Software Architecture

Lecture 2

Software Requirements

Néstor Cataño
Innopolis University

Spring 2016

Goals for this Lecture

1. Understand the differences between **functional** and **non-functional** software requirements
2. Understand how requirements may be organized in a **software requirements document**
3. Understand the concepts of **user** and **system requirements** and why these requirements should be written in different ways

Requirements

The **requirements** for a system are the **descriptions** of what the system should do, the services that it provides and the constraints on its operation.

These **requirements** reflect the **needs of customers** for a system that serves a certain purpose such as controlling a device, placing an order, or finding information.

Requirements

- **User Requirements** are statements in natural language plus diagrams of the functionality of the system and the constraints under which the system must operate
- **System requirements** are detailed user requirements

Requirements

- Functional Requirements →

  What the System Should do?

  - behavior of a system

- Non-Functional Requirements →

  How the System Should be?

  - quality attributes, quality goals, constraints

Functional requirements

- Services the system **should provide**
- How the system should **react / behave**
- What the system **should / should not do**

Acceptance Criteria

- **Acceptance criteria** are the conditions that the implementation of a software product must satisfy to be accepted by the user
- **Acceptance criteria** are used to **confirm** when a **functional requirement** is completed and working as intended

Example / Banking System

LLoyds Bank Group has opened a programming contest for Innopolis (**INO**) students to design and implement an application for the management of bank accounts for professors and students at **INO**. **LLoyds** offers two types of bank accounts, **saving** and **checking** accounts. Both types of accounts allow users to **deposit** and **withdraw** money.

User Roles

Role	Description
Client	On-line banking account with general privileges on her personal bank account – this role does not have privileges on other user's accounts
Bank Agent	User with view and edit privileges on any Client account, this role does not have administrative privileges
Administrator	User with view and edit privileges on any Client account, this role has administrative privileges on functionality of the system, e.g. removing / adding a user account, rebooting the system, altering the log of user transactions.

Functional Requirements as User Stories

US-xxx	<Name>
Description	As a <Role>, I want to <goal/desire>
Acceptance Criteria	<Criteria>

Client's Functional Requirements / User Stories

US-001	Depositing Money
Depositing money into a bank account	As a Client, I want to deposit money on my personal bank account
Acceptance Criteria	• Operation cannot be completed without the user entering the amount to be deposited • The current balance is modified in the database according to the amount deposited • An acknowledgement message is shown to the user (client) after submission

Client's Functional Requirements / User Stories

US-002	Withdrawing Money
Withdrawing money from a personal bank account	As a Client, I want to withdraw money from my personal bank account
Acceptance Criteria	• Operation cannot be completed without the user entering the amount to be debited from her account • Withdrawing funds are checked in the database • The current balance is modified in the database according to the amount debited • An acknowledgement message is shown to the user (client) after submission

Client's Functional Requirements / User Stories

US-003	Credit Card Payment
Payment of credit card	As a Client, I want to make payments to my credit card
Acceptance Criteria	• Operation cannot be completed without the user (client) entering the amount to be paid and the account to be debited • Amount to be paid must be greater than the minimum payment • Account balance is modified in the database according to the payment • An acknowledgement message is shown to the user after data submission

Client's Functional Requirements / User Stories

US-004	Bank Transfer
Transfers between accounts	As a Client, I want to transfer money from my saving (checking) account to my checking (saving) account
Acceptance Criteria	• Operation cannot be completed without the user entering the amount to be transferred, the client's account to be debited, and the IBAN and SWIFT of the target account. • Client's account balance in the database is modified accordingly • An acknowledgement message is shown to the user (client) after data submission

Client's Functional Requirements / User Stories

US-005	Personal Info
Updating personal info	As a Client, I want to update my personal information associated to any of my bank accounts
Acceptance Criteria	• Operation cannot be completed without the user (client) entering info about mandatory fields • An acknowledgement message is shown to the user (client) after data submission

Client's Functional Requirements / User Stories

US-006	Pin Code
Updating pin code	As a Client, I want to update my credentials (pin-code)
Acceptance Criteria	• Operation cannot be completed without the user (client) entering her old and new pin-code • An acknowledgement message is shown to the user (client) after data submission

Bank Agent's Functional Requirements / User Stories

US-007	Banking Movements
Checking user's information	As a Bank Agent, I want to access a client's personal and banking information (e.g. current balance, the list of banking movements, etc.)
Acceptance Criteria	• Operation cannot be completed without the Bank Agent entering the client's bank account info (ID or bank account number) • Information about the client's is shown to the Bank Agent after the transaction is completed

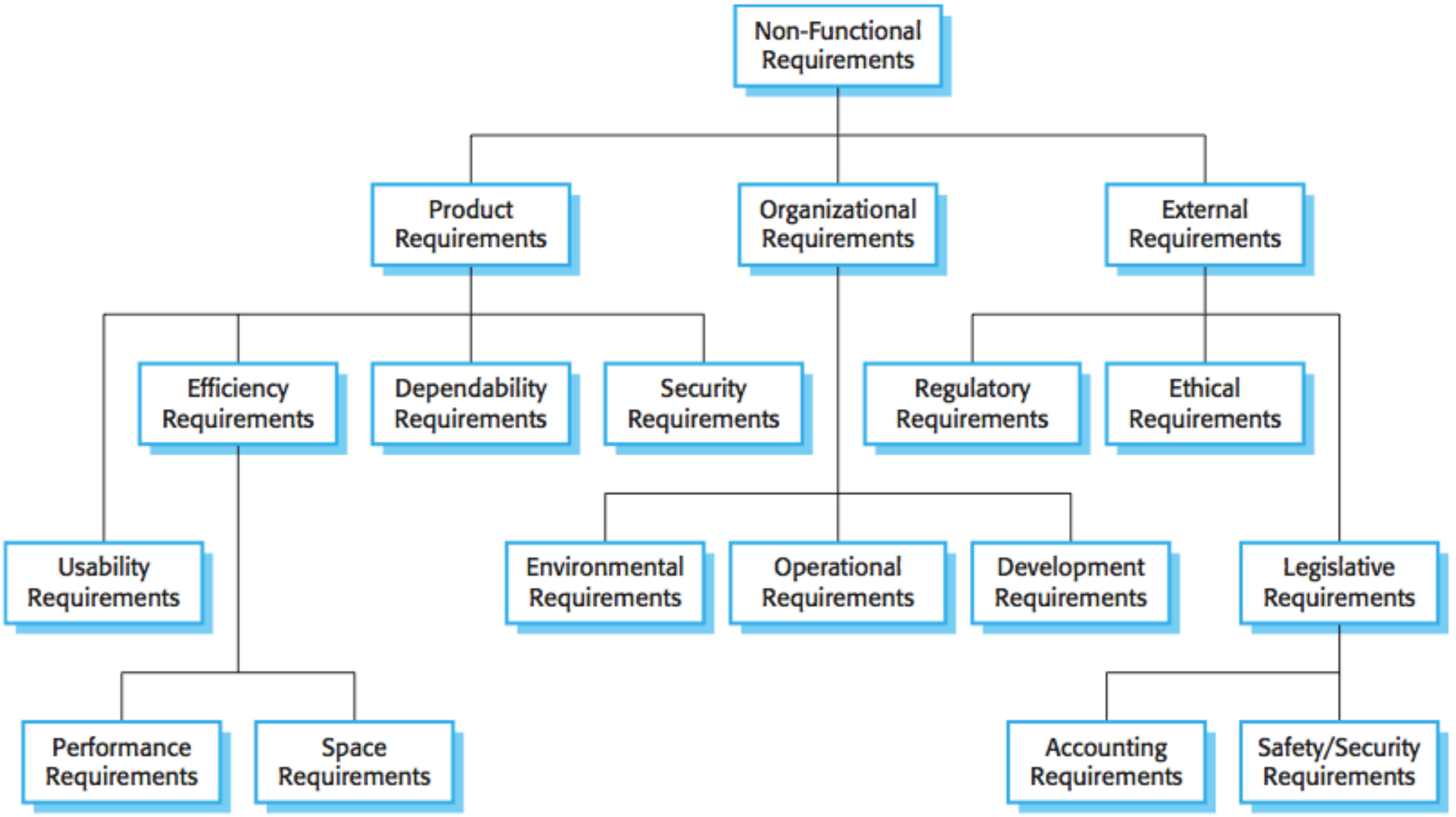
Administrator's Functional Requirements / User Stories

US-008	Creating User Account
Creating and Setting Up a New User Account	As an Administrator, I want to be able to create and set up a client's bank account
Acceptance Criteria	• Operation cannot be completed without Administrator entering client's mandatory fields including login, and password. • An acknowledgement message is shown to the user after creating the client's bank account.

Non-Functional Requirements

- **Non-functional requirements** are constraints on the services or functions offered by the system.
- **Non-functional requirements** include timing constraints, constraints on the development process, and constraints imposed by standards.
- **Non-functional requirements** often apply to the system as a **whole**, rather than individual system features or services.

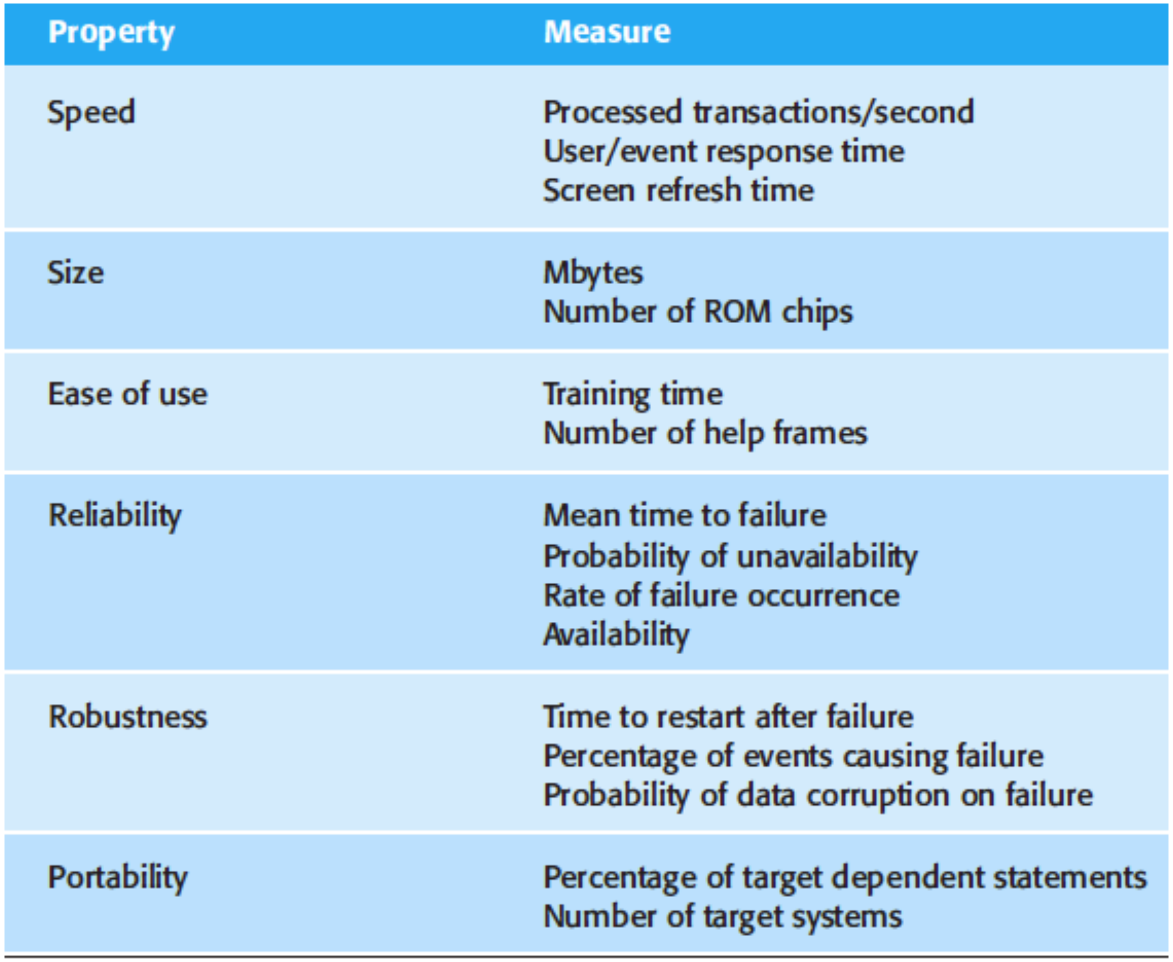
Types of Non-Functional Requirements



Types of Non-Functional Requirements

- **Usability**: is the system usable?
- **Portability**: usability of software through different environments
- **Security**: who can use the system?
- **Maintainability**: how easy is to keep the system working?
- **Extensibility**: what is the ability of the system to be extended?

Metrics for non-functional requirements



Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

What is Usability?

- **Learnability** – Minimal training
- **Efficiency** – Productivity in performing real tasks
- **Predictability** – You know what comes next
- **Memorability** – Little re-learning required
- **Satisfaction** – Pleasurable
- **Flexibility** – Adaptable

Why are Interfaces Important

- Sit-down-and-use computers and software
  - **don't read the manuals**
- Usability critical to **software sales**
- **HCI-trained** people build better interfaces
  - Programmers don't think like end-users
  - Exposure to different kinds of interfaces, problems

Who are Users?

- People who will use a computer system, device, or web site
- **As contrasted with Designers**
  - People who create the system
  - You
  - Designers ≠ Users

What is a User Interface?

- Everything a **user** encounters
  - Functionality
  - Content
  - Labels
  - Presentation
  - Layout
  - Navigation
  - Speed of response
  - Documentation and help

Non-Function Requirements – (Quality Attributes)

QA-xxx	<Type of the Quality Attribute>
Description	<The system should …>
Importance	<High/Medium/Low>
Measure	<Quantitative Measure>

Non-Function Requirements
Usability

QA-01	Usability
Description	Users will have a high satisfaction with the use of the online banking system
Importance	High
Measure	Measured by a Survey on customers (clients)

Non-Function Requirements
Usability

QA-02	Usability
Description	Transferring money between user accounts can be learnt in less than two minutes
Importance	High
Measure	Measured by a Survey on customers

Non-Function Requirements
Usability

QA-03	Usability
Description	Creating and setting up a user's bank account takes less than 10 minutes
Importance	High
Measure	Training time as measured by a survey conducted on system administrators

Non-Function Requirements
Security

QA-04	Security
Description	A user will have access to her own account information only
Importance	High
Measure	# of identified access permission violations

Non-Function Requirements
Security

QA-05	Security
Description	Only users with correct login/password will be given access to the online banking system
Importance	High
Measure	% login attempts with correct/incorrect passwords

Non-Function Requirements
Portability

QA-06	Portability
Description	The online banking system will run on Windows, Linux and Mac OS X platforms
Importance	High
Measure	% of supported platforms

Non-Function Requirements
Portability

QA-07	Portability
Description	The online banking system will run on Android mobile devices
Importance	High
Measure	% of supported platforms

Non-Function Requirements
Extensibility

QA-08	Extensibility
Description	The online banking system will be extended to support the payment of electricity bills
Importance	High
Measure	The average person/day to extend the system shall not exceed 30 days

Check List for Good Requirements

- **Clarity** → requirements should not be ambiguous or allow misinterpretations
- **Completeness** → all the desires of the product owner are considered
- **Consistency** → requirements do not contradict each other

Check List for Good Implementation

- **Traceability** → every implemented functionality can be traced back to a list of requirements
- **Verifiability** → does the system implement the functionality right?
- **Testability** → does the system implement the right functionality?

Business Rules

- **Business rules** define **restrictions** on some aspect of the business structure that may influence the behaviour of the system

Business Rules

- **BR1** – Users will have exactly two types of bank accounts, **saving** and **checking**.
- **BR2** – Minors will not be permitted to open a bank account without adult permission.

Software Constraints

- **Constraints** are restrictions on the degree of freedom in providing a solution for a system
  - They can be Economical, Political, Technical, Environmental
- **Constraints** are written the same way as standard requirements and sometimes can be **identified** as **non-functional requirements** or **business rules**

Constraints

CONS-xxx	<Name of the Constraint>
Description	<The system should be …>
Importance	<High/Medium/Low>
Justification	

Constraints

CONS-01	Existing Infrastructure
Description	The banking system will work on the existing technical infrastructure
Importance	High
Justification	Lloyds wants to extend the existing user’s services

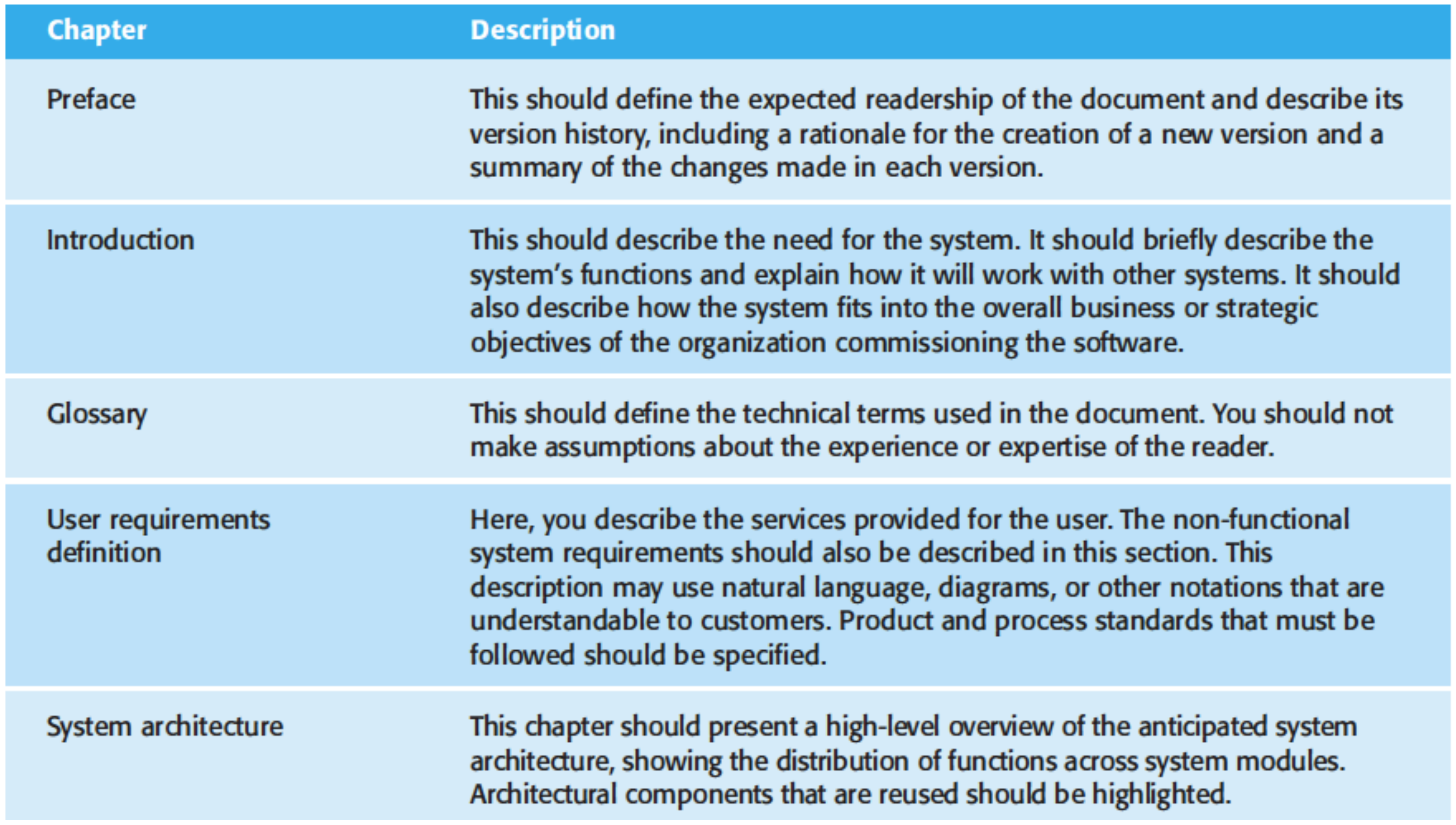
Constraints

CONS-02	Corporate Database Information
Description	The banking system will only use data contained in the banking database
Importance	High
Justification	Access to the database of other banking systems is protected by copyright

What's a Requirement Document?

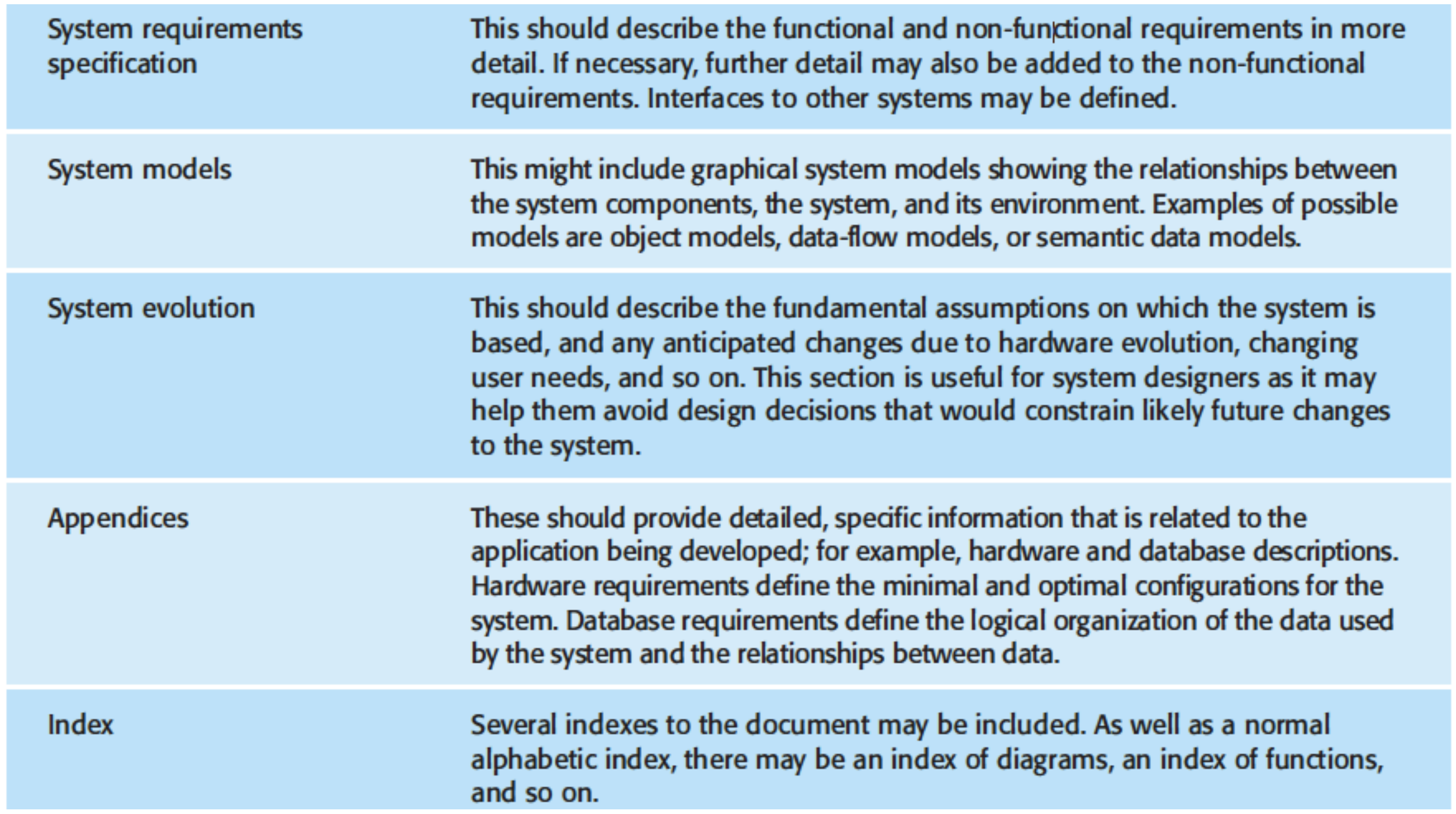
- When a company wishes to contract for a software development project, a software requirements document should be written that defines the **company needs**.
- Once the contract has been awarded the contractor must write a **system definition** with **more details** so that the client can understand and validate **what the software will do**

The Software Requirements Document
Part 1



Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The Software Requirements Document
Part 2



System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Ways of Writing a System Requirements Specification

1. Natural language sentences (yes--previously)
2. Structured natural language (yes)
3. Design description language (no)
4. Graphical notation (no)
5. Mathematical specification (yes)

Structured Requirements

- Structured natural language is a way of writing system requirements where the freedom of the requirements writer is limited and all the requirements are written in a standard way.
- The specification may use programming language constructors to show alternatives and iteration, and may highlight key elements using shading or different fonts.

Structured Requirements

REQ-xxx	<Function Name>
Description	<Text>
Pre-Condition	<Cond>
Post-Condition	<Cond>
Frame Condition	<Side Effects>

Lloyds Banking System
(CONT 1 …)

Saving accounts. Saving accounts for professors are created with an initial balance of 200 Euros. Saving accounts for students are created with an initial balance of 100 Euros. Professors and students have an overdraft protection of 200 Euros and 100 Euros, respectively. If a debit operation would result in an overdraft beyond that protection, then the bank will not permit the operation.

Structured Requirements

REQ-001	CreateProfSavingAccount
Description	Creates a saving account for professors
Pre-Condition	None
Post-Condition	balance == 200
Frame Condition	{ balance }

Structured Requirements

REQ-002	CreateStdntSavingAccount
Description	Creates a saving account for students
Pre-Condition	None
Post-Condition	balance == 100
Frame Condition	{ balance }

Structured Requirements

REQ-003	DebitSavingAccount(amount)
Description	It debits certain amount from the saving account
Pre-Condition	amount >= 0

Post-Condition	if balance + overdraft >= amount
	then balance = \old(balance) - amount
	else balance = \old(balance)

Frame Condition	{ balance }

Lloyds Banking System (CONT 2 …)

Checking accounts are created with an initial balance of 0 Euros for both professors and students. Checking accounts do not enjoy an overdraft protection. There is a limit of two on the number of consecutive (without crediting the account first) withdrawals a bank user can make on checking accounts. Thus, a third withdrawal from a checking account is allowed only if a deposit operation happens between the second and third withdrawal.

Structured Requirements

REQ-004	CreateCheckingAccount
Description	Creates a checking account
Pre-Condition	true
Post-Condition	balance == 0
Frame Condition	{ balance }

Structured Requirements

REQ-004	DebitCheckingAccount(amount)
Description	It debits certain amount from a checking account
Pre-Condition	amount >= 0

Post-Condition	If balance >= amount and numDebits < 2
	then
	    balance = \old(balance) – amount
	    numDebits = numDebits + 1
	else balance = \old(balance)

Frame Condition	{ balance, numDebits }

Mathematical Specification
High Secure Systems - Tokeneer

```
machine abstract_machine sees context
 variables certificates publicKeys
      isValidatedBy validityPeriods
      certificateID
 invariants
  @inv0_1 certificates ⊆ CERTIFICATES
  @inv0_2 publicKeys ⊆ KEYS
  @inv0_4 validityPeriods ∈ certificates ↔ ℕ
```

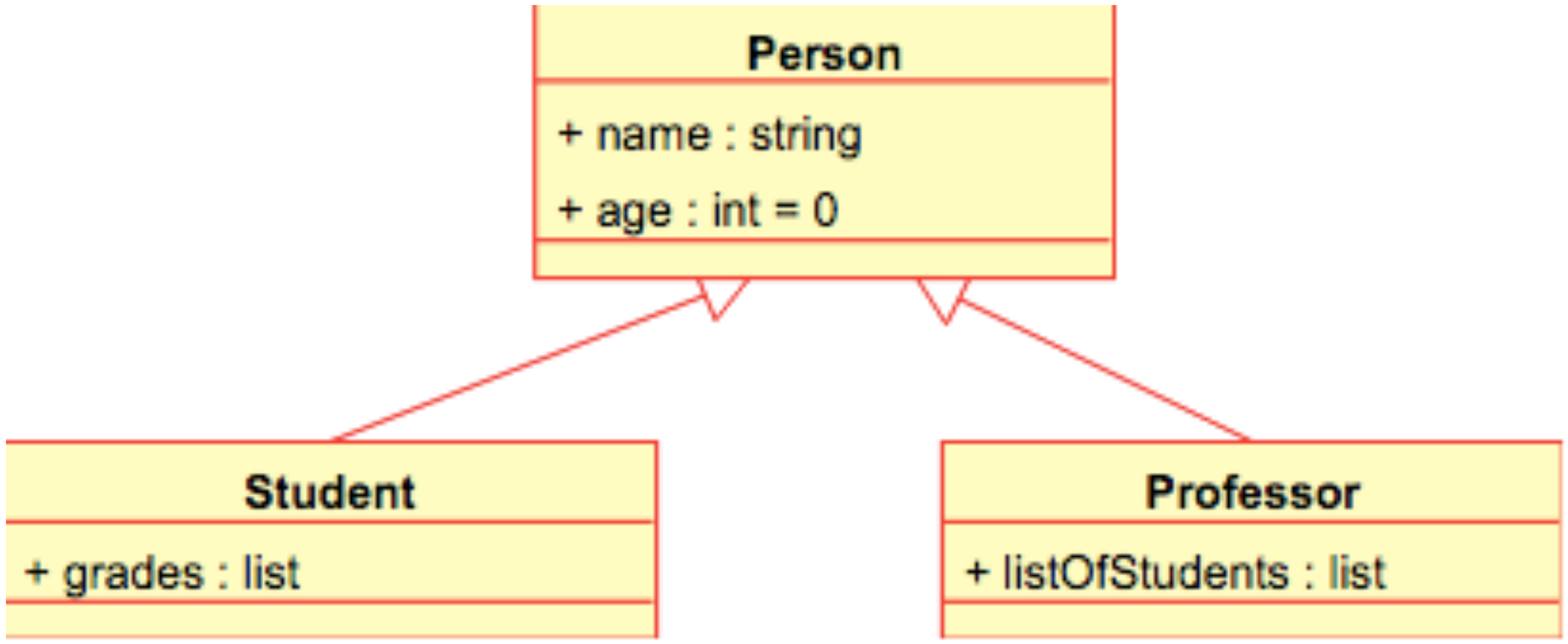
Mathematical Specification
High Secure Systems - Tokeneer

```
event addCertificate
 any ce period pubkey where ... then
  @act1 certificates := certificates ∪ {ce}
  @act2 validityPeriods :=  validityPeriods ∪
                           {ce ↦ period}
  @act3 isValidatedBy := isValidatedBy ∪
                            {ce ↦ pubkey }
  @act4 certificateID := certificateID ∪
                            {ce ↦ cid}
end
```

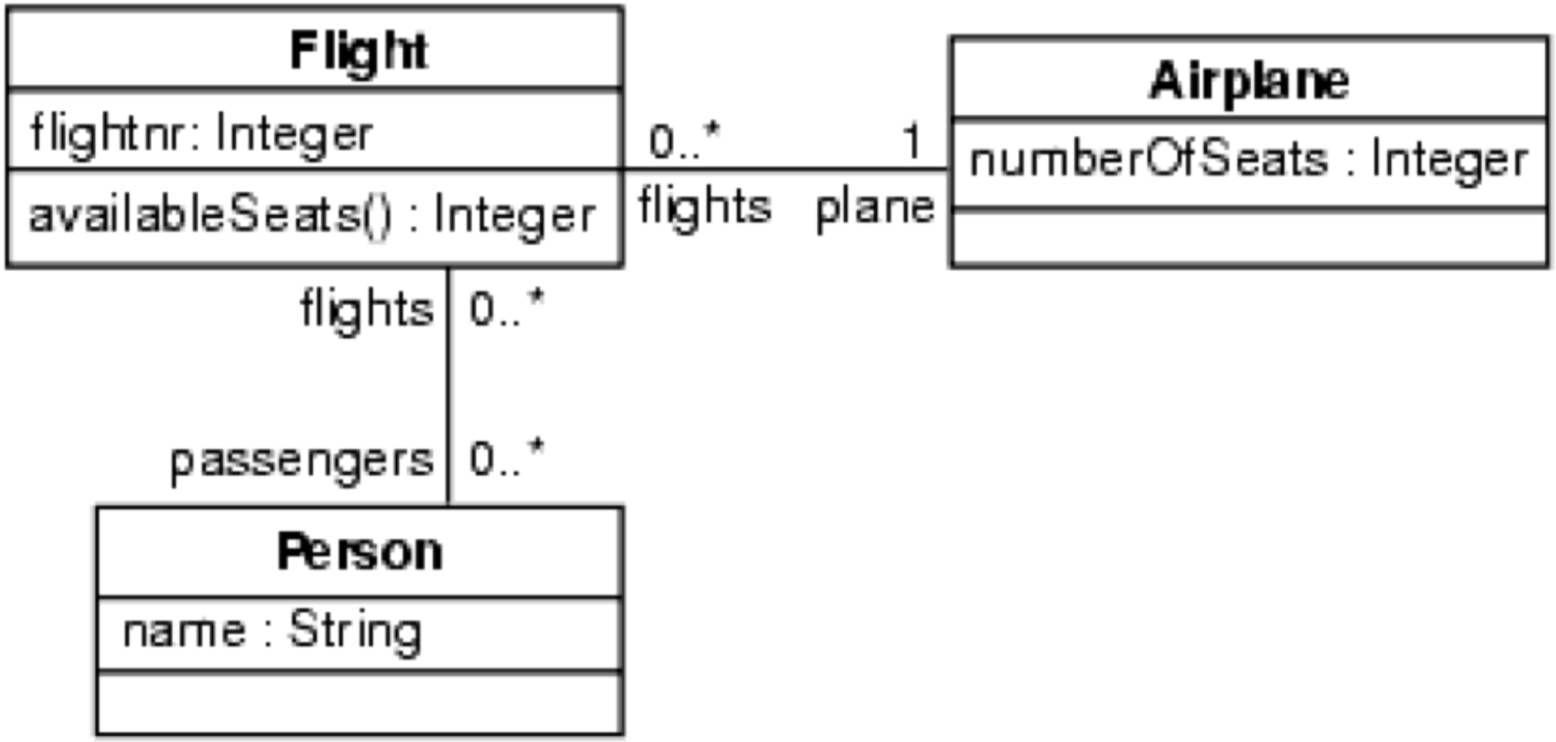
Mathematical Specification
Facebook

```
machine permissions refines Facebook sees ctx
variables person content owner page viewp editp
invariants
   @invr1 viewp ∈ content ↔ person
   @invr2 editp ∈ content ↔ person
   @invr3 editp ⊆ viewp
   @invr4 owner ⊆ viewp
   @invr5 owner ⊆ editp
   @invr6 viewp ⊆ page
event upload extends upload then
 @act1 viewp := viewp ∪ {c ↦ p}
 @act2 editp := editp ∪ {c ↦ p}
end
```

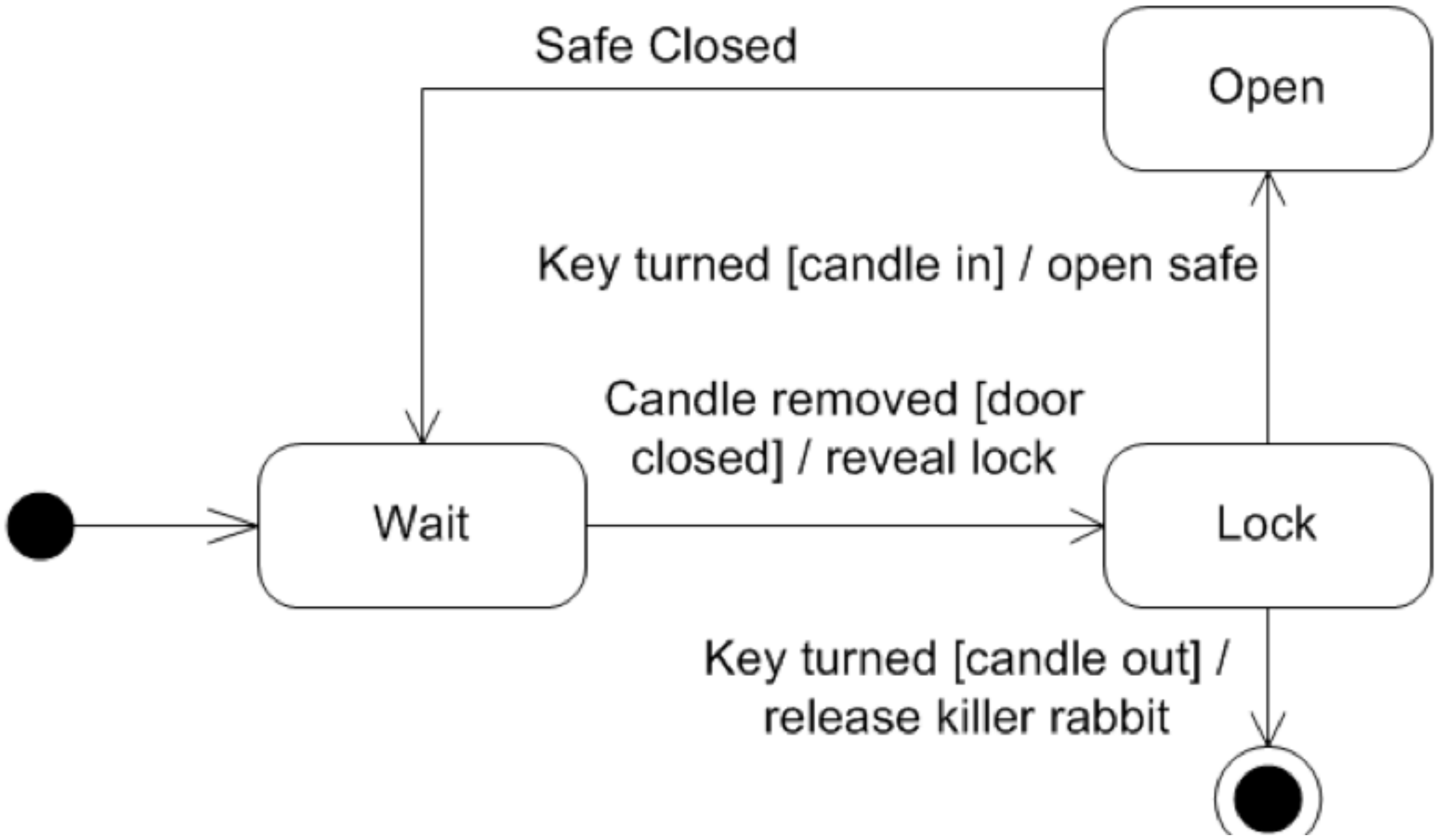
Graphical Notation – Class Diagram (taken from Wikipedia)



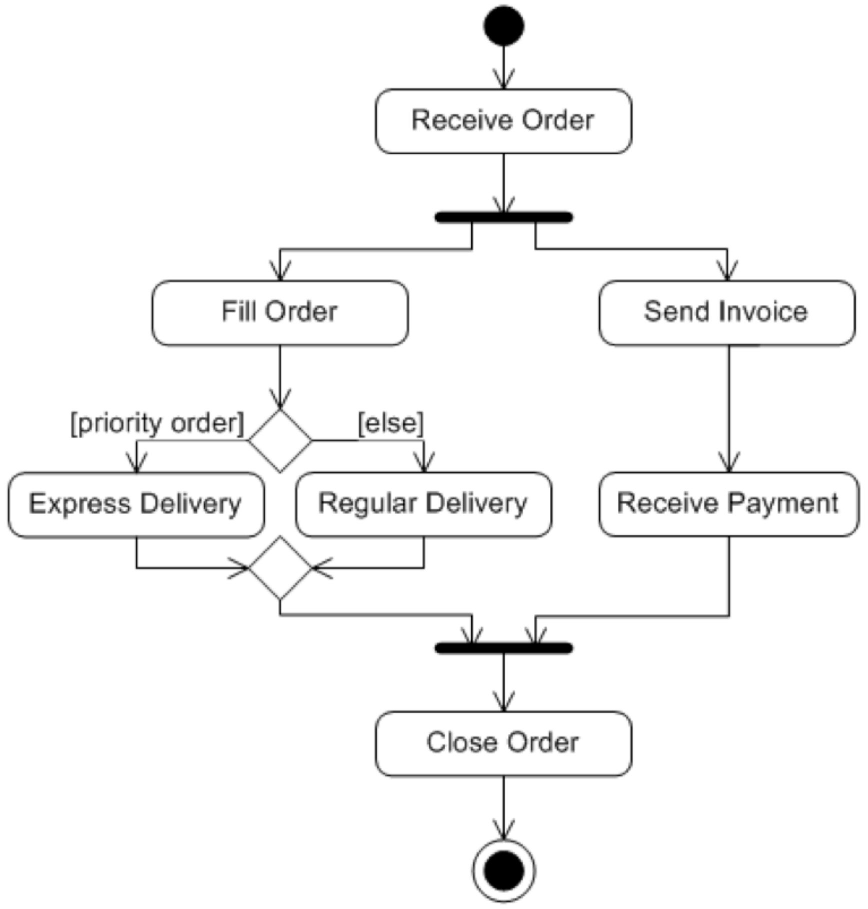
Graphical Notation – Class Diagram



Graphical Notation – State Machine


Safe Closed
Open
Key turned [candle in] / open safe
Candle removed [door closed] / reveal lock
Wait
Lock
Key turned [candle out] / release killer rabbit

Graphical Notation – Activity Diagram



Further Reading

- Software Engineering, 9th Edition, by I. Somerville, **Chapter 4**.
- Agile Principles, Patterns, and Practices in C♯, By Robert C. Martin and Mica Martin. Prentice Hall.

Next Class

- Software development models
- Software life-cycle activities
- Methodological frameworks

Today's Lab

- **Homework** : software requirements for a Lift Controller
- **Due: 01/Feb at 9AM**
- Turn in your homework via **Moodle**!!