INTRODUCTION TO SOFTWARE ENGINEERING

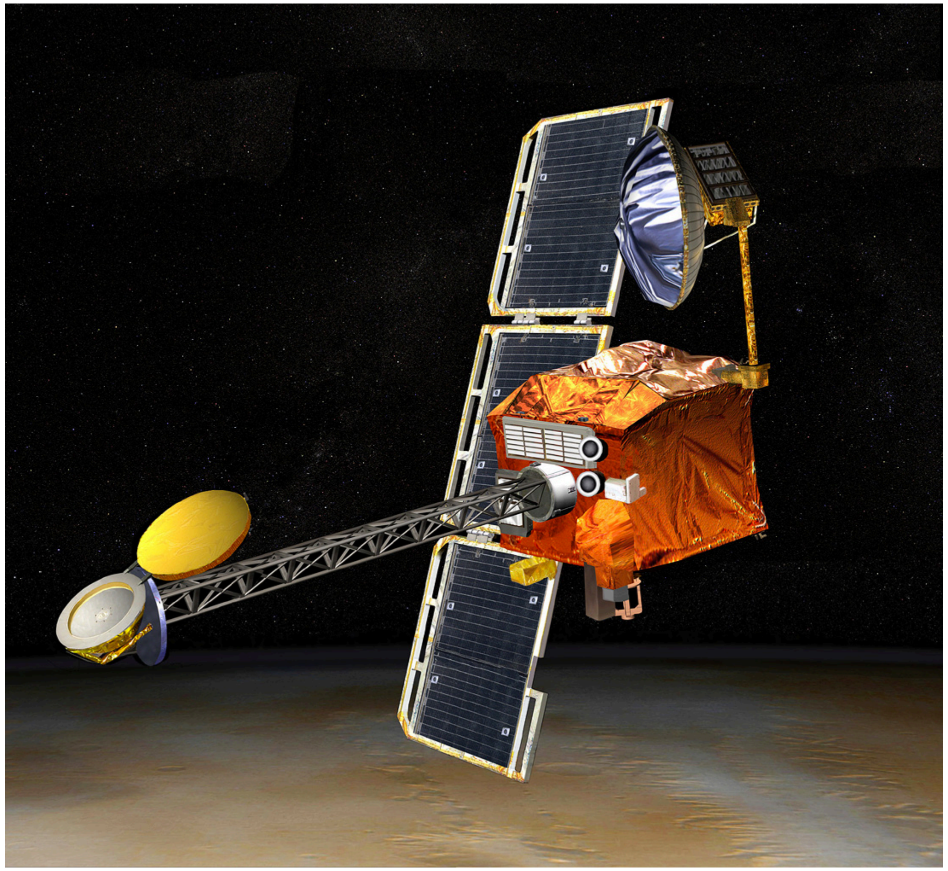
6. REQUIREMENT ENGINEERING

Nguyen Thanh Hùng
hungnt@soict.hust.edu.vn

ĐẠI HỌC
BÁCH KHOA

Mars Climate Orbiter

- In 1999, the Mars Climate Orbiter disappears around Mars
- Cost: about $125 millions
- Problem caused by a misunderstanding between a team in Colorado and one in California
- One team used the metric system while the other used the English system for a key function…



GIRES

- GIRES1 (Gestion intégrée des ressources)
  - Integrated management of resources
  - To replace >1.000 existing systems
  - In 140 organisations / departments
  - Affecting 68.000 employees!
- 8-year project of the Quebec government, started 1998
- $80 million budget
- Could not cope with changes to the requirements…
  - Cost of $400 millions after 5 years, and very late
  - Project cancelled in 2003

[1] http://radio-canada.ca/nouvelles/Index/nouvelles/200303/04/012-GIRES.shtml

Content

Content

According to IEEE 830-1993

- A requirement is defined as:
  - A condition or capability needed by a user to solve a problem or achieve an objective
  - A condition or a capability that must be met or possessed by a system … to satisfy a contract, standard, specification, or other formally imposed document …

Requirements

- Capturing the purpose of a system
- An expression of the ideas to be embodied in the system or application under development
- A statement about the proposed system that all stakeholders agree must be made true in order for the customer's problem to be adequately solved
  - Short and concise piece of information
  - Says something about the system
  - All the stakeholders have agreed that it is valid
  - It helps solve the customer's problem

Requirements Engineering

- The activity of development, elicitation, specification, analysis, and management of the stakeholder requirements, which are to be met by a new or evolving system
- RE is concerned with identifying the purpose of a software system… and the contexts in which it will be used
  - How/where the system will be used
  - Big picture is important
- Captures real world needs of stakeholders affected by a software system and expresses them as artifacts that can be implemented by a computing system
  - Bridge to design and construction
  - How to communicate and negotiate?
  - Is anything lost in the translation between different worlds?

General Problems with Requirement

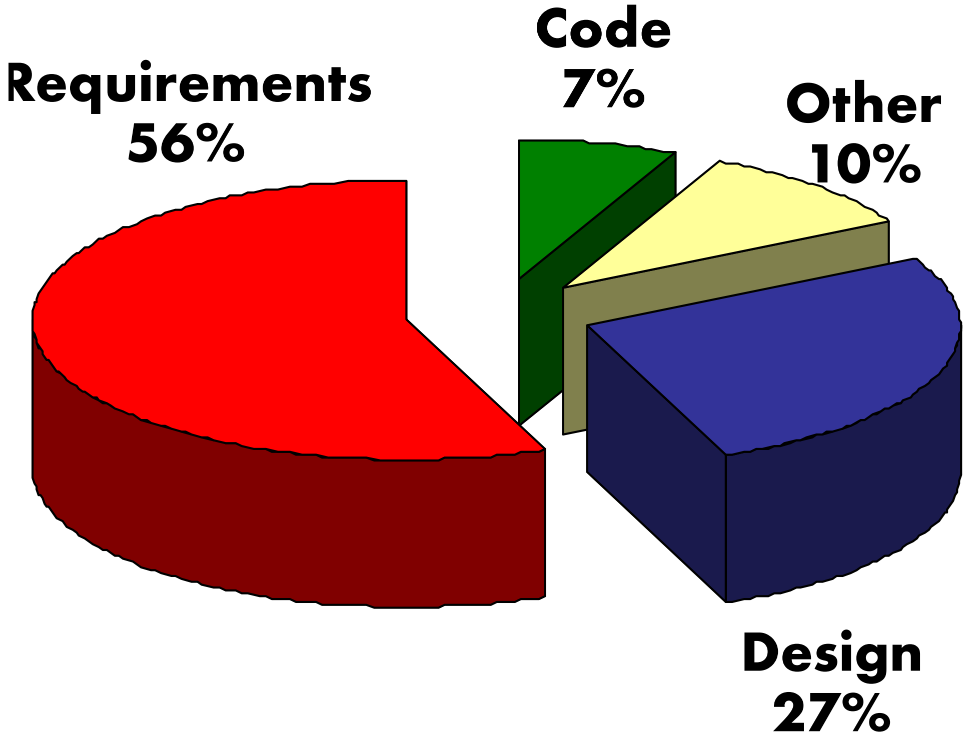
- Lack of the right expertise (software engineers, domain experts, etc.)

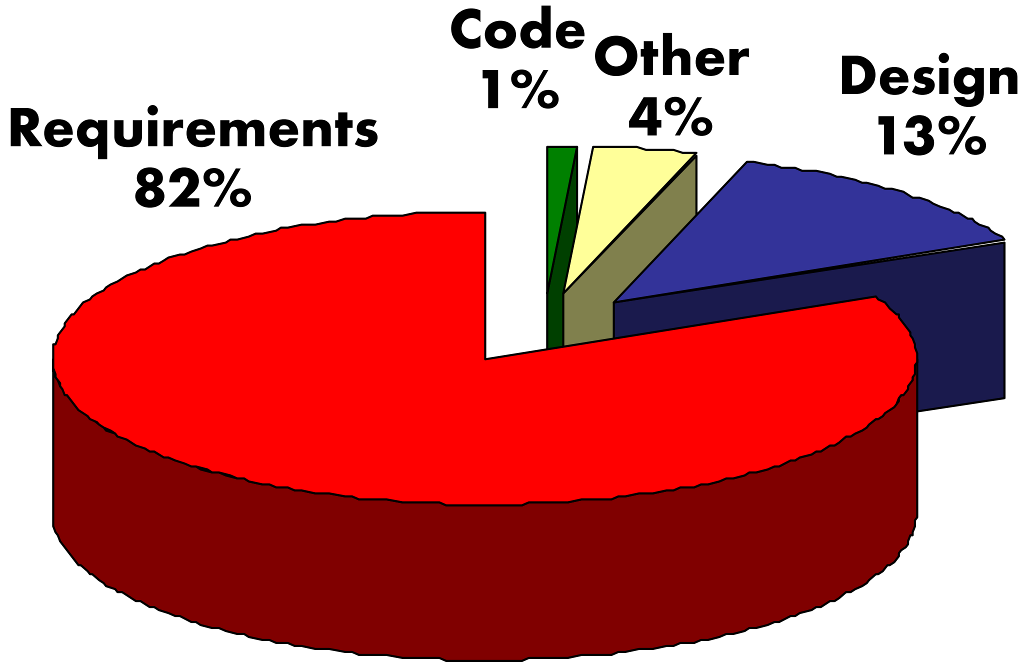
- Initial ideas are often incomplete, wildly optimistic, and firmly entrenched in the minds of the people leading the acquisition process

- Difficulty of using complex tools and diverse methods associated with requirements gathering may negate the anticipated benefits of a complete and detailed approach

Why Focus on Requirements ?

- Distribution of Defects
- Distribution of Effort to Fix Defects



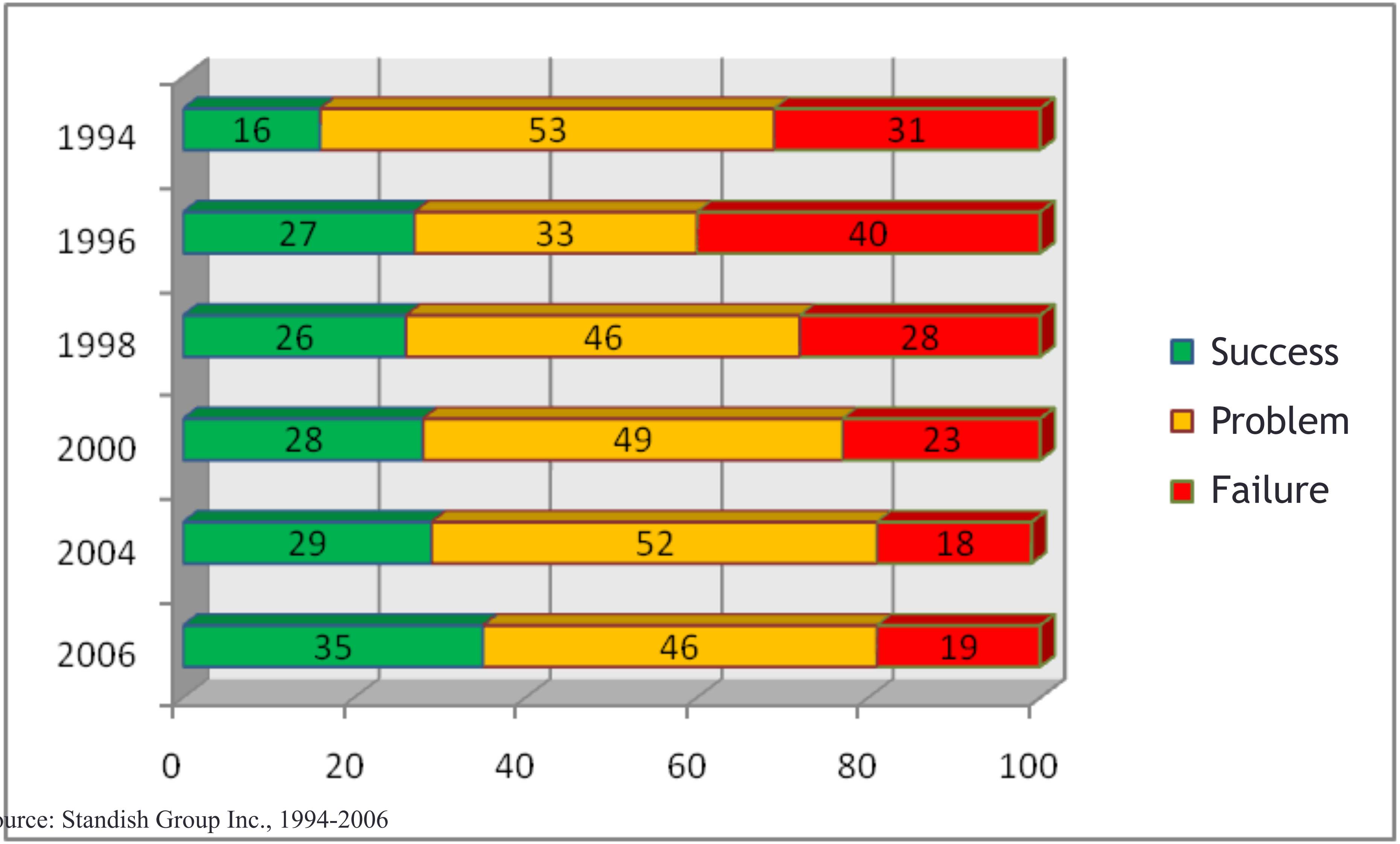


Source: Martin & Leffinwell

Software Engineering Institute (SEI) Vision

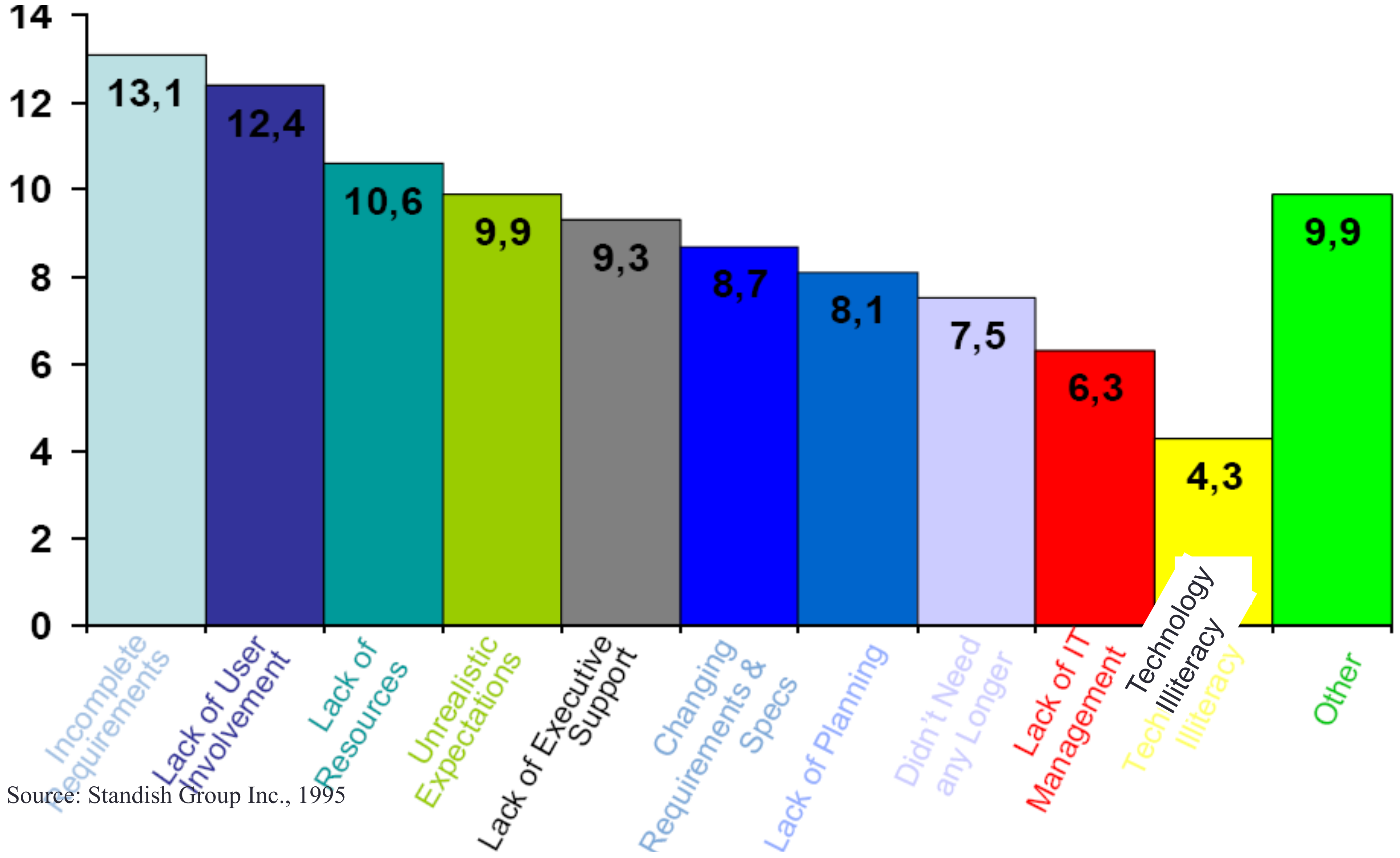
- The right software, delivered defect free, on time & on cost, every time
    - “Right software” implies software that satisfies requirements for functionality and qualities (e.g., performance, cost…) throughout its lifetime
    - “Defect free” software is achieved either through exhaustive testing after coding or by developing the code right the first time

Project success since 1994



Source: Standish Group Inc., 1994-2006

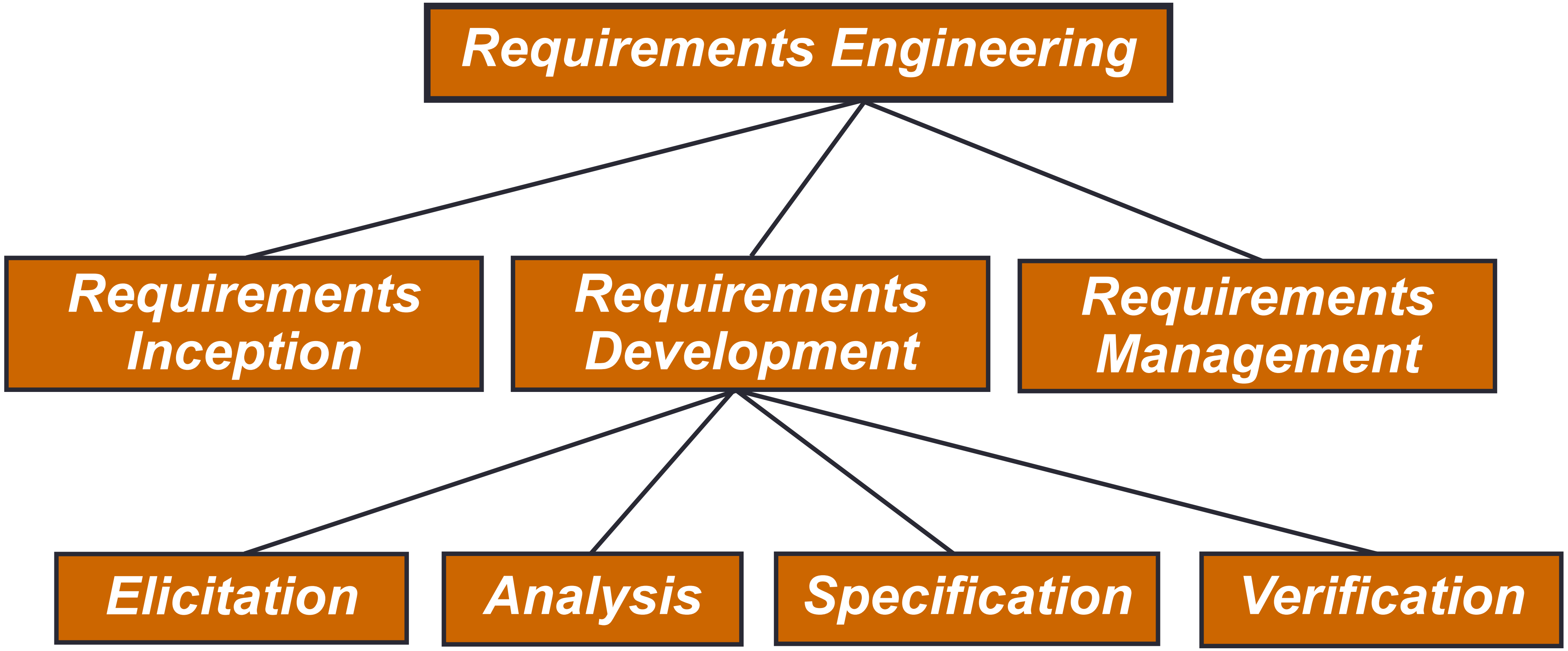
Problem Causes to Fail Project



Source: Standish Group Inc., 1995

Content

Requirements Engineering Activities



Source: Larry Boldt, Trends in Requirements Engineering People-Process-Technology, Technology Builders, Inc., 2001

Requirements Inception

- Start the process
  - Identification of business need
  - New market opportunity
  - Great idea
- Involves
  - Building a business case
  - Preliminary feasibility assessment
  - Preliminary definition of project scope

- Stakeholders
  - Business managers, marketing people, product managers...
- Examples of techniques
  - Brainstorming, Joint Application Development (JAD) meeting…

Requirements Elicitation

- Gathering of information
  - About problem domain
  - About problems requiring a solution
  - About constraints related to the problem or solution
- Questions that need to be answered
  - What is the system?
  - What are the goals of the system?
  - How is the work done now?
  - What are the problems?
  - How will the system solve these problems?
  - How will the system be used on a day-to-day basis?
  - Will performance issues or other constraints affect the way the solution is approached?

Requirements Analysis

- The process of studying and analyzing the needs of stakeholders (e.g., customer, user) in view of coming up with a “solution”. Such a solution may involve:
  - A new organization of the workflow in the company.
  - A new system (system-to-be, also called solution domain) which will be used in the existing or modified workflow.
  - A new software to be developed which is to run within the existing computer system or involving modified and/or additional hardware.
- Objectives
  - Detect and resolve conflicts between requirements (e.g., through negotiation)
  - Discover the boundaries of the system / software and how it must interact with its environment
  - Elaborate system requirements to derive software requirements

Requirements Specification

- The invention and definition of the behavior of a new system (solution domain) such that it will produce the required effects in the problem domain
- Requirements Analysis has defined the problem domain and the required effects

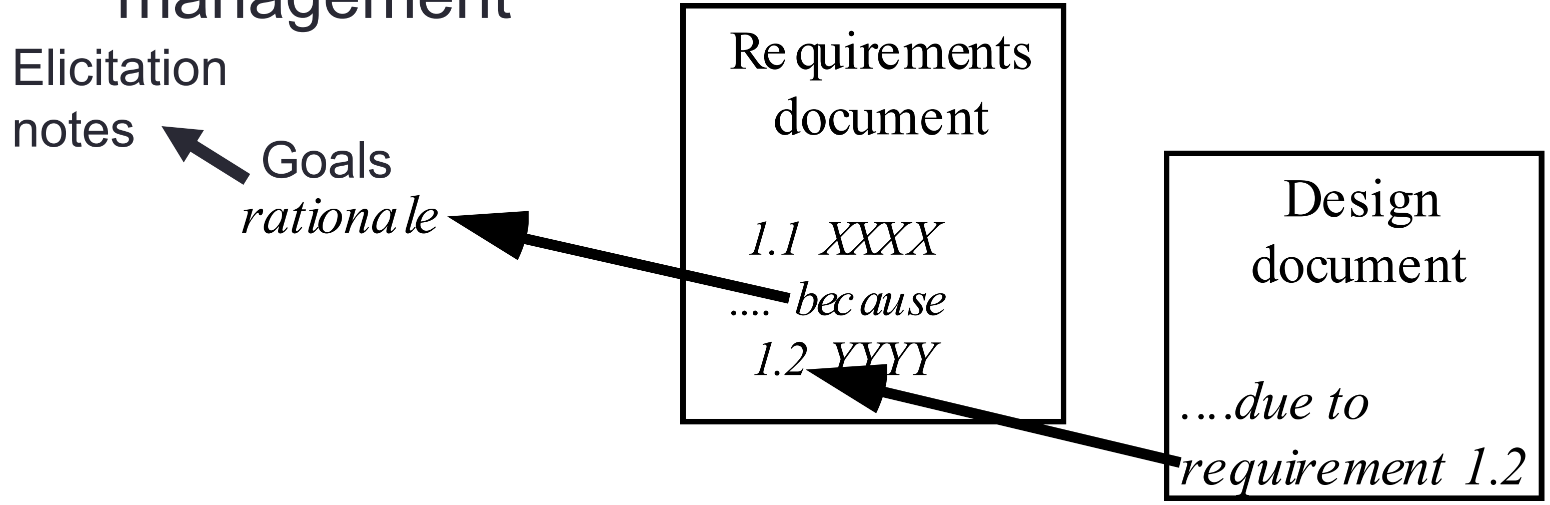
- Specification Document
  - A document that clearly and precisely describes, each of the essential requirements (functions, performance, design constraints, and quality attributes) of the software and the external interfaces
  - Each requirement being defined in such a way that its achievement is capable of being objectively verified by a prescribed method (e.g., inspection, demonstration, analysis, or test)
  - Different guidelines and templates exist for requirements specification

Requirements Verification and Validation

- Validation and verification
  - Both help ensure delivery of what the client wants
  - Need to be performed at every stage during the process
- Validation: checks that the right product is being built (refers back to stakeholders – main concern during RE)
- Verification: checks that the product is being built right
  - During design phase: refers back to the specification of the system or software requirements
  - During RE: mainly checking consistency between different requirements, detecting conflicts

Requirements Management

- Necessary to cope with changes to requirements
- Requirements change is caused by:
  - Business process changes
  - Technology changes
  - Better understanding of the problem
- Traceability is very important for effective requirements management



Requirement definition

- Structure-based: DFD (Data Flow Diagram)
- Entity Relation Diagram
- Finite State Machine
- Object-oriented: Use case diagram

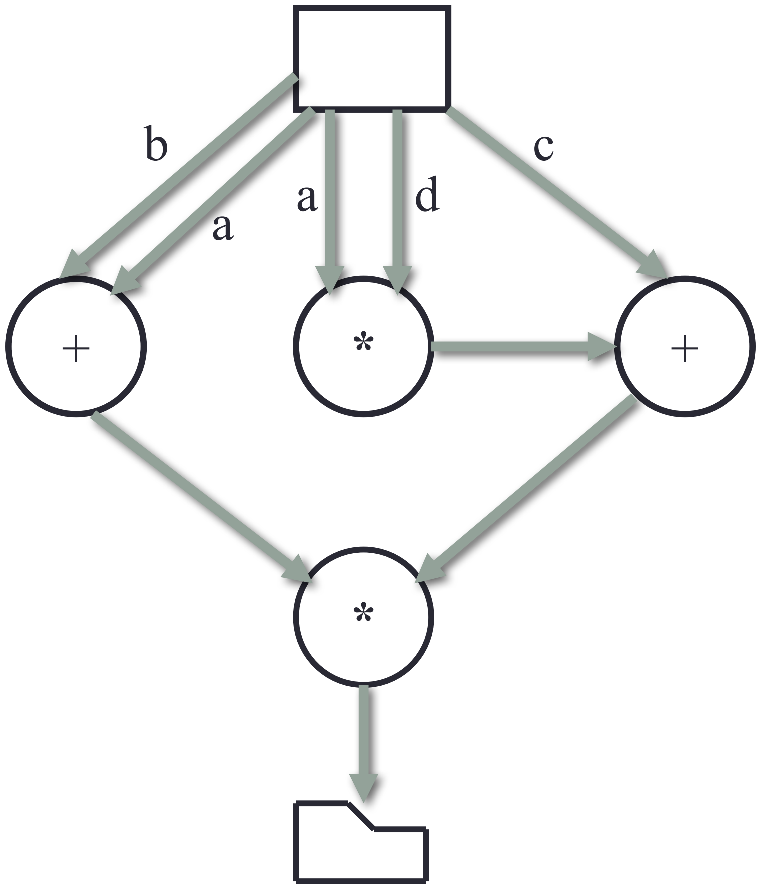
1. Requirement Specifcation with DFD

- System: collection of data that is processed by corresponding functions.
- Syntax:

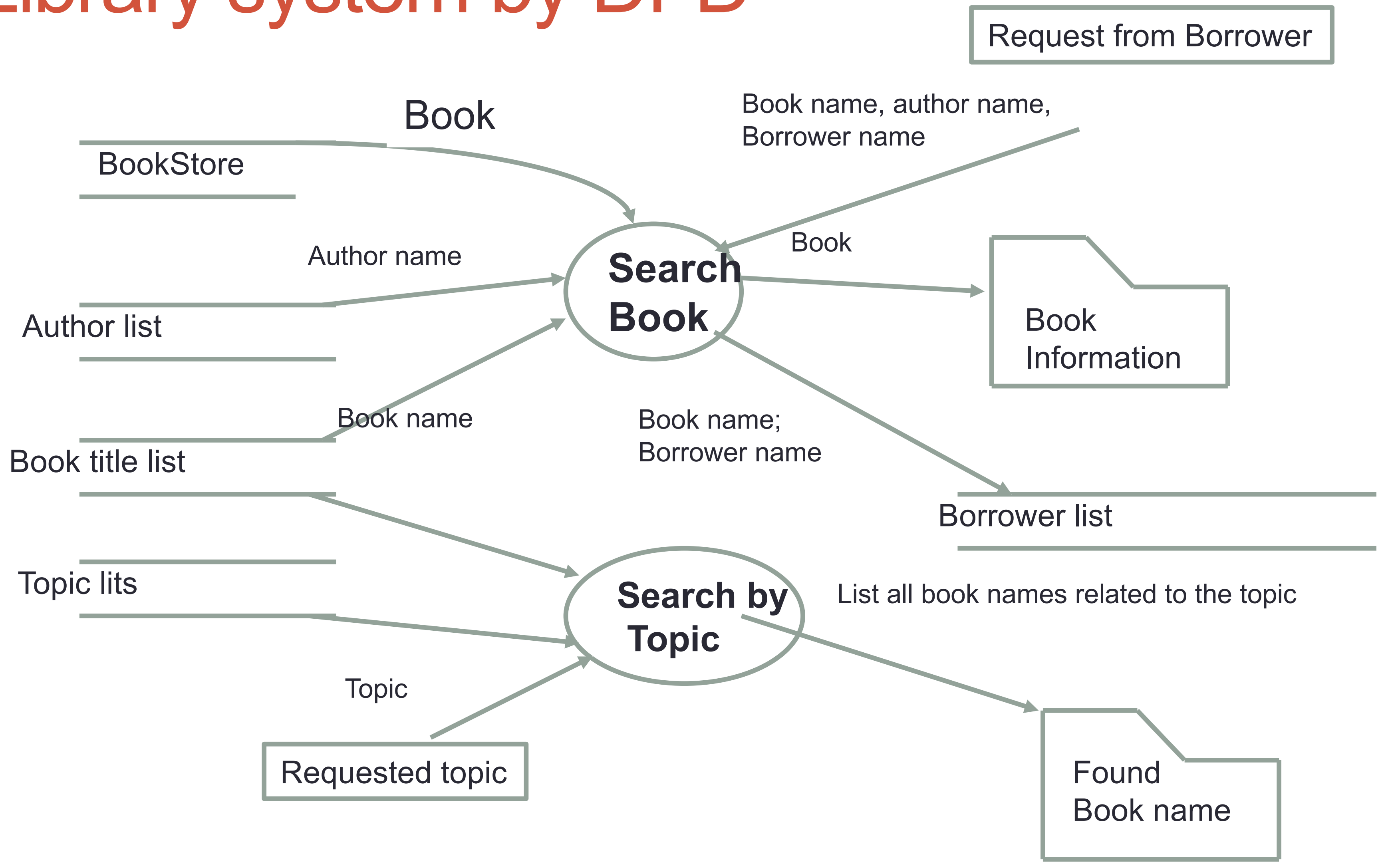
○	functions
→	Data flow
=	Data store
□ 🗀	Data input/ouput and interaction between system and users

Example of Math formula by DFD

(a+b)*(c+a*d)-e*(a+b)

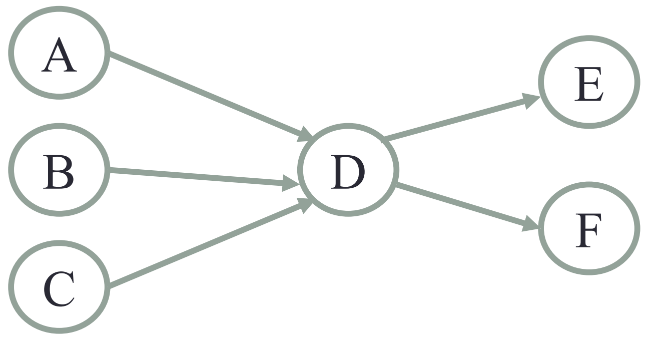


Example of Requirement specifcation of a Library system by DFD



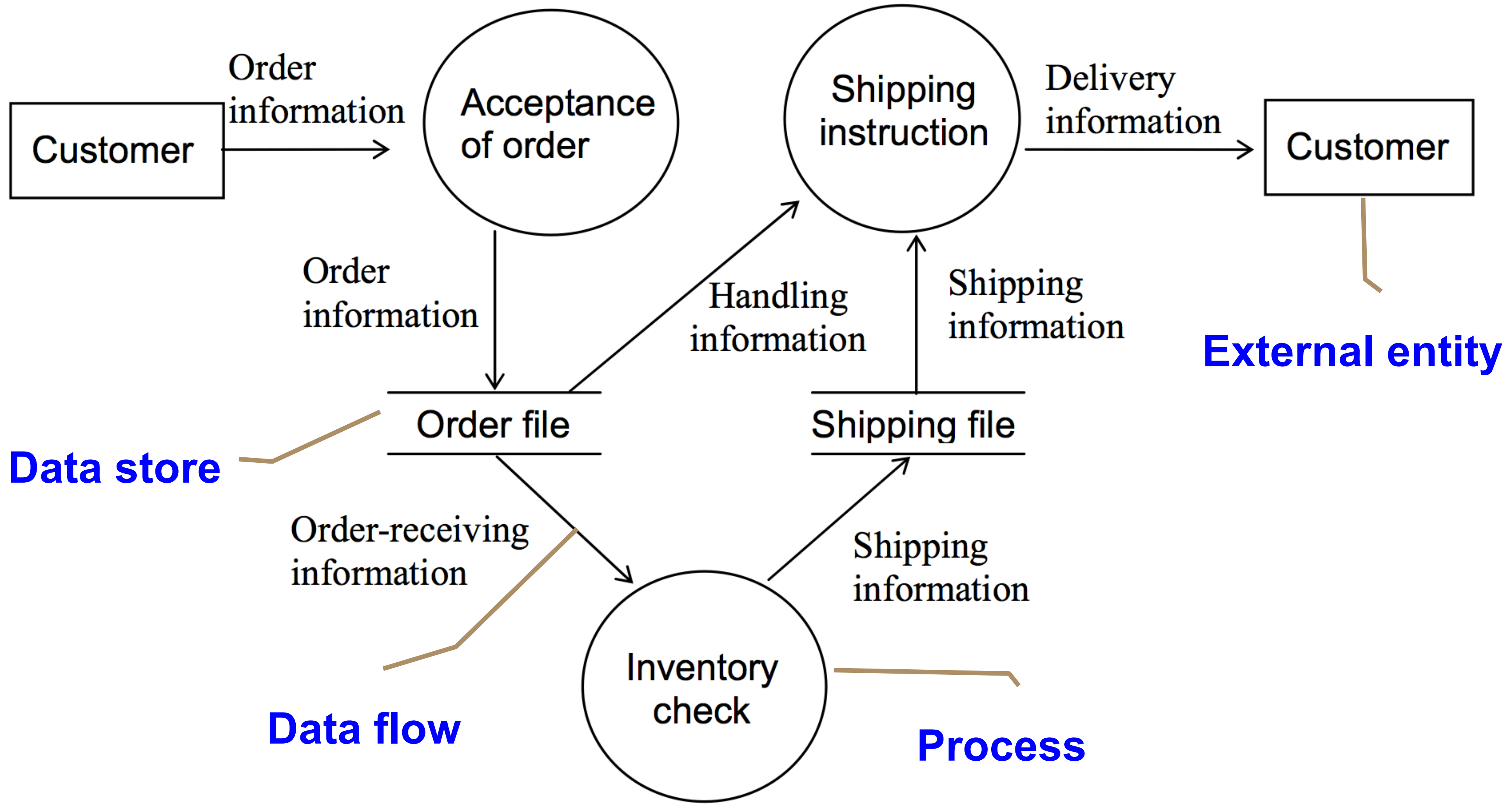
DFD disavantages

- DFD does not specify flow direction (control aspects)



- No synchronization mechanism

DFD (Data Flow Diagram)

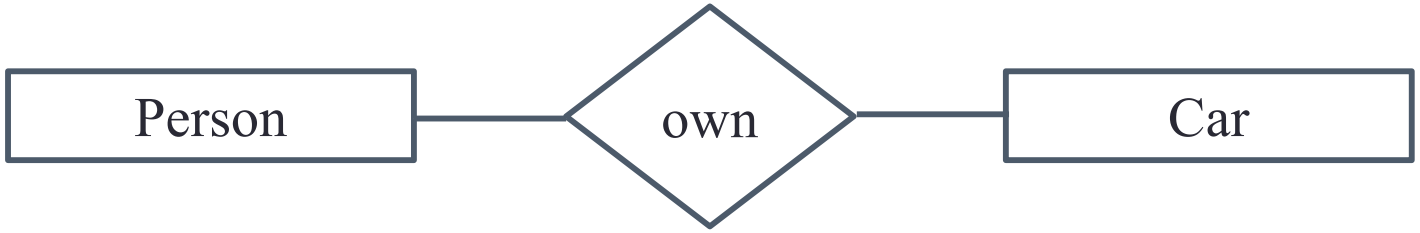


3. Entity Relation Diagram

- Concetual model for specifying logic requirements of systems, usually used for large-scale data system.
- ER Model
  - Entity
  - Relation
  - Attribute

Entity

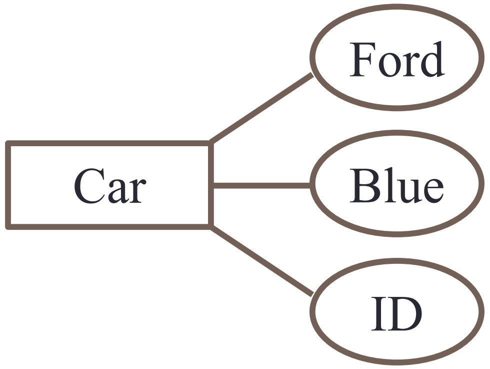
- Entity: a set of involved information that need to be processded in software
- Entity might have a relation:
  - Person owns a car

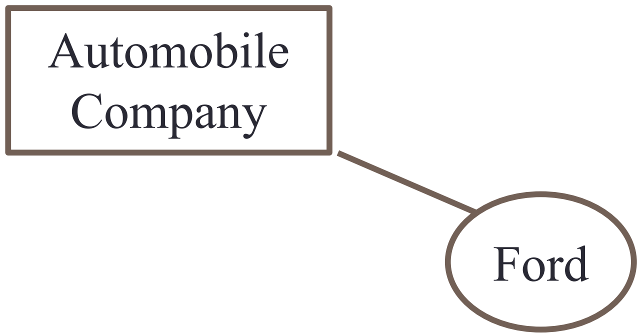


- Entity might have attributes

Attributes

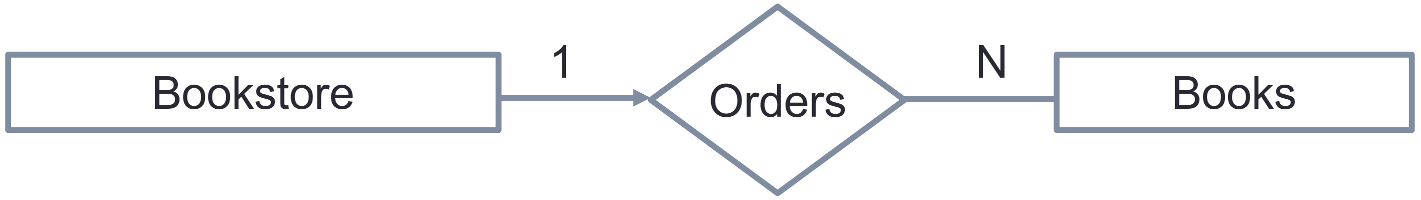
- Attributes of an entity or a data object





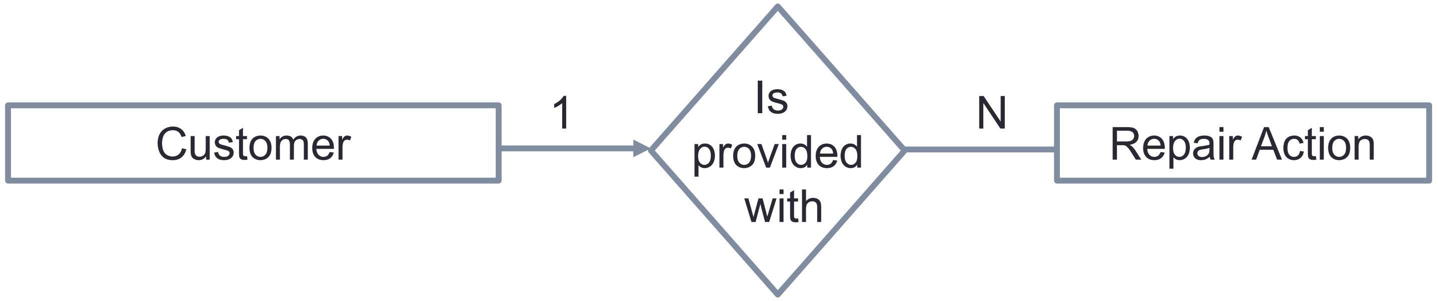
Relationship

- Specify relationship between data objects

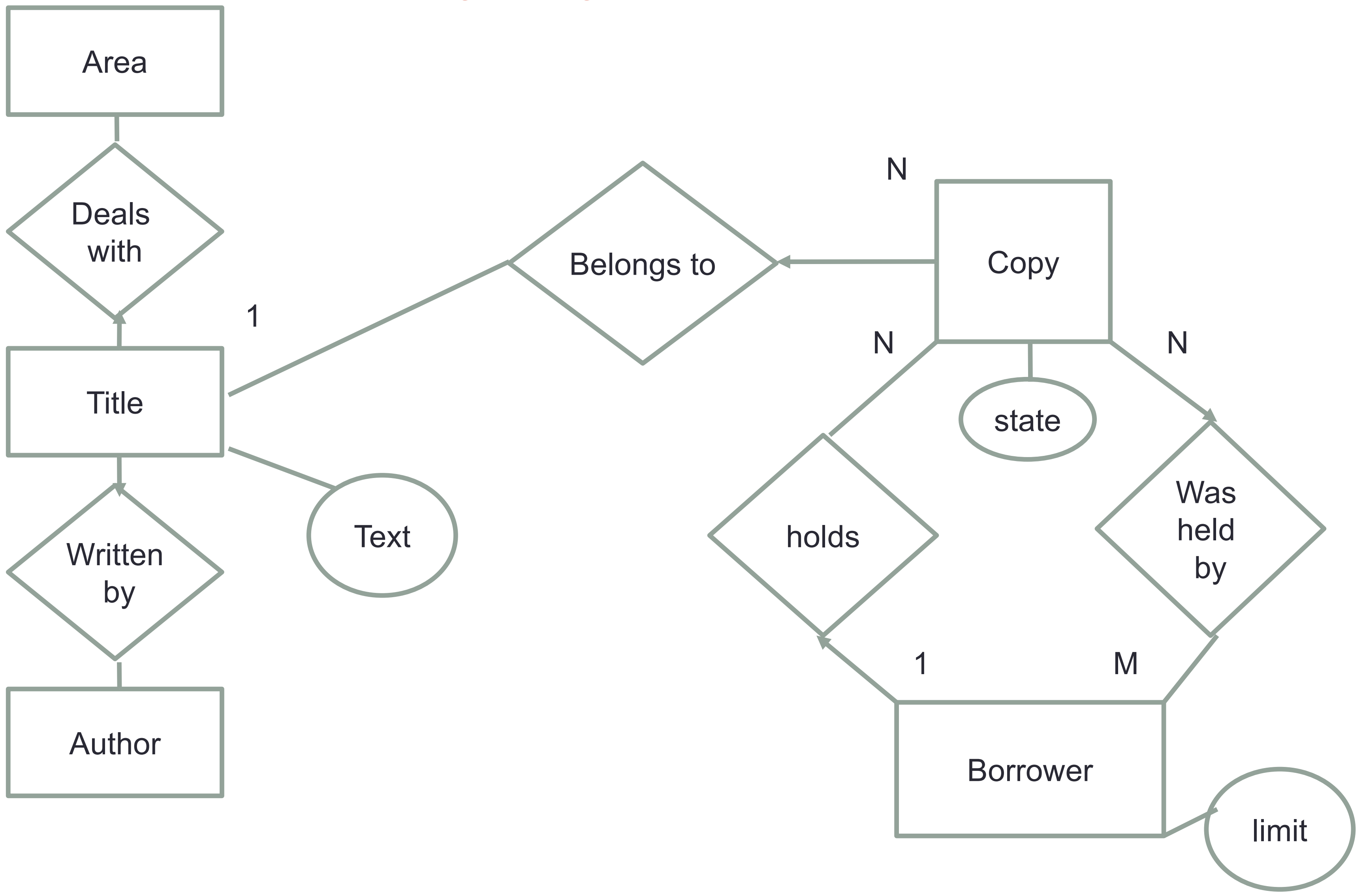


- ➢ Cardinality :

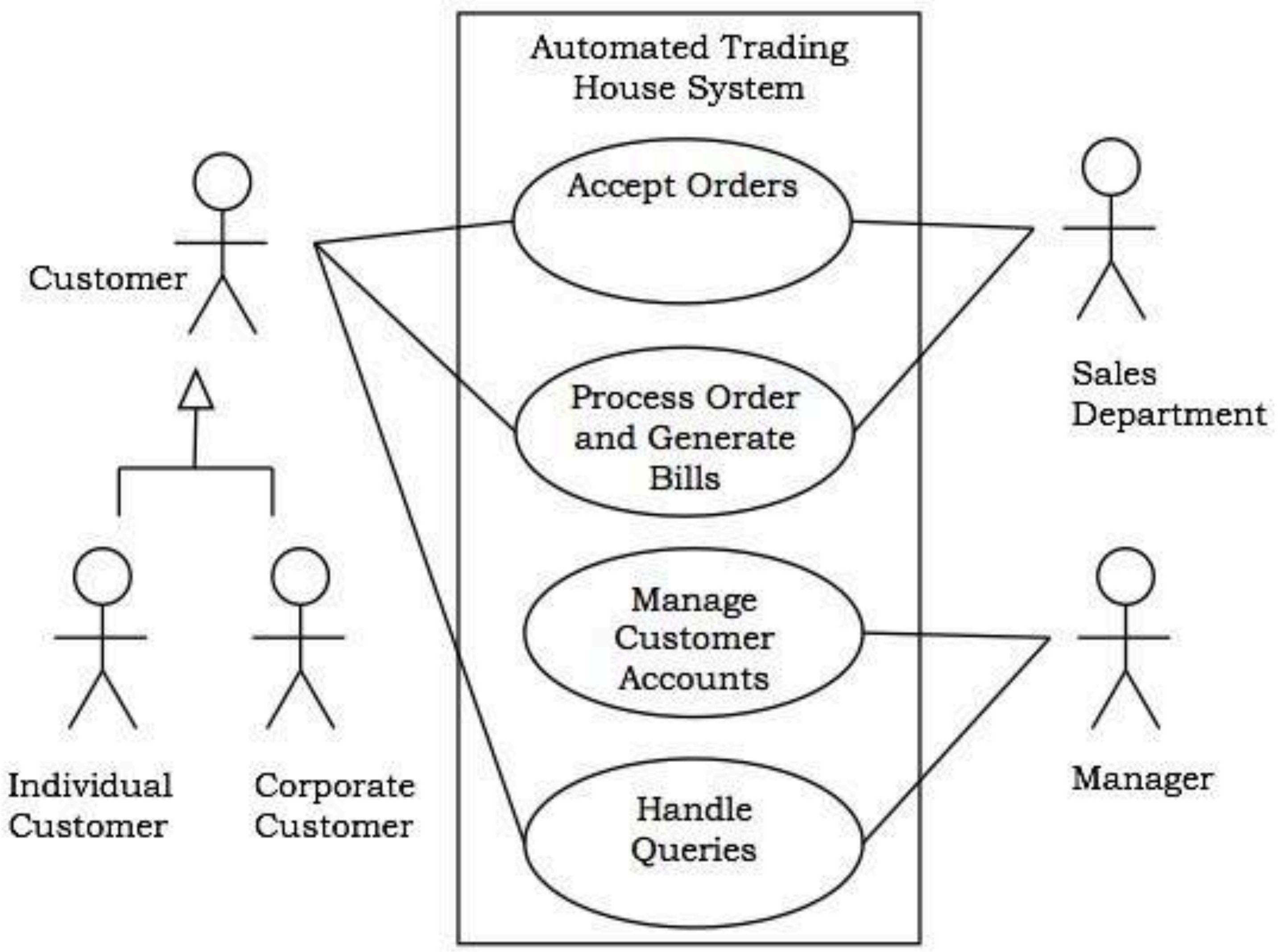
  1:1 one-to-one   1:N one-to-many M:N many-to-many



ERD of Library System



Use case diagram

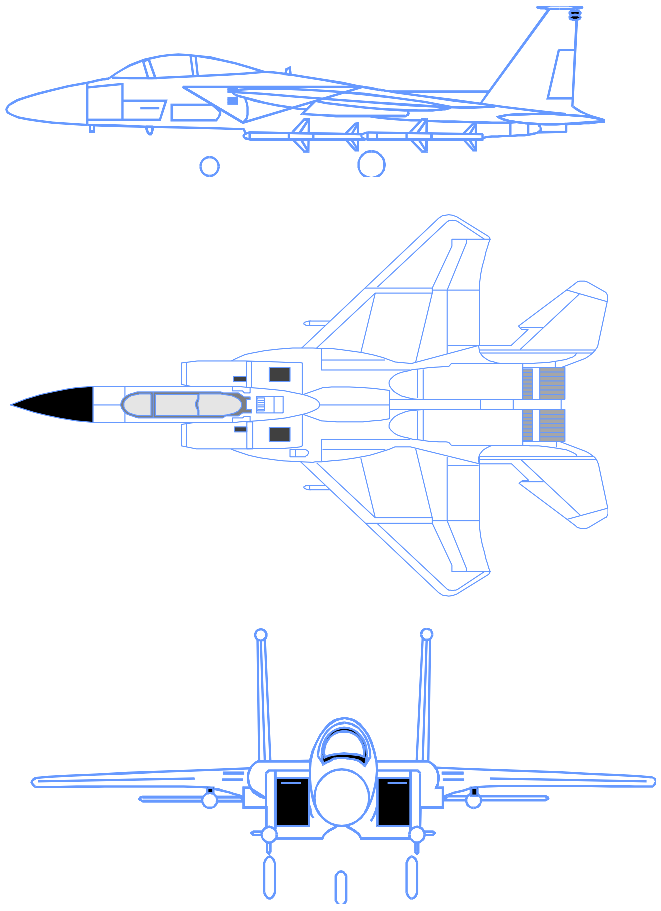


Content

1. What is Requirement?
2. Requirement Engineering
3. Unified Modeling Language (UML)

3.1. Modeling

- A model is a simplification of reality.



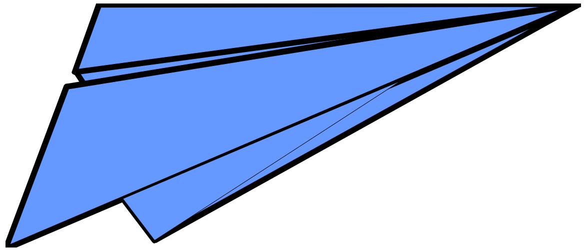
Why Model?

- Modeling achieves four aims [1]:
  - Helps you to “visualize a system as you want it to be”.
  - Permits you to “specify the structure or behavior of a system”.
  - Gives you “a template that guides you in constructing a system”.
  - “Documents the decisions you have made”.
- You build models of complex systems because you cannot comprehend such a system in its entirety.
- You build models to better understand the system you are developing.

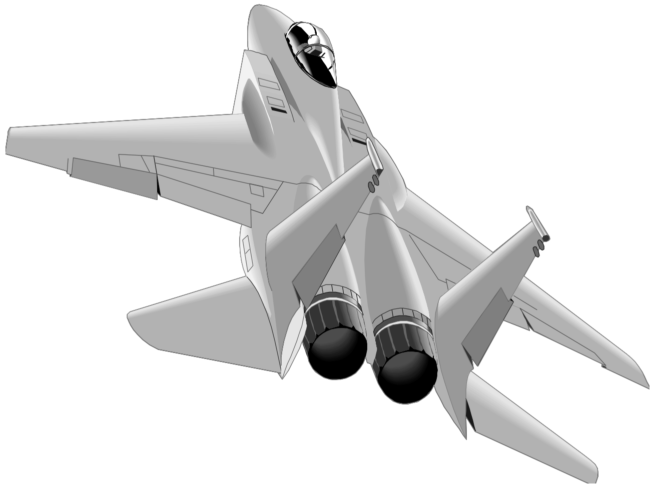
[1]: Chapter 1, Section 1.1

The Importance of Modeling

Less Important ⟷ **More Important**



Paper Airplane



Fighter Jet

Software Teams Often Do Not Model

- Many software teams build applications approaching the problem like they were building paper airplanes
    - Start coding from project requirements
    - Work longer hours and create more code
    - Lacks any planned architecture
    - Doomed to failure
- Modeling is a common thread to successful projects

3.2. Unified Modeling Language (UML)

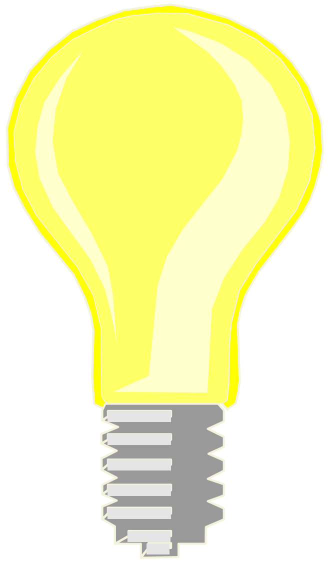
- “The UML is a language for
  - Visualizing
  - Specifying
  - Constructing
  - Documenting

  the artifacts of a software-intensive system”.

[1]: Chapter 2, Section 2.1

The UML Is a Language for Visualizing

- Communicating conceptual models to others is prone to error unless everyone involved speaks the same language.
- There are things about a software system you can't understand unless you build models.
- An explicit model facilitates communication.



The UML Is a Language for Specifying

- The UML builds models that are precise, unambiguous, and complete.

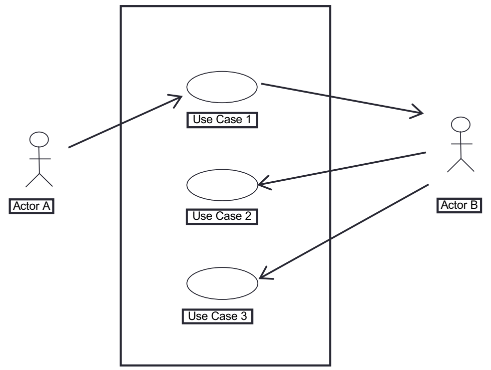
The UML Is a Language for Constructing

- UML models can be directly connected to a variety of programming languages.
  - Maps to Java, C++, Visual Basic, and so on
  - Tables in a RDBMS or persistent store in an OODBMS
  - Permits forward engineering
  - Permits reverse engineering

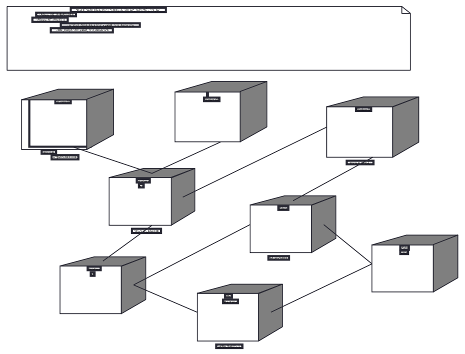
The UML Is a Language for Documenting

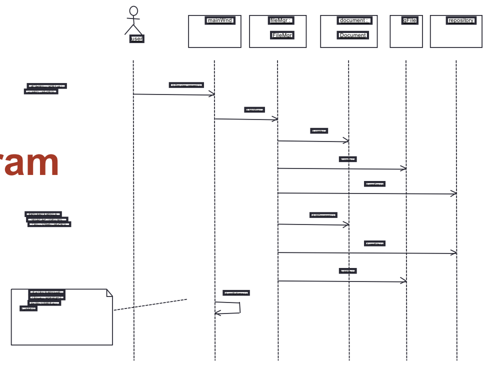
- The UML addresses documentation of system architecture, requirements, tests, project planning, and release management

Use Case Diagram

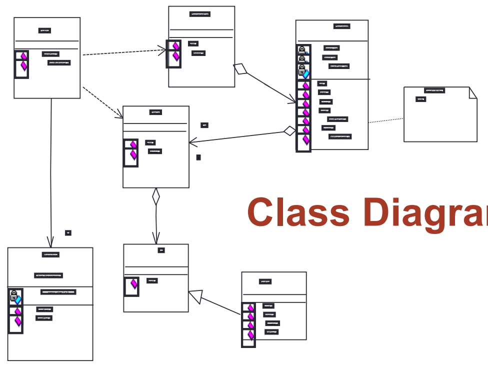


Deployment Diagram



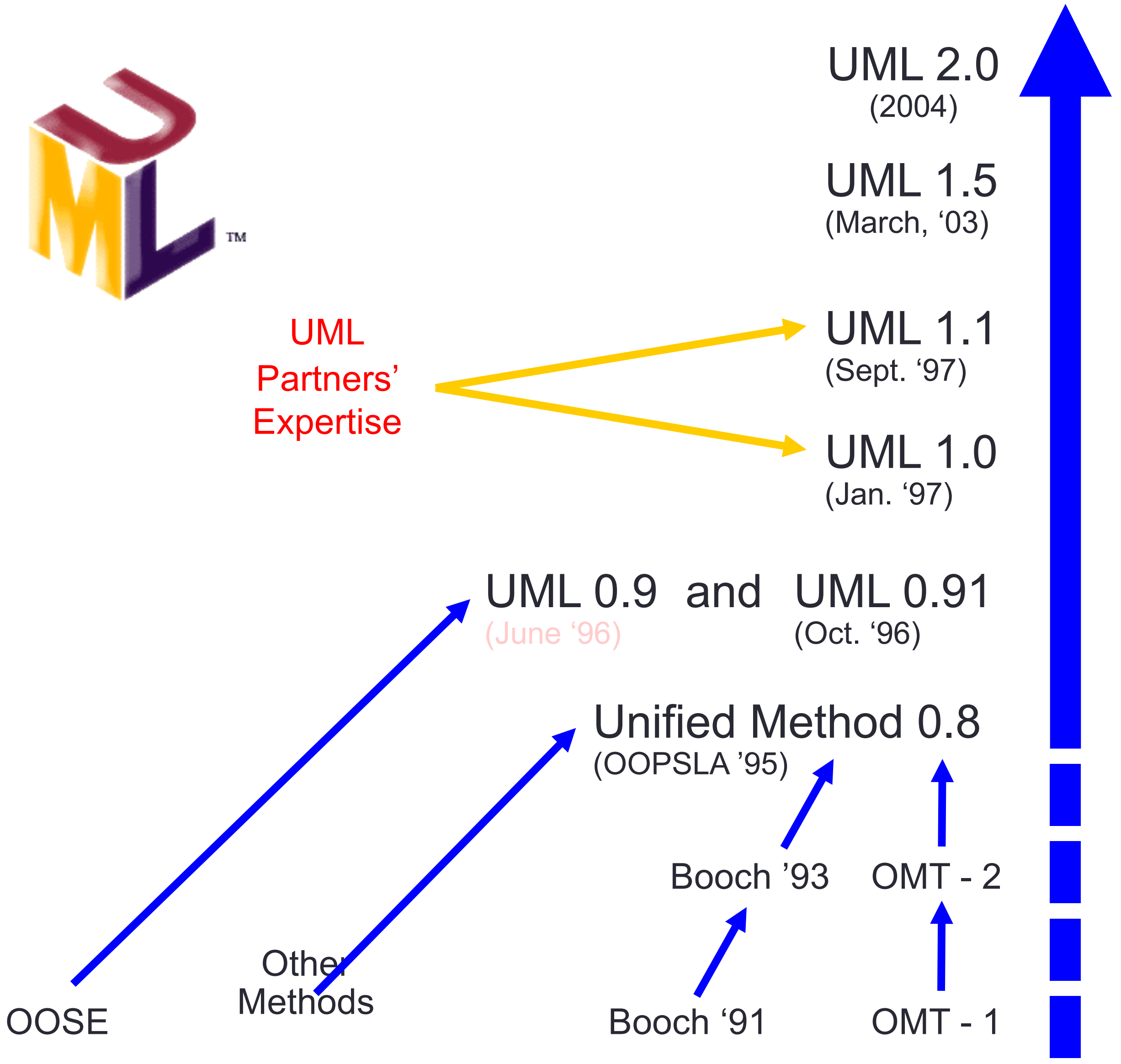


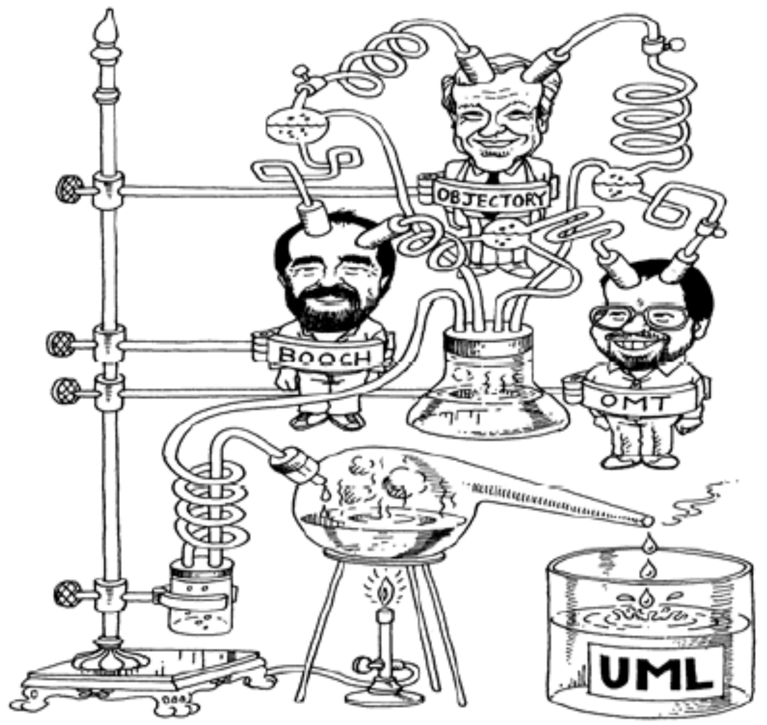
Sequence Diagram



Class Diagram

History of the UML



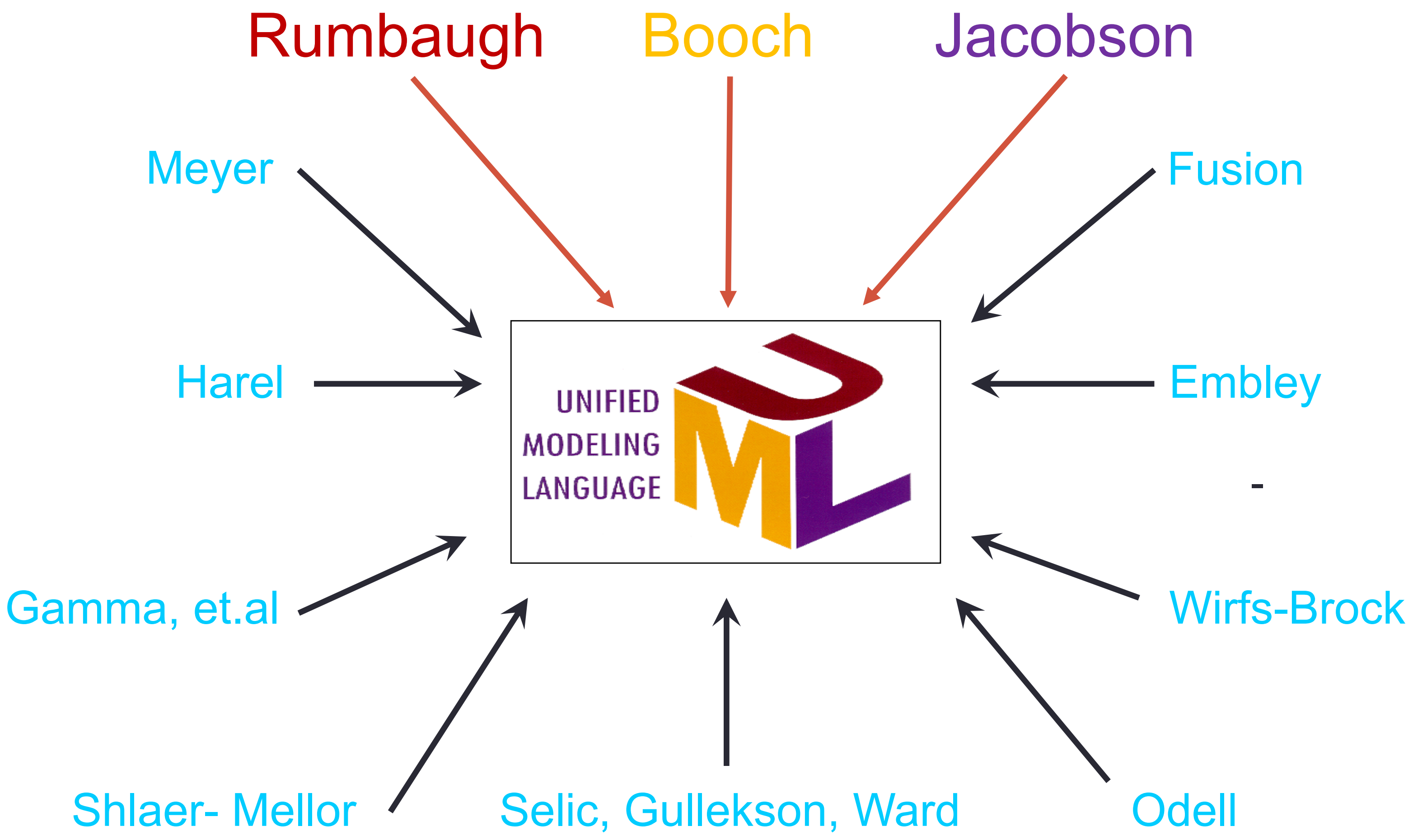


Booch91 (Grady Booch):
Conception, Architecture

OOSE (Ivar Jacobson): Use cases

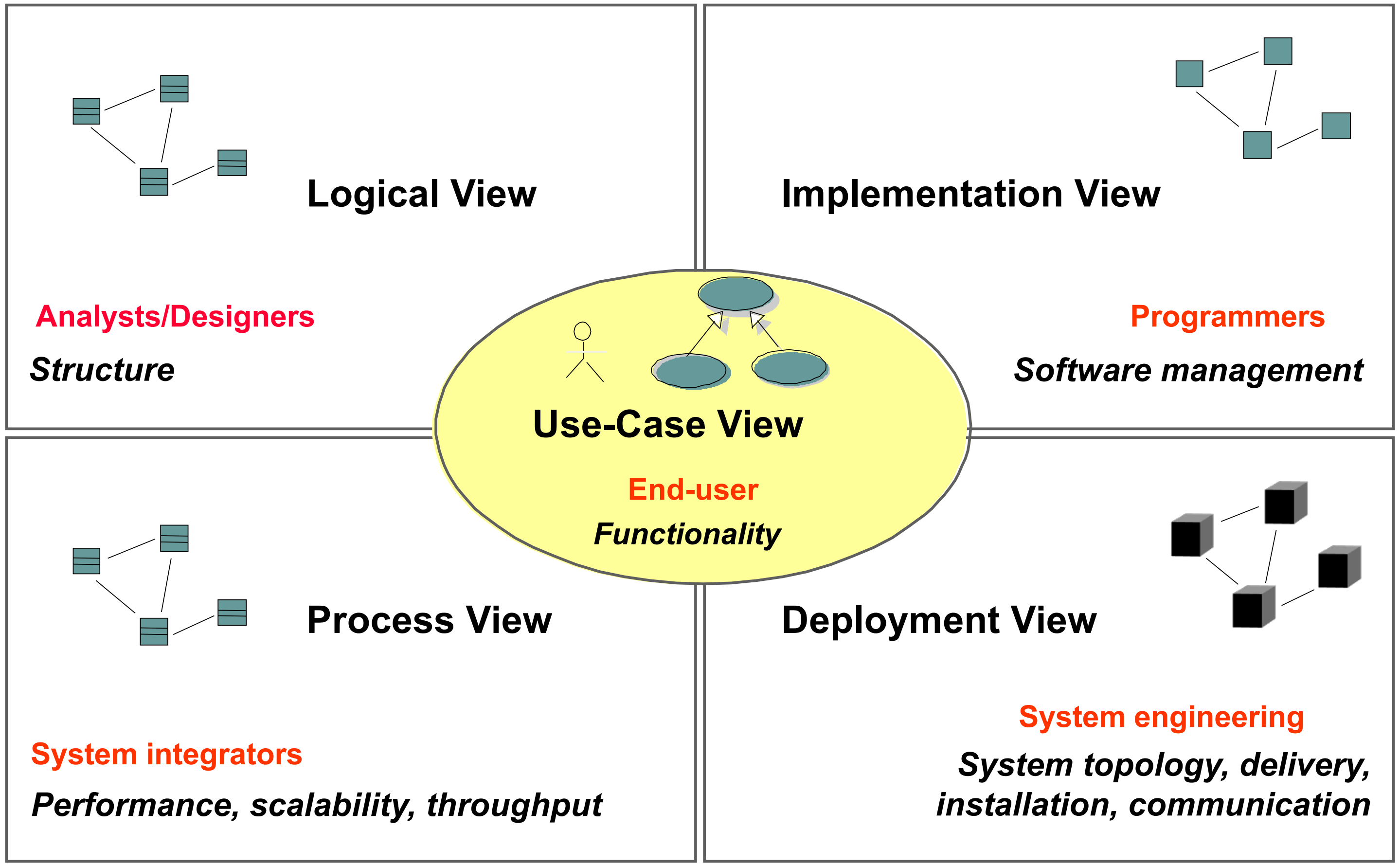
OMT (Jim Rumbaugh): Analysis

Inputs to the UML

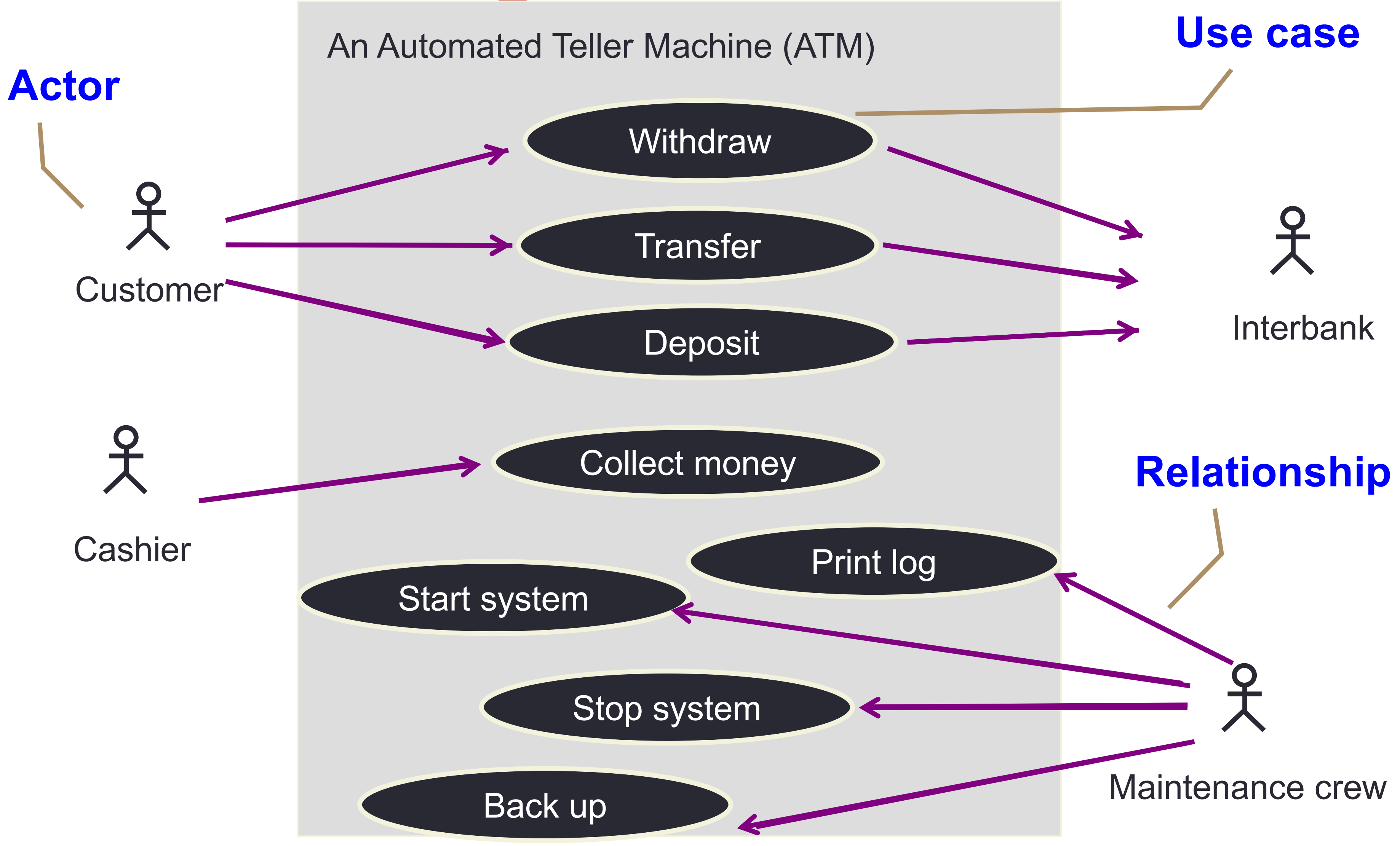


UML 4+1 Views

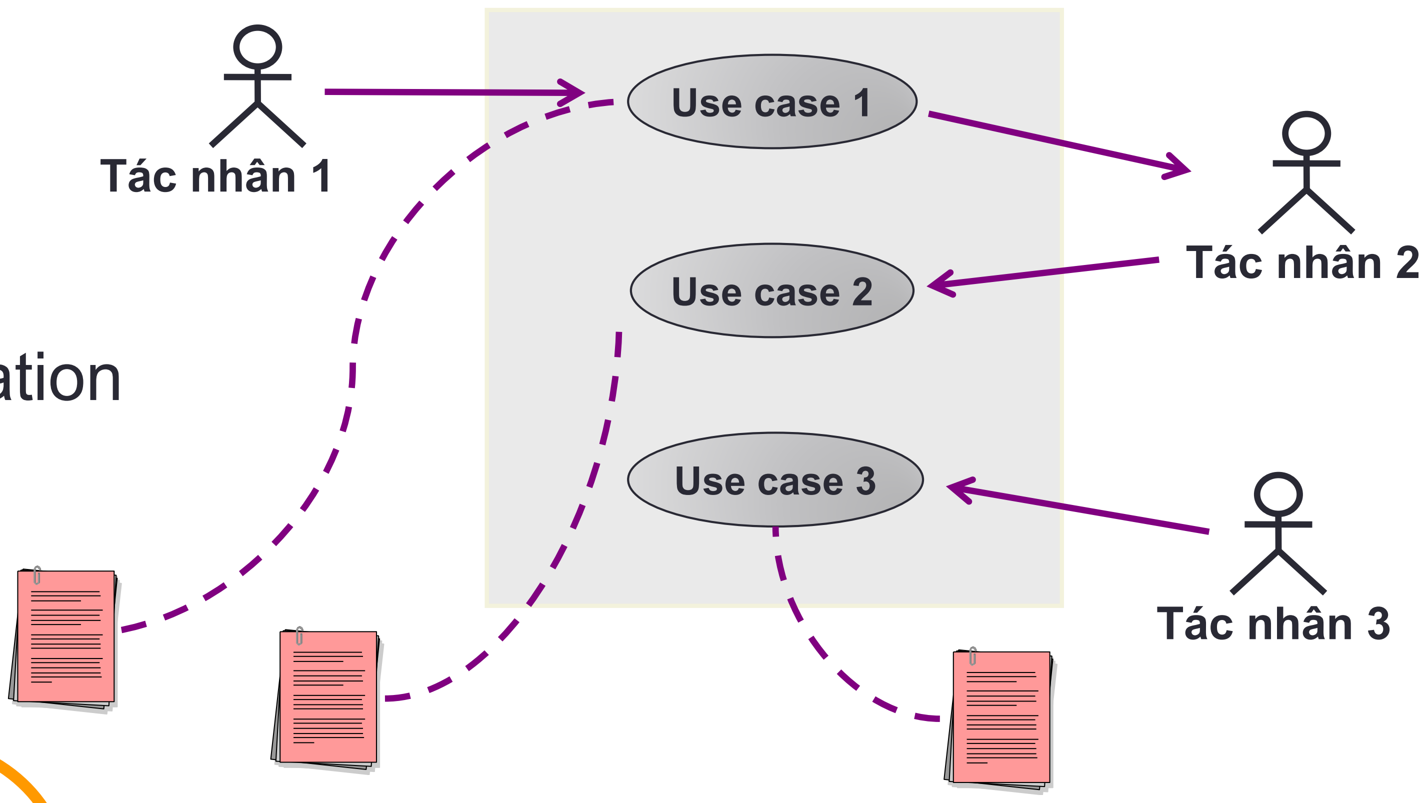
- “A view is simply a subset of UML modeling constructs that represents one aspect of a system



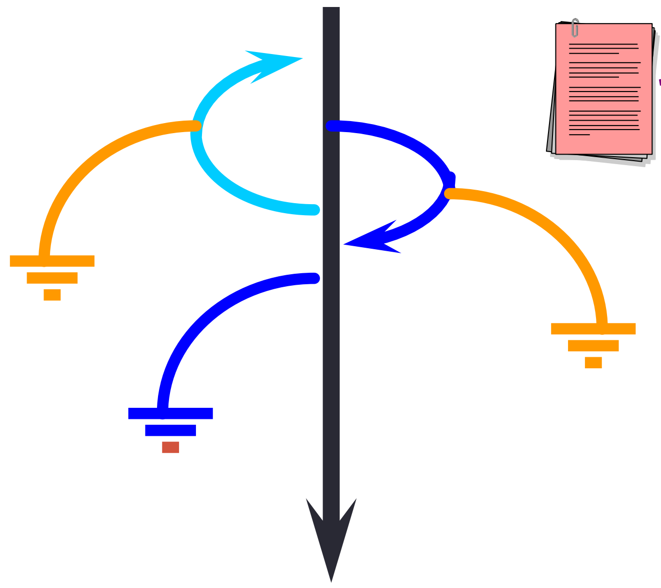
Use case diagram



Use-Case specification



- General Information
- Event flow



+ Regular variants
+ Odd cases
+ Error/exceptional flows

E.g. Specification for UC Log in

- Main flows of event: The use case starts when system prompts the Customer for a PIN Number. The Customer can now enter a PIN number. The Customer commits the entry. The system then checks this PIN number to see if it is valid. If valid, the system acknowledges the entry, thus ending the UC
- Regular variants: The Customer cancel a transaction at any time, thus restart the UC. No changes are made to the Customer’s account.
- Exceptional flow of events: The Customer clear a PIN number anytime before committing it and re-enter a new PIN number
- Exceptional flow of events: If the Customer enter an invalid PIN number, the UC restarts. If this happens 3 times in a row, the system cancel the entire transaction, preventing Customer from interacting with ATM for 60s