaluating and choosing among each task alternative.

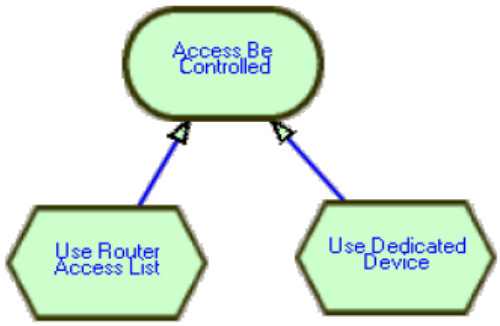


Figure 4.1 Goal Means-ends Structure

Graphically, a means-ends link connects an end node with the means node achieving it. In GRL, only goals are originally applicable to means-ends link. As shown in Figure 4.1.

However, in short hand forms for a combined structure, tasks and resources could also be connected by means-ends links. A Task Means-ends Structure connects a task with the means (tasks) to achieve it directly, which is a short hand form of one Task Decomposition Structure and the related Goal Means-ends Structure as explained in Figure 4.2.

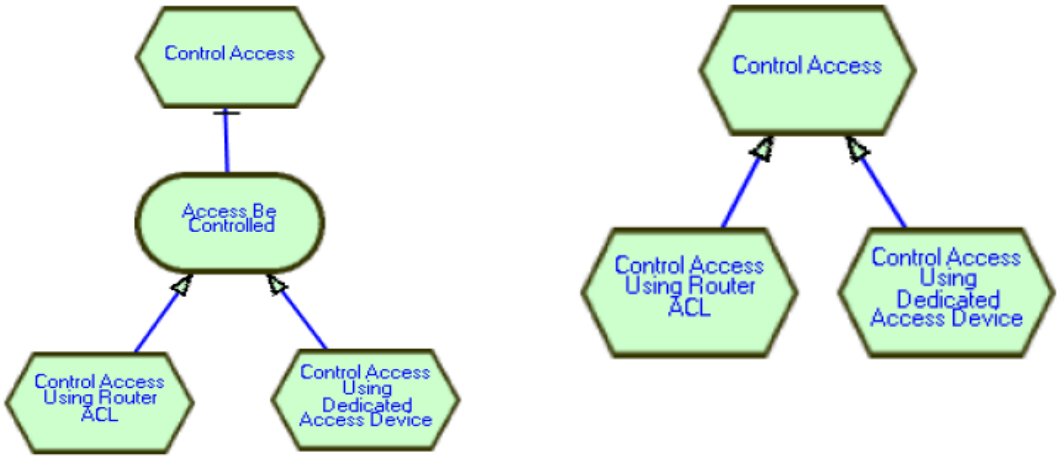


Figure 4.2 Task Means-ends Structure

A resource Means-ends structure connects a resource with the means (tasks) to make it available, which is a short hand form of one <softgoal Contribution Structure>, as explained in Figure 4.3.

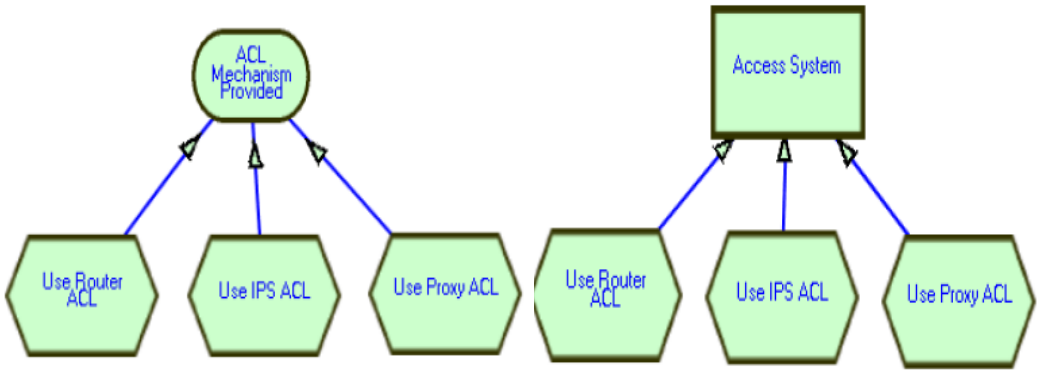


Figure 4.3 Resource Means-ends Structure

Decomposition Relationship

GRL Decomposition relation statement defines other elements needed to be achieved or be available in order for a task to perform.

A decomposition link connects a node with its sub-components. In GRL, only tasks are originally decomposable, but for convenience, in short hand forms, a combined structure goals could also be connected by decomposition links. The sub-components of a task can be goal, task, resource, and softgoal. A Task Decomposition Structure shows the essential components of a task, which include subtasks that must be performed, subgoals that must be achieved, resources that must be accessible, and softgoals that must be satisfied as shown in Figure 4.4.

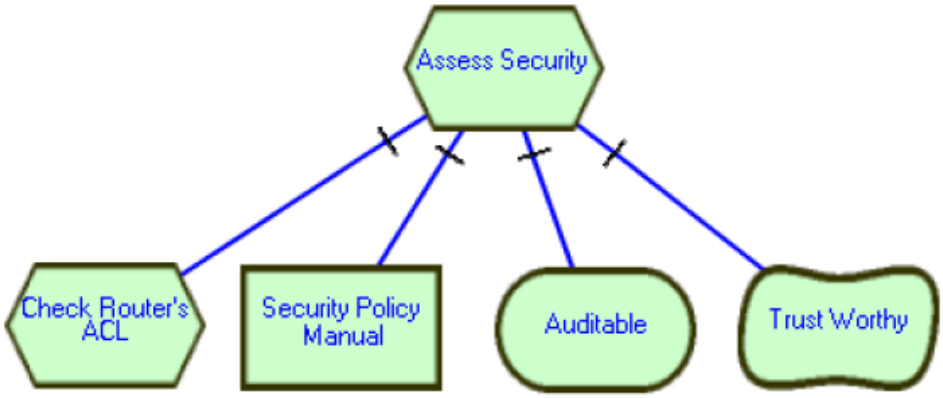


Figure 4.4 Task Decomposition Structure

Also Goal Decomposition Structure connects a goal with its sub-goals directly, which is a short hand form of one Goal Means-ends Structure and the related Task Decomposition Structure as shown in Figure 4.5.

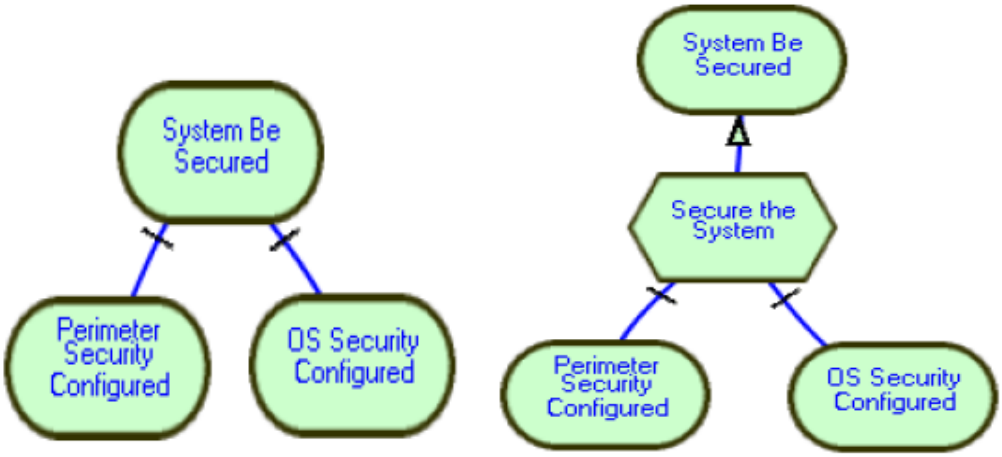


Figure 4.5 Goal Decomposition Structure

Contribution Relationship

The Contribution relationship statement describes how softgoals, task, believes, or links contribute to others. A contribution is an effect that is a primary desire during modeling. A contribution link describes how one intentional element contributes to the satisfying of another intentional element.

AND contributing relation means all of the sub-components are positive and necessary.

OR contributing relation means each of the sub-components is positive and sufficient.

MAKE: The contribution of the contributing element is positive and sufficient.

BREAK: The contribution of the contributing element is negative and sufficient.

HELP: The contribution of the contributing element is positive but not sufficient.

HURT: The contribution of the contributing element is negative but not sufficient.

SOME+: The contribution is positive, but the extent of the contribution is unknown.

SOME-: The contribution is negative, but the extent of the contribution is unknown.

EQUAL: There is equal contribution in both directions.

UNKNOWN: There is some contribution, but the extent and direction of which is unknown at the modeling phase.

The graphical representation of the contribution elements is shown in Figure 4.6.

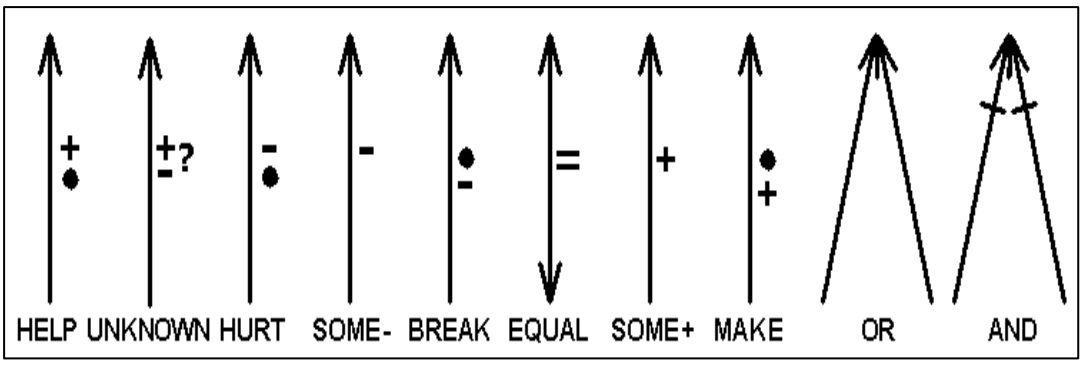


Figure 4.6 Contribution Relationships Graphical Representation (Reproduced from [27])

A Softgoal Contribution Structure shows the contributions of softgoals or tasks towards a softgoal, Figure 4.7 shows more details.

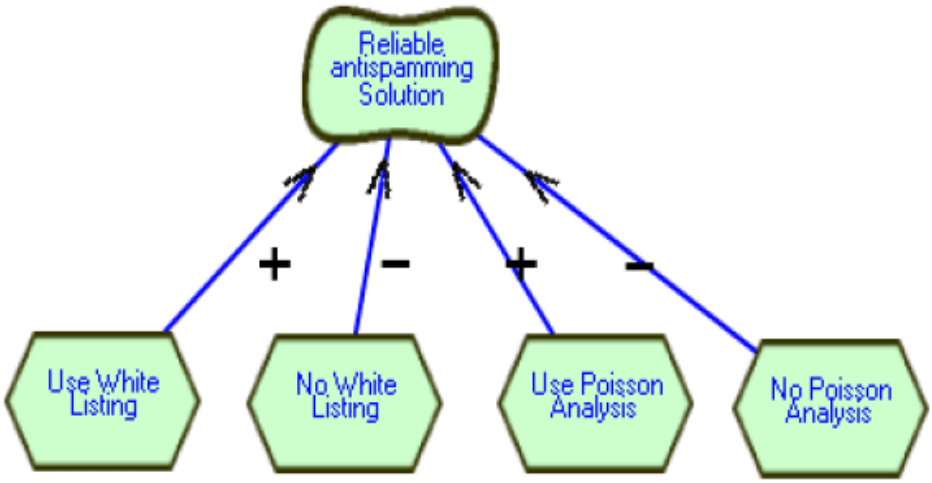


Figure 4.7 Softgoal Contribution Structure

A belief can be attached to any link or node of a model as argumentation structure, which denotes the contribution of a belief node to the link or node it is attached to. It also gives some argument for future review and justification. For example in Figure 4.8, there is a belief that Poisson analysis decreases false positives when it is used as a reliable anti-spamming solution.

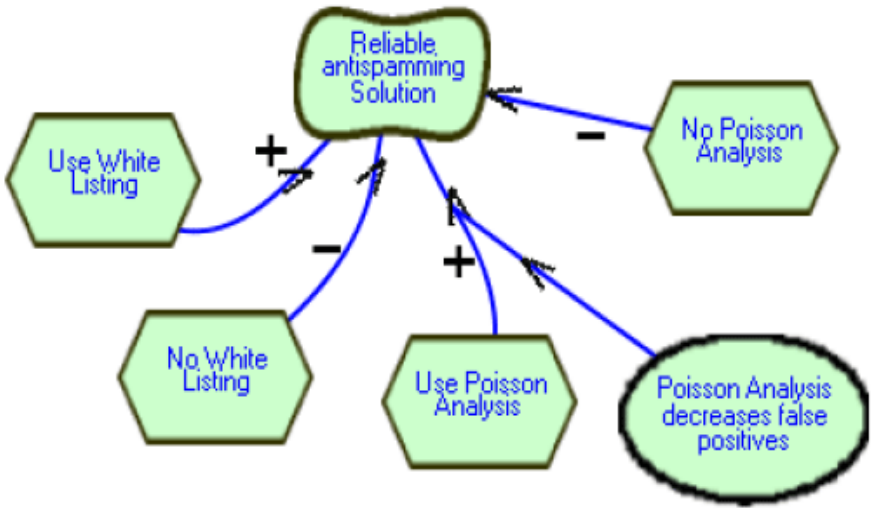


Figure 4.8 Argumentation Structure

4.3.3. Actor

An actor is an active entity that carries out actions to achieve goals by exercising its know-how. Graphically, an actor may optionally have a boundary, with intentional elements inside.

Actor Boundary

Actors are shown as circles with the name of the actor inside in graphical representations. The boundary of an actor is shown as a gray shadow with the actor icon inside or nearby. For example, within a Security Assessment environment, an actor “Security officer” in the assessment process decides to what extent the softgoals has been met. Figure 4.9 depicts the graphical representation of an actor structure and its boundary.

Figure 4.9 Actor and Boundary Structure

Dependency

A dependency statement in GRL describes an intentional relationship between two actors; Depender depends on another actor (Dependee) on something (Dependum). The symbol of dependency relation is shown in the side figure.

Correlations

The correlation relation allows to express knowledge about interactions between intentional elements in different categories, and to encode such knowledge. A correlation link is the same as a contribution link except that the contribution is not an explicit desire, but is a side effect.

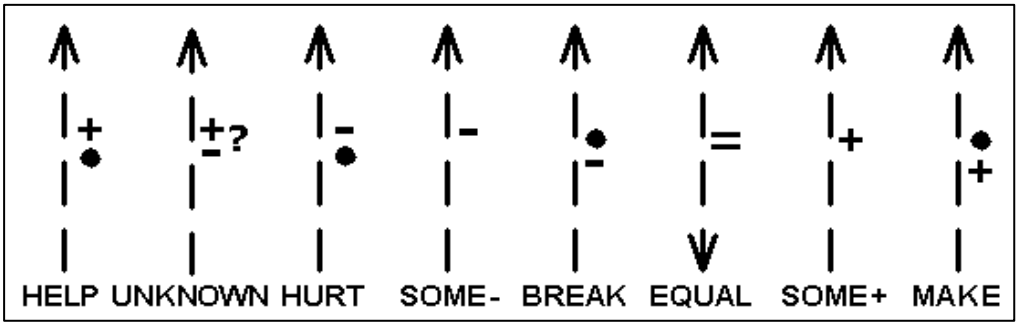


Figure 4.10 Correlation Relationships Graphical Representation (Reproduced from [27])

Examples

A basic goal model example is demonstrated in Figure 4.11. It consists of softgoals and contribution relationships.

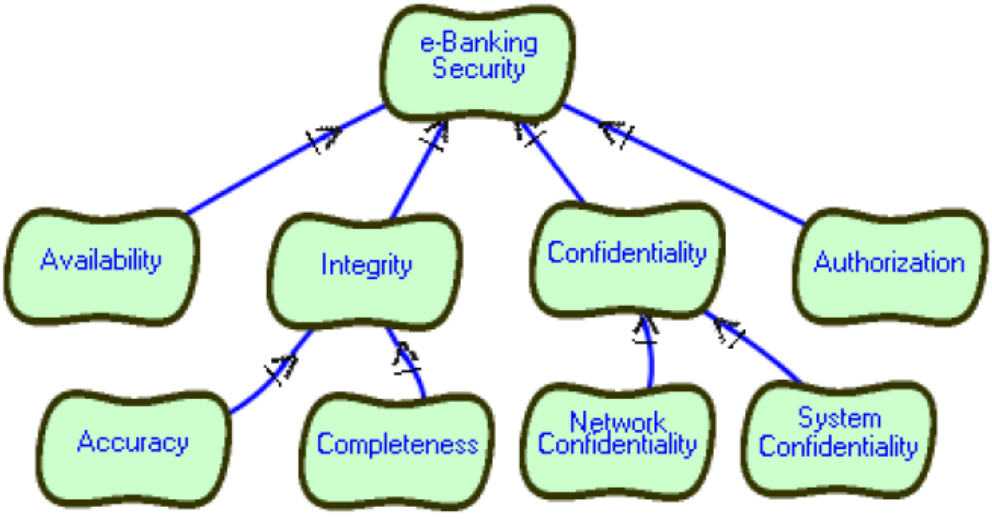


Figure 4.11 Basic Model Structure

A model can also consist of several goal model structures. Each structure is a requirement category as shown in Figure 4.12.

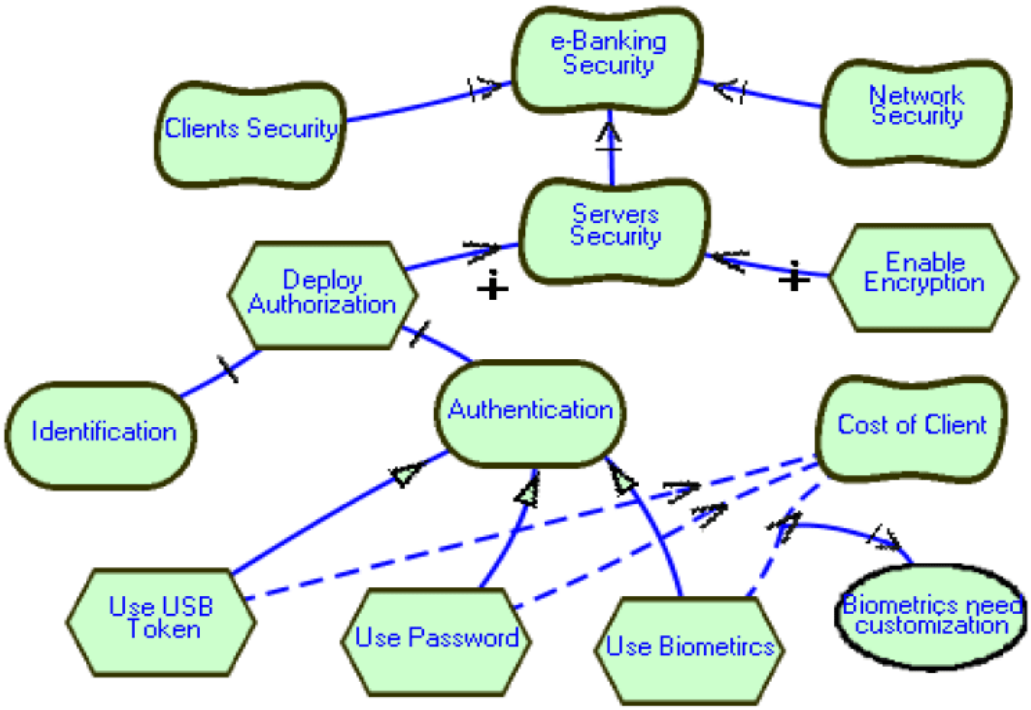


Figure 4.12 Correlated Series of Model Structure

CHAPTER 5

UML AND AN INTRODUCTION TO UMLsec

Most business units consider their customers as stakeholders in which their satisfaction in needed. In order to meet or exceed the customer's satisfaction, customer's exact expectation needs to be captured. Most technical engineers are dealing with systems in an abstract level. There is a need to establish a common language to describe the systems to the stakeholders in a clear and non-ambiguous way. The Unified Modeling Language (UML) is a tool for building the bridge between stakeholders and technical people. It helps capturing the vision for a system and then enables communicating the vision to anyone who has a stake in the system. It does this using a set of symbols and diagrams. Each diagram plays a different role within the system's building process [22].

5.1. UML in Security Engineering Modeling

The Unified Modeling Language (UML) notations can be used as an approach for modeling the security engineering process. Starting by constructing a model for a system, which should be as close to human intuition as possible. Then an implementation is derived from that model.

For security systems this method allows one to consider security requirement in the development phase. Using a model based approach, a system meets the relevant security requirements at the design level can be established by analyzing the model, then checking whether the system meets these requirements. The test is generated using model based test sequences.

There is an opportunity of using the UML as notation for a high-quality model based security systems since it has the following characteristics:

UML provides graphical description techniques with different kinds of diagrams; it has standard extension mechanisms like stereotypes, tags, profiles and constraints which can be used to accommodate different systems requirements.

Several tools are available that provide the basic functionality required to use the UML.

The UML is relatively precisely defined by the Object Management Group (OMG) compared to other previous object-oriented notations [10].

The UML is considered as the de facto standard in industrial modeling notations, a large number of engineers are trained to use the UML. The number is growing since the UML is taught at the universities. This makes UML specifications to be available to security analysts, and developers.

The UML is mainly a visual language. A picture is worth thousand words.

5.2. UML Diagrams

UML includes twelve types of diagrams [22, 28] divided into three categories.

Requirements Elicitation Diagrams - Functionality: Use case diagram.

System Behavior: Activity diagram, State diagram, Timing diagram[1], and Interaction diagram. Interaction diagrams are divided into Sequence and Communication diagrams. Communication diagrams were known as Collaboration diagrams.

Structure and Implementation Diagrams: Class diagram, Object diagram, Component diagram, Deployment diagram, Package diagram[1], and Composite structure diagram[1].

In the following section the UML diagrams are described.

Use Case Diagram: A Use Case diagram describes an abstract view of the functionality offered by a system by specifying typical interactions with the user. It is often used in an informal way for negotiation with a customer before a system is designed. It is used to clarify the user requirement from user's perspective and to show the functional behavior of the system as seen by the user.

---

[1] Added as diagrams in UML 2.0 version.

Class Diagram: A class diagram gives general definition information. It defines the static class structure of the system: classes with attributes, operations and signals, and relationships between classes. At the instance level, the corresponding diagrams are called object diagrams.

Object Diagram: An object diagram is a class diagram at one point in time. It shows objects and their relationships with one another. An object is an instance of a class-a specific thing that has specific values of the class's attributes. In other words an object diagram is a snapshot that shows how instances of classes are linked together in an instant of time.

State Diagram: State diagram also known as statechart diagram describes the dynamic behavior of an individual object or component: events may cause a change in state or an execution of an action, It shows a sequence of states that an object or an interaction goes through during its lifetime, in response to events, and also the responses that the given object or interaction makes to those events.

Sequence Diagram: A sequence diagram describes interaction between objects or system components via message exchange, and focuses on the time-ordering of messages between objects. Sequence diagram can have top level system sequence diagram, then more specific sequence diagrams.

Timing Diagram: A timing diagram, introduced in UML 2.0's, is designed to show how long an object stays is in a state. This is needed since in the sequence diagrams, there is duration of each state, and those durations are never explicitly being shown.

Activity Diagram: An activity diagram shows the dynamic behavior of a system. In particular, it specifies the control flow among several components within the system, usually at a higher degree of abstraction than statecharts and sequence diagrams. An activity diagram focuses on processes and organizations that perform them. An activity diagram can be used to put objects or components in the context of overall system behavior or to explain use cases in more details.

Communication Diagram: Communication diagrams were known as collaboration diagrams. They are an extension of the object diagram. In addition to the links among objects, the communication diagram shows the messages the objects send each other. One way to think of the relationship between the object diagram and the communication diagram is to imagine the difference between a snapshot and a movie. The object diagram is the snapshot: It shows how instances of classes are linked together in an instant of time. The communication diagram is the movie: It shows interactions among those instances over time. Also a communication diagram has nearly the same information as a sequence diagram. Both the sequence diagram and the communication diagram show interactions among objects. For this reason, the UML refers to them collectively as interaction diagrams.

Component Diagram: A component diagram shows how the system is logically composed of software files from packages, other supporting software, along with interfaces and relationships. Normally, a component diagram focuses on a system's software architecture. A component defines a system's functionality. The interface is a set of operations that a class presents to other classes. The important point about components and interfaces is that a component's operations can only be reached through its interfaces. As is the case with a class and its interface; the relation between a component and its interface is called realization.

Deployment Diagram: A deployment diagram describes the mapping of the system components to the physical structure of the system. A deployment diagram shows which components run on which physical processors.

Package Diagram: A package diagram shows how classes and other packages are grouped into packages.

Composite Structure Diagram: A composite structure diagram visualizes the internal structure of a class by showing classes nested inside that class. The composition is one way to show the components of a class.

Other UML Related Diagrams

Model Management Diagrams: Model management diagrams help in describing how and where the objects are implemented.

Subsystem Diagrams: Subsystem diagrams shows how logically related sets of components form subsystems.

Model Diagram: Model diagram includes all of the other diagrams to show how the system is structured and functions.

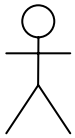
In the following sections, use case diagrams, activity diagrams, and sequence diagrams will be discussed in details.

5.3. Use Case Diagrams

The use case diagrams describe the system from a user's perspective. A use case diagram is an excellent tool for security analyst for gathering the system requirements from the stakeholders' point of view, which is important for building the system that the concerned personnel need. The following points describe the use case diagrams components, advantages, and their usage [22, 28].

- Use cases are used in the system general requirements specification phase.

- Use cases document the behavior of the system from the users' perspective.
  - User means anything externally interacting with the system. It can be another system, software, or person. An actor is a role played by an outside entity that interacts directly with the system.
  - An actor can be a person, a machine or a program.
  - An Actor is drawn as stick figures in use case diagrams. **Actor**



- A use case describes the possible interactions among the system and one or more actors who request service from the system.
  - Service link is referred to as a use case.
  - A use case is a description of a set of potential scenarios like Check balance, Account withdrawal, Account deposit, etc.

- Individual use cases are represented by named ovals in use case diagrams.

- A use case involves a sequence of interactions between the actor/actors and the system.
- In a use case, the system is considered as a black-box, since the concern is the external system's behavior.
- A use case **extend** another use case when it embeds new optional behavior into a complete base case.
  - Utility payments, in Figure 5.1, extend the base case Account transactions.
  - It is not necessarily to pay utilities whenever account transaction is needed.

- A use case **include** another use case when it embeds a subsequence as a necessary part of a larger case.
  - Include relationship permits the same behavior to be embedded in many otherwise unrelated use cases.
  - For example, Account transactions use case **include** Print receipt use case.
- The differences between extend and include are:
  - In the extend, the extended use case is a valid use case by itself.
  - In the include, the use case which is using the other use case is not complete without it.

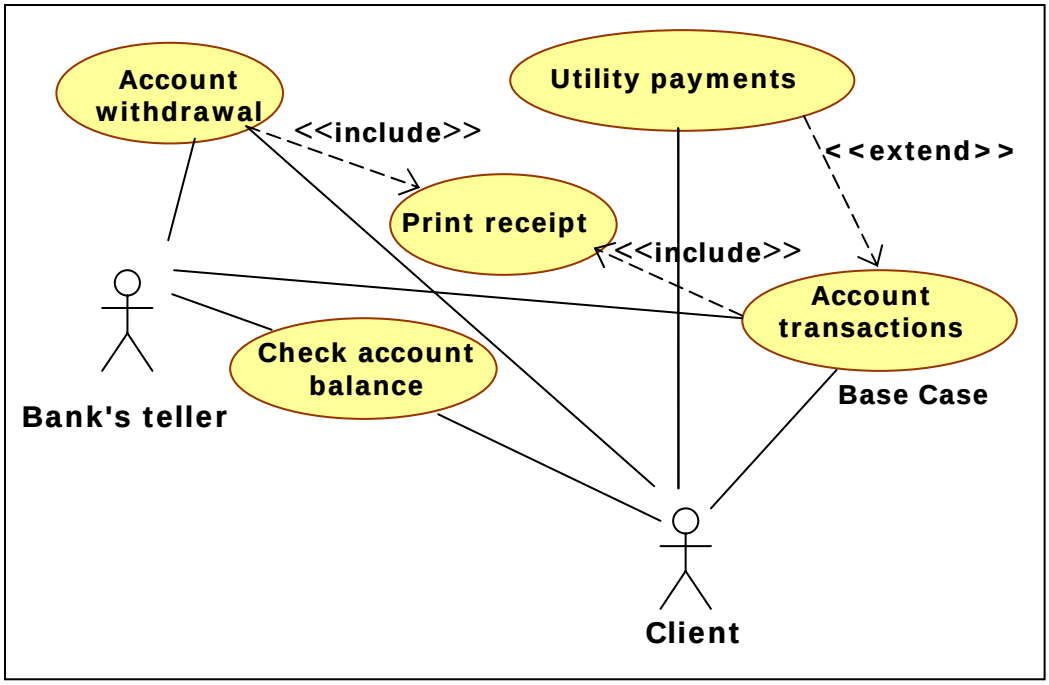


Figure 5.1 Use Case Diagrams: Combining Use Cases

Figure 5.2 shows a practical example of an Automated Teller Machine (ATM). is presented. The ATM machines allow bank's customers to perform several financial processes such as, view account balance, withdraw money from account, deposit money into account, and pay bills. The use case example views the system from the user's perspective. It shows the functional requirements.

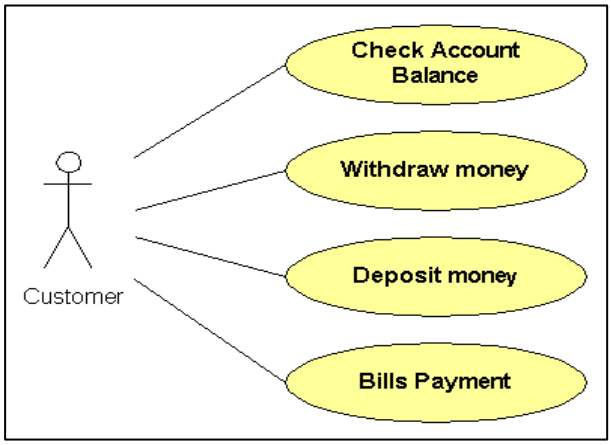


Figure 5.2 Use Case Diagram for the ATM System from User's Perspective.

When it comes to security requirements, the normal use cases need to be modified to incorporate the security aspects in the model.

5.4. Security Use Cases and Misuse Cases

Security use cases are used to specify security requirements that the system shall successfully protect itself from by considering the relevant security threats. The security requirements should be based on an analysis of the assets, services to be protected, and the security threats on these assets and services.

Figure 5.3 shows a security use case example that incorporates security requirements such as, control access, privacy, non-repudiation and integrity. The security requirements are not normally seen explicitly by the user, therefore they are called non-functional requirements [29].

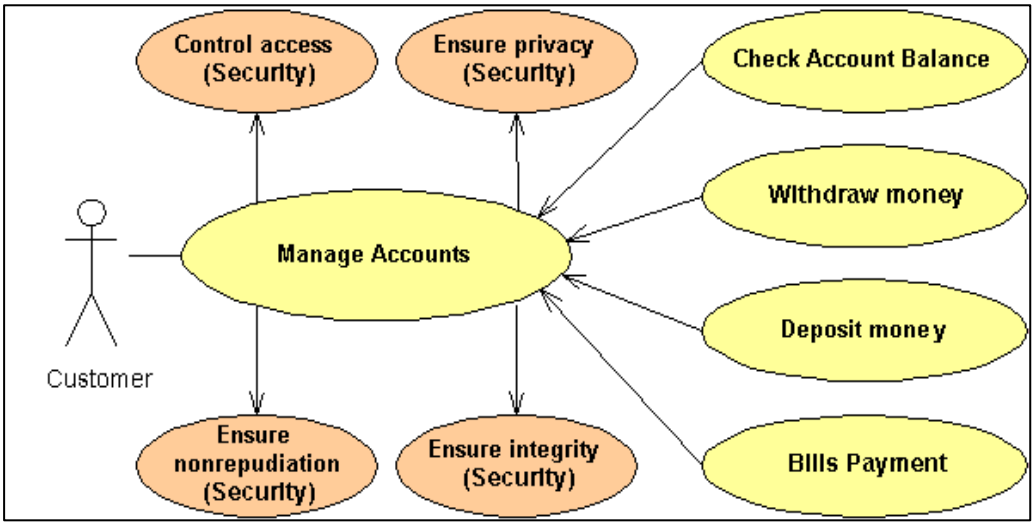


Figure 5.3 Security Use Cases for ATM - Elicit Security Requirement (Reproduced with modification from [29])

In most cases, the elicited security requirements are highlighted after discovering system assets and services that are vulnerable to certain security threats. Security mechanisms are needed to counter the threats and protect these assets. Figure 5.4 shows Assets and Services, Security threats, Security requirements, and Security mechanism relationships [29].

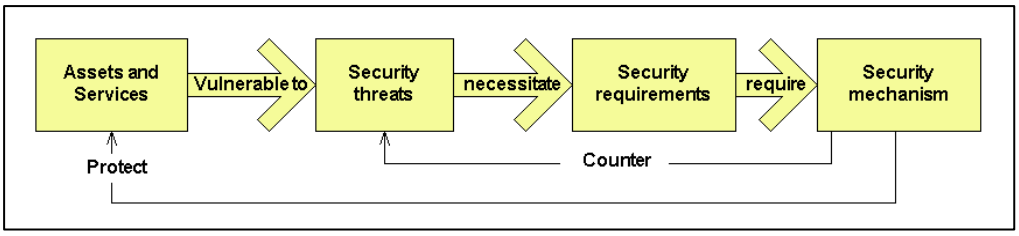


Figure 5.4 Assets and Services, their Security Threats, Requirements and Mechanisms (Reproduced from [29])

A new approach for addressing security threat analysis is the development of misuse cases, also known as abuse cases [29]. Misuse cases are special scenarios of use cases used to analyze and elicit security threats. Misuse cases concentrate on the interactions between the application and the misuser who seek to affect the security. As explained in Figure 5.5, the security use cases are driven by the misuse cases [29].

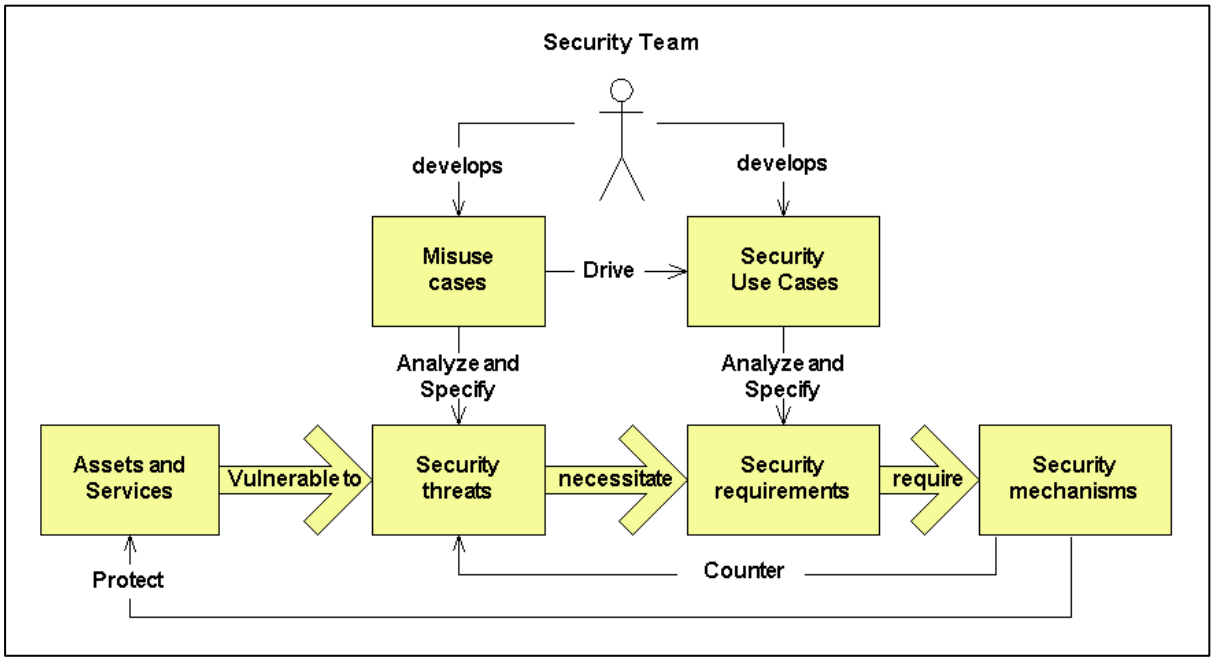


Figure 5.5 Misuse Cases vs. Security Use Cases (Reproduced from [29])

Misuse cases are highly effective ways of analyzing security threats, however use cases are highly effective in analyzing the specification of security requirement. Table 5.1 shows the primary differences between security use cases and misuse cases.

Table 5.1 Security Use Cases and Misuse Cases Comparison. (Reproduced with modification from [29])

	Security Use Cases	**Misuse Cases**
Purpose	Specify and analyze security requirements	Specify and analyze security threats
Success criteria	System succeeded	Misuser Succeeds
Produced by	Security Team	Security Team
Driven by	Misuse Cases	Asset Vulnerability Analysis, and Threat Analysis
Used by	Requirements Team	Security Team
Actors	User	Misuser, User

To show the association between normal use cases, security use cases and misuse cases, Figure 5.6 shows the normal use cases for ATM systems, which include balance check, bills payment, deposit notes or withdrawals. Security use cases include access control; ensure privacy, integrity, and non-repudiation. These security use cases specify the security requirements that protect the system including the user from three security threats tackled by thieves or crackers. Cracker and thief are the misusers [29].

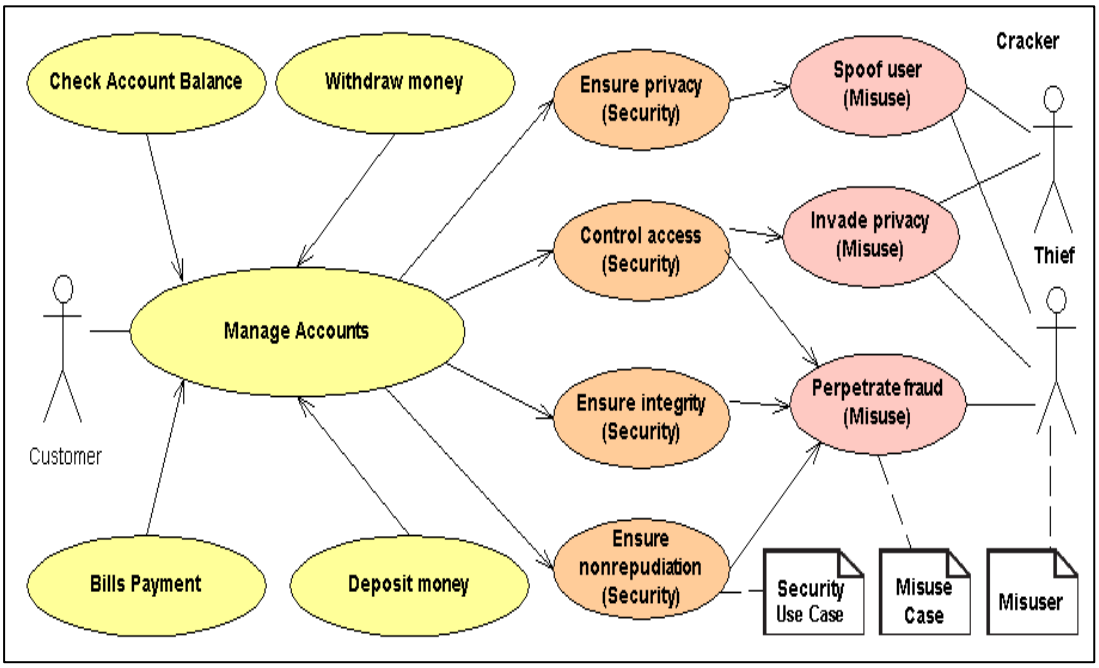


Figure 5.6 Security Use Cases and Misuse Cases Example
(Reproduced with modification from [29])

So far, the use cases are demonstrated as a good tool for describing the system's functional requirements.

5.5. Activity Diagrams

Activity diagrams are normally used to explain the details of use cases. They also model the workflow of system activities including the actions, activities and the order in which involved objects will perform them.

Figure 5.7 summarizes the activity diagrams icons, their use in showing the flow among activities, and actions associated with each object [28].

- Activity: represents a task that has to be performed.
- Action: changes the state of the system or returns a value.

- Transitions: can have guard conditions.
- Conditional branches: correspond to if-then-else scenarios.
- A branch is shown as a diamond.
- A branch can have one incoming transition and two or more outgoing.
- Forks and joins are used to model concurrent execution paths.
- Allows parallelism and synchronization
  - forks create concurrent paths
  - joins merge different paths
  - forks and joins allow synchronization

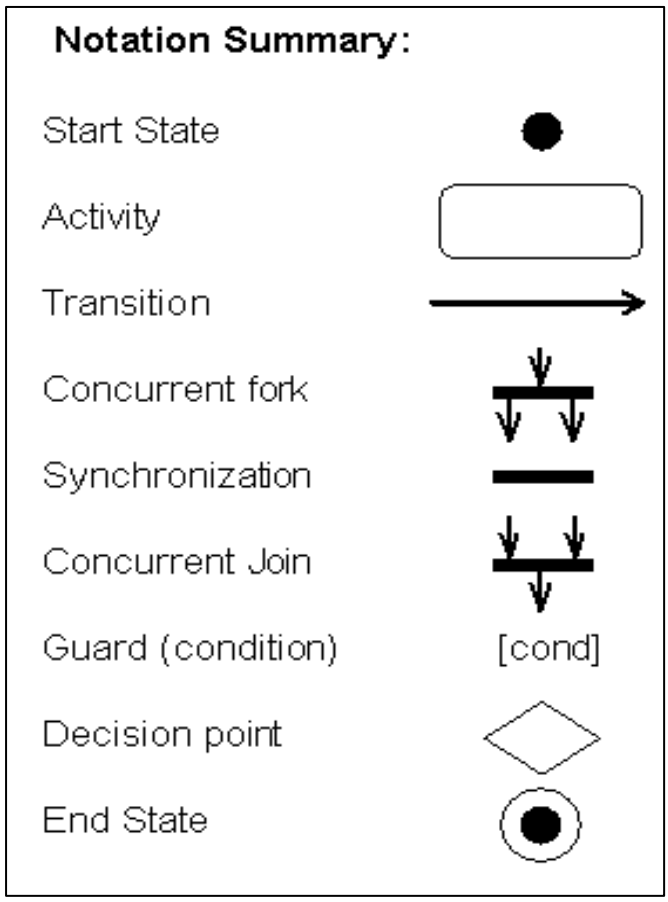


Figure 5.7 Activity Diagram Symbols and their Corresponding Use

Figure 5.8 shows the activity behavior of the ATM machine after authentication and selecting check account balance from the main ATM menu. This example use Start, Activity, and End symbols.

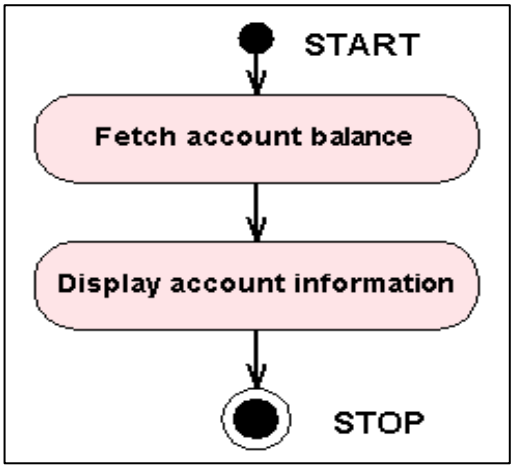


Figure 5.8 Activity Diagram for ATM Account Balance Request.

The second activity diagram example describes the actions/activities involved in money withdrawal. It includes the same symbols used in check account balance scenario in addition to the decision points and guarded conditions icons as shown in Figure 5.9.

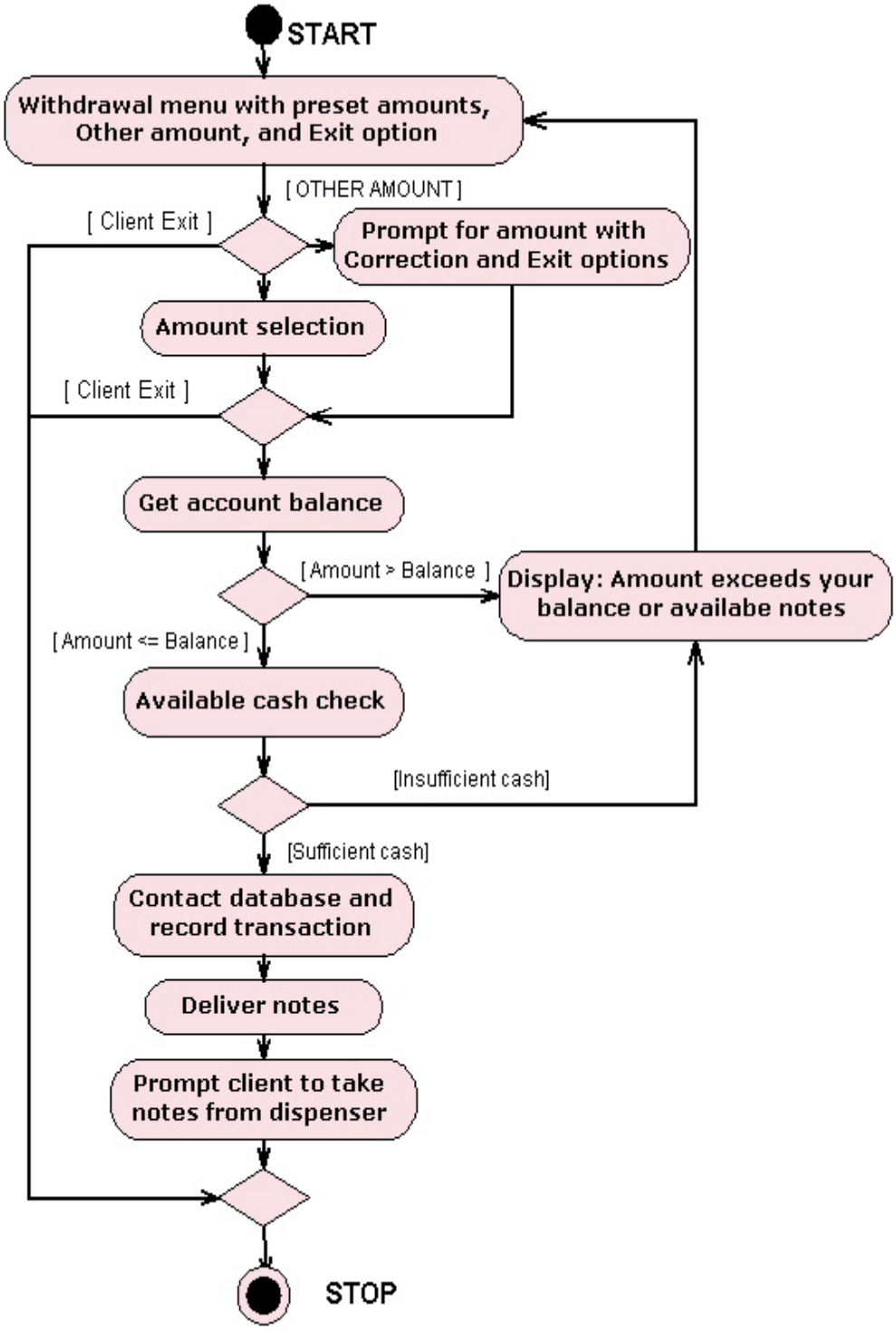


Figure 5.9 Activity Diagram for ATM Account Withdrawal Process

The third activity diagram example describes deposit money in an account scenario. It uses the same icons used in account withdrawal process as shown in Figure 5.10.

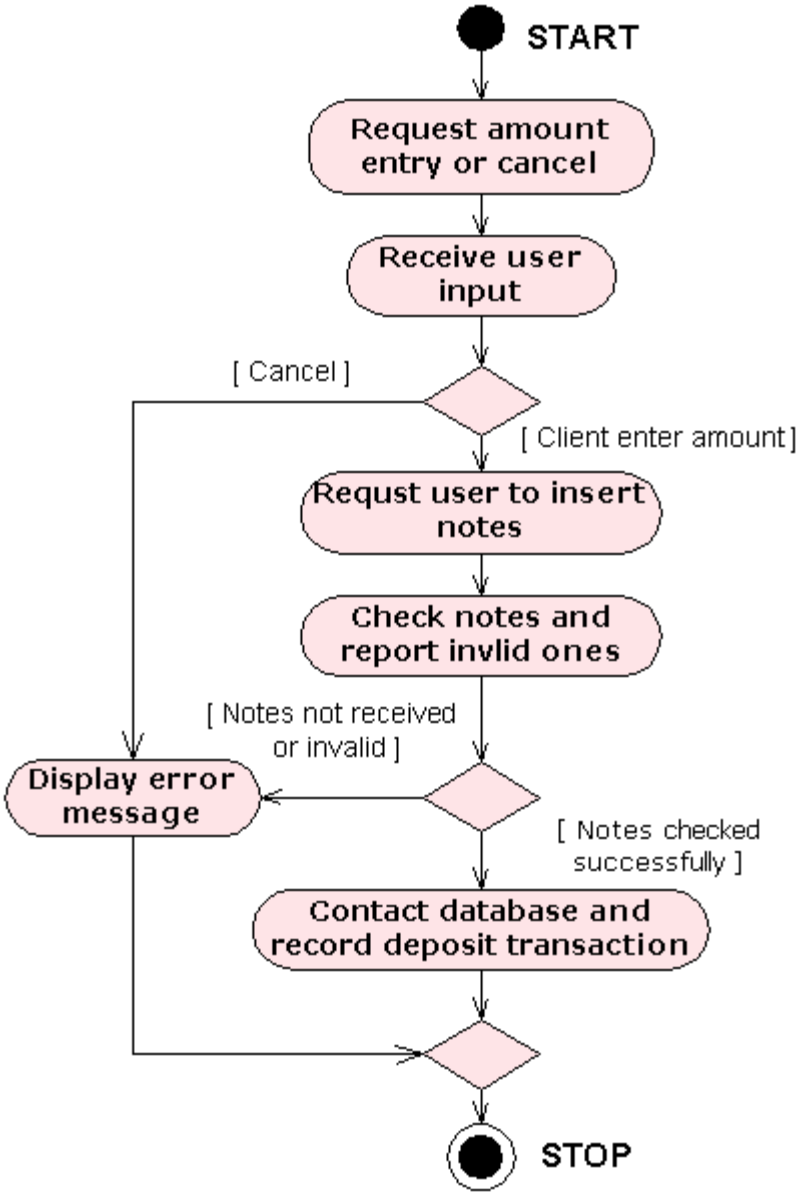


Figure 5.10 Activity Diagram for ATM Account Deposit Process

The fourth Activity diagram example is a modified version of the previous money withdrawal activity diagram, by including the card authentication phase. New symbols are introduced in this example such as fork and join synchronization symbols as depicted in Figure 5.11.

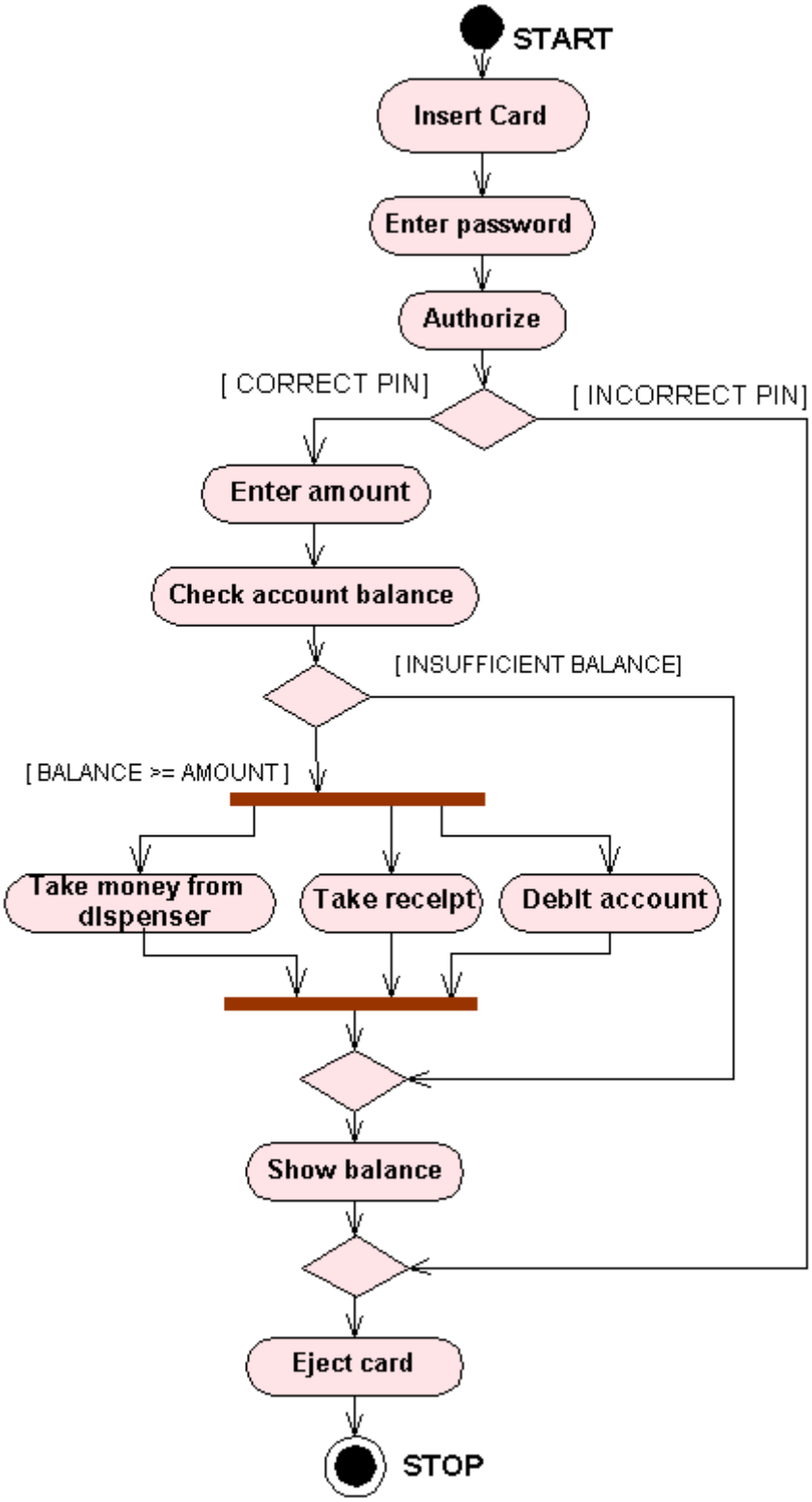


Figure 5.11 Activity Diagram for ATM Account Authentication and Money Withdrawal

5.6. Sequence Diagrams

The sequence diagram consists of objects represented by named rectangles with the name underlined, messages represented as solid-line arrows, and time represented as a vertical progression as shown in Figure 5.12. A sequence diagram shows sequences of messages with the emphasis on the order in which they are exchanged [28].

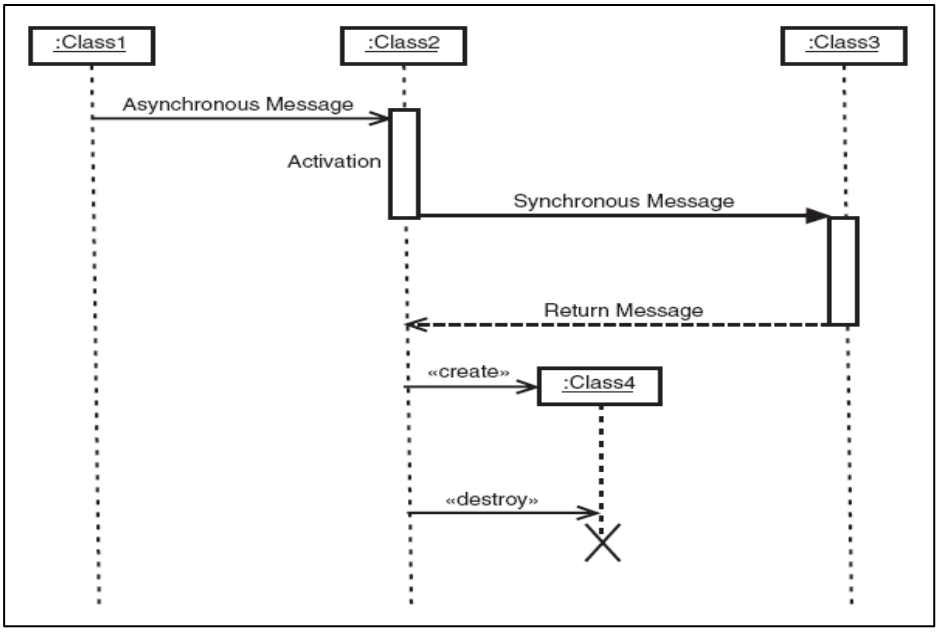


Figure 5.12 Sequence Diagram Illustration

(Reproduced from [28])

Objects

The objects are laid out near the top of the diagram from left to right. They're arranged in any order that simplifies the diagram, such as Class 1 to 4 in Figure 5.12. Extending downward from each object is a dashed line called the object's lifeline. Along the lifeline is a narrow rectangle called activation. The activation represents an execution of an operation the object carries out. The length of the rectangle signifies the activation's duration. Duration and time in general, are represented in a rough, ordinal way.

Messages

A message starts from one object's lifeline to the other object's lifeline. An object can also send a message to itself, from its lifeline back to its own lifeline.

One type of message is a call. This is a request from the object sending the message to the object receiving the message. The request is for the receiver to carry out one of its

operations. Usually, this entails the sender waiting for the receiver to carry out that operation. Because the sender waits for the receiver acknowledgement, this message is also referred to as synchronous.

UML signifies this message type with a filled arrowhead at the end of a solid line.

It's typically the case that a call involves a return message from the receiver, although modelers often omit the symbol for the return message. The symbol for the return message is an open-stick arrowhead with a dashed line. Figure 5.12 also shows these symbols.

Time

The diagram represents time in the vertical direction: Time starts at the top and progresses toward the bottom. A message that's closer to the top occurs earlier in time than a message that's closer to the bottom.

Thus, the sequence diagram is two-dimensional. The left-to-right dimension is the layout of the objects, and the top-to-bottom dimension shows the passage of time.

Figure 5.12 shows the essential symbol set of the sequence diagram, with the symbols working together. The objects are laid out across the top. Each object's lifeline is a dashed line extending downward from the object. A solid line with an arrowhead connects one lifeline to another and represents a message from one object to another.

Figure 5.13, shows a sequence diagram representation of an ATM deposit money scenario.

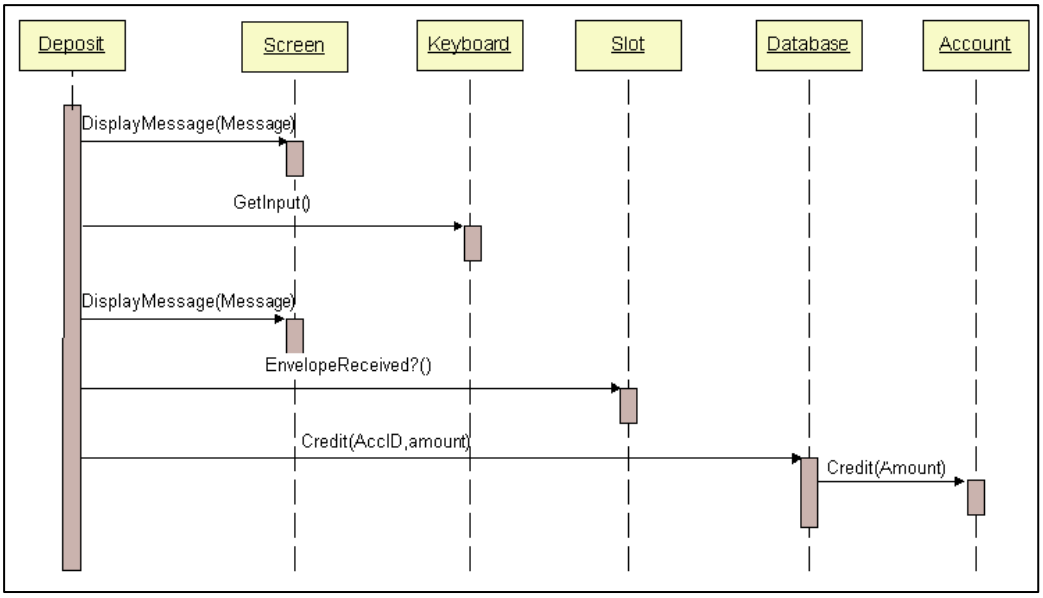


Figure 5.13 Sequence Diagram for ATM Account Deposit Process

The label in the rectangular shape represents the objects. The horizontal arrows are the messages passed between the instances.

Although the UML is developed for object-oriented system, it can be used to analyze non-object oriented systems by dealing with objects as components.

Still there is a challenge in projecting the UML to the security domain so there is a need to extend UML and develop advanced tool support for designing secure systems with the UML including, analyzing UML specifications with respect to the security requirements.

5.7. The UML Extension for Security: UMLsec

Security goals such as Confidentiality, Integrity, and Availability (CIA), in addition to accountability are offered as specification elements and stereotypes by the UMLsec extension [2]. UML sec properties are used for evaluating different kinds of security-related diagrams. The UMLsec encapsulates knowledge on prudent security engineering and makes it available to system and networking engineers who may not be expert in security. UMLsec extensions are based on the standard UML extension mechanism. Stereotypes are used with tags for security requirements formulation. The constraints criteria determine whether the requirements are met or not. The UMLsec extension has been developed by Jurjens [2] based on his security-critical system development experience in industrial projects involving government agencies, banks, insurance companies, smart cards and other companies. The extension provides core profile that includes the main security requirements. A detailed description of UMLsec is provided in Chapter 6 of this thesis.

CHAPTER 6

SECURE UNIFIED MODELING LANGUAGE (UMLsec)

So far, the UML and the security requirements are discussed as prerequisite for UMLsec, an extension of the Unified Modeling Language that permits the expression of security relevant characteristics within the model in a system specification. Modelers will be able to check whether the constraints associated with the UMLsec stereotypes are fulfilled in a given specification model. Stereotypes are extending the UML language by allowing the creation of new elements out of existing ones. There are three extension mechanisms for extending the UML: stereotypes, constraints, and tagged values [2].

Stereotypes: appearing inside guillemets << Stereotype>> or double angle brackets, define new types of modeling elements extending the semantics of existing types in the UML metamodel. The new element can capture some aspect of a new area in ways that standard UML elements can't. In addition to creating additional stereotypes, the UML includes an extensive set of ready-made stereotypes.

Constraints: Constraints supply conditions and restrictions for UML model elements. A constraint can be specified in any format as long as it is written inside braces. If, for example, a class has *password* as one of its attributes, constraint can be applied as {password can't be blank}

Tagged Values: are a name-value pairs in curly brackets {} associating data with model elements. A tagged value is designed to explicitly define a property. For example, If nodeName represents the node where the component resides, {location = nodeName} may be attached to a component.

A UML extension collects stereotypes, tagged values, and constraints into a profile. For UMLsec, validation rules evaluating a model against included security requirements are given by extending formal semantics for the used fragment of UML in a modular way with a formal notion of an adversary.

The following are examples of encapsulated UML stereotypes and tags in the UMLsec profiles.

Fair exchange: When trading electronically, the requirement *fair exchange* stipulates that the trade is performed in a way that prevents parties from cheating. What is specifically meant here is the requirement that after a prepayment, the buyer either receives the purchased good, or the money can be reclaimed.

Secrecy/confidentiality: One of the main data security requirements is secrecy or confidentiality, meaning that some information will become known only to legitimate parties.

Secure information flow: Sometimes partial leakage of information must be prevented. The notion of *secure information flow* ensures that when trusted parts of a system interact with untrusted parts, there is no partial leakage of secret information from the trusted to the untrusted part.

Secure communication link: This requirement ensures that the physical communication links between different parts of the system provide the required security guarantee regarding a given adversary model. For example, a local area network (LAN) is a secure link with respect to secrecy against *outsider* attackers.

6.1. UMLsec Examples

6.1.1. UML Model Extension and Fair Exchange Example

Figure 6.1 describes a customer buying a good from a business. The semantics of the stereotype ‹‹fair exchange›› has two tags: {start} and {stop}, which take the state names as values. A subsystem S meets the conditions of the stereotype ‹‹fair exchange››, if accomplishing an action in {start}, leads ultimately to the execution of a {stop} tag. The subsystem meets the conditions as soon as the customer pays. Either the order is delivered on time or the customer reclaims his money.

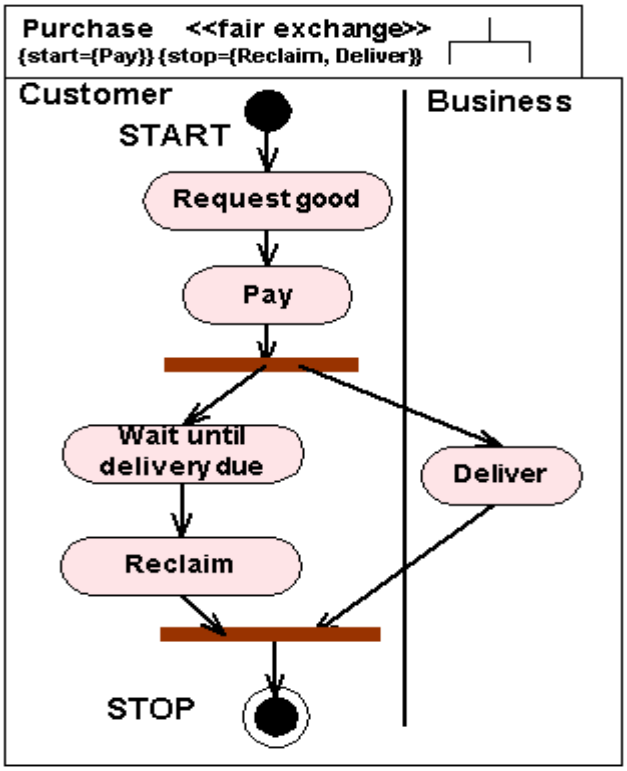


Figure 6.1 Fair Exchange Activity Diagram (Reproduced from [30])

In Figure 6.1, the condition for the stereotype <<fair exchange>> is violated; if the action "Pay" in {start} is accomplished, then action "Deliver" or "Reclaim" are both accomplished too in {stop} state.

The model is corrected by adding the condition after "Wait until delivery due" action as in Figure 6.2. The customer can only reclaim after a specific waiting time and the goods are not delivered.

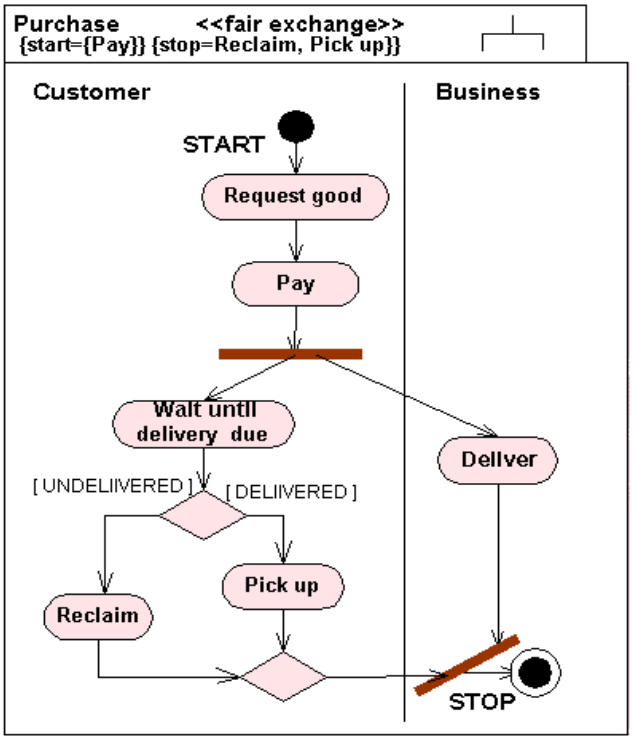


Figure 6.2 Corrected Fair Exchange Activity Diagram (Reproduced from [2])

6.1.2. Deployment Diagram and E-Banking Example

The deployment diagrams show the configuration of run time processing nodes, the component's processes and objects that reside on them. The nodes are represented by cubes, which may be connected by solid line to represent communication links. A node could contain components shown as rectangles with two smaller rectangles inserted on the left side. Components may be connected by broken arrows representing communication dependencies. They could have interfaces and contain class or subsystem models. Links represent physical communication links between different nodes in a system. Dependencies describe logical connections between components.

The e-Banking web service connectivity example Figure 6.3 shows a deployment diagram with four node instances: Customer machine, e-Banking Web Server, e-Banking Application Server, and e-Banking Database Server.

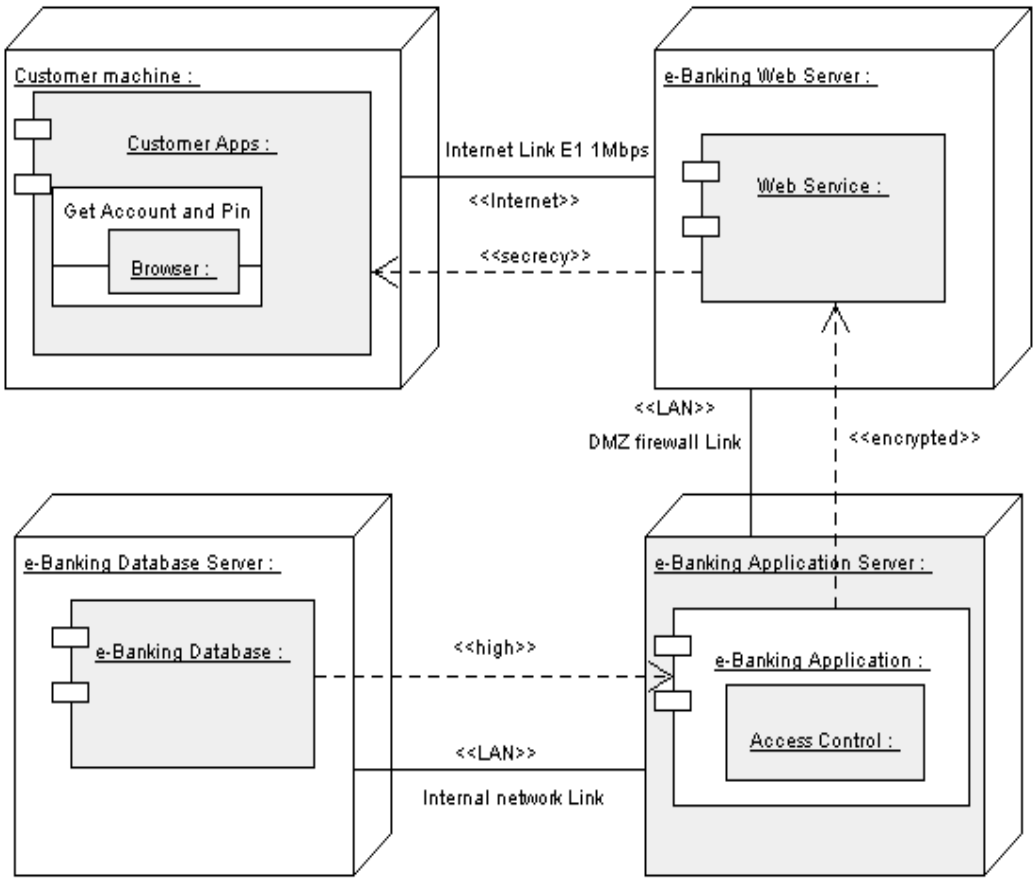


Figure 6.3 E-Banking Web Connectivity Deployment Diagram

The node instance Customer machine contains a component instance Customer Apps with "Get Account and Pin" which contain a Browser object. The node instance e-Banking Web Server contains a component instance Web Service. The node instance e-Banking Application Server contains a component instance e-Banking Application which contain Access Control object. Finally the node e-Banking Database Server contains e-Banking Database instance.

Customer machine and e-Banking Web Server nodes are connected via a link stereotyped <<Internet>>, and there is a <<secrecy>> stereotype dependency from the e-Banking Web Server component to "Get Account and Pin", thus the e-Banking Web Server is able to communicate with the Browser for getting the account and pin code.

<<Secrecy>> and similar dependency stereotypes like <<integrity>>, and <<high>>, denote dependencies that are supposed to provide the respective security requirement for the data that is sent along them as arguments or return values of operations or signals. These stereotypes are used in the constraint for <<secure links>> stereotype.

6.2. UMLsec Stereotypes

UMLsec stereotypes, as mentioned earlier, are used to stipulate security requirements in access control, confidentiality, integrity, etc. In this section, the UMLsec stereotypes are explained briefly. The author proposes new stereotypes needed for auditing, immunity and survivability security requirements. They include <<*audit log*>>,<<*multiplicity*>> and <<*no misuse*>>. These new stereotypes, as well as the original UMLsec stereotypes defined by Jurjens [2, 30, 31, 32], will be explained in the following.

***Audit Log*:** <<*audit log*>> stereotype can be used by the node elements to completely record any transaction information or activity. The logging fields include: event ID, event date, event time, event user, and event action (allowed, or denied). Audit logs records are tamperproof since they include the integrity tag. The author introduces <<*audit log*>> stereotype to fulfill several security requirements such as audit log, non-repudiation, and intrusion detection system security requirements.

Critical: This stereotype labels objects that are critical in some way. They are specified in more detail using the corresponding tags. The tags are {secrecy}, {integrity}, {authenticity}, {high}, and {fresh}, representing a specific security requirements as explained in Chapter 3. The values of {secrecy} tag are the names of attributes, expressions, or message argument variables of the current object the secrecy of which is supposed to be protected. The {integrity} tag has 'v' and 'E' pair values where. 'v' is a variable of the object in which its integrity need to be protected. 'E' is a set of acceptable expression that can be assigned to 'v'. {authenticity} tag is also pair of values 'a' and 'o' of attributes of the <<critical>> object. 'a' stores the data whose authenticity should be provided and 'o' stores the origin of the data. Tag {high} values are the names of messages that are supposed to be protected with respect to secure information flow as enforced by other stereotypes <<no down-flow>>, and <<no up-flow>>. Tag {fresh} has automatic data from the Keys and Data sets that should be freshly generated.

Data Security: Data security requirements on «critical» classes are met as follows. <<data Security>> subsystem stereotype respects the data security requirements given by the stereotypes <<critical>> and the associated tags contained in the system with the

following four <<data security>>. {secrecy} ensures that the subsystem preserves the secrecy of the data designated by tag {secrecy} against adversaries A. {integrity} of <<critical>> with values 'v', and 'E', the subsystem preserves the integrity of the variable 'v' against adversaries 'A', with respect to 'E' admissible expressions. {authenticity} is provided for attribute 'a' with respect to its origin 'o' against adversaries 'A'. {freshness} means that any data within subsystem S of <<data security>> stereotype should be fresh.

Encrypted, Internet, LAN, *Multiplicity*, and Wire: Those links stereotypes are used in deployment diagrams to denote the corresponding types of communication links. It is required that each link carries at most one of these stereotypes. For each adversary type 'A', the $Threats_A(s)$ function for each stereotype is described as follows.

$S \in$ {<<issue node>>,<<LAN>>, <<*multiplicity*>>, <<POS device>>, <<smart card>>} to a set of $Threat_A(s) \subseteq$ {delete, read, insert}

The <<*multiplicity*>> link stereotype is introduced to fulfill the survivability security requirements. <<*multiplicity*>> stipulates multiple network links per device.

Issuer Node, *Multiplicity*, LAN, POS Device, and Smart Card: Those node stereotypes are used in deployment diagrams to denote the respective kinds of system nodes. It is required that each node carries at most one of these stereotypes. For each adversary type A, the $Threats_A(s)$ function for each stereotype is described as follows.

$S \in$ {<<encrypted>>, <<internet>>, <<LAN>>, <<*multiplicity*>>, <<wire>>} to a set of $Threats_A(s) \subseteq$ {access}.

The <<*multiplicity*>> node stereotype is introduced to fulfill the survivability security requirements. <<*multiplicity*>> stipulates multiple active devices such as servers, routers, switches, or storage.

Fair Exchange: The <<fair exchange>> stereotype is applied to an activity diagram subsystem with {start}, {stop}, and {adversary} tags. When trading goods electronically, the requirement <<fair exchange>> postulates that the trade is performed in a way that prevents both parties from cheating. What is meant here, more specifically the requirement that after a prepayment, the buyer either receives the purchased good or is able to reclaim the money. This is controlled when the tags

{start}, and {stop} take pairs (good, state), where good is the name of the good to be sold and state is the name of a state. If there is only one good to be sold in a given system specification, the value good can be omitted. The associated constraint requires that, whenever a start state in the contained activity diagram is reached, then eventually a stop state will be reached. The tag {adversary} specifies an adversary type relative to which the security requirement should hold.

Guarded Access: Guarded access stereotype means that each <<guarded>> object in the subsystem can only be accessed through the object specified by the tag {guard} attached to the <<guarded>> object.

Guarded: Guarded stereotype labels objects that are supposed to be guarded in the scope of <<guarded access>> stereotype. It has a tagged value {guard} which defines the name of the corresponding guard object.

High, Integrity, and Secrecy: are used as dependency constraint for <<secure links>> stereotype in static structure or component diagrams. Dependencies stereotypes, denote dependencies that are supposed to provide the respective security requirement for the data that is sent along them as arguments or return values of operations or signals.

***No Misuse*:** <<*no misuse*>> can be used with link or node to detect, alert, prevent any misuse of the system, malicious software, Trojans, viruses, and repeated request for system component in an irregular manner.
The <<*no misuse*>> stereotype is introduced to fulfill the immunity security requirements.

No Down-flow: <<no down-flow>> subsystems stereotypes prevent indirect leakage of sensitive data and enforce secure information flow by making use of the tag {high} associated with the stereotype <<critical>>. The constraint for <<no down-flow>> stereotype is that the UML machine for subsystem S prevents down-flow with respect to the message specified in {high} and their return message.

No Up-flow: <<no up-flow>> subsystems stereotypes prevent indirect leakage of sensitive data and enforce secure information flow by making use of the tag {high} associated with the stereotype <<critical>>. The constraint for <<no up-flow>> stereotype is that the UML machine for subsystem S prevents up-flow with respect to the message specified in {high} and their return message.

Provable: <<provable>> stereotype for a subsystem S with {action}, {cert}, and {adversary} associated tags mostly are used in e-commerce systems. The tag {cert} contains an expression which serves as proof that the action as the state given in the tag {action} was performed. The tag adversary specifies an adversary type relative to which the security requirement should hold.

RBAC: <<rbac>> subsystems stereotype, normally used in activity diagrams to enforce Role-based Access Control (RBAC) in the business process specified in the activity diagram. {protected}, {role}, and {right} are the <<rbac>> associated tags. The {protected} tag has as its values the states in the activity diagram and the access to whose activities should be controlled. The {role} tag may have as its value a list of pairs (*actor, role*) where *actor* is an actor in the activity diagram, and *role* is a role. The {right} tag has as its value a list of pairs (*role, right*) where *role* is a role and *right* represents the right to access a protected resource. The associated constraint requires that the actors in the activity diagram only perform activities for which they have the appropriate rights.

Secure Links: <<secure links>> stereotype are used in subsystems containing deployment diagrams. This stereotype is used to ensure that security requirements on the communication are met by the physical layer, given the adversary type A that is specified in the tag {adversary} associated with it. When attached to the UML subsystem S, the constraint enforces that for each dependency *d* with stereotype *s* $\in$ {<<secrecy>>,<<integrity>>, <<high>>} between subsystems or objects on different nodes *m, n* via the link *l*, such that:

when s=<<high>>, then $\text{Threats}^{S}_{A}(l)=\varnothing$,

when s=<<secrecy>>, then read $\notin$ $\text{Threats}^{S}_{A}(l)$ and

when s=<<integrity>>, then insert $\notin$ $\text{Threats}^{S}_{A}(l)$.

Secure Dependency: <<secure links>> stereotype on subsystems containing static structure diagrams ensures that the <<call>> and <<send>> dependencies between objects or subsystems respect the security requirements on data that may be communicated across them. This is stipulated by the tags {secrecy}, {integrity} and {high} of the stereotype <<critical>>.

6.3. UMLsec Stereotypes, Tags and Constraints Summary

As a summary, Table 6.1 lists the UMLsec stereotypes along with the tags and constraints associated with them [2, 31].

The italicized stereotypes were introduced in the work to enhance the expressive power of the UMLsec.

Table 6.1 UMLsec Stereotypes [2], with the additional stereotypes

Stereotype	Base Class	Tags	Constraints	Description
critical	Object, subsystem	secrecy, Integrity, authenticity, high, fresh,		Critical object
audit log	Subsystems, node	integrity		Log event activity
data security	Subsystem	adversary, integrity, authenticity	Provides secrecy, integrity, authenticity, freshness	Basic data security requirements
encrypted	Link			Encrypted connection
fair exchange	Subsystem, package?	start, stop, adversary	After start eventually reach stop	Enforce fair exchange
guarded access	Subsystem		Guarded objects accessed through guards	Access control using guard objects
guarded	Object	guard		Guarded object
high	Dependency			High sensitivity
integrity	Dependency			Assumes integrity
internet	Link			Internet connection
issuer node	Node			Issuer node
LAN	Link, node			LAN connection
multiplicity	Link, node			Load balancing and redundancy requirements
no misuse	Subsystem		Prevent misuse	Antivirus, and network traffic
no down-flow	Subsystem		Prevents down-flow	Information flow condition
no up-flow	Subsystem		Prevents up-flow	Information flow condition
POS device	Node			POS device
provable	Subsystem	action, cert, adversary	Action is non-deniable	Non-repudiation requirement
rbac	Subsystem	protected, role, right	Only permitted activities executed	Enforces role-based access control
secure links	Subsystem	adversary	Dependency security matched by links	Enforces secure communication links
secrecy	Dependency			Assumes secrecy
secure dependency	Subsystem		<<call>>, <<send>> respect data security	Structural interaction data security
smart card	Node			Smart card node
wire	Link			Wire

6.4. UMLsec Stereotypes Threat Modes

UMLsec stereotypes are designed to protect systems against two categories of attackers: the default and insider attackers [2].

6.4.1. Threats from Default Attacker

Default attacker represents an outsider adversary with modest capability. Such attacker can read, delete and insert messages on the Internet link. On an encrypted link like Virtual Private Network (VPN), the attacker might still be able to delete messages without knowing their encrypted content. Also the attacker can stop the service by launching a Denial of Service (DOS) attack against a legitimate service. However, an average adversary would not be able to read the plaintext messages or insert messages encrypted with the right key. It is assumed that the encryption is setup in a way that the adversary does not get hold of the secret key. It is also assumed that the default attacker does not have direct access to the Local Area Network (LAN), and therefore, is not able to eavesdrop or sniff on the connection, nor on wires connecting security critical devices such as smart card readers or Point of Sale (POS) devices. It is also assumed that the default attacker is not able to access card issuer systems and POS devices. Table 6.2 summarizes default attacker threats against different types of stereotypes. The threats {delete, read, and insert} allow the adversary to delete, read, and insert messages on a communication link, respectively.

Table 6.2 Default Attacker Threats (Reproduced from [2])

Stereotype	**Threats** $_{default}$**()**
no misuse	∅
internet	{delete, read, insert}
encrypted	{delete}
LAN	∅
wire	∅
smart card	∅
POS device	∅
issue node	∅

However, an external adversary may be able to access local connections. But this is assumed to be beyond the default capabilities.

6.4.2. Threats from Insider Attacker

An insider attacker represents an insider adversary or disgruntled employee with modest capability. Consider an insider attacker as an employee working at a card issuer institution. Such attacker can read, delete and insert messages on the Internet link, encrypted link, LAN link, and wire link. As mentioned before, the threats {delete, read, and insert} allow the adversary to delete, read, and insert messages on a communication link, respectively. There is an access threat representing the possibility that an adversary may directly access a physical system node. Access threat with respect to a node contained in the system part is further broken down into delete, read and insert threats.

Table 6.3 considers that the insider attacker may access the encrypted Internet link, knowing the corresponding key, and the local system components.

Table 6.3 Card Issuer *Insider* attacker threats (Reproduced from [2])

Stereotype	**Threats $_{insider}$()**
no misuse	{delete, read, insert}
internet	{delete, read, insert}
encrypted	{delete, read, insert}
LAN	{delete, read, insert}
wire	{delete, read, insert}
smart card	∅
POS device	{access}
issue node	{access}

So the issuer node insider can also access the issuer node.

For a node stereotype s $\quad Threats_A(s) \subseteq \{access\}$

For link stereotype s $\quad Threats_A(s) \subseteq \{read, insert, delete\}$

CHAPTER 7

SECURITY STANDARDIZATION

An overview of the security standards such as ISO/IEC 17799:2000[3], Common Criteria (CC) Redbook [6], and COBIT [4, 5] is illustrated in this chapter.

7.1. BS 7799/ ISO/IEC 17799:2000

ISO/IEC 17799:2000 is an international standard covering most information security aspects with, a comprehensive set of controls comprising best practices in information security. This standard is intended to serve as a single reference point for identifying the range of controls needed in most situations where information systems are used in industry and commerce, and used by large, medium and small organizations. It includes equipment, management policies, human resources, and legal aspects [33, 34].

International Standard ISO/IEC 17799 was prepared by the British Standards Institution (BSI), as BS 7799. It was adopted by Joint Technical Committee ISO/IEC JTC 1, Information technology, in parallel with its approval by national bodies of International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). ISO 17799 - part 1 is a guide containing controls and recommendations by which an organization can ensure the security of its information. Part 2 proposes measures for an efficient information security management framework. BS 7799-2 helps an organization establish an information security management system (ISMS) [33].

ISO/IEC 17799 historically started in 1995 with BS 7799 Part 1, followed by BS 7799 Part 2 in 1998, followed by updated version of BS 7799 Parts 1 and 2 in 1999, then ISO/IEC 17799 in 2000.

There are several advantages of being ISO/IEC 17799:2000 certified. These include improving privacy practices and compliance with privacy laws, structuring security methodology and having international recognition, reduce hacker attacks risk, easier and faster recovery from attacks, compliance with governance rules for risk

management, better protection of the company's confidential information, potentially lower the premiums for computer risk insurance, and increase mutual confidence between parties.

BS 7799/ISO/IEC 17799:2000 ensures the information security goals Confidentiality, Integrity, and Availability (CIA triad), in addition to accountability by the control areas listed in Table 7.1.

Table 7.1 ISO/IEC17799:2000 (BS7799-1:2000) Contents

ISO/IEC 17799:2000	Description
1	Scope
2	Terms and definitions
3	Security policy
4	Organizational Security / Security organization
5	Asset classification and control
6	Personnel Security
7	Physical and environmental security
8	Communication and operation management
9	Access Control
10	System development and maintenance
11	Business Continuity Management
12	Compliance

Since Information Technology (IT) security includes all aspects related to defining, achieving and maintaining the five security services of identification, authentication, authorization, confidentiality, integrity and non-repudiation in addition to availability. BS 7799/ISO/IEC 17799:2000 can be used by any organization or company that are using computer systems internally or externally, possesses confidential data, depends upon information systems in the context of its business activities, or simply wants to adopt a high level of security while complying with standards. Several companies, organizations and institutions are adopting ISO/IEC 17799:2000. Most of these companies are headquartered in the United States and Europe [35].

In the system security reengineering processes and standardization mapping, the reference will be to the ISO/IEC 17799:2000 clause. For detailed and complete ISO/IEC 17799:2000 clauses see [3].

7.2. Control Objectives for Information and Related Technologies (COBIT)

COBIT: Control Objectives for Information and related Technology has been developed as a generally applicable and accepted standard for good Information Technology (IT) security and control practices that provide a reference framework for management, users, and IS audit, control and security practitioners. Originally released as an IT process and control framework linking IT to business requirements, and initially used by the assurance community in conjunction with business and IT process owners. Beginning with the addition of Management Guidelines in 1998, COBIT is now being used more and more as a framework for IT governance, providing management tools such as metrics and maturity models to complement the control framework [36].

7.2.1. COBIT Mission

The COBIT mission is to research, develop, publicize and promote an authoritative, up-to-date, international set of generally accepted information technology control objectives for day-to-day use by business managers, IT professionals and assurance professionals.

7.2.2. COBIT Advantages

COBIT has several advantages and characteristics such as:

- It is compliant with ISO/IEC 17799:2000 and maps onto many related standards.
- It is increasingly accepted internationally, based on the professional and practical experiences of worldwide experts.
- COBIT is a way to bridge the communication gap between IT functions, the business and auditors, by providing common understandable approach.
- COBIT is management-oriented, actionable and easy to use.
- COBIT provides strong support for IT audit, reduces the cost of audit risk assessment, and enables a higher quality of audit and related opinion.
- COBIT is a flexible and adaptable approach to suit every organization’s unique cultures, size and specific requirements.

- COBIT is complete, objective and continually evolving and is maintained by a reputable non profit-oriented organization.
- COBIT avoids reinventing the wheels and shortens the time required to implement effective practices.

7.2.3. COBIT Components

The COBIT is categorized into seven components: Executive Summery, Framework, Control Objectives, Control Practices, Management Guidelines Audit Guidelines, and Quickstart. They are briefly described below.

Executive Summary

COBIT Executive Summary explains COBIT key concepts and principles.

Framework

COBIT Framework is the basis of the COBIT approach and the foundation for all the other COBIT elements. The process model is organized into four domains: Plan and Organize, Acquire and Implement, Deliver and Support, and Monitor and Evaluate [4].

Control Objectives

COBIT's Control Objectives component provides more than 300 generic control statements that define what needs to be managed in each IT process to address the business requirements of ensuring IT delivers value, risks are managed and requirements are met.

Control Practices

Control Practices provide guidance on why controls are needed and what the best practices are for meeting specific control objectives. Control Practices help ensuring that solutions put forward are complete and successfully implemented.

Management Guidelines

COBIT Management Guidelines provide tools to help IT managers improve IT performance and link IT objectives to business objectives.

Audit Guidelines

Audit Guidelines outlines and suggests which assessment activities should be performed for each of the 34 high levels IT control objectives, providing helpful

guidance on who to interview, what questions to ask, and how to evaluate control, assess compliance and finally, substantiate the risk of the controls not being met.

COBIT Quickstart

COBIT Quickstart is specifically designed to assist in rapid and easy adoption of the most essential elements of COBIT. Quickstart was designed as a baseline for many Small to Medium Enterprises (SMEs). Nevertheless, it is also suitable for large organizations as a useful tool to accelerate adoption of governance best practices.

7.2.4. Business Drivers for Implementing COBIT Guidance

The following business cases usually trigger COBIT implementation.

- Services delivered by IT are to be aligned with business goals.
- There is a need for IT governance.
- IT processes are to be standardized/automated.
- A framework for overall IT processes is needed.
- Processes are to be unified.
- There is a need of a framework for a quality management system.
- A structured audit approach is to be defined.
- Mergers and acquisitions are occurring.
- IT cost-control initiatives are desired.
- Part or all of the IT function is to be outsourced.
- Compliance with external requirements is of concern

7.2.5. Noncompliance Related Risks

Companies that are not adopting COBIT may incur the following risks.

- Weak support of business goals due to misalignment
- Misaligned IT services, divergence
- Wasted opportunities due to misalignment
- Persistence of the perception of IT as a black-box
- Know-how tied to key individuals, not to the organization
- Excessive IT cost and overheads
- Erroneous investment decisions and projections
- Shortfall between management’s measurements and expectations

7.2.6. Control Objectives

COBIT provides a set of 34 high-level control objectives, one for each of the IT processes, grouped into four domains: planning and organization, acquisition and implementation, delivery and support, and monitoring. This structure covers all aspects of information and the technology that supports it. By addressing these 34 high-level control objectives, the business process owner can ensure that the control system provided for the IT environment is adequate.

7.2.7. Management Guidelines

The COBIT management guidelines provide a link between IT control and IT governance. They are action oriented and generic, and provide management specific guidance and direction for getting the enterprise's information and related processes under control, monitoring achievement of organizational goals, monitoring and improving performance within each IT process and benchmarking organizational achievement. They help provide answers to the following typical management questions [5]:

- What are the key performance indicators?
- How far should the IT be controlled, and is the cost-benefit justifiable?
- What are the critical success factors?
- What do others do?
- How the organization's maturity is measured and compare to the current status of (best-in-class in) industry and the current status of international standards?
- What is the organization's improvement strategy?
- What are the risks of not achieving the objectives?

7.2.8. Control Practices

The COBIT IT processes, business requirements and detailed control objectives define what needs to be done to implement an effective control structure. The IT control practices provide the more detailed how and why needed by management, service providers, end users and control professionals to implement highly specific controls based on an analysis of operational and IT risks.

7.2.9. Audit Guidelines

Analyze, assess, interpret, react and implement. To achieve the desired goals and objectives, the enterprise must constantly and consistently audit its procedures. Audit Guidelines outline and suggest actual activities to be performed, corresponding to each of the 34 high-level IT control objectives, while substantiating the risk of control objectives not being met.

7.2.10. Information Criteria

Information delivered to the core business processes has to fulfill certain criteria, which are characterized as follows:

- Quality Requirements
  - **Effectiveness** Deals with information being relevant and pertinent to the business process as well as being delivered in a timely, correct, consistent and usable manner.
  - **Efficiency** Concerns the provision of information through the optimal (most productive and economical) use of resources.

- Security Requirements
  - **Confidentiality** Concerns protecting sensitive information from unauthorized disclosure.
  - **Integrity** Relates to the accuracy and completeness of information as well as to its validity in accordance with business values and expectations.
  - **Availability** Relates to information being available when required by the business process now and in the future. It also concerns the safeguarding of necessary resources and associated capabilities.

- Fiduciary Requirements
  - **Compliance** Deals with complying with laws, regulations and contractual arrangements to which the business process is subjected to.
  - **Reliability** Relates to the provision of appropriate information for management to operate the entity and for management to exercise its financial and compliance reporting responsibilities.

7.2.11. IT Resources

The definitions of IT resources according to COBIT perspective are as follow.

Data: Objects in their widest sense (i.e., external and internal), structured and non-structured, graphics, sound, etc.

Application Systems: Understood to be the sum of manual and programmed procedures.

Technology: Covers hardware, operating systems, database management systems, networking, multimedia, etc.

Facilities: All the resources to house and support information systems.

People: Includes staff skills, awareness and productivity to plan, organize, acquire, deliver, support, monitor and evaluate information systems and services

COBIT contains 302 specific control measures for 34 IT processes, of which Information Security is one of them.

Regarding the IT process of Information Security, COBIT specifies 21 detailed management actions (Clauses DS5.X) which must be performed for security assurance, and 12 IT Services Continuity actions (Clauses DS4.X). The following is a sample security management action.

Clause DS 5.2 Identification, Authentication and Access

The logical access to and use of the information services function's computing resources should be restricted by the implementation of an adequate authentication mechanism of identified users and resources associated with access rules. Such mechanisms should prevent unauthorized personnel, dial-up connections and other system (network) entry ports from accessing computer resources and minimize the need for authorized users to use multiple sign-on. Procedures should also be in place to keep authentication and access mechanisms effective (e.g., regular password changes).

A complete listing of COBIT clauses can be found in [5].

7.3. Common Criteria, Redbook

The Common Criteria (CC) is an international security evaluation certification that is designed to improve and replace the various national evaluation schemes. Work to improve the evaluation criteria has been sponsored by the Communications Electronics Security Group (CESG), UK and its counterparts in Canada, France, Germany, the Netherlands and the USA, with the aim of replacing existing criteria with a single, international standard, the Common Criteria (CC). The existing criteria are Canadian Trusted Computer Product Evaluation Criteria (CTCPEC), Trusted Computer System Evaluation Criteria (TCSEC), Federal Criteria for Information Technology Security (FC), and FC Federal Information Processing Standard (FC-FIPS) [33].

The agreement to develop the CC grew out of a February 1993 European Commission-sponsored workshop in Brussels on the FC-FIPS that was presented by National Institute for Standards and Technology (NIST) and attended by many European security professionals. The USA and Canada had already agreed to align FC and CTCPEC criteria, and the effect of the Brussels meeting was a general agreement that, the time was right to harmonize European and North American security evaluation criteria.

The final objective is that certificates of evaluation issued by any of the sponsoring countries should be recognized by all co-sponsors - reducing the cost for developers and increasing the choice for users. Work began in 1993 and the project produced version 1.0 of Common Criteria in January 1996. These were refined by an extensive period of review and user trials; version 2.0 (final draft) was published in December 1997. Version 2.0 of Common Criteria was finalized in May 1998 and was soon afterwards submitted to the International Standards Organization (ISO) for acceptance as an international standard. Version 2.1 of the Criteria is equivalent to ISO's International Standard 15408, which consist of the following three parts. 1) Part 1, Introduction and general model: An introduction to the CC defines general concepts and principles of IT security evaluation and presents general model of evaluation. Part 1 also presents constructs for expressing IT security objectives, for selecting and defining IT security requirements, and high-level specifications for products and systems description. 2) Part 2, Security functional requirements: Establishes set of

security functional components as a standard way of expressing the security functional requirements for Targets of Evaluation (TOE) or systems. 3) Part 3, Security assurance requirements: Establishes set of assurance components as a standard way of expressing the assurance requirements for TOE. Part 3 also defines evaluation criteria for Protection Profiles (PPs) and Security Targets (STs) and presents evaluation assurance levels that define the predefined CC scale for rating assurance for TOEs, which is called the Evaluation Assurance Levels (EALs), Table 7.2 list EAL seven levels [25].

However, in the domain of e-commerce systems, the Common Criteria Redbook (CC) is intended to be used to evaluate system security only after the system is implemented. It cannot be used directly during the system design phase.

The Common Criteria (CC) Redbook [6] was one of the first attempts to standardize security assessment/evaluation requirements for Information Technology (IT) systems.

Table 7.2 Evaluation Assurance Levels, Seven Levels Listing (Reproduced from [25])

	Common Criteria Assurance Levels
EAL1	functionally tested
EAL2	structurally tested
EAL3	methodically tested and checked
EAL4	methodically designed, tested and reviewed
EAL5	Semi-formally designed and tested
EAL6	Semi-formally verified design and tested
EAL7	formally verified design and tested

CC is intended for security evaluation in IT systems. [6] While it is possible to use CC as a reference for designing security in Electronic Commerce (EC) systems, the corresponding design process is not a systematic one and relies completely on the individual security designer expertise. A possible risk is not introducing the proper security countermeasures during the design phase. In this case, defects will either be discovered during the system-testing phase, requiring an expensive fix, or not discovered at all.

Table 7.3 Summarize common criteria Assurance level comparison to CC predecessor [25].

Table 7.3 Assurance Levels Comparison (Reproduced from [25])

	Assurance Level Comparison						
Common Criteria:	EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
ITSEC:	-	E1	E2	E3	E4	E5	E6
CESG CompuSec Manual No.3:	-	UKL1	UKL2	UKL3	UKL4	UKL5	UKL6
TCSEC:	-	C1	C2	B1	B2	B3	A1

7.3.1. Common Criteria Predecessors Description

In this section, the predecessors of common criteria security assurance evaluation references, such as Trusted Computer System Evaluation Criteria (TCSEC), Federal Criteria for Information Technology Security (FC) TCSEC, The UK ITSEC Scheme, and Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) will be discussed [25].

7.3.1.1. Trusted Computer System Evaluation Criteria (TCSEC)

Trusted Computer System Evaluation Criteria: A document developed by the U. S. Department of Defense in 1985, containing the evaluation criteria for assessing degrees of assurance in the security features of hardware and software systems. The document is frequently referred to as 'The Orange Book'. It defines security in classes ranging from D (minimum) to A1 (highly-secure). These classes define the security functions required to meet a specified level of trust, and while most mainframe security systems will meet the class B criteria it is more common to see commercial products being submitted at class C2 functionality. An example of a product classified to C2 level is Windows NT.

7.3.1.2. Federal Criteria for Information Technology Security (FC)

US draft security criteria for trusted systems. This was an attempt by the US to develop trusted system evaluation criteria to replace the US TCSEC and become an element in a suite of Information Processing Standards under the management of

NIST. While it is still possible to find the original draft documents published in December 1992, the project has effectively been shelved.
FC Federal Information Processing Standard (FC-FIPS) would have been an improvement on TCSEC, but was overtaken by the international agreement to develop common criteria stemming from a meeting in Brussels, February 1993.

7.3.1.3. TCSEC, The UK ITSEC Scheme

Information Technology Security Evaluation Criteria, a scheme for the evaluation of security products run in the UK by the Department of Trade and Industry (DTI) and CESG.
ITSEC was probably the most successful computer security evaluation criteria of the 1990s. It offers greater flexibility than TCSEC and is easier and cheaper to use. It has also benefited from being developed after Orange Book and being closer to modern technology.

Another factor in its success has been 'mutual recognition'. Being European criteria, and having an established common methodology, it has been comparatively straightforward to negotiate recognition agreements. This process started in 1996 with a bilateral agreement between the UK and Germany. It was added to by an agreement between the UK and France in 1997. And in March 1998, an agreement was signed by the UK, France, Finland, Germany, Greece, the Netherlands, Norway, Portugal, Spain, Sweden and Switzerland, whereby ITSEC certificates issued by any of the Qualifying Certification Bodies would be recognized in all other countries. The initial Qualifying Certification Bodies were those of the UK, France and Germany.

"In May 1990 France, Germany, the Netherlands and the United Kingdom published the Information Technology Security Evaluation Criteria (ITSEC) based on existing work in their respective countries. Following extensive international review, Version 1.2 was subsequently published in June 1991 by the Commission of the European Communities for operational use within evaluation and certification schemes.
ITSEC is a structured set of criteria for evaluating computer security within products and systems. Each evaluation involves a detailed examination of IT security features culminating in comprehensive and informed functional and penetration testing. This

work is undertaken using an agreed Security Target as the baseline for ensuring that a product or system meets its security specification. ITSEC operates the concept of assurance levels E0 to E6. This scale represents ascending levels of confidence that can be placed in the TOEs security functions and determines the rigor of the evaluation. Since the launch of ITSEC in 1990, a number of other European countries have agreed to recognize the validity of ITSEC evaluations."[25].

7.3.1.4. Canadian Trusted Computer Product Evaluation Criteria (CTCPEC)

Canadian Trusted Computer Product Evaluation Criteria: Canadian secure products criteria. It was the agreement between the Canadians and the USA to harmonize the criteria of CTCPEC and FC/TCSEC that kick-started the development of the Common Criteria [25].

The previous three standards are examined. The author found that the Common Criteria (CC) is suitable for testing product compliance. However, ISO/IEC 17799:2000 and COBIT are both concerned with security management process and security measures. The author chooses to adopt ISO/IEC 17799:2000 in this research for its international visibility.

CHAPTER 8

SYSTEMS MAPPING

Security systems can be described in terms of the requirements, adopted standards, deployment mechanisms, provided services, products category, or products brands. Since this research includes the security requirements elicitation, security standards, and security mechanisms, it is convenient to map each category to its equivalent part.

8.1. Mapping Security Requirements to ISO/IEC 17799:2000 Security Standard

As discussed in Chapter 3, there are 13 security requirements subcategories, like Identification, Authentication, Authorization Integrity, Non-repudiation, Privacy, etc. Those requirements can be mapped to ISO/IEC 17799:2000 [3] clause or clauses category standards.

For example identification and authorization requirements can be mapped to ISO/IEC 17799:2000 Clause 9.5.3 User identification and authentication. One security requirement can be mapped to one or more ISO/IEC 17799:2000 Clause. Also one ISO/IEC 17799:2000 clause can be mapped to one or more security requirements.

Example: Mapping Identification requirements and Authorization requirements to ISO/IEC 17799:2000 Clause 9.5.3

Clause 9.5.3 User Identification and Authentication

"All users (including technical support staff, such as operators, network administrators, system programmers and database administrators) should have a unique identifier (user ID) for their personal and sole use so that activities can subsequently be traced to the responsible individual. User IDs should not give any indication of the user's privilege level, e.g. manager, supervisor.

In exceptional circumstances, where there is a clear business benefit, the use of a shared user ID for a group of users or a specific job can be used. Approval by management should be documented for such cases. Additional controls may be required to maintain accountability.

There are various authentication procedures, which can be used to substantiate the claimed identity of a user. Passwords are a very common way to provide

identification and authentication (I&A) based on a secret that only the user knows. The same can also be achieved with cryptographic means and authentication protocols.

Objects such as memory tokens or smart cards that users possess can also be used for I & A. Biometric authentication technologies that use the unique characteristics or attributes of an individual can also be used to authenticate the person's identity. A combination of technologies and mechanisms securely linked will result in stronger authentication."

Similarly, other security requirements can be mapped to its equivalent ISO/IEC 17799:2000 standard as explained in Table 8.1.

Table 8.1 Mapping Security Requirements to ISO/IEC 17799:2000 Clauses

Security Requirement	ISO/IEC 17799:2000
Identification Requirements	Clause: 9.1.X, 9.2.X, 9.5.X
Authentication Requirements	Clause: 9.1.X, 9.3.X, 9.4.X, 9.5.X, 10.2.3
Authorization Requirements	Clause: 9.5.X, 9.6.X
Immunity Requirements	Clause: 7.X, 8.3.X, 11.X, 12.1.X
Integrity Requirements	Clause: 8.6.X; 8.7.X, 10.4.X
Intrusion Detection Requirements	Clause: 9.7.X, 12.3, 8.4.X
Intrusion Prevention Requirements	Clause: 7.2.X, 7.3.X
Non-repudiation Requirements	Clause: 10.3.X
Privacy/Secrecy Requirements	Clause: 8.7.X, 10.3.X, 12.1.X
Security Auditing Requirements	Clause: 8.4.X, 11.3.X, 9.7.X, 12.3.X
Survivability Requirements	Clause: 7.X, 11.X
Physical Protection Requirements	Clause: 7.X, 8.6.X
System Maintenance	Clause: 10.5.X, 11.1.5

Detailed security standards discussion was discussed in Chapter 7. A complete listing of ISO/IEC 17799:2000 clauses can be found in ISO/IEC 17799: *Code of Practice for Information Security Management* [3].

8.2. Mapping Security Requirements to Security Mechanisms

In order to fulfill the security requirements in a modeled based system, Unified Modeling Language with its Security Extensions UMLsec [2] can be used with some modification. UMLsec allows one to stipulate security mechanisms within the diagram in a UML system specification. The extension is given in form of the UML profile using standard UML extension mechanisms. Stereotypes are used together with tags to formulate security requirements and assumptions on the system

environment; Constraints give criteria that determine whether the requirements are met by the system design. Security mechanisms can be also structured as a stereotype values attached to the security system node, link, or component. In fact, one UMLsec stereotype may map to many security mechanisms at the implementation of the required security measures.

For example, email privacy/confidentiality between the email client and server security requirements in the network layer can be achieved by deploying encryption security mechanism in the network link between the email server and client, this mechanism can be represented by <<encrypted>> dependency stereotype.
Similarly other security requirements can be mapped to its equivalent UMLsec stereotypes as described in Table 8.2. Newly added stereotypes are italicized.

Table 8.2 Mapping Security Requirements to UMLsec Stereotypes

Security Requirement	**UMLsec Stereotypes**
Identification Requirements	guarded access
Authentication Requirements	guarded access
Authorization Requirements	guarded access, rbac
Immunity Requirements	*no misuse*
Integrity Requirements	integrity, critical
Intrusion Detection Requirements	*no misuse, audit log*
Intrusion Prevention Requirements	*no misuse*, no down-flow, no up-flow, *audit log*
Non-repudiation Requirements	provable, fair-exchange, *audit log*
Privacy/Secrecy Requirements	encrypted, secure links, data security, .
Security Auditing Requirements	provable, *audit log*
Survivability Requirements	*multiplicity*
Physical Protection Requirements	secure links, LAN, wire, *multiplicity*
System Maintenance	*no misuse, multiplicity*

There are predefined stereotypes which can be used as security mechanisms to fulfill the desired security requirements. However they are not covering all security requirement categories. Therefore other stereotypes are introduced like <<*audit log*>>, <<*multiplicity*>>, and <<*no misuse*>>. Also one stereotype can be mapped to one or more security requirements, and one requirement can be achieved by one or more stereotypes.

CHAPTER 9

SYSTEM SECURITY TESTING METHODOLOGY

The dependency on Internet services has increased dramatically. Several institutions, companies, universities, government agencies, as well as individuals are offering services on the Internet. Consequently efforts from both academia and industry are striving to create technologies and standards that meet the sophisticated requirements of today's e-services and users. In many situations, security remains a major roadblock to universal acceptance of the e-services for all kinds of transactions. Computer Emergency Response Team (CERT) [40] statistical reports show an increase in the security vulnerabilities rate, which implies that there are several computer systems that have been deployed without proper security mechanism, or deployed without adequate security testing. In this chapter, security and testing methodologies are discussed.

9.1. Penetrate and Patch

In Penetrate and Patch methodology the system is tested against known vulnerabilities checks based on a trial and error basis. The discovered vulnerabilities will be fixed. However Penetrate and Patch methodology has some disadvantages such as:

- The system could be insecure during the penetration test.
- Disruption to the service during the test costs money, impacts confidence, and annoys the customers.
- Expensive, since it needs high skilled engineers, and costly training.

Also designing secure systems manually for normal computer practitioners, is difficult due to the lack of knowledge, time pressure, and the methodologies for applying security mechanisms.

9.2. Testing in UML Diagrams

The current practice in UML design is to use inspections and walkthroughs [16]. Given that UML models for many designs can become large and complex, inspections and walkthroughs can be ineffective and problematic. In addition, the UML provides a variety of notational views. These views in isolation cannot provide a representation

of the cross notational aspects in the UML. Resolving notational inconsistencies requires transformation of the model into a representation that allows verification of integrity between views. When views are integrated, types of relationships can be discovered and validated which are not obvious in a single UML notational view. Since the integration of different UML views requires altering the system component codes which could not be feasible, a model-based testing mechanism is needed.

9.3. Security by Design

Security by Design is performed according to best practices and standards, with a proper automated security mechanisms, and checks starting from the system initiation phase, within building context, until retirements. Security by design ensures meeting the current security challenges.

The Security by Design methodology detects security related problems in early stages, and includes automated verification and certification components. Suggested mechanisms and checks can be achieved by UMLsec stereotypes.

9.4. Testing of UMLsec Diagrams

UMLsec as explained earlier is an extension to the UML with encapsulated knowledge on prudent security engineering. This knowledge is made available to system builders that may not be specialized in security. Also the system could be checked for the conformance to a specific security requirement, represented by constraints associated with UMLsec stereotypes. A UML extension collects stereotypes, tagged values, and constraints into a profile. For UMLsec, the model is validated against included security requirements which are achieved by extending the formal semantic of the original UML in a modular way with a formal notion of an adversary or misuser.

9.4.1. UMLsec Security Checks and Support Tools

In order to facilitate UMLsec security analysis approach in the industry, automated tools for the analysis of UML models using the suggested semantics are required. Figure 9.1 describes UMLsec framework which meet the requirements [31, 38].

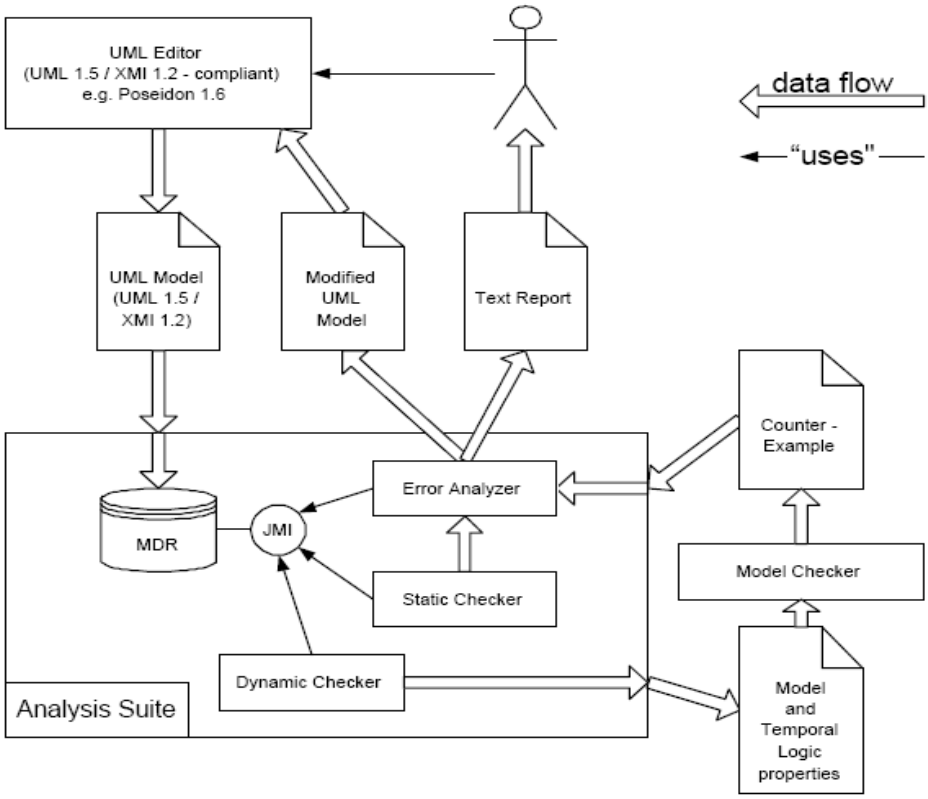


Figure 9.1 UMLsec Analysis Tool Suite
(Reproduced from [38])

UMLsec Tools Suite Function

1. The system builder creates a model and stores it in the UML 1.5/XMI 1.2 (XMI XML metadata Interchange) format by a number of existing UML design tools like Poseidon [39]. To avoid processes UML model directly on the XMI level, the MDR (Meta Data Repository http://mdr.netbeans.org) repository is used, which allows one to directly operate on the UML concept level. The MDR library implements repository for any model described by a modeling language compliant to the Meta Object Facility (MOF).

2. Then the file is imported by the tool into the internal MDR repository. The tool accesses the model through the Java Metadata Interface (JMI) interfaces generated by the MDR library.

3. The checker parses the model and checks the constraints associated with the stereotype.

4. The results are delivered as a text-based report for the system builder, describing the discovered problems. In addition, a modified UML model is delivered, where the stereotypes whose constraints are violated are highlighted.

UMLsec Webinterface Tools

Automated UMLsec security checks tools are developed by Jan Jurjens [37]. They can be used for testing UMLsec models through the Internet, using the following steps.

1. The UML secure system diagram is created by the system builder using Poseidon [39] with the required security constraints and stereotypes.
2. The design file should be saved in .zargo or .xmi format, with Poseidon 1.6.
3. Access to the Webinterface tools [37] site, with proper authentication.
4. Upload the UMLsec model file into the framework.
5. Select the diagram type and tool, then click submit.
6. The Webinterface tool will check the system.
7. A text based test report will be generated, describing whether the system is satisfying or violating the security requirements according to the adversary mode.

9.5. Test Cases and UMLsec Security Checks Webinterface Tool

In this section, security test cases are developed with UMLsec tool examples.

Figure 9.2 represents a blank copy of the suggested test cases template for testing security-relevant scenarios.

Security test case template includes the following fields:

TC #:	Represent the test case unique identifier.
TC Description:	Provide a brief description of the test case rational.
Objectives:	The test case objectives, and the security requirement it addresses.
Live:	To be used in limited cases when live systems' inspection is needed as part of the assessment phase of the system security reengineering process.
Model:	Normally selected, since the test is performed in a pre-deployment stage.
Test impact:	To be used in limited cases when live system’s inspection is needed as part of the assessment phase in the process, and there is an associated impact due to the test execution, such as service interruption.
Security requirement:	The security requirement that need to be checked.
Preconditions:	Assumption prior to running the test.
Test input:	Interaction input that will run the test.
Expected output:	Status of the test and relevant output report.
Test observed output:	The exact test output after running the test in live environment.
Test verdict:	Analysis of the test observed output to produce a final test judgment, Pass, Fail or Inconclusive.

The Test observed output and Test verdict fields are only needed in the live test mode.

TC #		TC Description	
Objectives			
Live ☐	Model ☐	Test Impact	
Security requirements			
Preconditions	Test Input	Expected Output	
Test observed output		Test verdict	

Figure 9.2 A proposed Security Test Case Template

E-Banking web application is a good example for contemporary e-service scenarios. Consider an e-Banking application in which the client surfs the bank web page in the Web Server. Then the traffic is forwarded to an Application Server, which communicates with the backend Database Server.

Each UMLsec diagram can better relate to a different security mechanism. For example, deployment diagrams are concerned with the network layer security and component physical layout. On the other hand, a sequence diagram can enforce interaction flow mechanism such as encryption keys exchange and access control.

9.5.1. Checking UMLsec Deployment Diagram Models

Link level security mechanisms are enforced and verified using UMLsec stereotypes attached to deployment diagram models. The following scenarios consider different security requirements enforced in the deployment diagram models. The deployed mechanisms are tested using the UMLsec Webinterface tool.

Scenario 1: Test Case 1, Basic Model

Figures 9.3 and 9.4 depict e-Banking example elements and the corresponding test case respectively.

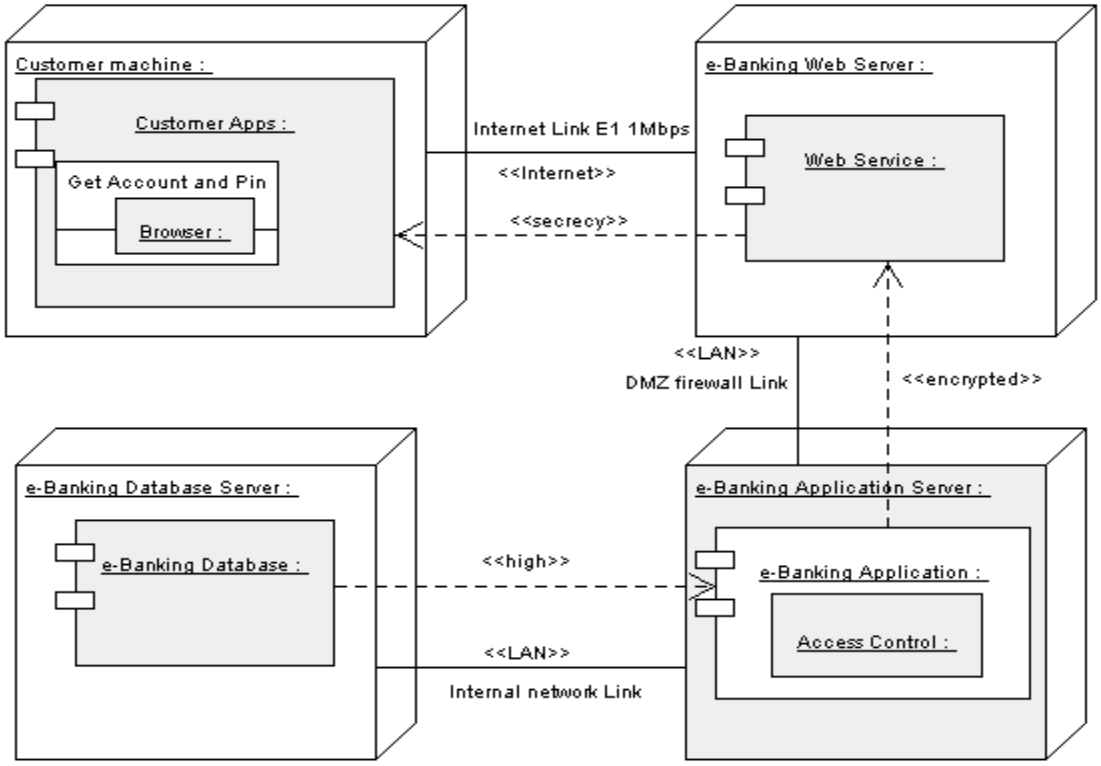


Figure 9.3 Scenario 1, E-Banking Deployment Diagram, <<secrecy>> Internet Link

TC #	1/1/9.5	TC Description	E-Banking web service - Basic model
Objectives	Verify the security measures for client, Web to Application servers, and Application to backend Database servers links. Check e-Banking model structure.		
Live ☐	Model ☒	Test Impact	No Impact in model mode
Security requirements	All network traffic must be encrypted (Privacy/ Confidentiality requirements).		

Preconditions	Test Input	Expected Output
Adversary does not know the encryption keys. Insider attacker knows the encryption keys.	E-Banking model file in .zargo format corresponding to Figure 9.3 model setup.	Violation in the Internet link. Compliance in the link between Web and Application servers. Violation in the link between Application and backend Database servers.

Figure 9.4 Scenario 1, E-Banking <<secure link>>, Test Case 1

In order to run the test, e-Banking system is modeled using UML CASE tool Poseidon version 1.6, and is saved in .zargo format.

Webinterface tool site is accessed via http://www4.in.tum.de:8180/vikinew/vikiweb link, and proper credentials are submitted.

The e-Banking UMLsec module .zargo file is uploaded to the Webinterface tools Figure 9.5.

The required tool is selected. In this example, UMLsec Static Check I will be selected.

Submit button will be clicked.

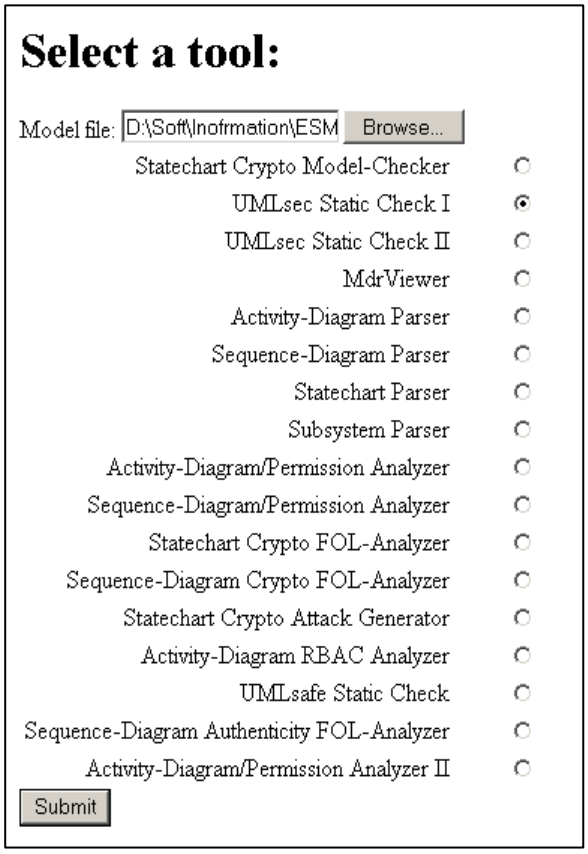


Figure 9.5 Webinterface Tool Options

Select a command window will appear, as shown in Figure 9.6, and then the required security measure needs to be selected.

In the current example, Secure Link will be selected.

Webinterface UMLsec tools will process the .zargo file and will report the analysis result as follows.

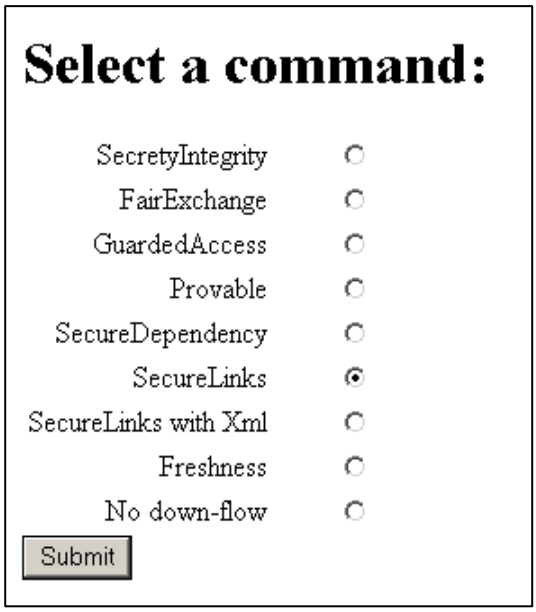


Figure 9.6 Webinterface Command Selections

Webinterface Secure Links Analysis Report:

==========running CheckerSecureLinks...
======= Dependencies, Component Instances, Node Instances and Communication Links
The name of the dependency is Web2Client
The name of the supplier component instance of the dependency is Customer Apps and it is resident in the node instance Customer machine
The name of the client component instance of the dependency is Web Service and it is resident in the node instance e-Banking Web Server

The name of the communication link is Internet Link E1 1Mbps
The stereotype of the communication link of the dependency with the name Web2Client is Internet

The name of the dependency is DB2App
The name of the supplier component instance of the dependency is e-Banking Application and it is resident in the node instance e-Banking Application Server
The name of the client component instance of the dependency is e-Banking Database and it is resident in the node instance e-Banking Database Server
The name of the communication link is Internal network Link
The stereotype of the communication link of the dependency with the name DB2App is LAN

The name of the dependency is App2Web
The name of the supplier component instance of the dependency is Web Service and it is resident in the node instance e-Banking Web Server
The name of the client component instance of the dependency is e-Banking Application and it is resident in the node instance e-Banking Application Server
The name of the communication link is DMZ firewall Link
The stereotype of the communication link of the dependency with the name App2Web is LAN

=======Here begins the verification
The name of the dependency is **Web2Client**
The stereotype of the communication link of the dependency with the name Web2Client is Internet

The stereotype of the dependency is secrecy

::::::::Against **Default Attacker**
The UML model **violates** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **violates** the requirement of the stereotype secure links.
The name of the dependency is **DB2App**
The stereotype of the communication link of the dependency with the name DB2App is LAN
The stereotype of the dependency is high

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **violates** the requirement of the stereotype secure links.
The name of the dependency is **App2Web**
The stereotype of the communication link of the dependency with the name App2Web is LAN
The stereotype of the dependency is encrypted

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

Report Analysis:

The test case report satisfies the secure link requirements against the default attacker as well as the insider attacker for the link between the Web and Application servers. However, violations to the <<secure link>> stereotypes security requirements are discovered in the Internet link against the default attacker as well as the insider attacker.

Regarding the link between the Application and the backend Database Servers, the test discovers satisfying the <<secure link>> security requirement against the default attacker. On the other hand, it discovers violation of the <<secure link>> security requirements against the insider attacker.

The analysis suggests checking the security mechanism between the client web Browser and the e-Banking Web Server. Also the security mechanisms in the link between the Application Server and backend Database Server need to be revised.

The modifications are reflected in Scenario 2, in which <<encrypted>> stereotype dependency is added to the Internet link.

Figures 9.7 and 9.8 describe e-Banking deployment diagram Scenario 2 and the corresponding test case respectively.

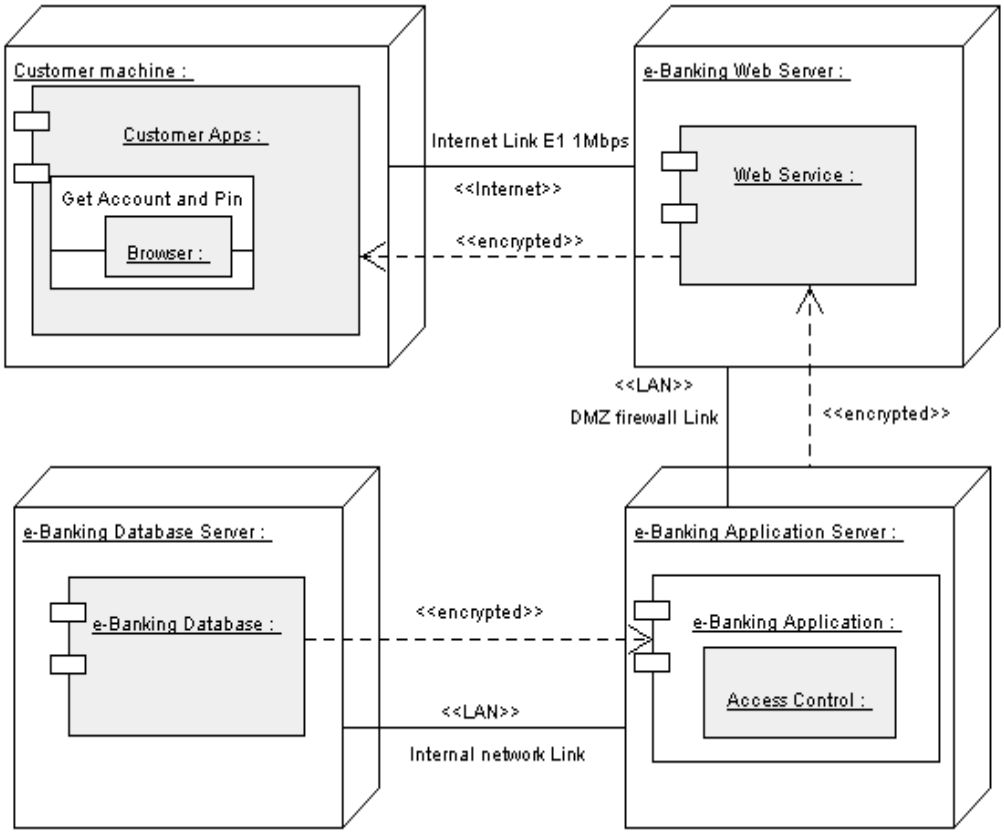


Figure 9.7 Scenario 2, E-Banking Deployment Diagram, <<encrypted>> Internet Link

TC #	1/2/9.5	TC Description	E-Banking web service - encrypted Internet link mode
Objectives	▪ Verify the security measures between the client and Web Server meet secure link security requirements. ▪ Check e-Banking model structure.		
Live ☐	Model ☒	Test Impact	No Impact in model mode
Security requirements	All network traffic to be encrypted (Confidentiality Requirements)		
Preconditions		Test Input	Expected Output
Adversary does not know the encryption keys. Insider attacker knows the encryption keys. Availability UMLsec Webinterface tools.		E-Banking model file in .zargo format corresponding to Figure 9.7 Scenario.	Compliance in the Internet link. Compliance in the link between Web and Application servers. Compliance in the link between Application and backend Database servers.

Figure 9.8 Scenario 2, E-Banking <<encrypted>> Internet Link, Test Case 1

E-Banking deployment diagram model is constructed using UML tools with the modified security mechanisms corresponding to Figure 9.7. The model is saved in .zargo file format, and uploaded into the Webinterface tools as explained in Scenario 1 earlier.

Webinterface Secure Links Analysis Report:

==========running CheckerSecureLinks...
======= Dependencies, Component Instances, Node Instances and Communication Links
The name of the dependency is Web2Client
The name of the supplier component instance of the dependency is Customer Apps and it is resident in the node instance Customer machine
The name of the client component instance of the dependency is Web Service and it is resident in the node instance e-Banking Web Server
The name of the communication link is Internet Link E1 1Mbps
The stereotype of the communication link of the dependency with the name Web2Client is Internet

The name of the dependency is DB2App

The name of the supplier component instance of the dependency is e-Banking Application and it is resident in the node instance e-Banking Application Server
The name of the client component instance of the dependency is e-Banking Database and it is resident in the node instance e-Banking Database Server
The name of the communication link is Internal network Link
The stereotype of the communication link of the dependency with the name DB2App is LAN

The name of the dependency is App2Web
The name of the supplier component instance of the dependency is Web Service and it is resident in the node instance e-Banking Web Server
The name of the client component instance of the dependency is e-Banking Application and it is resident in the node instance e-Banking Application Server
The name of the communication link is DMZ firewall Link
The stereotype of the communication link of the dependency with the name App2Web is LAN

=======Here begins the verification
The name of the dependency is **Web2Client**
The stereotype of the communication link of the dependency with the name Web2Client is Internet
The stereotype of the dependency is encrypted

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.
The name of the dependency is **DB2App**
The stereotype of the communication link of the dependency with the name DB2App is LAN
The stereotype of the dependency is encrypted

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.
The name of the dependency is **App2Web**
The stereotype of the communication link of the dependency with the name App2Web is LAN
The stereotype of the dependency is encrypted

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

Report Analysis:

The test case report satisfies the secure link requirements against the default attacker as well as the insider attacker for the three links:

- The Internet link between the client and Web Server.
- The link between the Web and Application servers.
- The link between the Application Server and the backend Database Server.

In Scenario 2, two link security mechanisms have been changed. The Internet link as well as the link between the Application Server and the backend Database Server.
In the latter, <<encrypted>> tag encrypts the traffic, however, it does not affect the flow of information. An appropriate tag in this link is <<high>>, provided that the servers' network segment is secured physically or application-aware encryption mechanism is deployed.
The modifications are reflected in Scenario 3, in which <<high>> dependency stereotype is added to the link between Database and Application servers.

Figures 9.9 and 9.10 describe e-Banking deployment diagram scenario 3 and its test case respectively.

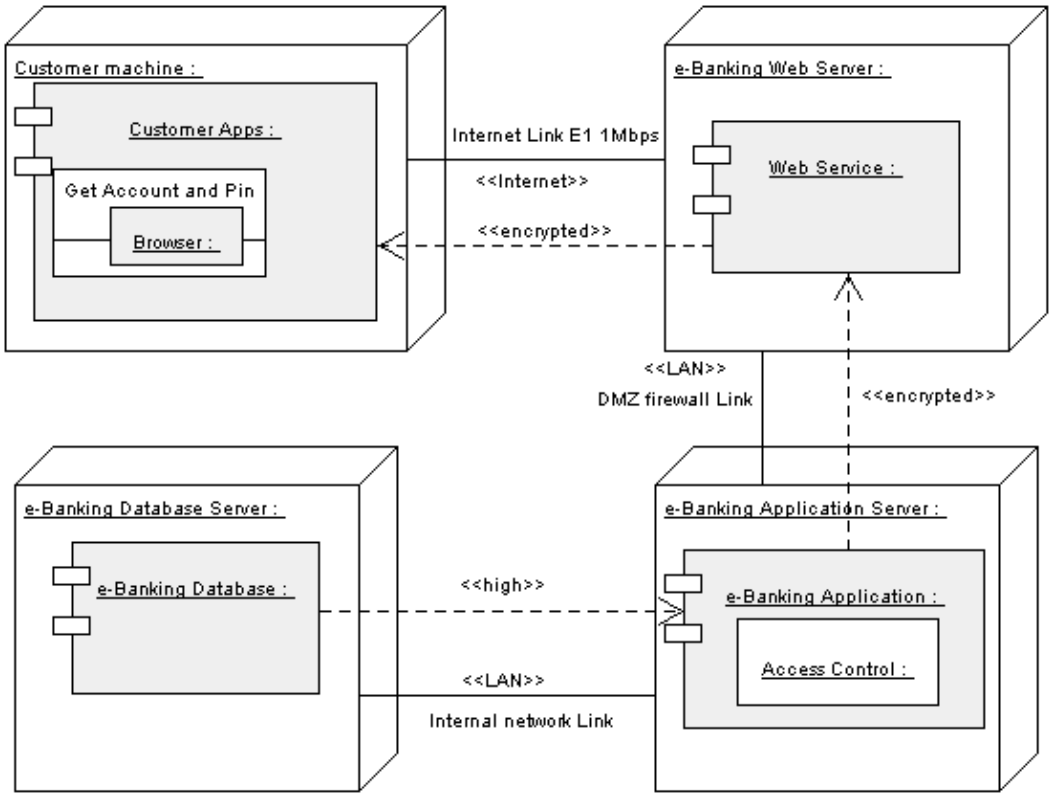


Figure 9.9 Scenario 3, E-Banking Deployment Diagram, <<high>> LAN Link

TC #	1/3/9.5	TC Description	E-Banking web service - <<high>> database and application servers.
Objectives	▪ Verify the security measures in the link between the e-Banking Application Server and Database Server meet the requirements. ▪ Check e-Banking model structure.		
Live ☐	Model ☒	Test Impact	No Impact in model mode.
Security requirements		All network traffic to be encrypted.	

Preconditions	Test Input	Expected Output
Adversary does not know the encryption keys. Insider attacker knows the encryption keys.	E-Banking model file in .zargo format, corresponding to Figure 9.9 Scenario.	Conformance in the Internet link. Conformance in the link between web and application servers. Violation in the link between application and backend database servers.

Figure 9.10 Scenario 3, E-Banking <<high>> LAN Link, Test Case 1

E-Banking deployment diagram model is constructed using UML CASE tool Poseidon version 1.6, reflecting the modified security mechanisms for Scenario 3 and uploaded into the Webinterface tools using the same steps explained in Scenario 1.

Webinterface Security Analysis Report:

==========running CheckerSecureLinks...
======= Dependencies, Component Instances, Node Instances and Communication Links
The name of the dependency is Web2Client
The name of the supplier component instance of the dependency is Customer Apps and it is resident in the node instance Customer machine
The name of the client component instance of the dependency is Web Service and it is resident in the node instance e-Banking Web Server
The name of the communication link is Internet Link E1 1Mbps
The stereotype of the communication link of the dependency with the name Web2Client is Internet

The name of the dependency is DB2App
The name of the supplier component instance of the dependency is e-Banking Application and it is resident in the node instance e-Banking Application Server
The name of the client component instance of the dependency is e-Banking Database and it is resident in the node instance e-Banking Database Server
The name of the communication link is Internal network Link
The stereotype of the communication link of the dependency with the name DB2App is LAN

The name of the dependency is App2Web
The name of the supplier component instance of the dependency is Web Service and it is resident in the node instance e-Banking Web Server
The name of the client component instance of the dependency is e-Banking Application and it is resident in the node instance e-Banking Application Server
The name of the communication link is DMZ firewall Link
The stereotype of the communication link of the dependency with the name App2Web is LAN

=======Here begins the verification
The name of the dependency is **Web2Client**
The stereotype of the communication link of the dependency with the name Web2Client is Internet
The stereotype of the dependency is encrypted

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.
The name of the dependency is **DB2App**
The stereotype of the communication link of the dependency with the name DB2App is LAN
The stereotype of the dependency is high

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **violates** the requirement of the stereotype secure links.
The name of the dependency is **App2Web**
The stereotype of the communication link of the dependency with the name App2Web is LAN
The stereotype of the dependency is encrypted

::::::::Against **Default Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

::::::::Against **Inside Attacker**
The UML model **satisfies** the requirement of the stereotype secure links.

Report Analysis:

The test case report satisfies the secure link requirements against the default attacker as well as the insider attacker for the Internet link between the client and e-Banking Web Server. Also the link between e-Banking Web Server and Application Server satisfies secure link requirements. However the e-Banking servers should be segregated from the institution's personnel machines to protect them against the inside attacker's risk.

9.5.2. Checking UMLsec Sequence Diagram Models

UMLsec stereotypes are attached to sequence diagram messages to address security requirmetns at the interaction level.

Scenario 4: Test Case 1, Account Balance Sequence Diagram, rbac

Figures 9.11 and 9.12 describe e-Banking sequence diagram scenario 4 and its test case, respectively. In this scenario, role-based access control (rbac) security mechanism is checked, in which a legitimate client requests account balance inquiry. The user is able to get the account status since account balance service is defined in the general clients' profile.

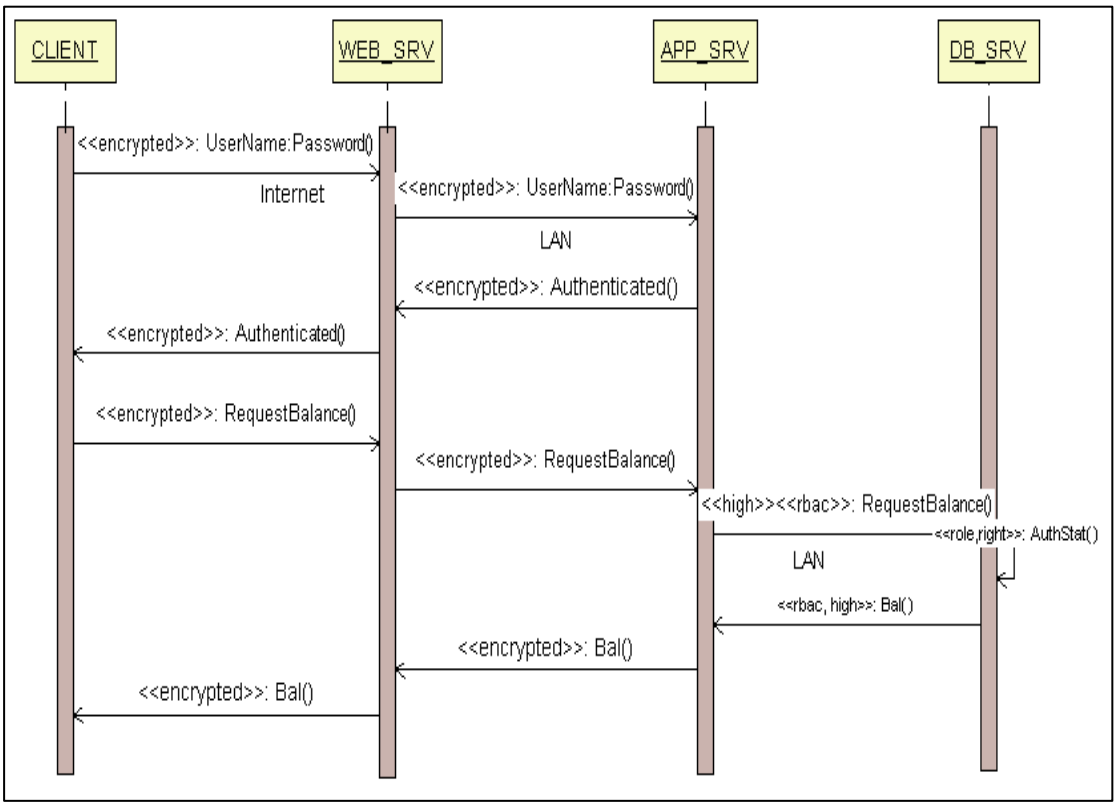


Figure 9.11 Scenario 4, E-Banking Sequence Diagram, <<rbac>> Balance request, Test Case 1

TC #	1/4/9.5	TC Description	E-Banking web service - <<rbac>>, Balance inquiry scenario.
Objectives	▪ Check the "role-based access control" security mechanism in inquiry mode. ▪ Check e-Banking model structure.		
Live ☒	Model ☐	Test Impact	No Impact, since the request is a legitimate user request.
Security requirements	Authorization Requirements: Role-based Access Control mechanism component.		

Preconditions	Test Input	Expected Output
The client has a valid user name and password. The client has access to the bank's web site.	The user request balance inquiry.	The user should get positive response related to the inquiry.

Test observed output	Test verdict
The user is authenticated successfully, and is authorized for getting the account balance.	PASS: The observed outputs are the same as the expected output.

Figure 9.12 Scenario 4, E-Banking <<rbac>> Account balance, Test Case 1

Scenario 5: Test Case 1, Transfer Money Sequence Diagram, rbac

Figures 9.13 and 9.14 describe e-Banking sequence diagram scenario 5 and its test case, respectively. In this scenario, role-based access control security mechanism is checked, in which a legitimate client requests transfer money service. Because transfer money service is not included in the user profile, the request is denied and the user is notified with an un-authorized user message.

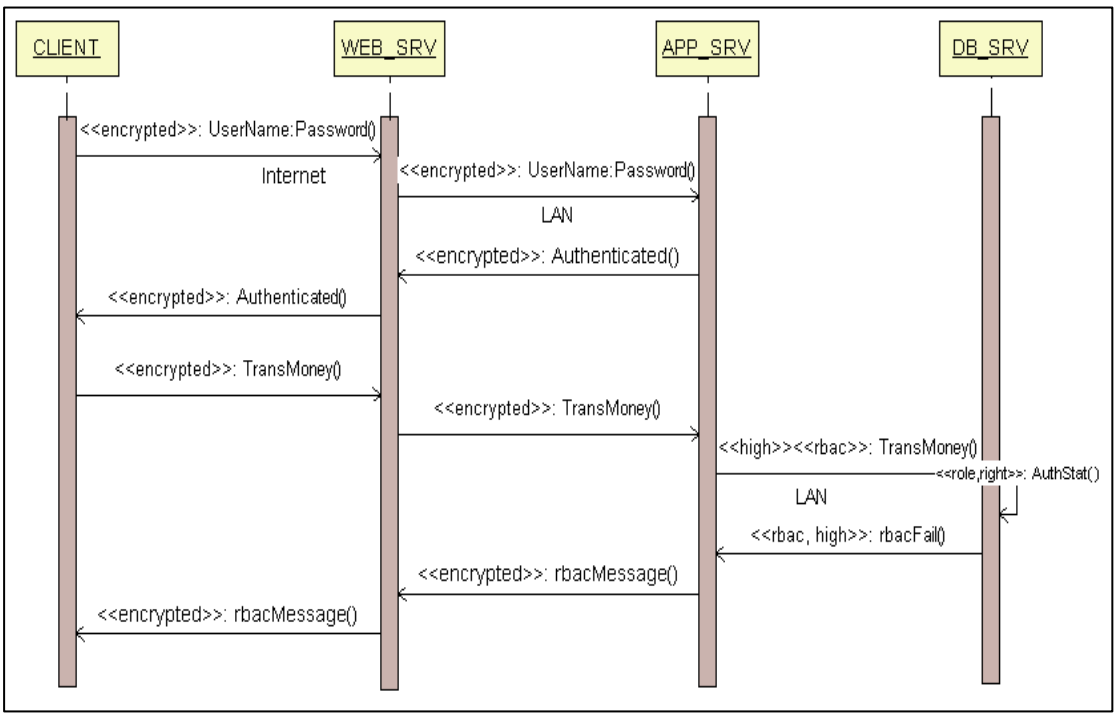


Figure 9.13 Scenario 5, E-Banking Sequence Diagram,

<<rbac>> Money Transfer, Test Case 1

TC #	1/5/9.5	TC Description	E-Banking web service - <<rbac>>, Transfer money request scenario.
Objectives	▪ Examine role-based access control mechanism in transfer money mode. ▪ Check e-Banking model structure.		
Live ☒	Model ☐	Test Impact	No Impact, since the request is a legitimate user request.
Security requirements	Role-based Access Control mechanism in place.		

Preconditions	Test Input	Expected Output
The client has a valid user name and password. The client does not have a transfer money privileges. The client has access to the bank's web site.	The user request transfer money.	The user should get an un-authorized user message.

Test observed output	Test verdict
The user is authenticated successfully, and transfer money request is rejected. The user receives an un-authorized user message.	PASS: The observed outputs are the same as the expected output.

Figure 9.14 Scenario 5, E-Banking <<rbac>> Money Transfer, Test Case 1

.

CHAPTER 10

SYSTEM SECURITY REENGINEERING PROCESS WITH CASE STUDY

The rapid changes in evolving technologies and the dynamic nature of the world necessitate the need for establishing product-independent and mechanism-independent system security reengineering management process. Standards and tools selection are depending upon the deployment phase. For example, electing ISO/IEC 17799:2000 as security standard and GRL with UMLsec as modeling tools was based on matching the current contemporary technologies. Also the security requirements vary from one system to the other according to the sensitivity and exposure of assets to risks. Risk is the possibility that any specific threat will exploit a specific vulnerability to harm the system. Vulnerability is the absence of security safeguard or countermeasure. It is also the weakness of a security counter-measure.

In summary security requirements, tools, and standards can be changed within the frame of the author's systematic approach to the management of system security reengineering process.

The system reengineering process consists of the following phases.

- Security Requirements Elicitation and standard Mapping.
- Security Acceptance Test Development.
- Acceptance Testing and Vulnerability Assessment.
- Security Design Modifications and Standard Mapping.
- Implementation of Modifications.
- Final Acceptance Testing and Vulnerability Assessment.

Figure 10.1 shows the phases of the author's systematic formal approach to the management of system security reengineering process.

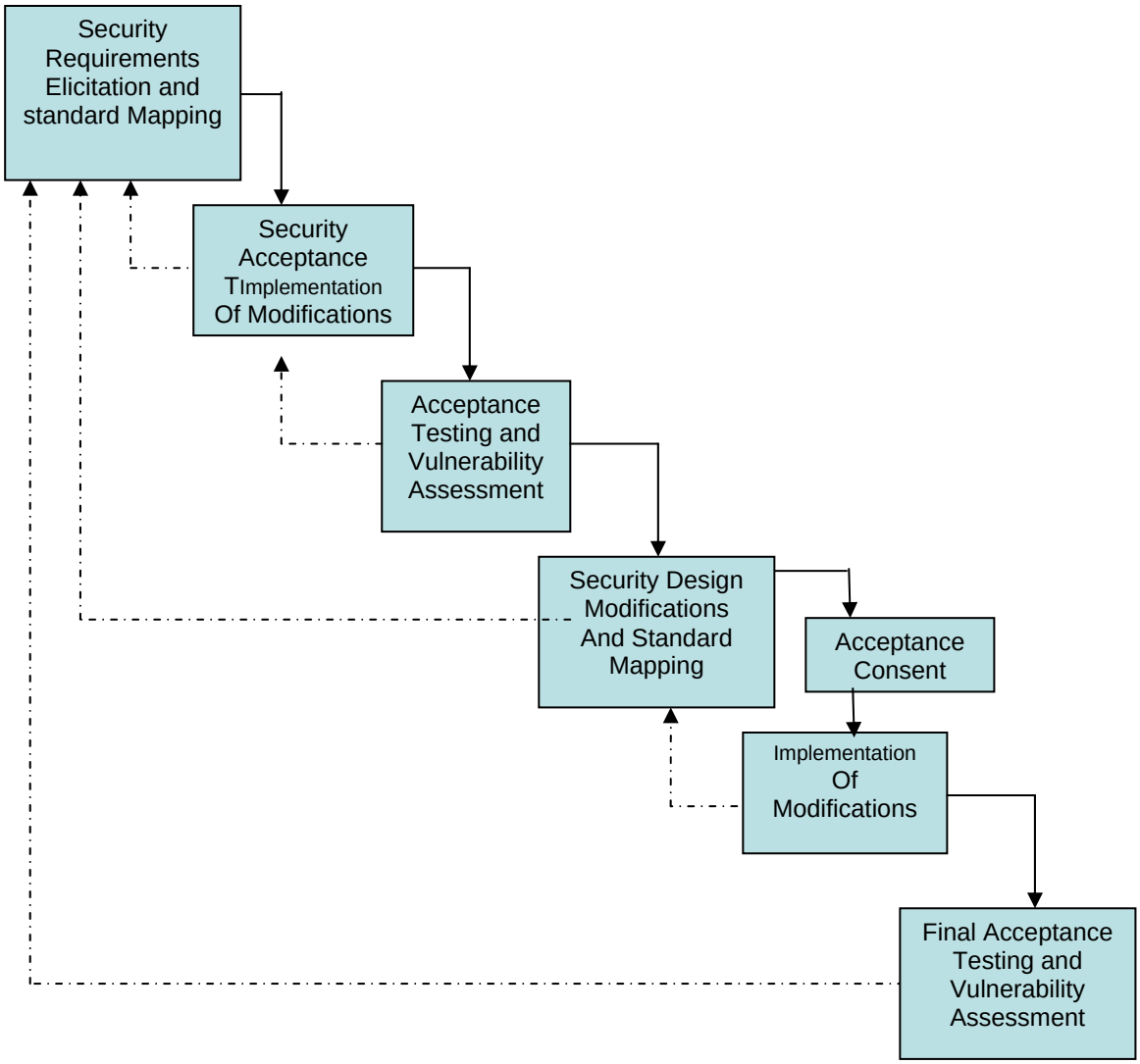


Figure 10.1 Formal System Security Reengineering Process.

Since this part represents the core of the research. A practical example is used to signify internal operations.

10.1. Security Requirements Elicitation and Standard Mapping

In Security requirements elicitation and standard mapping, the system security requirements are elicited from the business stakeholders perspective and mapped to the latest security standards, currently ISO/IEC 17799:2000 [3]. The requirements are adjusted to match the standard specifications while coordinating with the stakeholders. Then the system security requirements are depicted using the graphical Goal-Oriented Requirements Language (GRL) [1,27] and/or UMLsec, an extension to the Unified Modeling Language [2], explained in Chapter 6.

10.1.1. Practical Case Study

A local financial institution, consisting of head office and 20 branches, offers several banking services through branches, call centers, and Automatic Teller Machines (ATM). The bank business unit, sales department, would like to deploy e-Banking services to service their customers'; on the other hand they are faced with the risk element in terms of the e-Banking transactions security. Along with a wide variety of features in e-Banking services, today's customers also want assurance that their e-Banking services are completely secure.

The bank's business unit contacts an IT security consultant to help in introducing secure e-Banking services.

10.1.1.1. Case Study - Security Requirements Elicitation

The discussion between bank's stakeholders and the security consultant elicit the following e-Banking system’s security requirements.

1. The e-Banking system shall identify all of its human users, and client applications before any interaction.
2. The e-Banking system shall verify the identity of all of its users before allowing changes to user's personal profile.
3. The e-Banking system shall verify the identity of all of its client applications before any interaction.
4. The e-Banking system shall allow each customer to access his or her own personal account information.
5. The e-Banking system shall not allow any customer to access any other customers account information.
6. The e-Banking system shall not allow customer service agents to access the customer's accounts information.
7. The e-Banking system shall allow customer service agents to automatically email a new customer password to that customer’s email address.
8. The e-Banking system shall not allow customer service agents to access the customer new or old passwords.

9. The e-Banking system shall not allow one or more users to misuse the authorized services such as flooding the e-Banking system with multiple legitimate requests.
10. The e-Banking system shall clean any harmful object, if cleaning is possible, other wise it should be deleted.
11. The e-Banking system shall notify the security officer and the associated user, in case any harmful object is detected.
12. The e-Banking system shall protect itself from being infected by new infectious programs by deploying configurable automatic updates mechanism.
13. The e-Banking system shall prevent unauthorized corruption of customers and other external users' emails.
14. The e-Banking system shall prevent unauthorized corruption of all communications traffic through the networks that are external to the protected premises.
15. The e-Banking system shall prevent unauthorized corruption of customers and external users' data.
16. The e-Banking system shall detect and record all attempted accesses that fail identification, authentication, or authorization requirements.
17. The e-Banking system shall daily notify the concern security officer of all failed attempted accesses during the previous 24 hours.
18. The e-Banking system shall notify the concern security officer within five minutes of any repeated failed attempt to access the databases of employees and corporate financials.
19. The e-Banking system shall prevent and record all unauthorized attempted accesses.
20. The e-Banking system shall keep tamper-proof records related to client's transactions including, transactions amount, transactions date and time, and client's identity.
21. The e-Banking system shall not allow unauthorized programs or individuals access to any communications.
22. The e-Banking system shall not allow unauthorized programs or individuals access to any stored data.
23. The e-Banking system shall collect, organize, summarize, and regularly report its security mechanisms status including: 1) Identification, Authentication, and

Authorization, 2) Intrusion Detection, 3) Immunity, 4) Antivirus, 5) Privacy, 6) Intrusion Prevention.

24. The e-Banking system shall not have a single point of failure.
25. The e-Banking system shall continue to function for example, in degraded mode even if a data center is destroyed.
26. The e-Banking system shall protect its hardware components from physical damage, unplanned replacement, destruction, or theft.
27. The e-Banking system shall not violate its security requirements as a result of the upgrading its hardware, or software.
28. The e-Banking system shall not violate its security requirements as a result of a change to its existing requirements and their implementations.

10.1.1.2. Case Study - Security Requirements Categorization

These requirements can be included in the frame of security requirement categories, or security mechanism suites.

Security requirements 1 to 8 are part of the Access Control security mechanism suite, which includes the Identification, Authentication and Authorization security requirement categories.

Security requirements 9 to 12 are part of the Immunity requirements category.

Security requirements 13, 14, and 15 are part of the Integrity requirements category.

Security requirements 16, 17, and 18 are part of the Intrusion Detection System requirements category.

Security requirement 19 is part of the Intrusion Prevention System requirements category.

Security requirement 20 is part of the Non-repudiation requirements category.

Security requirements 21 and 22 are part of the Confidentiality/Privacy/Secrecy requirements category.

Security requirement 23 is part of the Security Auditing requirements category.

Security requirements 24 and 25 are part of the Survivability requirements category.

Security requirement 26 is part of the Physical Protection requirements category.

Security requirements 27 and 28 are part of the System Maintenance Security requirements category.

10.1.1.3. Case Study - Security Requirements Standardization

The elicited security requirement categories can be mapped to ISO/IEC 17799:2000 security standard clauses, by matching the requirement parameters to their corresponding clauses in ISO/IEC 17799:2000. For example, security auditing requirements are partially included in event logging Clause 9.7.1 as follows.

"Clause 9.7.1 Event Logging

Audit logs recording exceptions and other security-relevant events should be produced and kept for an agreed period to assist in future investigations and access control monitoring.
Audit logs should also include:

a) User IDs;
b) Dates and times for log-on and log-off;
c) Terminal identity or location if possible;
d) Records of successful and rejected system access attempts;
e) Records of successful and rejected data and other resource access attempts.

Certain audit logs may be required to be archived as part of the record retention policy or because of requirements to collect evidence (see also clause 12 in [3])"[3].

ISO/IEC 17799:2000 event logging clause 9.7.1 covers only part of the security auditing requirements. Consequently one security requirement can be covered by one or more security standard clauses, and one security standard clause can cover one or more security requirements.

Table 10.1 summarizes the Security Requirements and their corresponding clauses in ISO/IEC 17799:2000 security standard. Detailed description of security mapping was discussed in Chapter 8, Systems mapping.

Table 10.1 Mapping Case Study Security Requirements to ISO/IEC 17799:2000 Clauses

Security Requirement	ISO/IEC 17799:2000
Identification Requirements	Clause: 9.1.X, 9.2.X, 9.5.X
Authentication Requirements	Clause: 9.1.X, 9.3.X, 9.4.X, 9.5.X, 10.2.3
Authorization Requirements	Clause: 9.5.X, 9.6.X
Immunity Requirements	Clause: 7.X, 8.3.X, 11.X, 12.1.X
Integrity Requirements	Clause: 8.6.X; 8.7.X, 10.4.X
Intrusion Detection Requirements	Clause: 9.7.X, 12.3, 8.4.X
Intrusion Prevention Requirements	Clause: 7.2.X, 7.3.X
Non-repudiation Requirements	Clause: 10.3.X
Privacy/Secrecy Requirements	Clause: 8.7.X, 10.3.X, 12.1.X
Security Auditing Requirements	Clause: 8.4.X, 11.3.X, 9.7.X, 12.3.X
Survivability Requirements	Clause: 7.X, 11.X
Physical Protection Requirements	Clause: 7.X, 8.6.X
System Maintenance Requirements	Clause: 10.5.X, 11.1.5

There are several benefits for incorporating the latest standards in the requirements elicitation phase. Examples of such benefits include 1) any institution electing to adopt this methodology will be certifiable from process perspective, and 2) avoiding reinventing the wheel since the best practices scenarios are embedded in the standards.

10.1.1.4. Case Study - Security Requirements GRL Representation

The e-Banking system security requirements are shown in Figure 10.2 from the system stakeholders perspective using the Goal Oriented Language (GRL) [1] E-Banking Security softgoal is the root of the required security components. There are several ways for representing each component depending upon the scope of the design. Some requirements can be grouped into one softgoal, for example, Identification; Authentication and Authorization are combined into the Access Control softgoal. Defense and Hardening softgoal combine the Intrusion Detection System, Physical Security, Intrusion Prevention System, Immunity (anti malicious software system), and System Maintenance security requirements. The other security requirements such as Confidentiality (for secrecy and privacy requirements), Integrity, Availability, and Survivability are represented individually. Non-repudiation and Security Auditing requirements are deployed as tasks contributing to the execution of Tracking softgoal.

Security requirements and mechanisms can be represented by different GRL elements according to the nature of each security requirement and to highlight the importance of selected components.

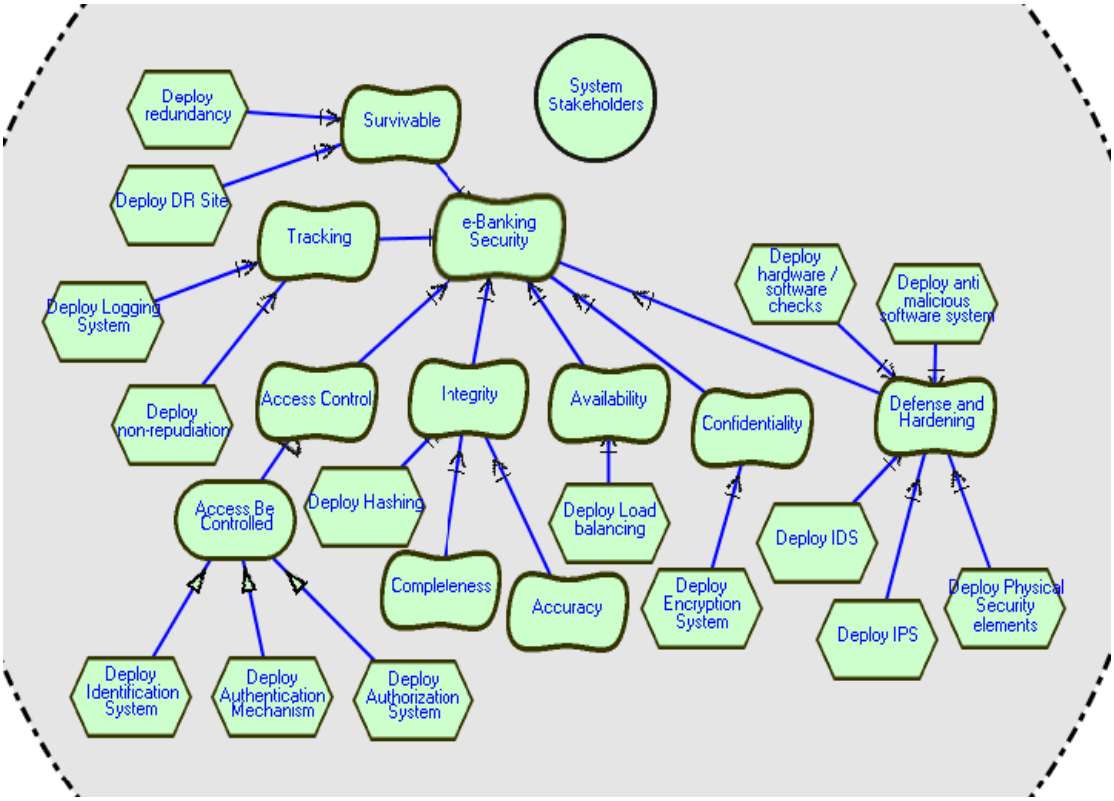


Figure 10.2 GRL Diagram for the E-Banking System Stakeholders' Security Requirements.

10.1.1.5. Case Study - Security Requirements UMLsec Representation

Representing the security requirements using UMLsec is different than representing them using the Goal Requirement Language (GRL). Single GRL diagram can describe all elicited security requirements, however, single UMLsec diagram can represent related security mechanism categories. For example, the UMLsec deployment diagram in Figure 10.3 describes the physical layout of the security mechanisms such as network link, type of traffic and medium, deployed modules, systems layout, and traffic flow security dependencies. Also other security mechanisms can be specified within other UMLsec diagrams. For example, sequence diagrams can address authentication mechanism, and role-based access control. Although GRL model can describe the security requirements and mechanisms in one

diagram, it does not enforce or validate the requirements. On the other hand, UMLsec diagrams permit the expression of security relevant characteristics within the model in the system specification and validate them.

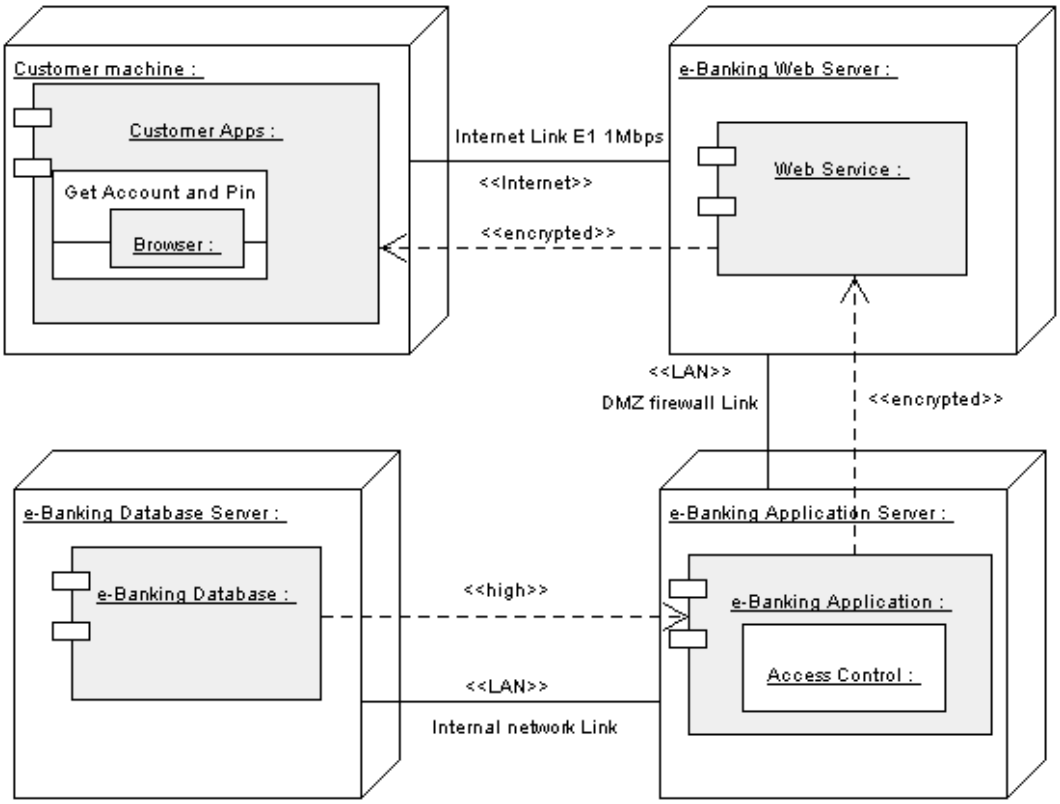


Figure 10.3 E-Banking System Deployment Diagram

10.2. Security Acceptance Test Development

Security acceptance and test development represent the second step of the systematic approach to the management of system security reengineering process. In this phase, security related test cases are developed to assess the existing system security related mechanisms. It is assumed that, other testing elements and external factors such as test environment have no effect on the test results.

10.2.1. Test Case Template Proposal

The test case template proposed in Figure 9.2 is used for assessing the existing system security mechanism, as well as testing the proposed security design model against the stakeholders' requirements. Before assessing the existing system security mechanism

using the proposed test cases templates, general stakeholders consent is taken. In case the test will run on a production environment and there is an associated impact on the offered services, such as interruption to account transfer processes, the test consequences will be highlighted to the stakeholders as part of seeking their explicit consent.

The security requirements are mapped to the security mechanisms or components in which the existing system is assessed against.

10.2.2. Deploying the Proposed Test Cases in the E-Banking Services Case Study

In order to assess the current security mechanisms for the local financial institution's system, the proposed test cases use the mapped security mechanisms as security requirement inputs.

10.2.2.1. Case Study - Access Control Suite Part-1 Test Case

Since identification and authentication security requirements are tied together, the first test case 1/10.2, shown in Figure 10.4 will combine the two requirements as part of Access control suite assessment – Part 1. The test is assessing an existing system's security mechanisms. And since the identification and authentication are normal system activities they will not interrupt the offered services. Therefore, there is no need for explicit stakeholders' consent. It is assumed that the existing system has the identification and authentication components, and the user has valid user name and password.

TC #	1/10.2	TC Description	Access Control Test – Part 1
Objectives	▪ Assess Identification mechanism. ▪ Assess Authentication mechanism.		
Live ☒	Model ☐	Test Impact	No Impact – Normal user side activities
Security requirements	The System should have the identification and authorization components for example, usernames and corresponding passwords database.		
Preconditions	Test Input	Expected Output	
The user has a valid username and password.	The user enters username and the valid password.	The user is granted general access to the system.	
Test observed output	Test verdict		
The user is identified and authenticated to the security system successfully.	PASS: The system's logs and behavior prove the test success.		

Figure 10.4 Access Control Part-1 Security Test Case

10.2.2.2. Case Study - Access Control Suite Part-2 Test Case

Since security authorization is included in the Access Control security suite, the test case 2/10.2, shown in Figure 10.5, with the objective to assess the security authorization mechanism. The test is used to assess the security mechanism in the existing system so it will be in the live mode. And since the authorization mechanism is a normal system security activity it will not interrupt the offered services, so there is no need for explicit stockholder's consent. It is assumed that the existing system has the security authorization mechanism such as role-based access control. In order to run the assessment test case successfully, the user needs to be identified and authenticated by the system. Since the test is executed in normal privileges level, the client services are allowed and the administrative services are denied.

TC #	2/10.2	TC Description	Access Control Test – Part 2
Objectives		Assess Security Authorization Mechanism	
Live ☒	Model ☐	Test Impact	No Impact – Normal user side activities
Security requirements		The System should have the authorization mechanism, such as role-based access control.	

Preconditions	Test Input	Expected Output
The user has been identified and authenticated	1- The user requests client side services. 2- The user requests admin side services.	The user is granted access to the client side services. The user's access to the admin services is denied.

Test observed output	Test verdict
The user is granted access to the client side services, and access to the admin services is denied.	PASS: The logs and system's behavior prove the test success.

Figure 10.5 Access Control Part-2 Security Test Case

In Section 10.1.1.4, Case Study Security Requirements GRL Representation, Defense and Hardening security suite includes five tasks corresponding to five mechanisms fulfilling Immunity, Intrusion Detection, Intrusion Prevention, Physical Protection, and System Maintenance security requirements. These five security requirements are included in the Defense and Hardening security suite test cases.

10.2.2.3. Case Study - Defense and Hardening Part-1 Test Case

Immunity security mechanisms are examined in test case 3/10.2, shown in Figure 10.6 as part of the Defense and Hardening security suite components. The test is used to assess the immunity security mechanisms in the existing system, so it will be in the live mode. It is assumed that the immunity security mechanism database has all the known virus signatures. Therefore, the test can be performed without explicit

stakeholders' consent. The test security requirements include Internet service-based antivirus protection, client-based antivirus protection, and server-based antivirus protection. Known virus or spyware is needed as input to the system.

It is expected that the immunity security mechanism will delete the infected email and web viruses successfully as well as notifying the security officer.

TC #	3/10.2	**TC Description**	Defense and Hardening Test – Part 1
Objectives	▪ Assess System Immunity Security Mechanisms.		
Live ☒	**Model** ☐	**Test Impact**	No Impact – Deployed using known viruses and Trojan horses.
Security requirements	The System should have three levels antivirus, and anti spyware protection. For example: services-based antivirus system(http, ftp, and smtp); Client-based antivirus system, and Server-based antivirus system		
Preconditions	**Test Input**	**Expected Output**	
The user can access known virus or spyware. The user has known virus.	1- The user sends virus-infected email. 2- The Internal user downloads virus from the Internet.	1- Email virus is deleted successfully. 2- Web surfing virus is deleted successfully. 3- The security officer receives antivirus notification message.	
Test observed output		**Test verdict**	
E-mail virus is deleted successfully. However, the web virus reaches the client. The virus is deleted by the client antivirus.		FAIL: The virus should have been deleted in the gateway before accessing the internal network.	

Figure 10.6 Defense and Hardening Part-1 Security Test Case

10.2.2.4. Case Study - Defense and Hardening Part-2 Test Case

Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) security mechanisms are examined in test case 4/10.2, shown in Figure 10.7 as part of the Defense and Hardening security suite components. The test is performed to assess the IDS and IPS mechanisms in the existing system components without interrupting the offered activates. Explicit stakeholder consent is not needed. The system should have logging and alerting mechanism and IPS as part of the security requirements.

It is expected that the IDS and IPS systems will log all identification, authentication, and authorization related activities. The IDS will notify repeated failed attempts within five minutes to the security officer. The IDS will also report failed attempts within 24 hours and the prevented unauthorized accesses.

TC #	4/10.2	TC Description	Defense and Hardening Test – Part 2
Objectives		▪ Examine the Intrusion Detection System. ▪ Examine the Intrusion Prevention System.	
Live ☒	**Model** ☐	**Test Impact**	No Impact – Offered services will not be affected.
Security requirements		The System should have complete logging and reporting mechanisms. For example: IPS, Server-based IDS system, or in house developed logging and alerting application.	

Preconditions	Test Input	Expected Output
The user knows the e-Banking web site. Applications client side module is installed.	1-User1 enters wrong password once. 2-User2 enters wrong password several times. 3-The user attempts to access unauthorized application.	1-The logging system (network and servers) records all User1 and User2 activities. 2-The Security officer is notified within five minutes about User2 activities. 3-Security officer receives reports about User1 activities, and all prevented activities on the next day.

Test observed output	Test verdict
Security officer receives user1, and user2 activity reports, however the logs are not complete.	FAIL: Network and security device logs are incomplete.

Figure 10.7 Defense and Hardening Part-2 Security Test Case

10.2.2.5. Case Study - Defense and Hardening Part-3 Test Case

Physical security mechanisms are examined in test case 5/10.2, shown in Figure 10.8 as part of the Defense and Hardening security suite components. Since human intervention is needed to assess the physical security mechanisms, there is a need for the stakeholders consent. The data center should be physically secured by having proper locked doors and security guards.

The stakeholders will arrange for including the user name in the trusted list, and then the user will try to access the data center.

It is expected that the user will be allowed to enter the premises. The user will fail to enter the data center since proper keys are needed.

TC #	5/10.2	**TC Description**	Defense and Hardening Test – Part 3
Objectives	Assess Physical Security Mechanisms.		
Live ☒	**Model** ☐	**Test Impact**	Personnel Impact – Stakeholders consent is needed to avoid personnel conflict.
Security requirements	The Data center should be protected physically, using locked doors and security guards for example.		

Preconditions	**Test Input**	**Expected Output**
The user knows the data center location. The user name is included in the trusted list. The user does not have valid keys.	The user tries to enter the data center.	1- Security guard allows the user to enter the data center building. 2- The user fails to enter the data center because valid keys are needed.

Test observed output	**Test verdict**
The user is identified to the guard, and failed to enter the data center.	PASS: The system's behavior meets the physical security objectives.

Figure 10.8 Defense and Hardening Part-3 Security Test Case

10.2.2.6. Case Study - Defense and Hardening Part-4 Test Case

System maintenance security mechanism is examined in test case 6/10.2, shown in Figure 10.9 as part of the Defense and Hardening security suite components. It is assumed that the existing system does not have any redundancy, so the offered services will be interrupted during the system maintenance. The test will run in the live mode and stakeholders consent is needed. Also It is assumed that the institution is adopting security regression testing as part of their security system maintenance and enhancement procedures. It is expected that the system engineer will follow the institution's prescribed procedure and will perform security regression test.

TC #	6/10.2	TC Description	Defense and Hardening Test – Part 4
Objectives	Assess the System Maintenance Security Mechanisms.		
Live ☒	Model ☐	Test Impact	Service Impact – Stakeholders consent and system downtime arrangements are needed.
Security requirements	The institution procedures should include one for the security system maintenance and enhancement. The institution should also adopt Security Regression Testing.		
Preconditions	Test Input	Expected Output	
Trusted system engineer will perform system upgrade.	New software release.	1-The system engineer follows the institution's procedure for system upgrade. 2- Security regression testing is performed.	
Test observed output		Test verdict	
The institutional policies are followed, and security regression testing is performed.		PASS: The Security measures meet the objectives.	

Figure 10.9 Defense and Hardening Part-4 Security Test Case

10.2.2.7. Case Study - Integrity and Privacy Test Case

Integrity and Privacy security mechanisms are assessed in test case 7/10.2, shown in Figure 10.10. Since the test will be performed as normal user side activities and will not interrupt the offered services, stakeholders' consent is not needed.

It is assumed that the security system will encrypt all communication traffic and data. Also it is assumed that proper authentication mechanism is adopted. The user will access the secure services while running network-capturing tools in the background. In addition, the user will try to access other user's data.

It is expected that the user will not be able to decrypt the captured network traffic due to strong encryption, and will not be able to access other user's data.

TC #	7/10.2	**TC Description**	Integrity and Privacy Test
Objectives	Assess the existing system's Integrity, and Privacy Security Mechanisms.		
Live ☒	**Model** ☐	**Test Impact**	No Impact - Normal user activities. The offered services will not be interrupted.
Security requirements	The security system should deploy strong hashing, encryption, and authorization system security mechanisms.		
Preconditions	**Test Input**	**Expected Output**	
The user has valid ID and password. The user is able to capture the network traffic.	Normal user authentication activities.	1-The user captures encrypted packets. 2- The user fails to decrypt the data.	
Test observed output	**Test verdict**		
The user is able to capture encrypted traffic. The user fails to decrypt the captured traffic. The user fails to access other user's data.	PASS: The user is not able to read encrypted data. The user is not able to access other user's data.		

Figure 10.10 Integrity and Privacy Security Test Case

10.2.2.8. Case Study - Auditing and Non-repudiation Test Case

Auditing and Non-repudiation security mechanisms are assessed in test case 8/10.2, shown in Figure 10.11. Since the test will be performed as normal user side activity and will not incur any services interruptions, stakeholders consent is not needed.
Assuming that the security system includes complete and tamperproof logging system for all transactions, as well as proper authorization mechanism is adopted. The user will perform a financial transaction and will later claim that he did not perform the transaction. It is expected that the security system will log the transactions completely, with proper ID records mapped to the user's digital signature.

<table>
<tr><td>TC #</td><td>8/10.2</td><td>TC Description</td><td colspan="2">Auditing and Non-repudiation Test</td></tr>
<tr><td>Objectives</td><td colspan="4">Assess Auditing and Non-repudiation Security Mechanisms.</td></tr>
<tr><td>Live ☒</td><td>Model ☐</td><td>Test Impact</td><td colspan="2">No Impact: Normal activities, the offered services will not be affected.</td></tr>
<tr><td colspan="2">Security requirements</td><td colspan="3">The security system should deploy proper non-repudiation and auditing mechanisms, such as complete and tamperproof logging system, and digital signature authentication mechanism.</td></tr>
<tr><td colspan="2">Preconditions</td><td>Test Input</td><td colspan="2">Expected Output</td></tr>
<tr><td colspan="2">The user has a valid ID and password.
The user performs financial transaction.</td><td>Normal authenticated user activities.</td><td colspan="2">1- Complete and tamperproof records are found in the logging system.
2- Proper user's digital signature records are logged.</td></tr>
<tr><td colspan="3">Test observed output</td><td colspan="2">Test verdict</td></tr>
<tr><td colspan="3">There are incomplete logs, and the users are identified by username and password only.</td><td colspan="2">FAIL: The observed output is different from the expected output.</td></tr>
</table>

Figure 10.11 Auditing and Non-repudiation Security Test Case

10.2.2.9. Case Study - Survivability Test Case

Survivability security mechanisms are assessed in test case 9/10.2 shown in Figure 10.12. Testing survivability could impact the offered services due to failure in switching from the primary component to the secondary component, so stakeholders consent is needed. It is assumed that, there is a redundancy in the system such as, server clusters, dual network links, dual network devices, and Uninterruptible Power Supply (UPS), as well as having a contingency disaster recovery site.

It is expected that the load balancing and redundancy system will takeover and handle the entire requests. Also it is expected that the UPS system will supply power to the servers and network devices.

TC #	9/10.2	**TC Description**	Survivability Test
Objectives	Assess Survivability Security Mechanisms.		
Live ☒	**Model** ☐	**Test Impact**	Impact may occur due to switching to the redundant components failure.
Security requirements	The security system should deploy redundancy in the servers, network links, and network devices, as well as having UPS. Includes Disaster Recovery site.		

Preconditions	**Test Input**	**Expected Output**
Stakeholders agree on performing the test.	One cluster node is switched off. Data center power supply is switched off.	1-The other cluster node handles all the requests. 2- The UPS system provides power to the servers and network devices.

Test observed output	**Test verdict**
The other cluster node handles the traffic, and the UPS takeover.	PASS: The observed output is the same as the expected output.

Figure 10.12 Survivability Security Test Case

10.3. Acceptance Testing and Vulnerability Assessment

Acceptance testing and vulnerability assessment is the third step of the author's systematic approach to the management of system security reengineering process. In this phase a gap analysis test is performed by applying the developed test cases to the existing system's security mechanisms. All inconclusive or failing test cases are identified and traced back to the specified requirements. This security requirements gap can be further confirmed by a vulnerability analysis covering the failing tests, and a verdict on the satisfaction of each of the stakeholders' requirements is produced.

10.3.1. Case Study - Testing the Existing System Security Mechanisms

The eight test cases developed in section 10.2 are used to assess the existing system security requirements against the corresponding security mechanisms. Table 10.2 summarizes the test cases result including the used security mechanisms and stakeholders' explicit/general consents.

Table 10.2 Test Cases Security Assessment Result

Security Requirement	Component or Mechanism	Consent	Test Result
Identification	Access control system.	General	Satisfy
Authentication	Access control system.	General	Satisfy
Authorization	Role-based access control.	General	Satisfy
Immunity	Three layers antivirus and anti spyware protection.	General	Inadequate
Integrity	Hashing and encryption systems.	General	Satisfy
Intrusion Detection	Host and network-based IDS systems. In house developed middleware alerting logic system.	General	Inadequate
Intrusion Prevention	IPS System, and in house developed middleware access logic.	General	Violate
Non-repudiation	Complete and tamperproof logging system, user's digital signature, encryption, and hashing.	General	Inadequate
Privacy/Secrecy	Encryption, and strong access control.	General	Inadequate
Security Auditing	Audit trials, and event logs.	General	Violate
Survivability	Load balancing and redundancy in servers, network links and data centers.	Explicit	Inadequate
Physical Protection	Locked doors, security guards and surveillance systems.	Explicit	Inadequate
System Maintenance	Change control polices and procedures, and security regression testing.	Explicit	Inadequate

10.4. Security Design Modifications and Standard Mapping

Security design modifications and standard mapping is the fourth step of the author's systematic approach to the management of security reengineering process, in this phase action plan and design modification with the needed security mechanisms are developed to accommodate the security requirements that were identified earlier in Section 10.1. In this step, these modifications must be inline with the formal requirement specifications obtained in the security requirements elicitation and standard mapping phase 1. Once approved by the stakeholders, this plan will be executed.

10.4.1. Case Study – Action Plan and Design Modifications.

According to the security assessment tests, the identification, authentication, authorization, and integrity security mechanisms are satisfying the stakeholders' requirements described in Section 10.1, Security Requirements Elicitation and standard mapping. And in accordance with ISO/IEC 17799:2000 standard clauses as listed in Table 10.3, so the action for these four requirements is to report satisfactory condition and no need for design modification.

Table 10.3 Satisfactory Security Mechanisms According to Stakeholders' Requirements as well as ISO/IEC 17799:2000 Standard Clauses

Security Requirement	Component or Mechanism	ISO/IEC 17799:2000 Compliance
Identification	Access Control System	Clause: 9.1.X, 9.2.X, 9.5.X
Authentication	Access Control system	Clause: 9.1.X, 9.3.X, 9.4.X, 9.5.X, 10.2.3
Authorization	Role-based access control	Clause: 9.5.X, 9.6.X
Integrity	Hashing and Encryption systems	Clause: 8.6.X; 8.7.X, 10.4.X

Regarding the missing and inadequate security mechanisms, the actions are to report the current condition, modify the existing design by including security mechanisms that satisfy the stakeholders' requirements described in Section 10.1, Security elicitation and standard mapping, as well as satisfying ISO/IEC 17799:2000 standard clauses.

Table 10.4 represents the security gap analysis of the inadequate and missing mechanisms in accordance to the stakeholders' requirements and ISO/IEC 17799:2000 standard clauses

Table 10.4 Security Gap Analysis for the Missing and Inadequate Security Mechanisms According to the Stakeholders' Requirements.

Security Requirement	Tested Mechanisms	Existing Mechanisms	Needed Mechanisms	Standard Mapping
Immunity	Three layers antivirus and anti spyware protection.	E-mail, client and server sides protection.	Web-based protection.	Clause: 7.X, 8.3.X, 11.X, 12.1.X.
Intrusion Detection	Host-based IDS System, network-based IDS system, and in house developed middleware alerting system.	Host-based IDS system in critical servers. In house developed middleware logic alerting.	Host-based IDS for all servers. Network-based IDS.	Clause: 9.7.X, 12.3, 8.4.X.
Intrusion Prevention	IPS System, and in house developed middleware access logic.	In house developed middleware access logic.	Network-based IPS.	Clause: 7.2.X, 7.3.X.
Non-repudiation	Complete and tamperproof logging system, user's digital signature, encryption, and hashing.	Logging, hashing and encryption are Limited to customers' ATM access, and banking core applications.	Complete logging and encryption for employees file and email systems. User's digital signature.	Clause: 10.3.X.
Privacy/Secrecy	Encryption, and strong access control.	Limited to ATM machines, and core banking applications.	Encryptions for employees file and email systems.	Clause: 8.7.X, 10.3.X, 12.1.X.
Security Auditing	Audit trials, and event logs.	Incomplete scattered logging and auditing systems.	Consolidated logging and auditing systems.	Clause: 8.4.X, 11.3.X, 9.7.X, 12.3.X.
Survivability	Load balancing and redundancy in servers, network links and data centers.	Deployed for core banking data and applications.	E-mail, Web, and telephone services, as well as employees file system load balancing and redundancy.	Clause: 7.X, 11.X.
Physical Protection	Locked doors, security guards and Surveillance system.	Buildings, ATMs and data centers.	Computerized surveillance, and electronic-based physical access control systems.	Clause: 7.X, 8.6.X.
System Maintenance	Change control polices and procedures, and security regression testing.	Controlled manually for critical systems.	Logical enforcement.	Clause: 10.5.X, 11.1.5.

10.5. Implementation of Modifications

Implementation of modifications is the fifth step of the author's systematic approach to the management of system security reengineering process. In this phase the security design modifications are implemented in the system according to the action plan after getting the stakeholders approval. The modifications will address the needed mechanisms identified in the gap analysis phase, Table 10.4.

10.6. Final Acceptance Testing and Vulnerability Assessment

Final acceptance testing and vulnerability assessment represents the final step of the author's systematic approach to the management of system security reengineering process. In this phase, the re-engineered system security is reassessed against the initial security requirements using the same test suites, and a final security acceptance test is performed.

The newly added mechanism could affect other requirements mainly non-functional requirements. For example, if additional audit log requirement is needed, the modifications performed will meet this non-functional security requirement. However, additional tests should be performed to retest the performance requirements since requiring the system to collect audit logs at a refined granularity levels may degrade some performance indicators in the system. To deal with the identification of affected functional or non-functional requirements, it is suggested to benefit from the theoretical work done in the area of system regression testing. The methodology does not address this issue of regression testing since it is beyond the scope of this thesis.

Ultimately, the reengineered system will be handed to the operations department to maintain the security status, and update the system security batches according to its security requirements whenever new vulnerabilities are discovered. Standard baseline is maintained and the gap will be closed according to the process shown in Figure 10.13. Closing the gap must be an ongoing process, and must be semi-automated after putting the management tools and procedures in place.

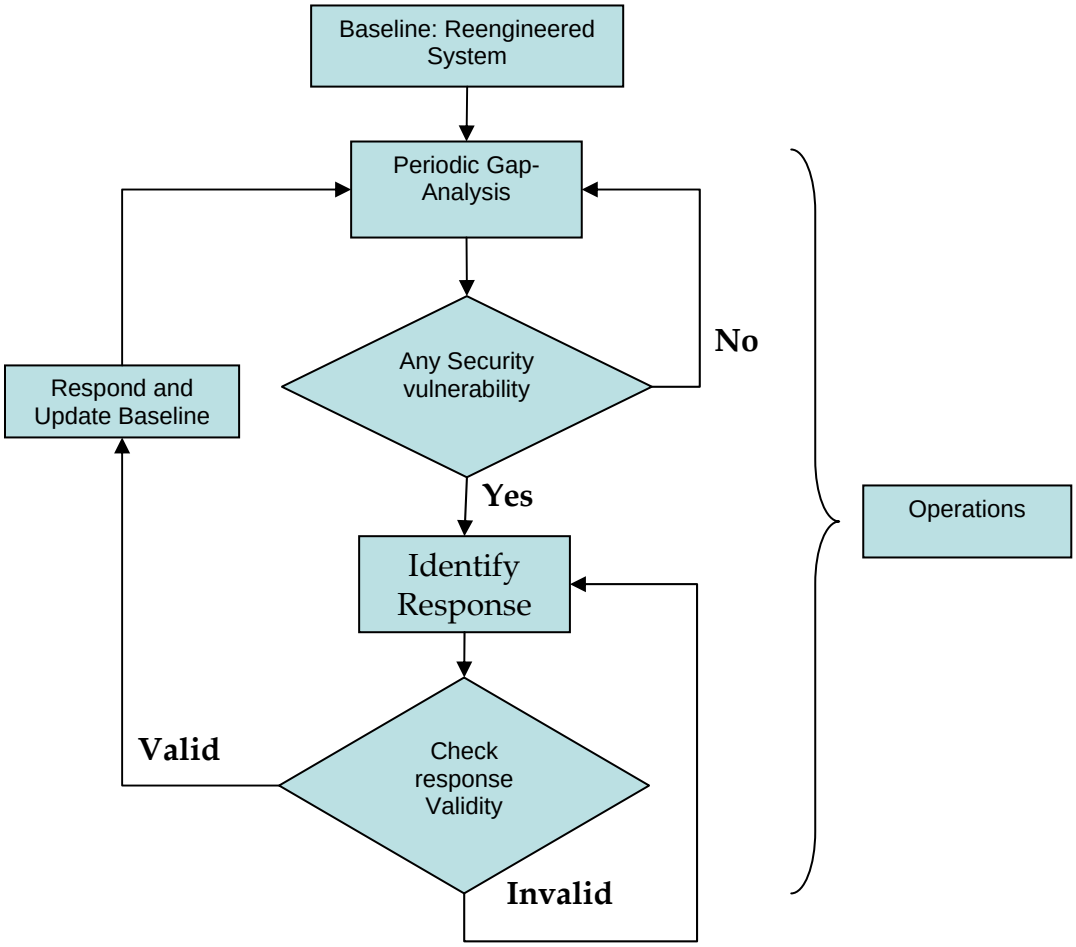


Figure 10.13 Continuous Process for Addressing Security Weaknesses

CHAPTER 11

CONCLUSION AND FUTURE WORK

11.1. Conclusion

The main objectives of this thesis are (1) to develop a systematic and formal approach to the process of reengineering security in an existing system, and (2) to use the recent advances in formal modeling methods and standard languages and extensions such as UML, UMLsec and GRL to achieve the main objectives.

Security requirements were discussed and grouped to make their elicitation easier. The current security standards have been examined. ISO/IEC 17799:2000 security standard was chosen since it addresses all types of security requirements, and for its international visibility. An overview of all known security requirements related to the four security goals, namely confidentiality, integrity, availability and accountability, were discussed. A comprehensive listing and mapping of all known types of security requirements are linked to the UMLsec security stereotypes, and as a result of this mapping, UMLsec have been extended. Finally, security requirements are mapped to the security mechanism using corresponding UMLsec serotypes.

A security reengineering approach have been considered to avoid expensive system development life cycles fixes. This is achieved by having an effective security reengineering design model specifying detailed requirements for the security needs. This approach focuses on satisfying legitimate user requirements while blocking malicious user requirements at system reengineering time.

11.2. Contributions of Thesis

The main contributions of this thesis are:

1. Extension of the UMLsec language to improve its generality, and its expressive power for the modeling of all security requirements types. These extensions included stereotypes like <<*audit log*>>, <<*multiplicity*>> and <<*no misuse*>>. These stereotypes are used to address additional availability, accountability and immunity requirements. These stereotypes can be used in UMLsec security use case, deployment and sequence diagrams.

2. Mapping the international standard on information systems security, ISO/IEC 17799:2000, to a classification of security requirements and UMLsec stereotypes.
3. Introducing an approach to develop security test cases starting from the formal security requirements specifications described using UMLsec and GRL specifications languages.
4. Incorporating the use of requirements models such as UMLsec and GRL, the mapping to ISO/IEC 17799:2000, and the security test case development process, into an overall security reengineering life cycle approach that can be applied any time security requirements change.

11.3. Future Directions

This work can be further enhanced by including in the author's approach other security-related features, such as impact of security mechanisms on performance, and other functional and non-functional system features. Also, a study should be done on the effect of security enhancements on the amount of retesting to perform in the last step of the reengineering methodology This proposed enhancement can benefit from the research work done in the regression testing area. Furthermore, the study of risks associated with non-conformity to the security requirements, and prioritizing the security requirements according to the risk associated with each of them, can be performed. Quantitative assessment of the cost of risk elimination or reduction and its relation to the increased trust in the security system needs to be further investigated. Moreover, new types of security requirements may require additional UMLsec stereotypes design models to analyze them before implementation, and to test their conformance once implemented.

Most of the above future enhancements were raised during this research. The author was not able to answer them because of necessary constraints on the scope of this work. However, the author believes that he has established a foundation for future research work in this area.

REFERENCES

[1] International Telecommunications Union ITU-T, Draft Recommendations Z.151 – Goal-oriented Requirements Language (GRL), Geneva 2002.

[2] J. Jurjens, Secure Systems Development with UML, Springer Verlag, 2005.

[3] ISO/IEC 17799:2000, International Organization for Standardization (ISO), Code of Practice for Information Security Management, Switzerland, 2000.

[4] COBIT 3.1 Edition: Framework, IT Governance Institute, Rolling Meadows, IL, USA, 2000.

[5] COBIT 3.1 Edition: Management Guidelines, IT Governance Institute, Rolling Meadows, IL, USA, 2000.

[6] Common Criteria (CC), Common Criteria for Information Technology Security Evaluation, August 1999, website: http://www.commoncriteria.org.

[7] E. Tittel, M. Chapple and J. Stewart, CISSP® Certified Information Systems Security Professional Study Guide, SYBEX, USA, 2003.

[8] K. Saleh and A. Al-Zarouni, Capturing non-functional software requirements using the user requirements notation, Proceedings of the 2004 International Conference on Innovations in Information Technology, Dubai, October 2004.

[9] K. Wiegers, Software Requirements, Second Edition, Microsoft Press, USA, 2003.

[10] Object Management Group (OMG) - Systems Modeling Language (SysML) Specification, Draft Version 0.9, January 2005.

[11] D. Firesmith, Engineering Security Requirements, Journal of Object Technology, Vol. 2, No. 1, Jan-Feb 2003.

[12] R. Anderson, Security Engineering, John Wiley, 2001.

[13] R. Henning and L. Garner, Using the System Security Life Cycle Plan to Enforce A System Security Engineering Process, Government Communications Systems Division, 1999.

[14] R. Krutz and R. Vines, The CISSP Prep Guide: Mastering the Ten Domains of Computer Security, John Wiley & Sons Inc, Canada, 2001.

[15] T.T. Dinh-Trong, A systematic approach to testing UML design models, 7th International Conference on the Unified Modeling Language (UML), Lisbon, Portugal, 2004.

[16] O. Pilskalns, A. Andrews, R. France and S. Ghosh, Rigorous testing by merging structural and behavioral UML representations, 6th International Conference on the Unified Modeling Language (UML), San Francisco, USA, 2003.

[17] P. Devanbu and S. Stubblebine, Software engineering for security: a roadmap, ICSE'2000, 2000, pp. 227-239.

[18] R. Linger, H. Lipson, J. McHugh, N. Mead and C. Sledge, Life-Cycle Models for Survivable Systems, Software Engineering Institute, TR-2002-026, October 2002.

[19] S. McConnell, Code Complete, Microsoft Press, Washington, 1993.

[20] J. Weinberg, Quality Software Management: Volume 1 Systems Thinking; Volume2 First-Order Measurement; Volume 3 Congruent Action; Volume 4 Anticipating Change, Dorset House, New York, 1992-97.

[21] S. Robertson and J. Robertson, Mastering the Requirements Process, ACM Press, New York, 1999.

[22] Object Management Group, OMG Unified Modeling Language Specification v1.5, OMG Document formal/03-03-01, March 2003.

[23] D. Amyot, Introduction to the user requirements notation: learning by example, Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 42, No. 3, 2003, pp. 285-301.

[24] International Telecommunications Union ITU-T, Recommendation Z.150, User Requirements Notation (URN) – Language Requirements and Framework, Geneva, 2003.

[25] Information Technology Security Related Definitions, website: http://www.itsecurity.com, Accessed on 28 December 2004.

[26] R. Turn, Security and Privacy Requirements in Computing, Proceedings of 1986 ACM Fall joint computer conference, IEEE Computer Society Press, California, 1986, pp. 1106-1114.

[27] Home of GRL, website: http://www.cs.toronto.edu/km/GRL, Accessed on 08 December 2004.

[28] J. Schmuller, SAMS Teach Yourself UML in 24 Hours, third edition, SAMS Publishing, USA, 2004.

[29] D. Firesmith, Security Use Cases, Journal of object technology, Vol. 2, No. 3, May-June 2003.

[30] J. Jurjens, UMLsec: Extending UML for secure systems development. In J.-M. The Unified Modeling Language, Vol. 2460 of LNCS, Springer, 2002, pp. 412-425.

[31] S. H. Houmb and J. Jurjens, Developing secure networked web-based systems using model-based risk assessment and UMLsec, In 10th Asia-pacific Software Engineering conference (APSEC 2003), IEEE Computer Society, New York, 2003, pp. 488ff.

[32] J. Jurjens and P. Shabalin, Automated verification of UMLsec models for security requirements, In J.-M. jezequel, H. Hubmann, and S. Cook, editors, UML 2004- The Unified Modeling Language, Vol. 2460, of Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2004, pp. 412-425.

[33] V. Sawma, E-Commerce Security, A New Methodology for Deriving Effective Countermeasures Design Modes, Master Thesis, University of Ottawa, Canada, 2002.

[34] National Institute of Standards and Technology (NIST), Special Publication 800-33, Underlying Technical Models for Information Technology Security, 2001, website: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf, Accessed on 17 February 2005.

[35] BS 7799-2:1999: Information Security Management – Part 2: Specification for information security management systems, British Standards Institution (BSI), July 2002.

[36] Control Objectives for Information and related Technology COBIT, Information Systems Audit and Control Association ISCA, website: http://www.isaca.org/cobit, Accessed on 24 March 2005.

[37] J. Jurgens, Secure Systems Development with UML, Viki UMLsec-Web interface tools, website: http://www4.in.tum.de:8180/vikinew/vikiweb, Accessed on 20 Jan 2005.

[38] J. Jurjens, Developing security-critical applications with UMLsec – a short walk-through, Novatica, Issue #168, March-April 2004.

[39] UML CASE tool Poseidon, website: http://www.gentleware, Accessed on 20 Jan 2005.

[40] Computer Emergency Response Team (CERT), vulnerabilities statistics, website: http://www.cert.org/stats/cert_stats.html, Accessed on 20 April 2005.

VITA

Ghanem Ibrahim Elshahry was born on January 10, 1969, in Khan Younis, Palestine. He was educated in Khan Younis High School in Palestine. In 1993 he graduated with honor as the top ranked engineering student at Ajman University of Science and Technology, Ajman, UAE. His degree was a Bachelor of Science in Electrical Engineering with a minor in Electronics. In 2002, Mr. Elshahry enrolled on Engineering Systems Management master program at the American University of Sharjah (AUS).

Ghanem's IT Security expertise includes government and private institutions such as American University of Sharjah, Ajman University of Science and Technology, and Ministry of Information and Culture.
Since September 2004, he is working at Abu Dhabi Islamic Bank as IT Security Specialist, and Head of IT Security Team.

Mr. Elshahry is a Certified Information Systems Security Professional (CISSP), and holds several other professional certificates, namely CCNA, CCSA, MCSE, SCNA, SCSA and more. In 2002, he received the AUS Chancellor's award for his distinguished service to AUS.