Security Engineering for Software

CS996 – CISM

Jia An Chen

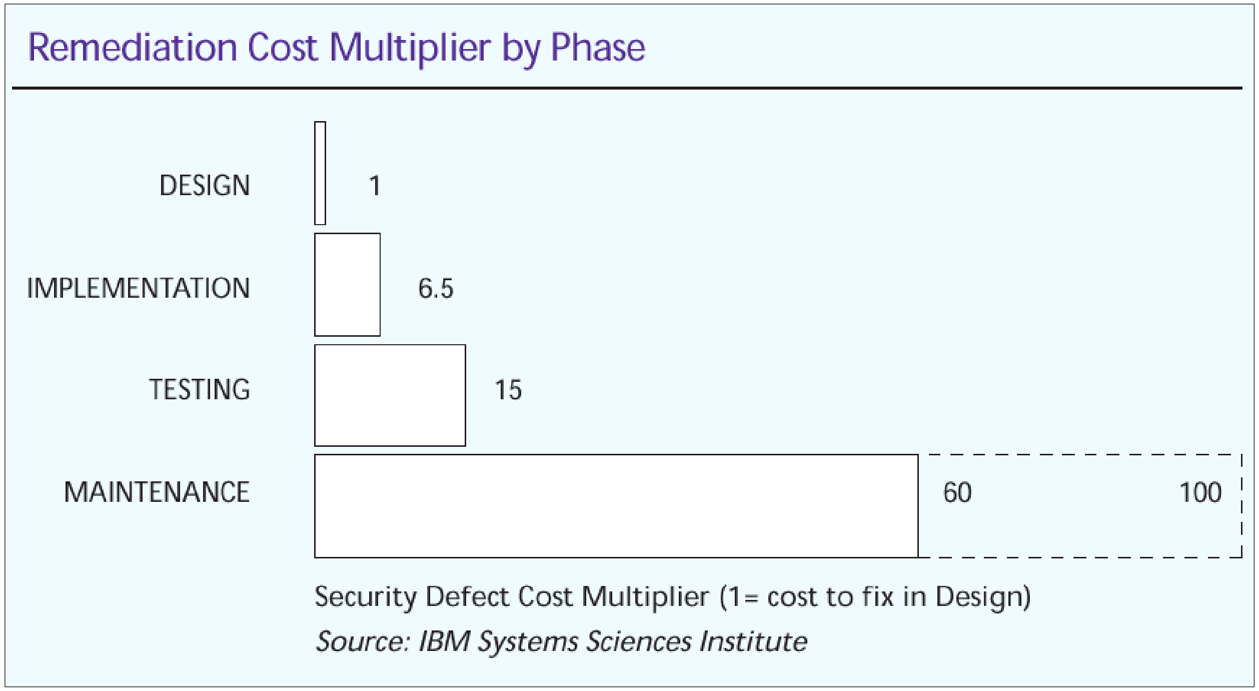
03/31/04

Current State of Software Security

- Fundamental lack of planning for security
- Most security issues come to light only after completion of the development. As a result, security is often managed in an ad-hoc fashion, as an afterthought
- Examples:
  - Almost every company buys some kind of security infrastructure, such as firewalls, VPNs, IDS
  - Many companies perform penetration tests and automated security scans

Security should be done EARLY

- @stake: “Findings indicate that significant cost savings and other advantages are achieved when security analysis and secure engineering practices are introduced early in the development cycle.”

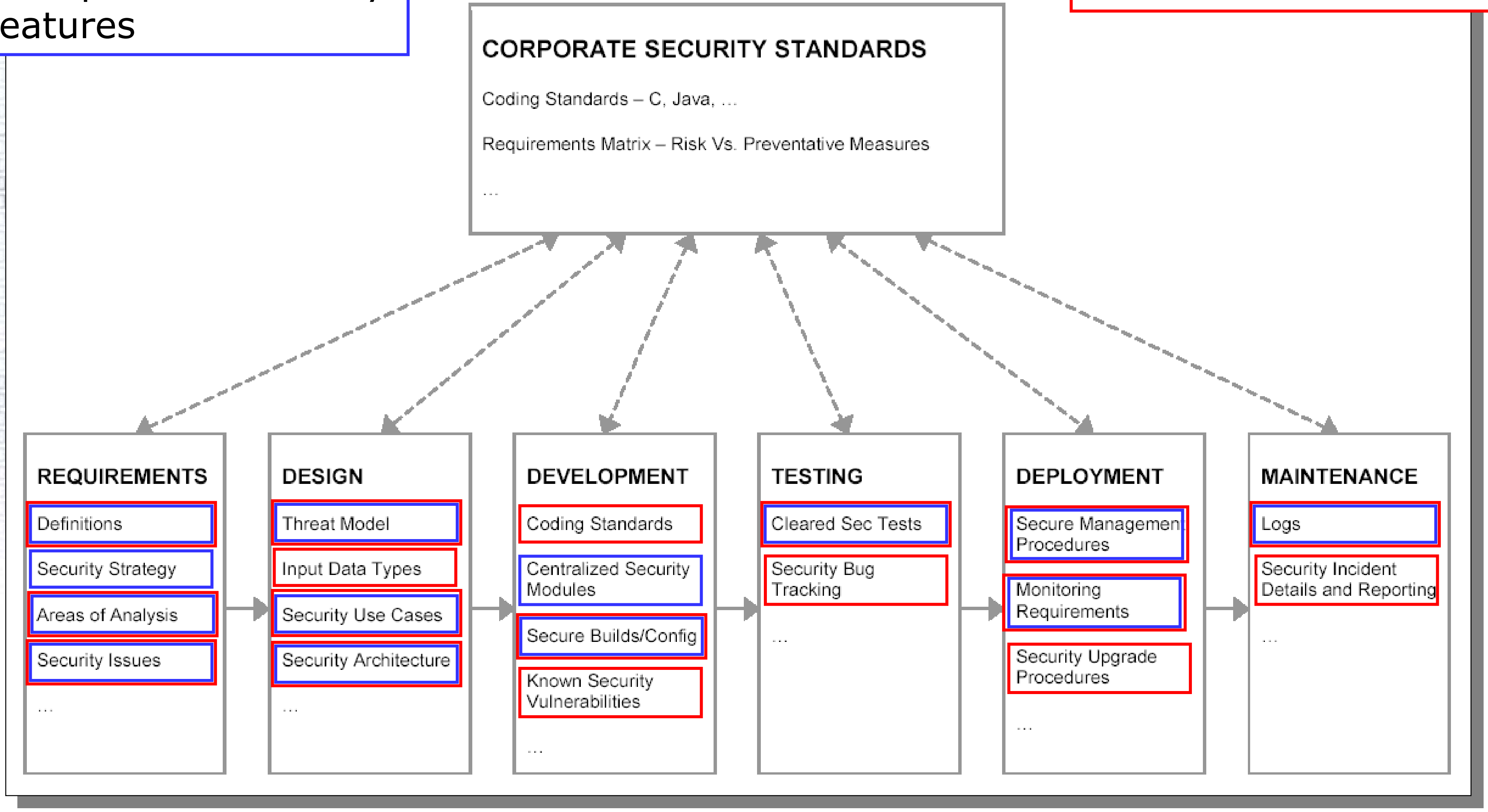


Integrating security into software lifecycle

Security Features Vs. Robustness

Engineering software to implement security features

Engineering software to be "safe"



Requirement Phase

"Security is like adding brakes to cars. The purpose of brakes is not to stop you: it's to enable you to go fast!"
--- Gene Spafford

- A security requirement is complementary to the functional requirement of a system.
- It is a manifestation of a high-level organizational policy into the detailed requirements of a specific system.
- Security requirements should be based on an analysis of the assets and services to be protected and the security threats from which these assets and services should be protected.

Quality Properties of Requirements

- **Design Independent** - A software requirement is free of design and implementation decisions except in the form of a constraint
- **Unambiguous** - A requirement is unambiguous if it has only one possible interpretation
- **Precise** - A requirement is precise if it exactly defines a behavioral aspect of the software including data sets and ranges for outputs and inputs
- **Understandable** - A requirement is understandable if it accurately conveys the requirement to its intended audience.
- **Traceable** - Any document that references the requirement must be identifiable
- **Verifiable** - A requirement is verifiable if there is a quantifiable or observable effect of the software that is directly expressed by the requirement.

Quality Properties of Requirements

- **Prioritized** - Unless all requirements are of equal importance a requirement should be ranked by priority.

- **Complete** – Include all the information needed to produce a design description of the software and no more.
  - everything that the software is supposed to do is represented in the document,
  - software responses to all possible classes of input in all possible situations is described,
  - all references are made, pages numbered and terms are defined.

- **Consistent** - There are no contradictory statements between the requirement statements contained within.

- **Organized** - Requirements are arranged in such a way that readers can easily locate information

- **Modifiable** - Requirements are organized in such a way that it is easy to change.

Requirement Engineering Process

- Requirements Elicitation
- Requirements Analysis
- Modeling the requirements and writing the specification
- Verification of the requirements

Requirements Elicitation

- Collect data about the overall system and its operational characteristics
- Negotiate with the customer on feasibility issues of attributes or other requirements
- Through surveys of, and interview with users of the software

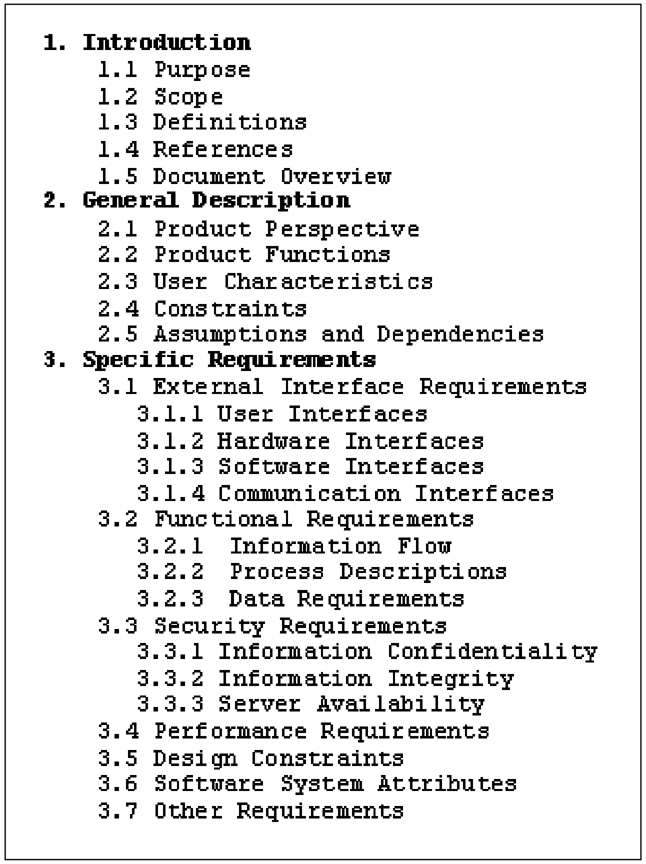
Requirements Analysis

- The process of evaluating existing conditions that represent the primary motivation of the problem domain as evidenced by data gathered during the requirements elicitation phase

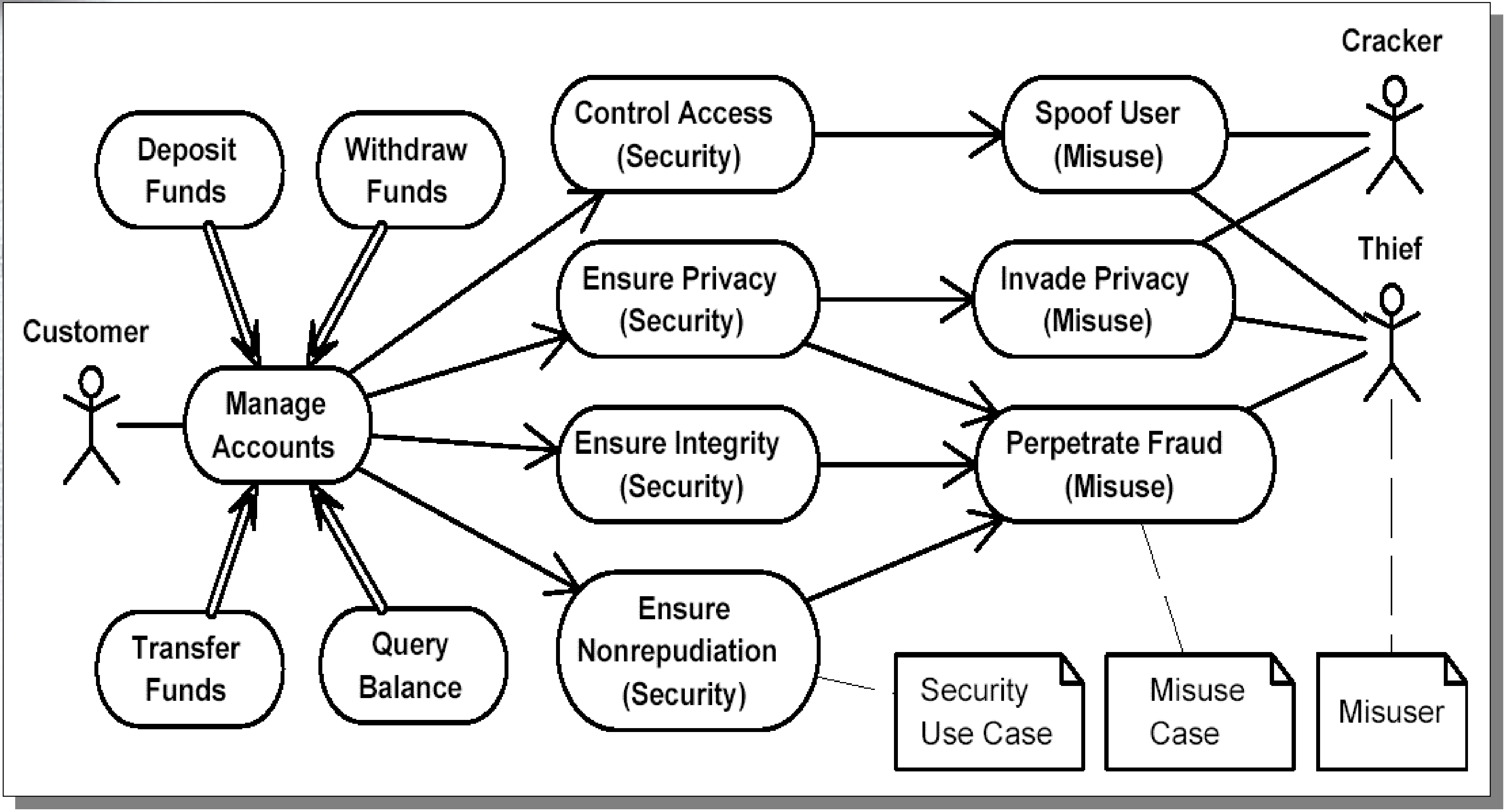
- Derive security requirements
  - Confidentiality
  - Integrity
  - Availability

Requirements Modeling and Specification

- The process of presenting and formalizing the requirements in a document format
- The goal is to communicate user needs to the system developers
- Security Requirements
  – Role requirements
  – Security Use Cases and Misuse Cases



Security Use Cases and Misuse Cases

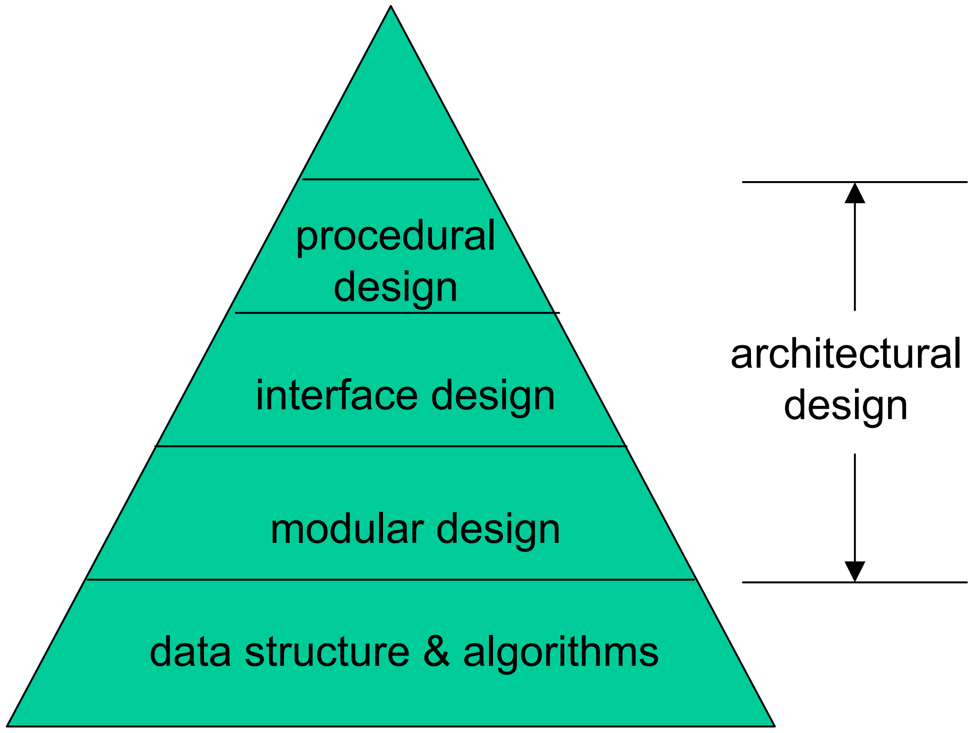


Requirements Verification

- The process of ensuring that the allocation of the overall system requirements is appropriate and correct
- Examination of the project documentation
- Evaluation of product testability
- Interface analysis

Design Phase

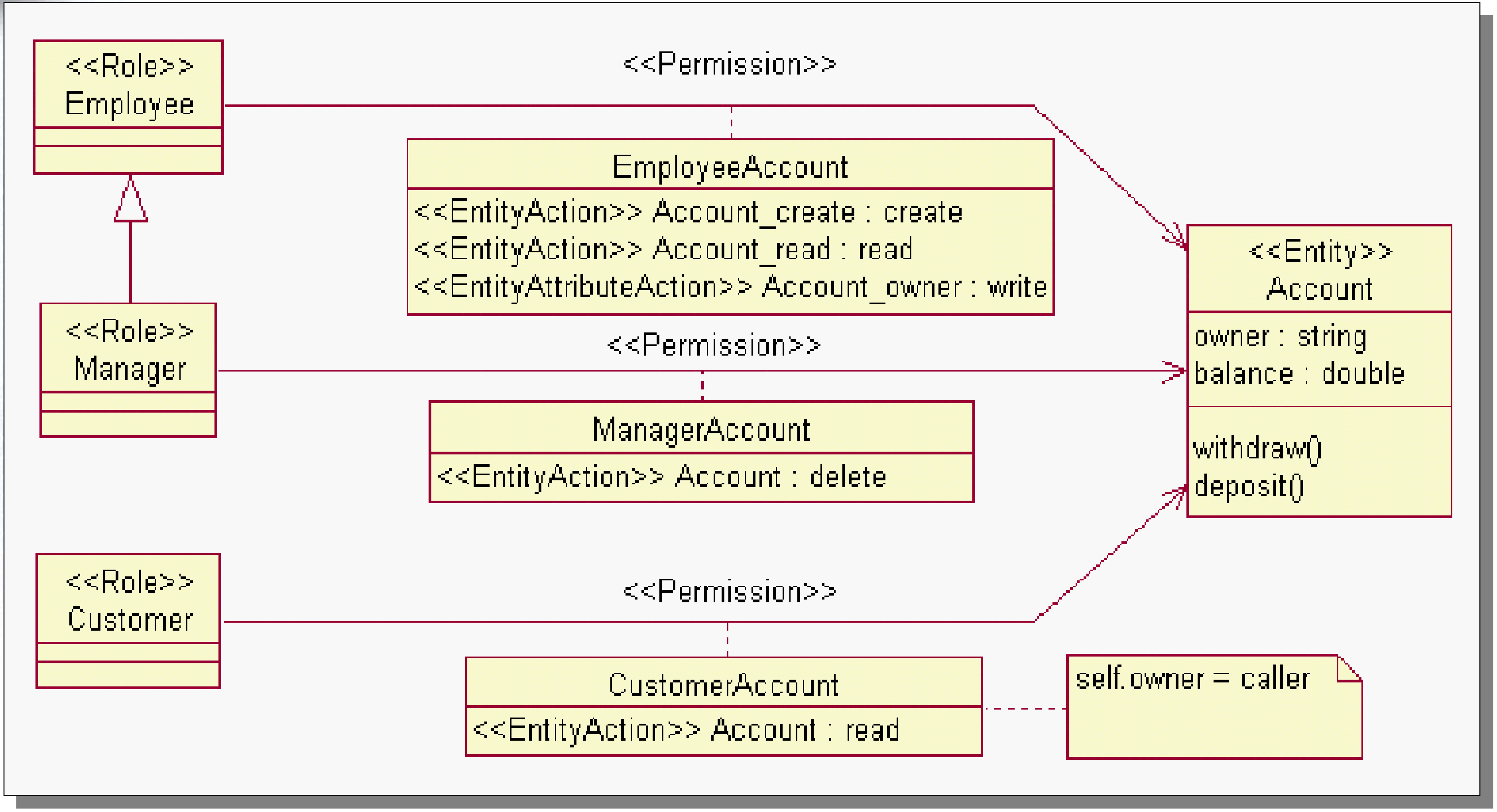
- A period of time during which the designs are made for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements.
- General Design Concepts
  - Data Abstraction
  - Procedural Abstraction
  - Stepwise Refinement
  - Modular Design
    - Cohesion
    - Coupling
  - Information Hiding
- Design testing
- Documentations



Model Driven Security – SecureUML

- Strives to integrate security requirements with the design of the system
- Extends UML to include the explicit modeling of security dimensions like authentication, access control, etc.
- Use the same principles of software design to solve security-related problems in making software systems more secure

Sample SecureUML



Design with security

- Defense in depth
  - Manage risk with multiple defensive strategies, so that if one layer of defense turns out to be inadequate, another layer of defense will, ideally, prevent a full breach
- Compartmentalization
  - Break the system up into as many isolated units as possible, in order to minimize the amount of damage that can be done to a system when an unit is compromised
- Least privilege
  - Only the minimum access necessary to perform an operation should be granted, and that access should be granted only for the minimum amount of time necessary

Implementation Phase

- Process of translating the detailed design into code
- Know what you do
  - Read documentation
  - Thorough testing
  - Understand underlying mechanisms
- Common security bugs
  - Buffer overflow
  - Format string attack
  - Race condition
  - False assumption

Implementation Phase

- Best programming practice
  - Be paranoia: don’t trust anything
  - Robustness: assume the user is an idiot, so protect everything… because the evil guy is not a common user. (check all user inputs)
  - Make no assumption: don’t suppose something would never happen, never say ‘never’

Software Security

- Configuration Management
  - Defined as the art of identifying, organizing, and controlling modifications to software
  - Programming standards and controls
  - Documentation
  - Change controls
  - Tools: Concurrent Versions System (CVS)
  - Example: Bugs sometimes creep in when software is modified, and you might not detect the bug until a long time after you make the modification.

Software Security

- Software security mechanisms to protect information
  - Internal labeling
  - Application security features
  - Audit trails and logging
  - Need-to-know controls

- Malicious logic protection

Maintenance Phase

- Maintenability has to be built into a product from the very beginning and must not be compromised at any time during the development process
- Three types of maintenance
  - Corrective
  - Perfective
  - Adaptive
- Management of Maintenance
  - Fault reports
  - Authorizing changes to the product
  - Ensuring maintenability

Software Testing

- Step 1 – Planning
  - Understanding the application and its requirements
  - Develop testing plan
  - Develop test cases
- Step 2 – Execution
  - Execute test cases
  - Performing identification and investigation
  - Performing fix validation on “resolved” issues
- Step 3 – Reporting
  - Document issues found

Software Testing

- Black–box testing
  - Test of the externally observable behavior of the system
  - No knowledge of internal structure needed
  - Tools: PROTOS
- White–box testing
  - Test the internal interaction between components of a system
  - Requires detailed knowledge of structure
  - Tools: RATS, FlawFinder, ITS4

Regression Testing

- Retesting of a software system that has been modified to ensure that any bugs have been fixed and that no other previously working functions have failed as a result of the bug fixes and that newly added features have not created problems with previous versions of the software
- Regression Test Framework
  - A repository which maintains the test cases and allows the test cases to be run

Regression Testing

- Comprehensive Regression Testing
  - A general regression test suite that provides for a comprehensive retest of most or all of the application
- Selective Regression Testing
  - Selectively retesting the system based on the modules that were changed or the modules to which an interface was changed

Selective Regression Testing

- The selection of test cases for regression testing
  - Requires knowledge on the bug fixes and how it affect the system
  - Includes the area of frequent defects
  - Includes the area which has undergone many/recent code changes
  - Includes the area which is highly visible to the users
  - Includes the core features of the product which are mandatory requirements of the customer

Mobile Code

- Technology which allows for the creation of executable information which can be delivered to an information system and then directly executed on any hardware/software architecture which has an appropriate host execution environment
- Malicious Mobile Code
- Trusted Source
  - Trusted networks, e.g. SIPRNET & JWICS
  - a digital signature over the mobile code itself using either DoD or IC-approved PKI certificate
  - a commercial certificate approved by either the DoD CIO or the IC CIO; or
  - authentication of the source of the transfer by public key certificate (e.g., S/MIME, SSL web server)

Mobile Code Technologies

- Category 1
  - Mobile code that can exhibit broad functionality using unmediated access to services and resources of workstations, hosts and remote systems (Active X, Visual Basic for Applications (VBA), Windows Scripting Host, when used as mobile code…)
- Category 2
  - Mobile code that has full functionality using mediated or controlled access to services and resources of workstations, hosts and remote systems (Java Applets and other Java Mobile Code, LotusScript, Postscript)

Mobile Code Technologies

- Category 3
  - Mobile code that has limited functionality, with no capability for unmediated or uncontrolled access to services and resources of workstations, hosts and remote systems (JavaScript, VBScript, PDF files, Shockwave/Flash…)
- Others that are not considered true mobile code
  - XML
  - QuickTime
  - Web server scripts that execute on a server (Java servlets, Java Server Pages, CGI, Active Server Pages, CFML, PHP, SSI, server-side JavaScript, server-side Lotus Script)
  - ……

Review

- Current State of Software Security
- Requirement Phase
- Design with security
- Software Testing
- Mobile Code

References

- @stake Strategic Security Resources (http://www.atstake.com/research/strategic_security/)
- Donald G. Firesmith, "*Security Use Cases*", JOURNAL OF OBJECT TECHNOLOGY, Vol. 2, No. 3, May-June 2003
- J. Chris Gibson, "*Quality Software Requirements*"
- appLab Technologies, (http://www.applabs.com/funtionality.htm)
- J. Chris Gibson, Quality Software Requirements, "*Regression Testing Framework and Strategy* "
- DOD – Malicious Code Prevention
- http://web.umr.edu/~umreec/web-courses/cs306/lecture-notes/Chapter13.pdf
- http://www-106.ibm.com/developerworks/security/library/s-fail.html
- http://www-106.ibm.com/developerworks/security/library/s-priv.html
- http://www.cvshome.org/