

Information & Communication Security (WS 18/19)

Security Engineering

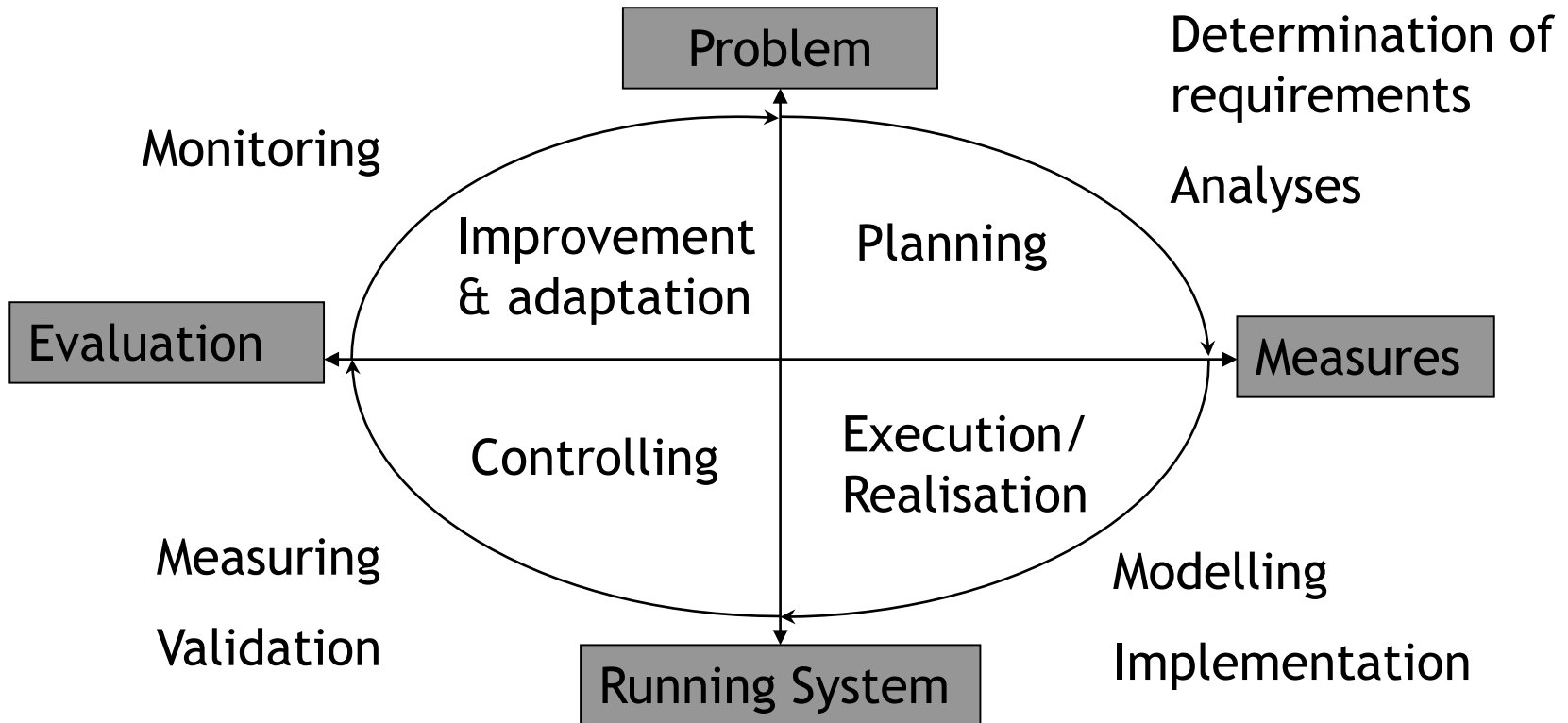
Prof. Dr. Kai Rannenber

Chair of Mobile Business & Multilateral Security
Goethe-University Frankfurt a. M.

- Introduction
- Secure System Development Process
- Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

- Security Engineering – a disciplined approach to build secure systems
- General methodical approach from Software Engineering

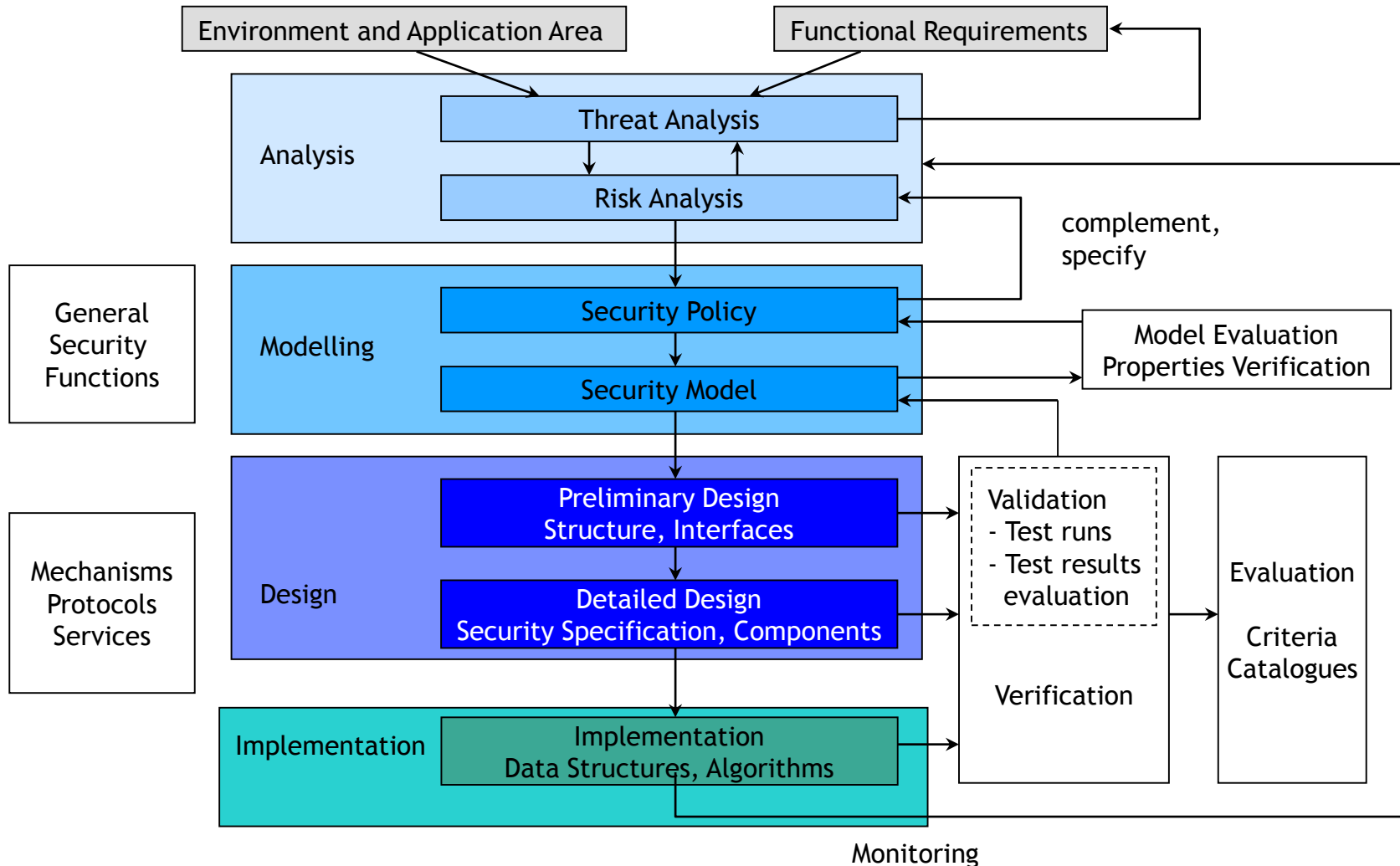
Iterative Engineering Process



- Introduction
- Secure System Development Process
- Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

- Planning phase
 - Structural analysis
 - Determination of requirements, especially for protection
 - Threat analysis
 - Risk analysis
- Realisation phase
 - Security policy
 - Strategy model
 - Implementation
- Controlling
- Improvement and adaptation

Secure System Development Process



- Introduction
- Secure System Development Process
- Analysis
 - **Structural Analysis**
 - Determination of protection needs
 - Threat Analysis
 - Risk Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

- System Requirements

- System Functions
- System Components
- System Purposes

=> Requirements specification

=> Net topology

- Connection of the Local Area Network to the outside world: ISDN, DSL, Satellite
- External connections (e.g. a mother company and a subsidiary company): Broadband LAN, leased lines

=> Properties of each component

- Unique ID
- Operating System
- IP Address
- Purpose

Requirements specification

- Functional requirements

 - [FR25] The OS should provide usual environment to users and developers

 - [FR26] The OS should provide a trusted way of user authentication

 - [FR27] User should be able to refine access permissions concerning their objects
without disturbing the security of the whole system

 - [FR28] Users should be able use the OS without experience in security

 - • •

- Performance requirements

- Quality requirements

- Introduction
- Secure System Development Process
- Analysis
 - Structural Analysis
 - Determination of protection needs
 - Threat Analysis
 - Risk Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

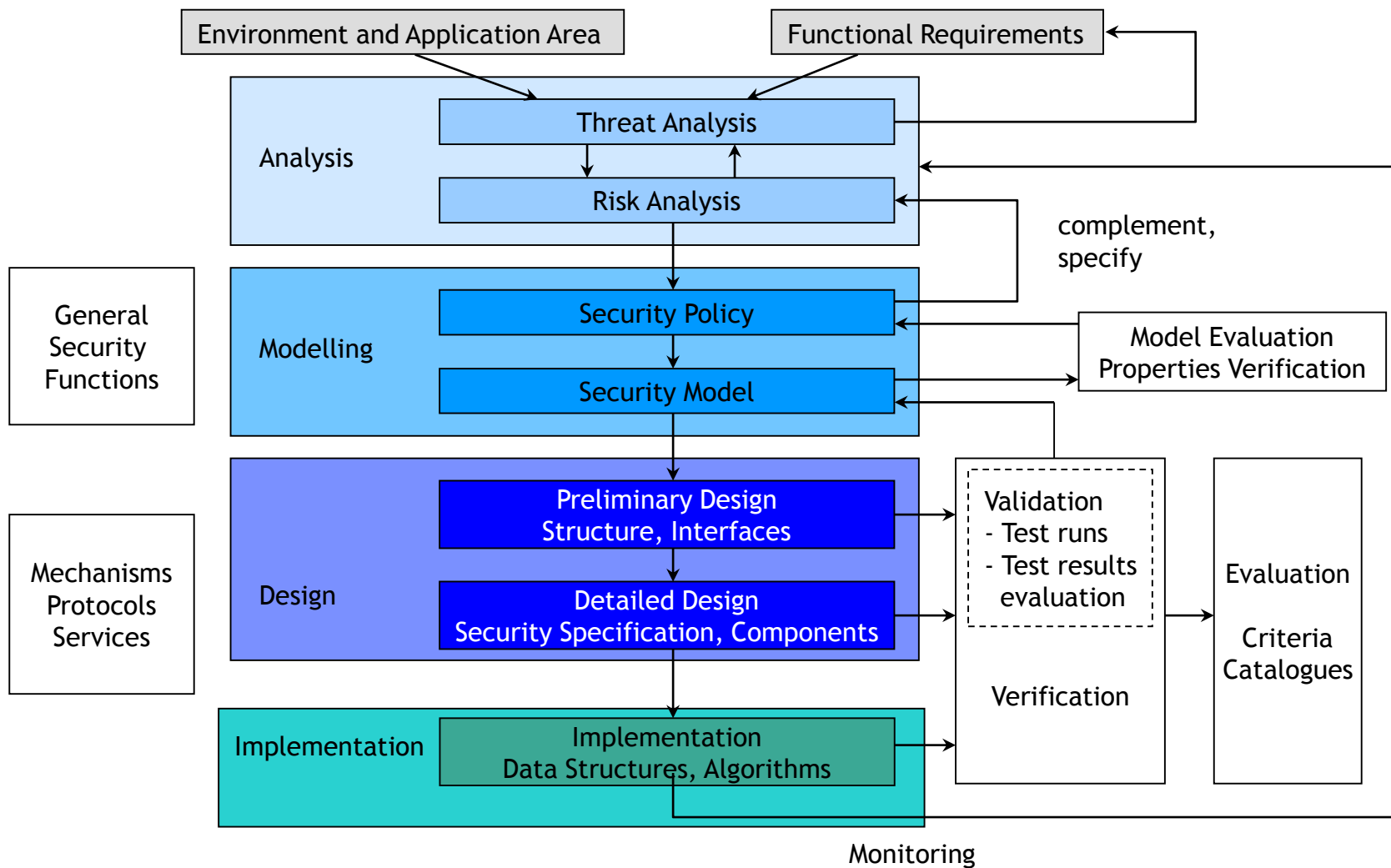
- Possible damage
 - Difficult to define in general
 - Assess based on scenarios

- Semi-quantitative categories for Protection Needs
 - Low to medium: Damage is manageable.
 - High: Damage could be considerable.
 - Very high: Damage may reach an existential dimension.

- Violation of laws, rules, or agreements
(e.g. the constitution, (German) BGB, etc.)
- Curtailing the right of informational self-determination
(e.g. unwarranted transfer of personal data)
- Curtailing of personal integrity
(e.g. medical systems (or databases))
- Curtailing task completion
(e.g. defective production because of wrong control data)
- Negative consequences for the reputation
(e.g. website deformation => prestige loss)
- Negative financial consequences
(e.g. unwarranted research findings transfer)

- Introduction
- Secure System Development Process
- Analysis
 - Structural Analysis
 - Determination of protection needs
 - Threat Analysis
 - Risk Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

Secure System Development Process - Threat Analysis

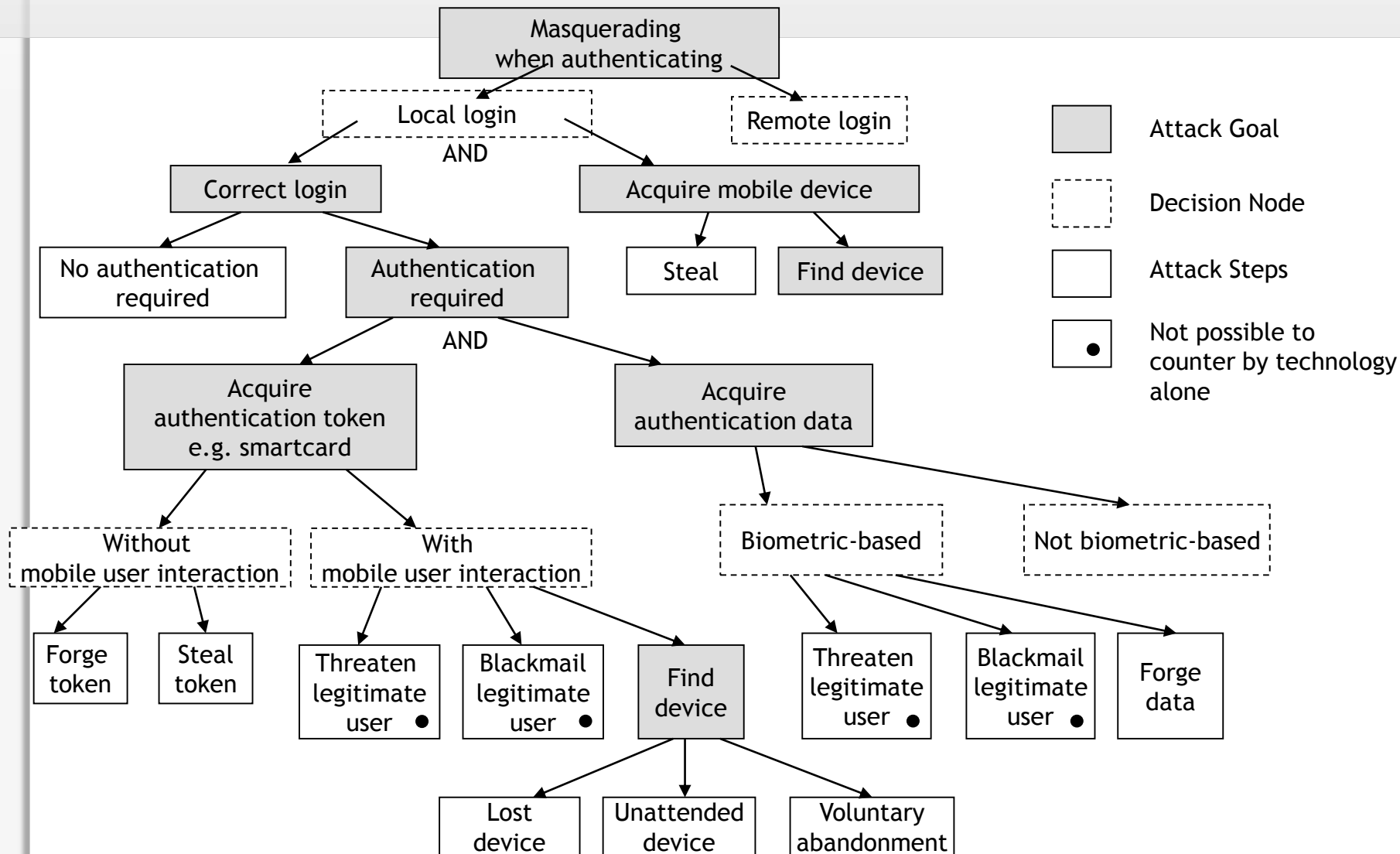


- The examination of threat sources against system vulnerabilities to determine the threats for a particular system in a particular operational environment
- Available approaches
 - Threat matrix
 - Attack tree

- System threats can be presented as an **attack tree**:
 - **Tree root**: symbolizes the attack goal
 - **Next level(s)**: contain(s) provisional goals (as nodes) required to reach the final attack goal
 - **Nodes**
 - “Or” nodes (standard): represent alternatives
 - “And” nodes: have to occur in common
 - **Leaves**: contain options to attack the goal

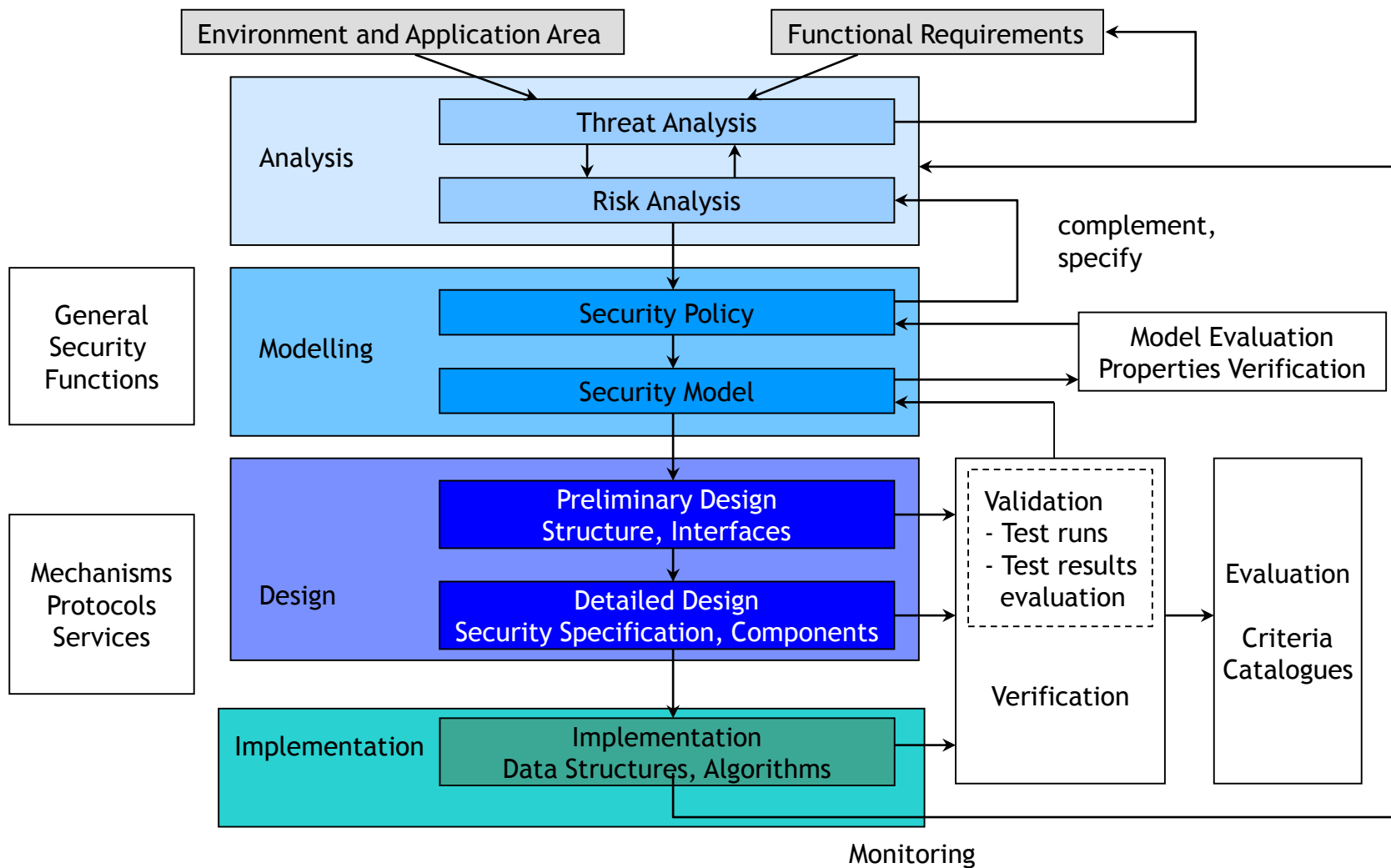
- Intruding into a system by using the identity of an authorized user
- Disabling required functions (e.g. for protection)
- Reading (and writing) of sensitive data
- Unauthorized changing and receiving of sensible information exchanged via (electronic) communication ways

Example: Attack Tree for the Mobile OS



- Introduction
- Secure System Development Process
- Analysis
 - Structural Analysis
 - Determination of protection needs
 - Threat Analysis
 - Risk Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

Secure System Development Process - Risk Analysis



- Threat rating
- Items to be considered
 - Who is the **attacker** (e.g. spy, hacker, co-worker, ...)?
 - Attacker's **knowledge** (newbie or IT Professional)
 - Estimation of possible **damage** (low to high)
 - Attacker's **final goal** (information, money, ...)
 - **Reasons** for attacking (experiment, revenge, gains, ...)
 - Attacker's **budget** (low to high)

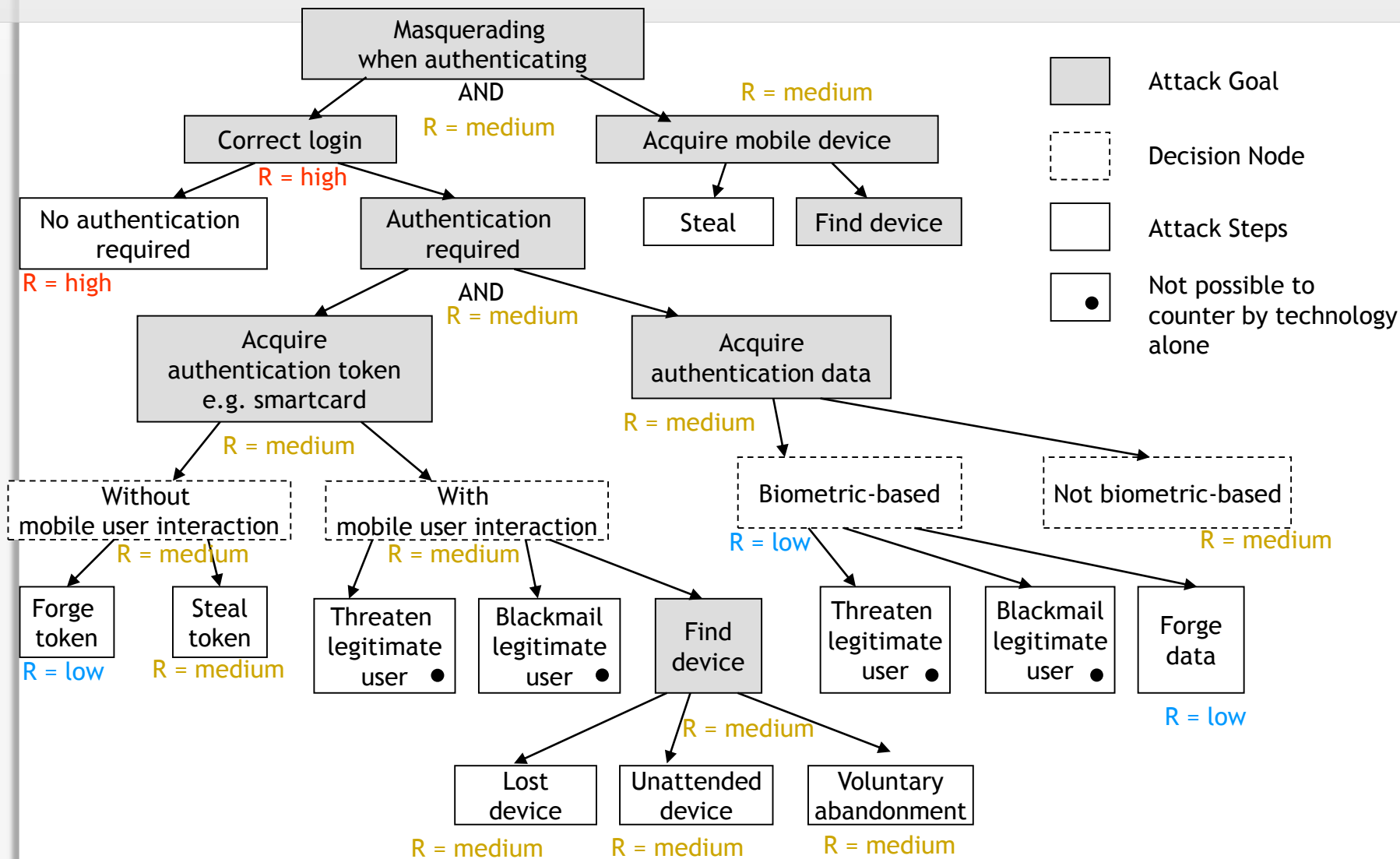
- **Quantitative Approach**

- Attempts to assign real and meaningful numbers to all elements of the risk analysis process
- Each element within the analysis is quantified and entered into equations to determine total and residual risks.
- Purely quantitative risk analysis is not possible, because
 - the method is attempting to quantify qualitative items
 - there are always uncertainties in quantitative values

- **Qualitative Approach**

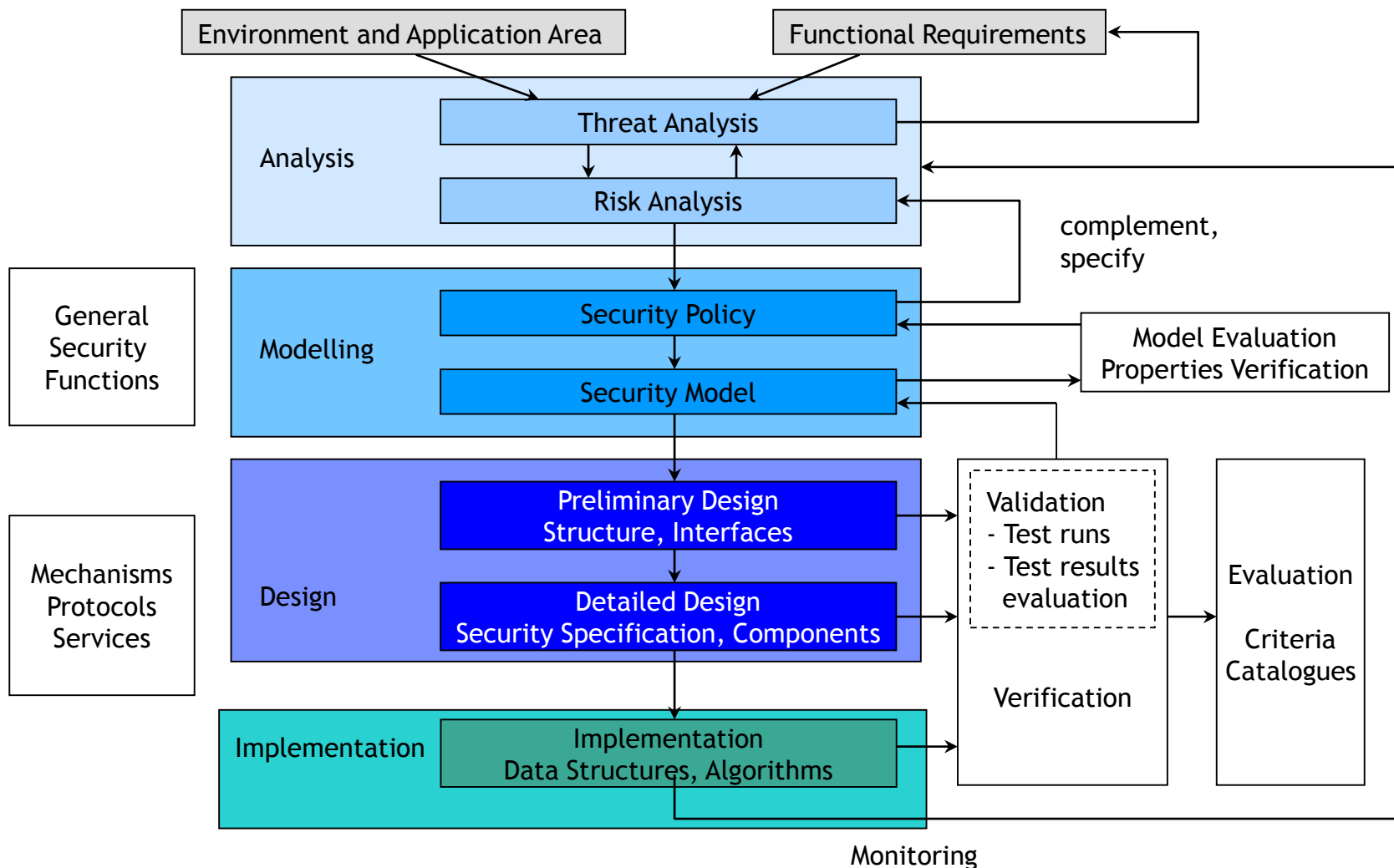
- Walks through different scenarios of risk possibilities
- Ranks the seriousness of the threats and the validity of the different possible countermeasures

Example: Risk Analysis on the basis of an Attack Tree



- Introduction
- Secure System Development Process
- Analysis
- Modelling
 - Security Policies
 - Abstract Security Models
 - Basic Security Functions
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

Secure System Development Process - Modelling



Modelling Security Policies?

- In a formally ideal world a complete workable security policy can be modelled formally ...
- ... but only in a formally ideal world.
- Therefore models model what can be modelled:
 - Abstract Security Requirements
 - Relations between
 - A concrete (but maybe incomplete) set of security policy elements and
 - Basic Security Functions

- Introduction
- Secure System Development Process
- Analysis
- Modelling
 - Security Policies
 - Abstract Security Models
 - Basic Security Functions
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

- Model formal aspects of a security policy
- Goal is to prove
 - Consistency of a system
 - Completeness of a system
- Typical examples are Access Control Models

- Introduction
- Secure System Development Process
- Analysis
- Modelling
 - Security Policies
 - Abstract Security Models
 - Basic Security Functions
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

- Identification and Authentication
- Administration of Rights
- Verification of Rights
- Conservation of Evidence
- Availability of the services

Identification and Authentication

- Both subjects (the entities who access) and objects (the entities which are accessed) have to be identified clearly.
- Subjects needs to prove their identity.
- Clarify if the subject has to authenticate for each action or only once (until session is closed), e.g.
 - Logon only once to an operating system
 - Authentication against a web-server only once
 - Authentication to an Anti-Theft-Device every time the engine is started
- Procedure in case of failure of identification or authentication:
 - Log files including ID, IP, time, date, ...
 - Disable the subject's account
 - Reset the password

- Defining access rights
- Access rights required for each object: to be defined in the security model
- Determination of who might change these rights:
 - Only the administrator
 - Users with super user rights
- Defining availability of rights
 - Always available
 - Only available for certain tasks

- Frequency of verification
 - Once per session
 - Every time an object is accessed
 - Often due to the costs: once per session
- Exceptions
 - System components found to be fully trustable:
 - Once per session
 - Never
 - Example: Kernel tasks

The following facts of an attack have to be recorded:

- Every information concerning the attacker
 - User name
 - From outside/inside
 - IP address
 - ...
- Objects and operations which have been affected:
 - Unique ID
 - IP address
 - User account
 - ...
- Time and date of the attack
- Possibilities which could allow the subject to change the recorded data
- Events leading to the recording of the attack
 - Wrong password
 - Non existing ID

or how to avoid denial-of-service attacks

Two properties have to be defined:

- Warranty
 - For every function of every component
 - The priority of the components
- Boundary conditions to be able to miss a component

Example:

- The function to verify an authentication should always be available.

Security Policy Elements

Basic Security Functions

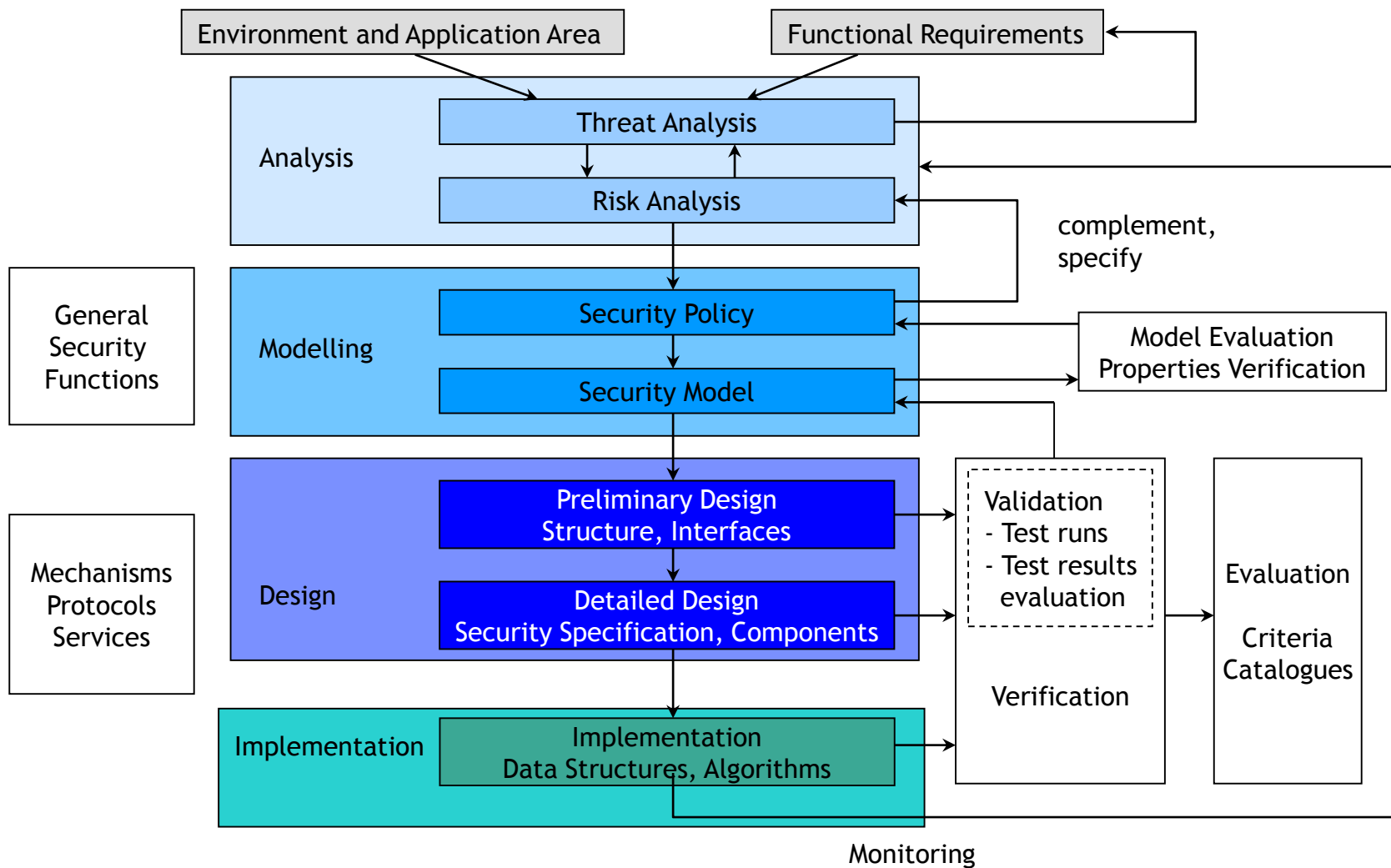
- A security token is required for user authentication.
- All incorrect authentication cases are recorded and notified.

Authentication

Conservation of Evidence

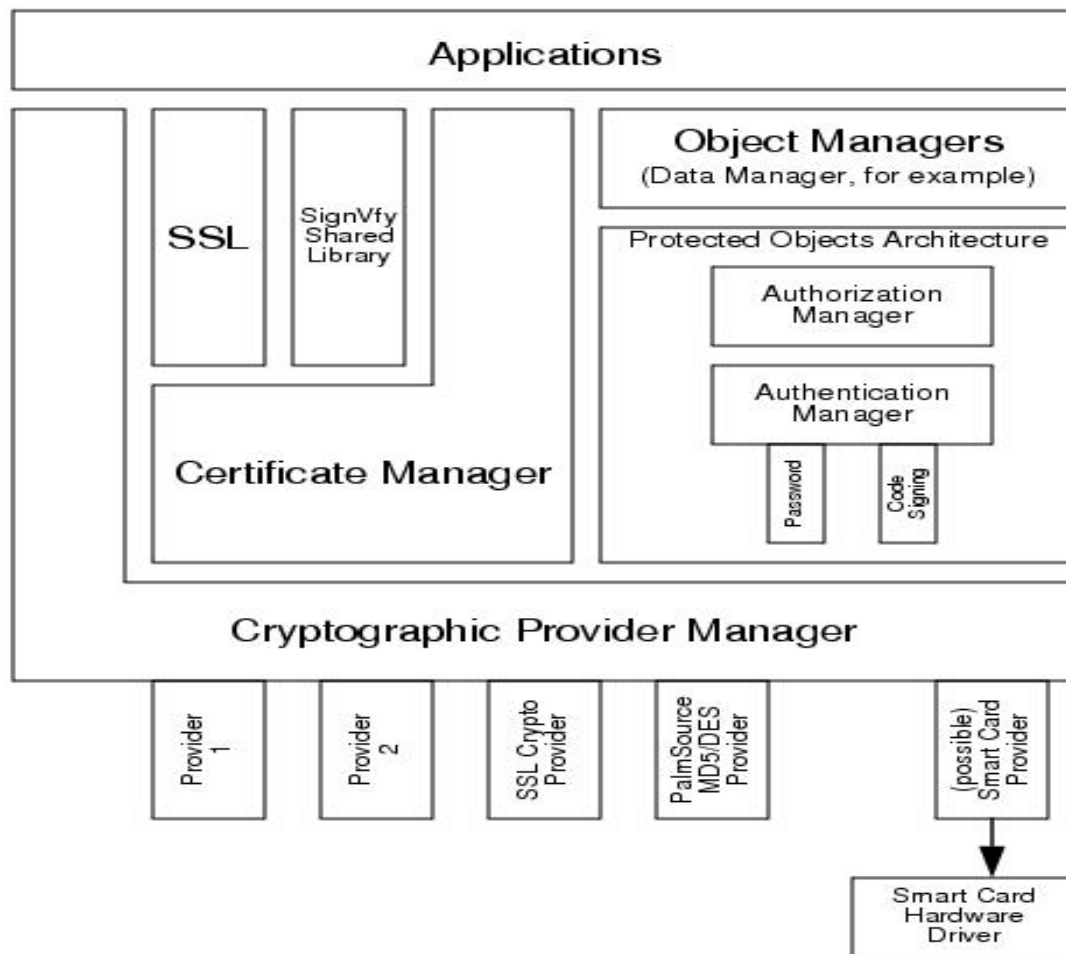
- Introduction
- Secure System Development Process
- Analysis
- Modelling
- Design
 - Security Architecture
 - General Design Principles
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

Secure System Development Process - Design



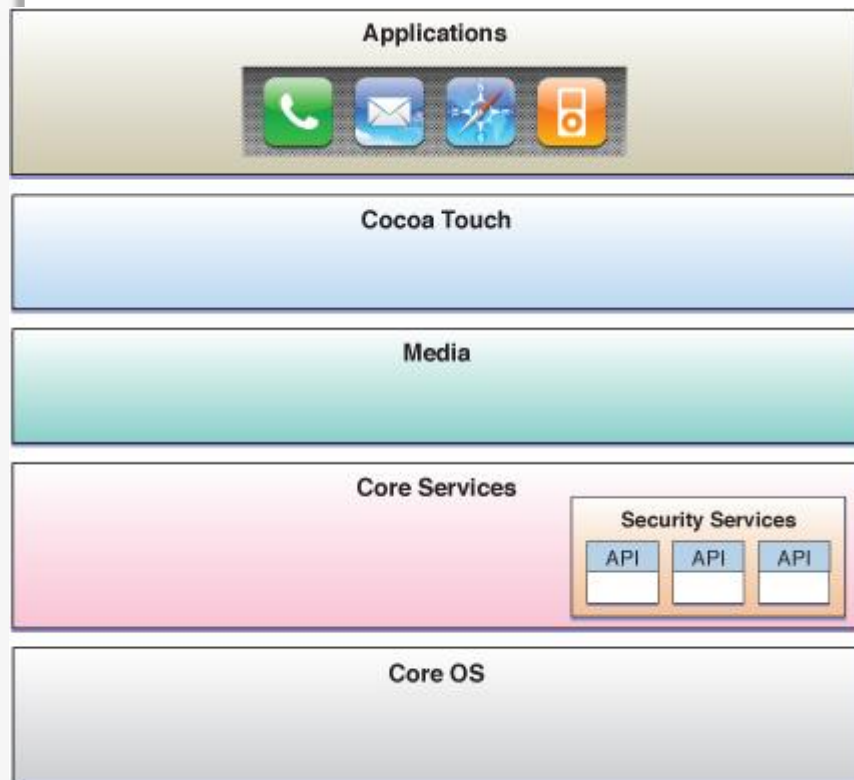
- Description of the architecture in a **coarse grained way**
 - Components which are to be protected
 - Components which are protecting
- Description of the architecture in a **fine grained way** – a closer frame for the implementation
 - Required coding tools
 - Used algorithms
 - Employed data structures

Example: Security Architecture of Mobile OS - 1

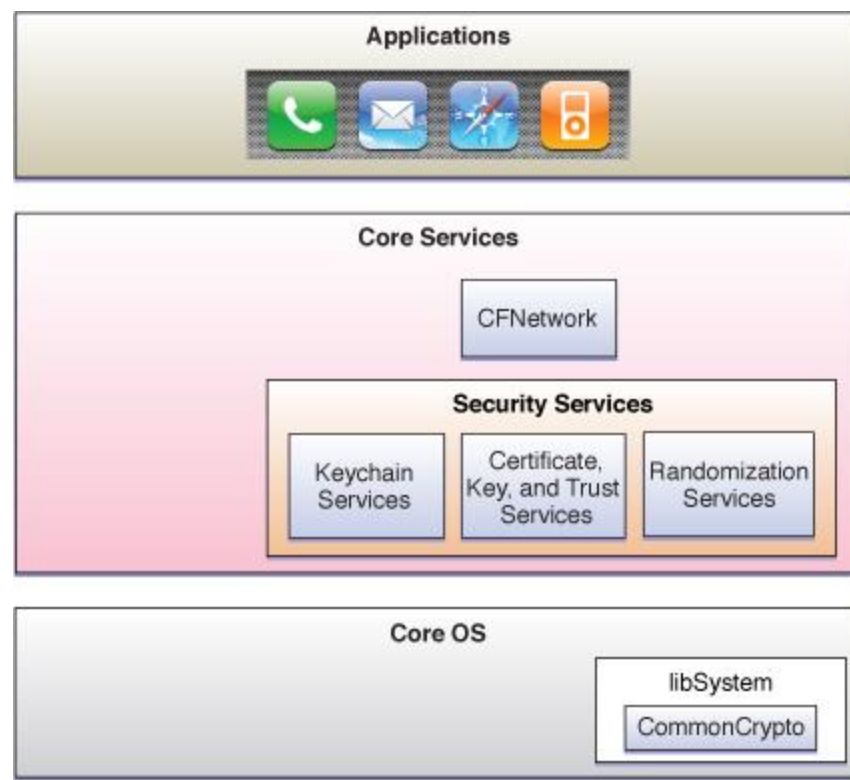


Security Components of Palm OS Cobalt [Palm OS]

Example: Security Architecture of Mobile OS - 2



iOS security architecture overview



iOS security APIs

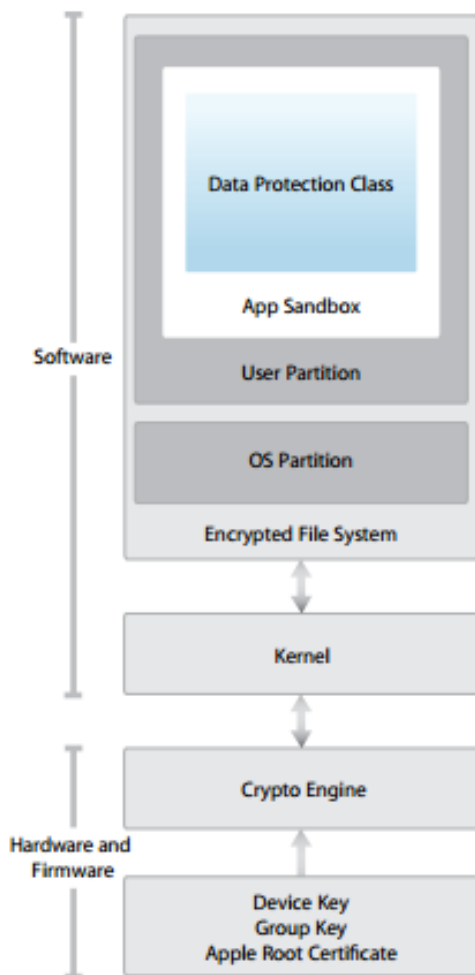
iOS Security Architecture visualized in 2010
[developer.apple.com 2010; disanji.net 2012]

Example: Security Architecture of Mobile OS - 3



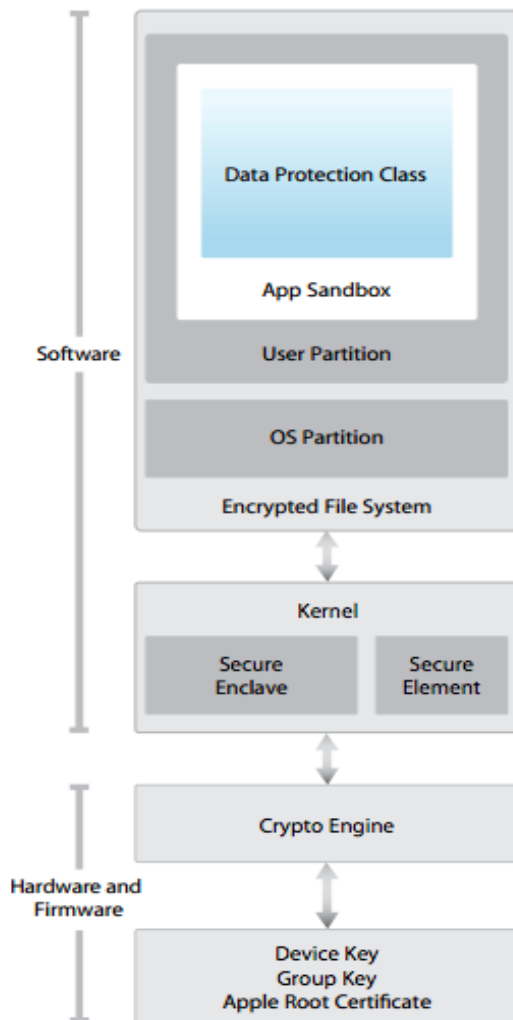
iOS security architecture overview with security services APIs

Example: Security Architecture of Mobile OS - 4



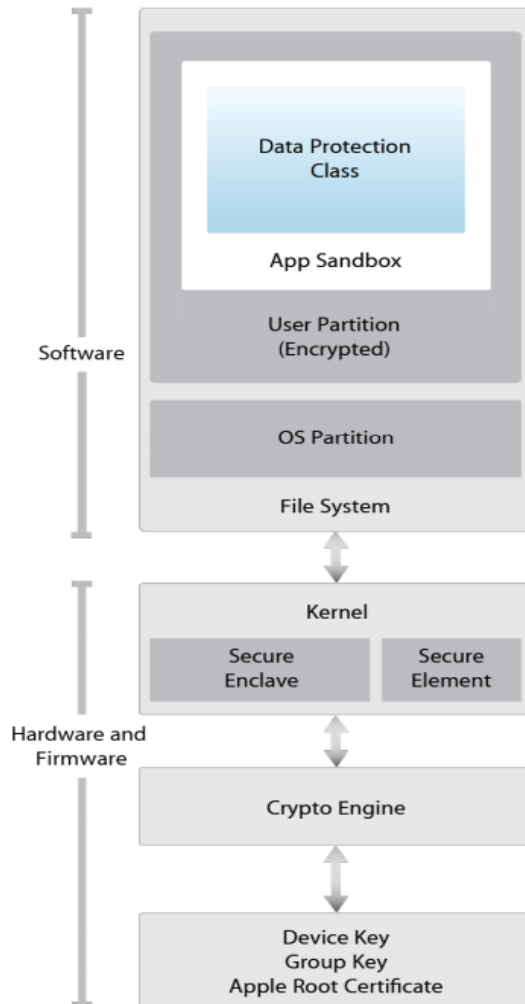
Security Architecture diagram of iOS
2012 [apple.com 2012]

Example: Security Architecture of Mobile OS - 5



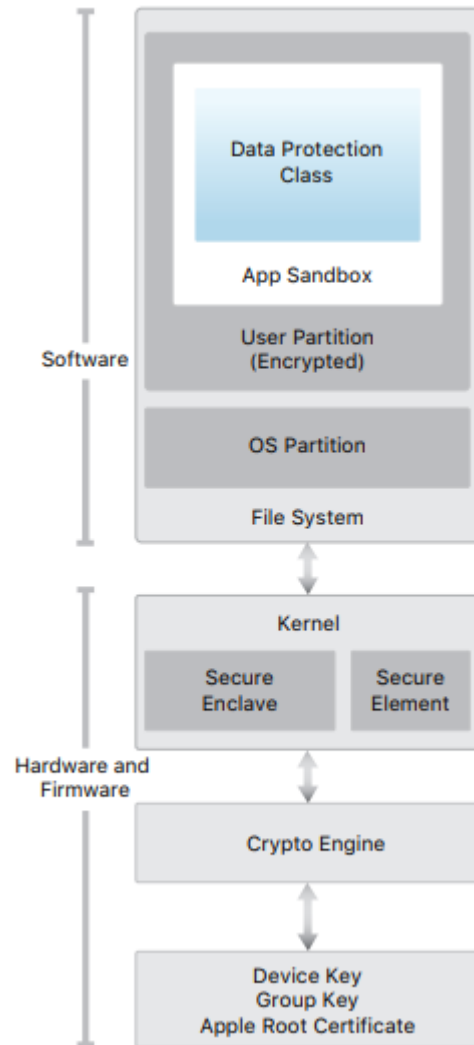
Security Architecture diagram of iOS 2014 [apple.com 2014]

Example: Security Architecture of Mobile OS - 6



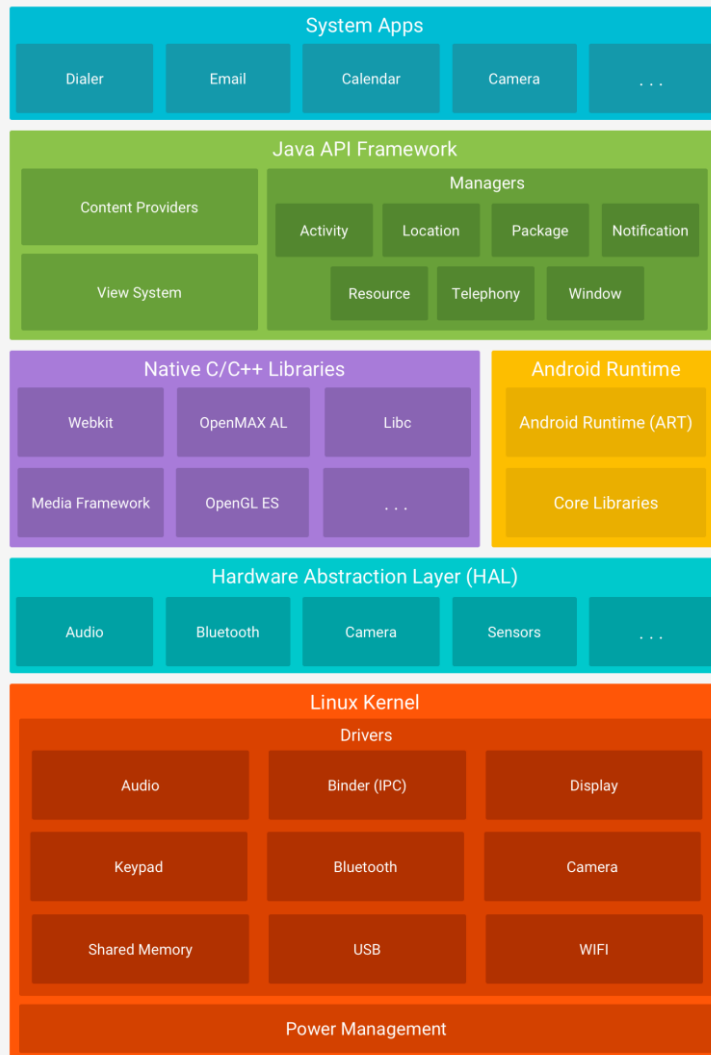
Security Architecture diagram of iOS 2016 [apple.com 2016]

Example: Security Architecture of Mobile OS - 6



Security Architecture diagram of iOS 2018 [apple.com Nov 2018]

Example: Security Architecture of Mobile OS - 7



Security components of various levels of the Android software stack
[developer.android.com]

- Introduction
- Secure System Development Process
- Analysis
- Modelling
- Design
 - Security Architecture
 - General Design Principles
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

General Design Principles (1)

- **Economy of Mechanism:** The protection mechanism should have a simple design without overhead.
- **Fail-safe Defaults:** The protection mechanism should deny access by default, and grant access only when explicit permission exists.
- **Complete Mediation:** The protection mechanism should check every access to every object.
- **Open Design:**
 - The strength of protection mechanisms should not depend on attackers being ignorant of their design.
 - It may however be based on the attacker's ignorance of specific information such as passwords or cipher keys.

General Design Principles (2)

- **Separation of Privilege:** The protection mechanism should decide on access based on more than one piece of information.
- **Least Privilege:** The protection mechanism should force every process to operate with the minimum privileges needed to perform its task.
- **Least Common Mechanism:** The protection mechanism should be shared as little as possible among users.
- **Psychological Acceptability:** The protection mechanism should be easy to use (at least as easy as not using it).

- Introduction
- Secure System Development Process
- Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

Validation

- Internal classification by documentation of test ...
 - Targets
 - Plans
 - Methods

Evaluation

- Often done by a 3rd party
 - Based on a(n) (inter)national criteria catalogue
 - the Common Criteria (IS 15408)
 - the European ITSEC
 - the German IT security criteria
- Assures a certain level of security

- Introduction
- Secure System Development Process
- Analysis
- Modelling
- Design
- Validation and Evaluation
- Security Monitoring
- Security Engineering with UML

- Done during operation
- Allows fast reaction on new incidents, especially if they are not covered by the security system
- Possibly use of tools, e.g. Intrusion Detection Systems

- Introduction
- Secure System Development
- Analysis
- Modelling
- Design
- System Evaluation and Validation
- Security Monitoring
- Security Engineering with UML

UML is an opportunity for secure systems development that is feasible in an industrial context:

- As UML is the de-facto standard in industrial modelling, a large number of developers is trained in UML.
- Compared to previous notations with a user community of comparable size, UML is relatively precisely defined.
- A number of analysis, testing, simulation, transformation and other tools are developed to assist the every-day work using UML.

UML in Security Engineering (2)

- Use Case Diagrams describe typical interactions between a user and a computer system (or between different components of a computer system).
- Activity Diagrams can be used e.g. to model workflows and to explain use cases in more detail.
- Class Diagrams define a static structure of the system.
- Sequence Diagrams describe interaction between objects via message exchange.
- Deployment Diagrams describe the underlying physical layer.

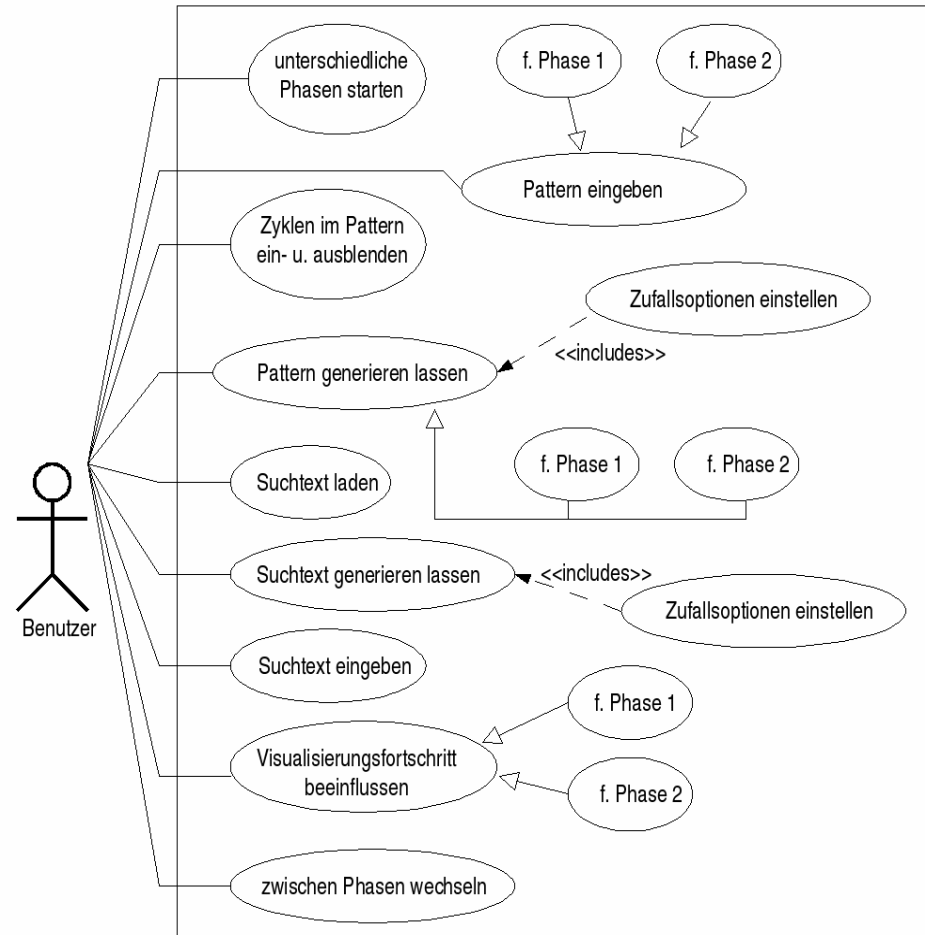
Security Requirements Capture

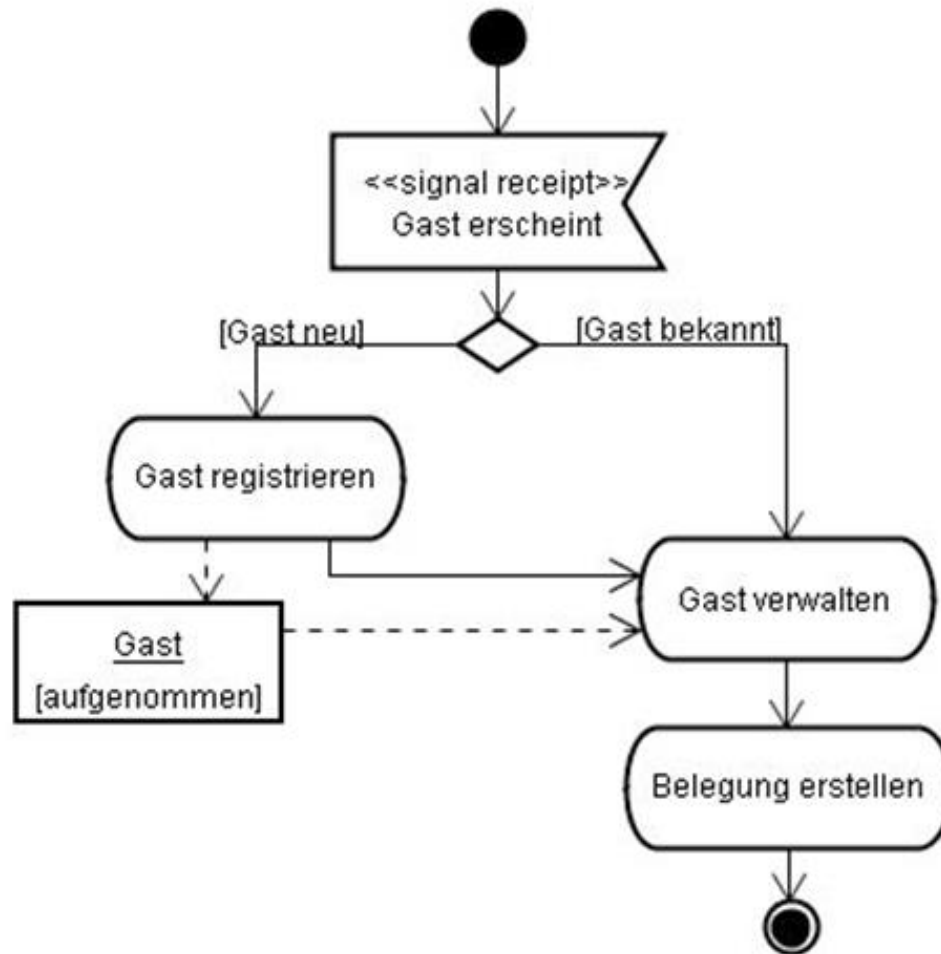
Secure Business Processes

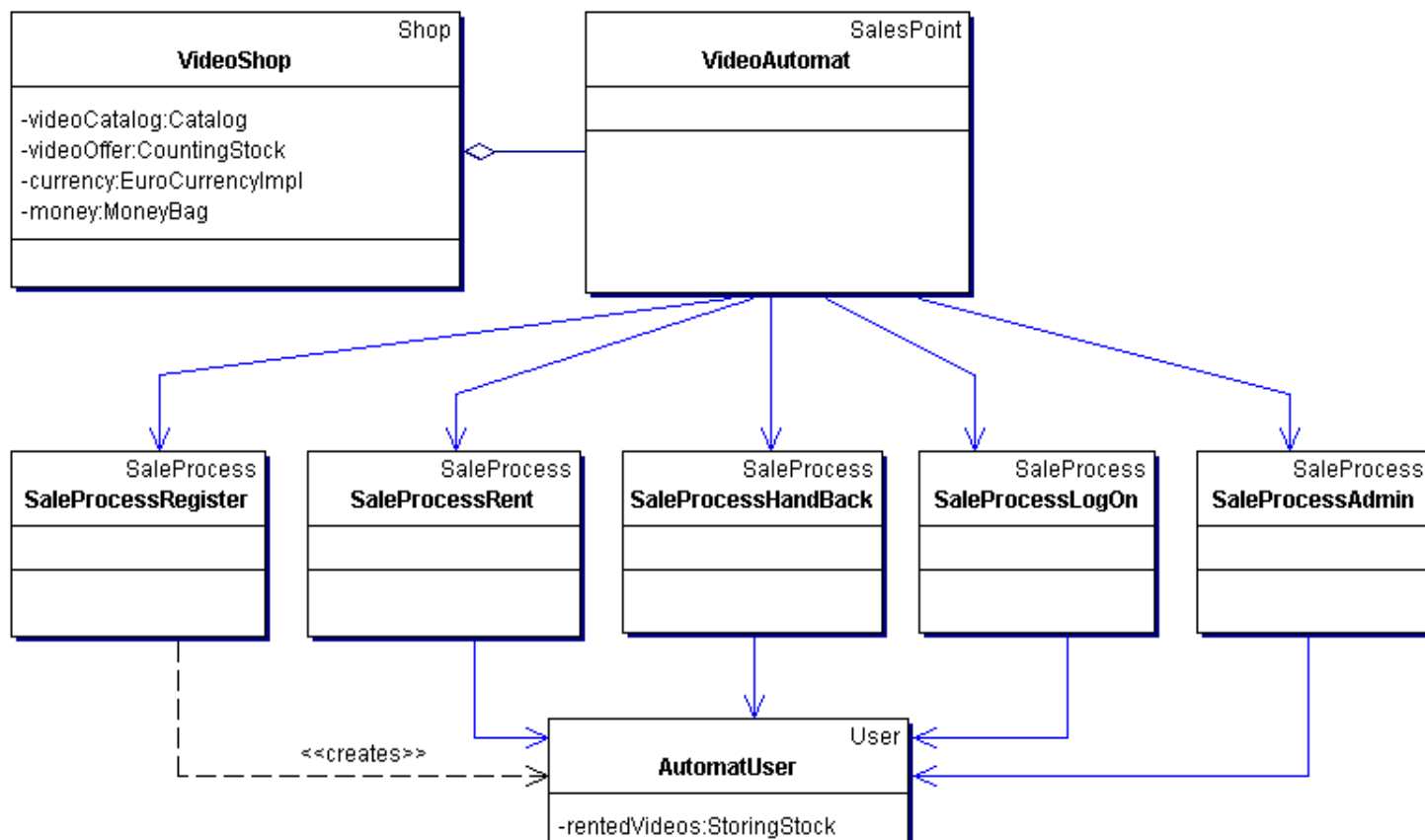
Structural Data Security

Security-Critical Interaction

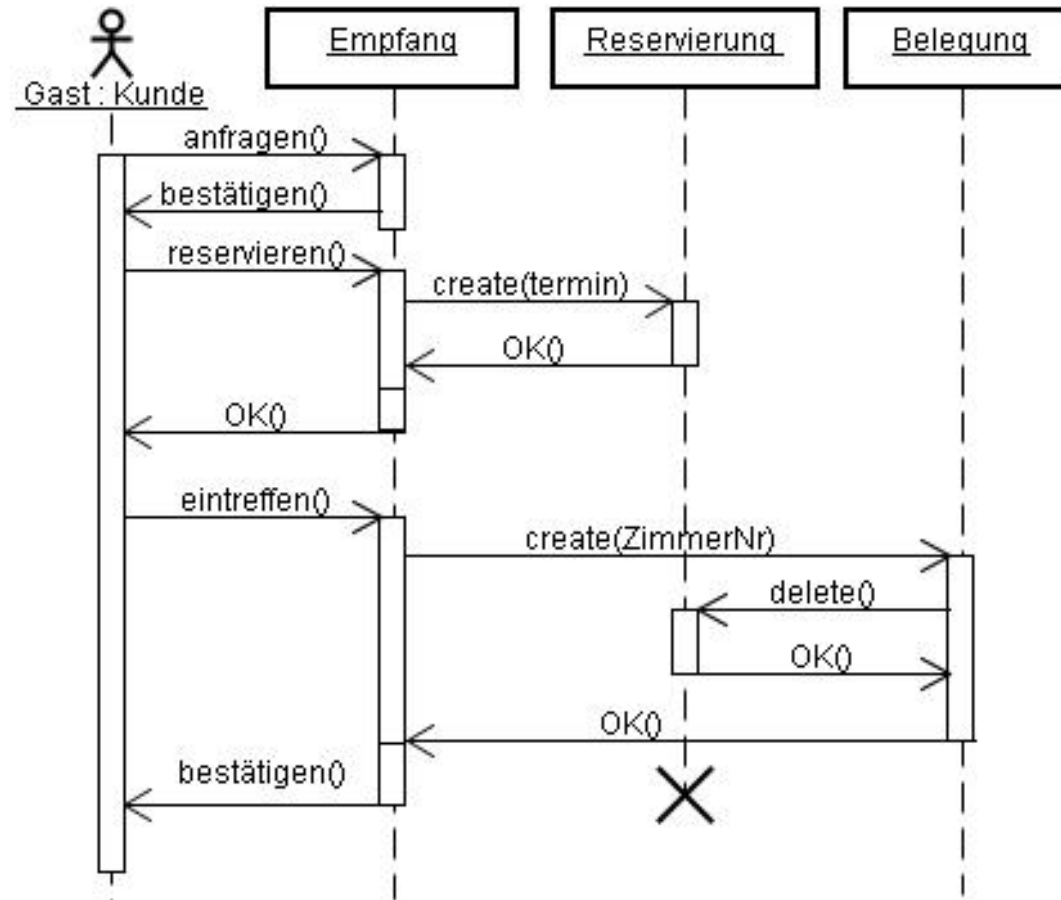
Physical Security



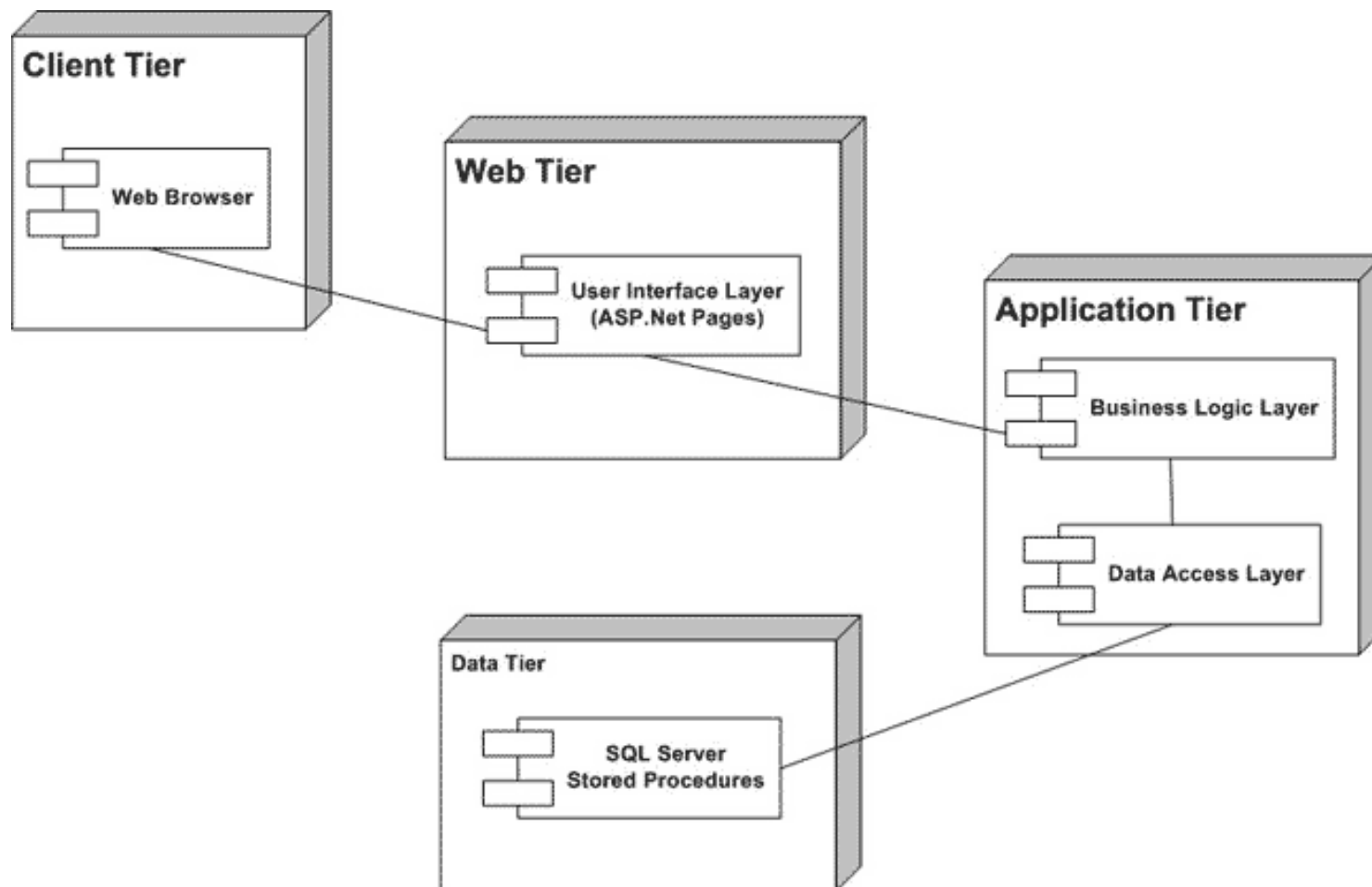




Sequence diagram

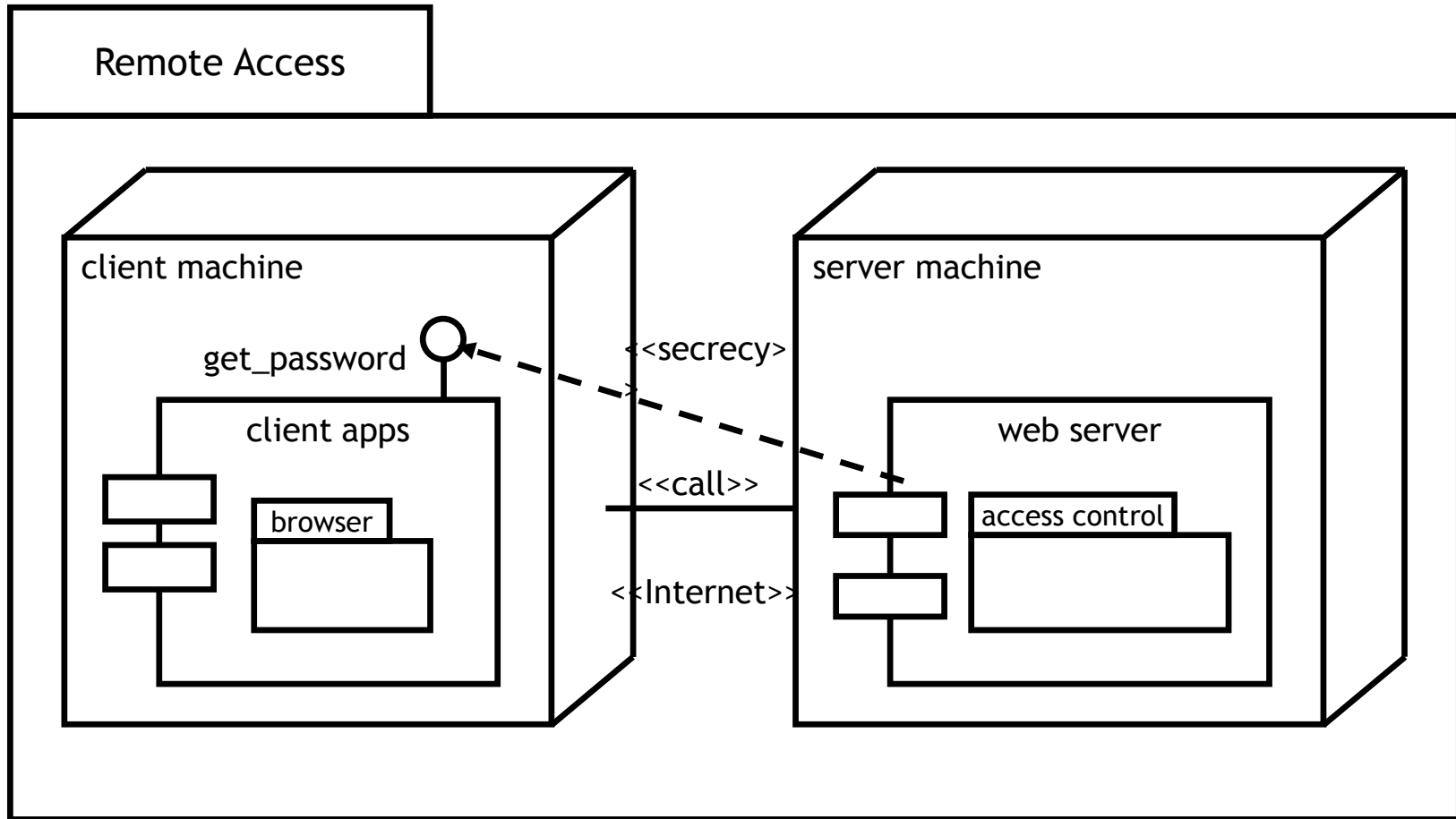


Deployment diagram



- Extension for secure systems development:
 - Evaluate UML specifications for weaknesses in design
 - Encapsulate established rules of prudent secure engineering as checklist
 - Make security considerations available to developers not specialized in secure systems
 - Consider security requirements from early design phases in system context
 - Make certification cost-effective

Example: Deployment Diagram



- **Anderson, Ross.** *Security Engineering: A Guide to Building Dependable Distributed Systems*: Wiley Computer Publishing, 2001. Ch.1,22
- **Bishop, Matt.** *Introduction to Computer Security*. Boston: Addison Wesley, 2005. Ch.12
- **Eckert, Claudia.** *IT-Sicherheit: Konzepte, Verfahren, Protokolle*, 4. überarbeitete Auflage, R. Oldenbourg Verlag, 2006, ISBN 3-486-57851-0
- **Jürjens, Jan.** *Developing Secure Systems with UMLsec - From Business Processes to Implementation*: Vieweg-Verlag, VIS 2001, Kiel (Germany), 12-14 Sept 2001
<http://ls14-www.cs.tu-dortmund.de/main2/jj/umlsec/index.html>
- **Palm OS.** *Security and Cryptography, Exploring Palm OS*
www.palmos.com/dev/support/docs/protein_books/Security_and_Cryptography/SecurityAndCryptographyTOC.html