OOAD: Requirements

Presenter: Dr. Ha Viet Uyen Synh.

Part A

GENERAL SYSTEMS THEORY

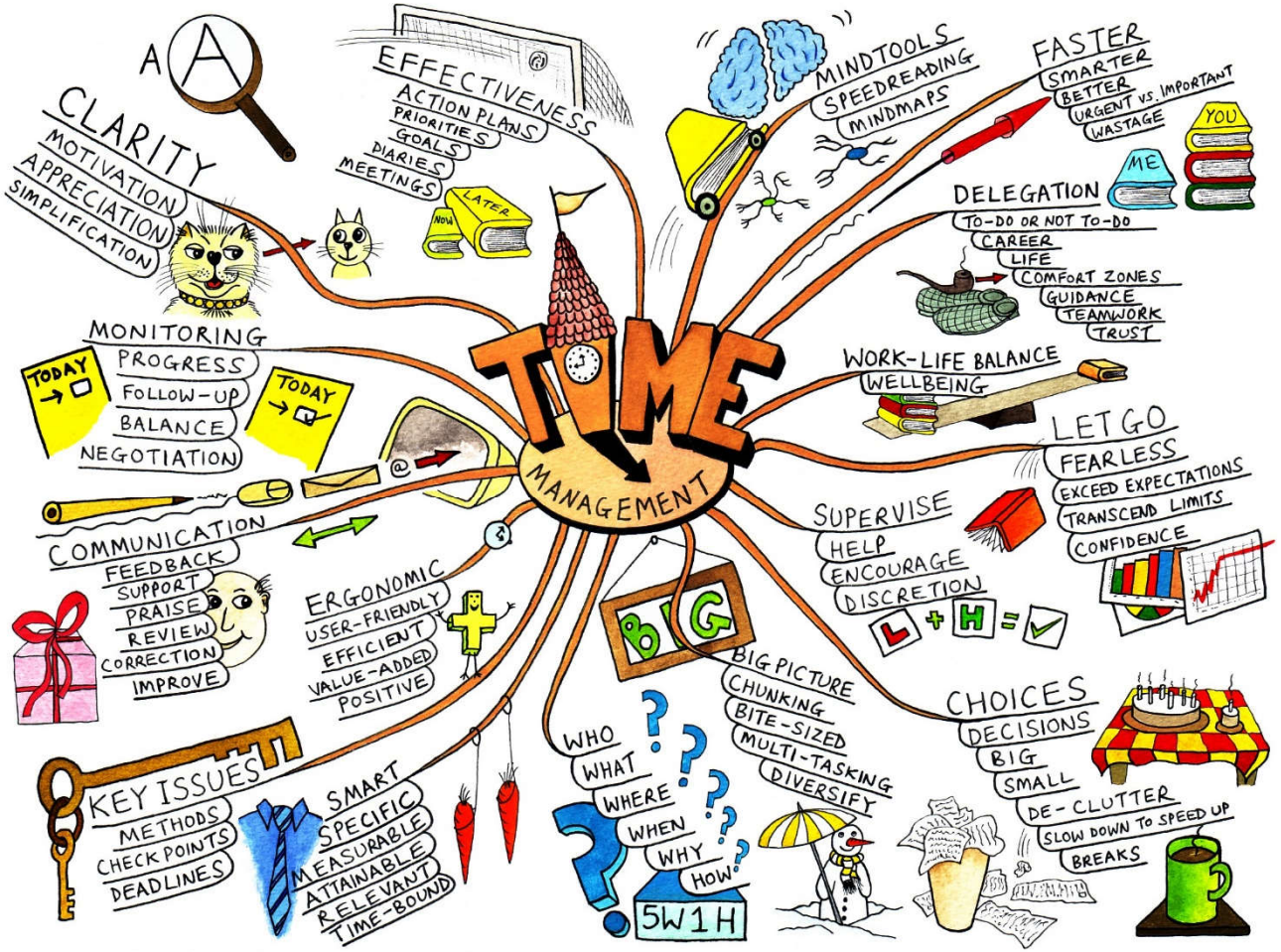
System's Concept

Def. A system is a set of components that interact with other ones and serve for a common purpose or goal.

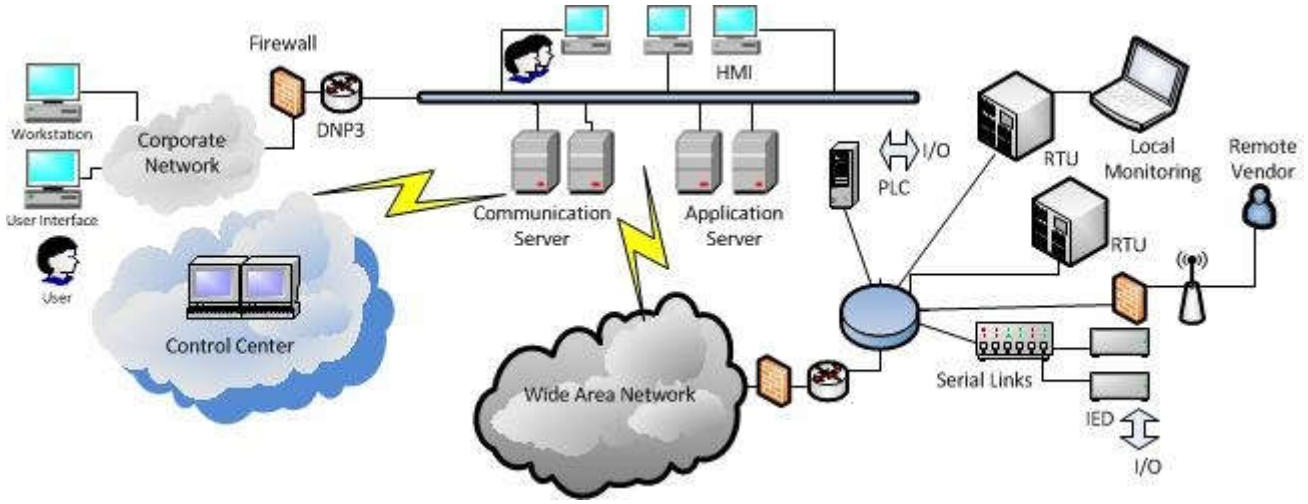
Systems may be: (1) abstract or (2) physical

- An abstract system is conceptual, a product of a human mind. That is, it cannot be seen or pointed to as an existing entity. However, they do exist and can be discussed, studied and analyzed.
- A physical system, in contrast, has a material nature. It is based on material basis rather than on ideas or theoretical notions.

Example _ an abstract system

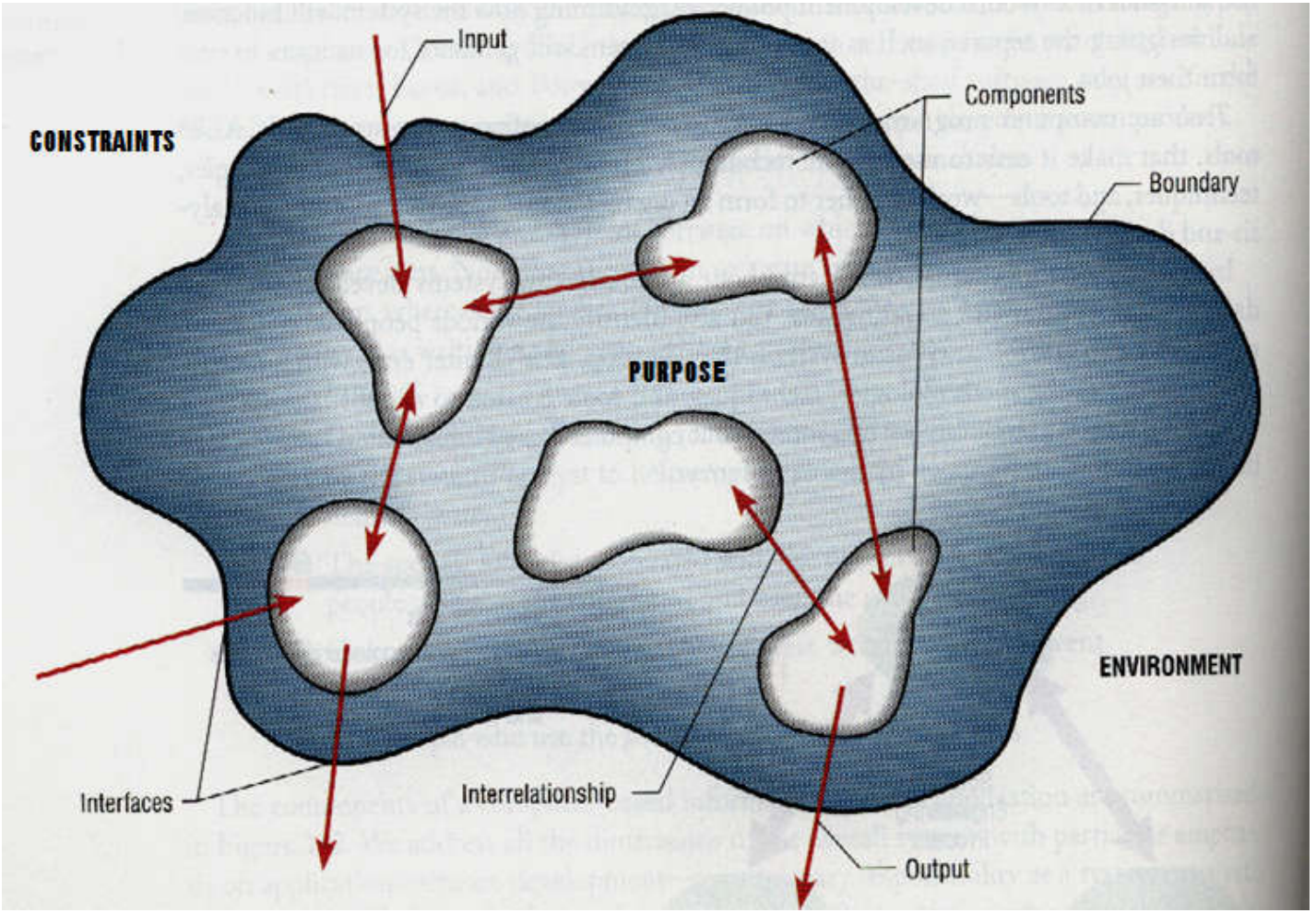


Example _ a physical system



SCADA System

Nine System's Characteristics



“Systems” Thinking

- Identify something as a system
- Involves being able to identify subsystems
- Identifying system characteristics and functions
- Identifying where the boundaries are (or should be)
- Identifying inputs and outputs to systems
- Identifying relationships among subsystems

Information Systems

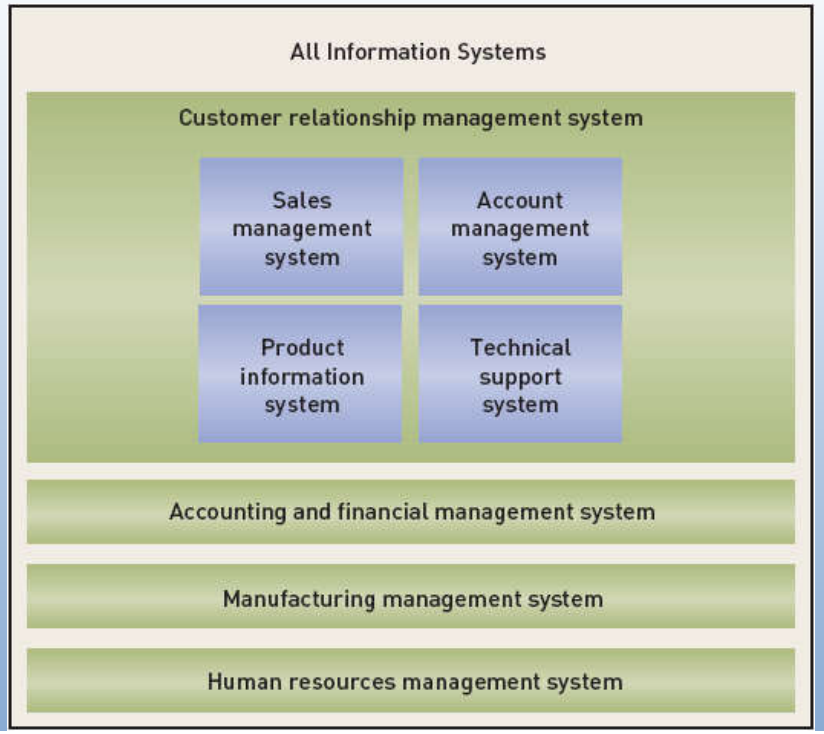
Def. An Information System (IS) is a collection of interrelated components that collect, process, store, and provide as output the information needed to complete a business task.

Example: A payroll system collects information on employees and their work, processes and stores that information, and than produces paychecks and payroll reports for the organization. Then information is provided to manufacturing so the department can schedule production.

Hierarchy of Information Systems

- *Hierarchy*: System, Subsystem, Supersystem
  Ex: customer support system: order entry subsystem, order fulfillment subsystem, shipping and back order subsystem, product catalog maintenance, etc.

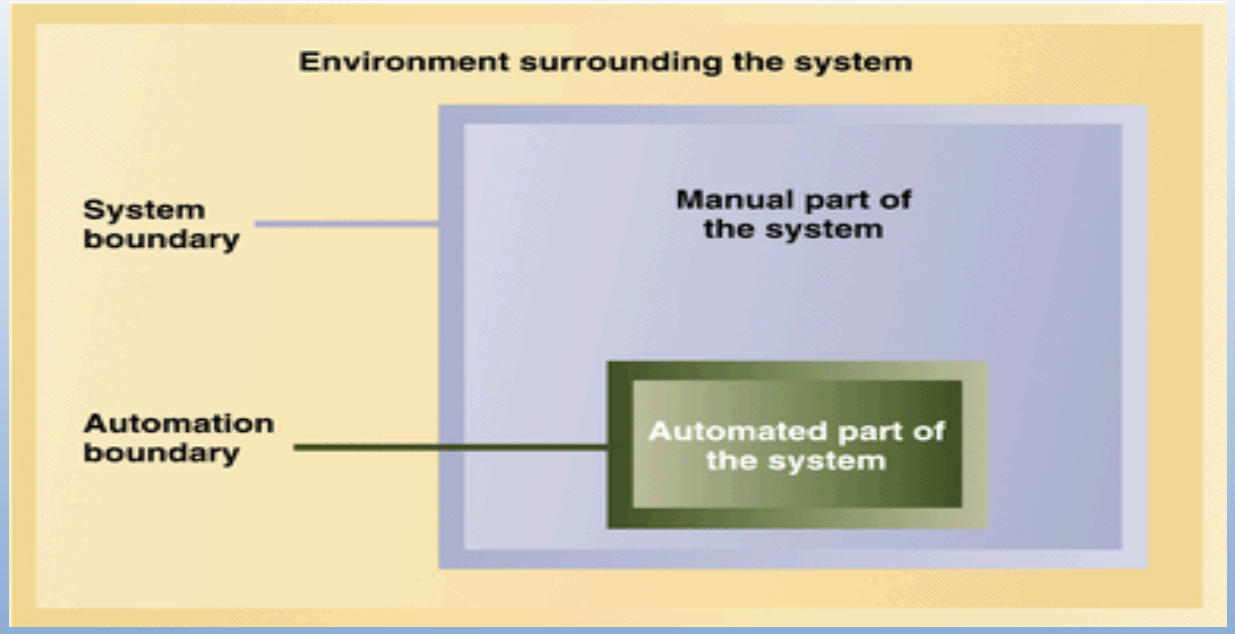
- *Supersystem* – a larger system that includes the system. The system is a subsystem of the larger supersystem
  Ex: production system includes other systems, such as inventory management and manufacturing and customer support system.



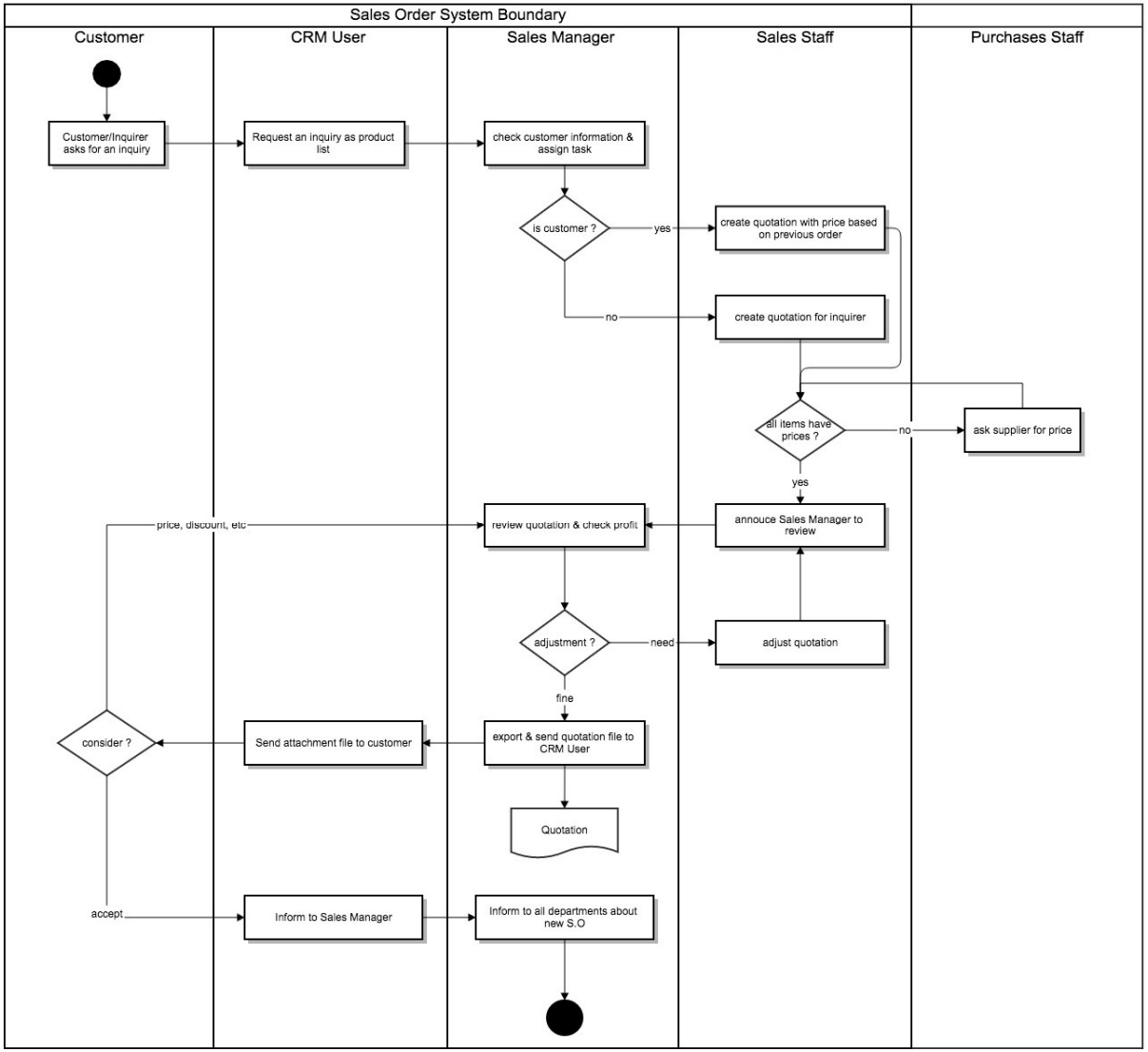
System Boundary vs. Automation Boundary

- System boundary: separates system from other systems and from its environment

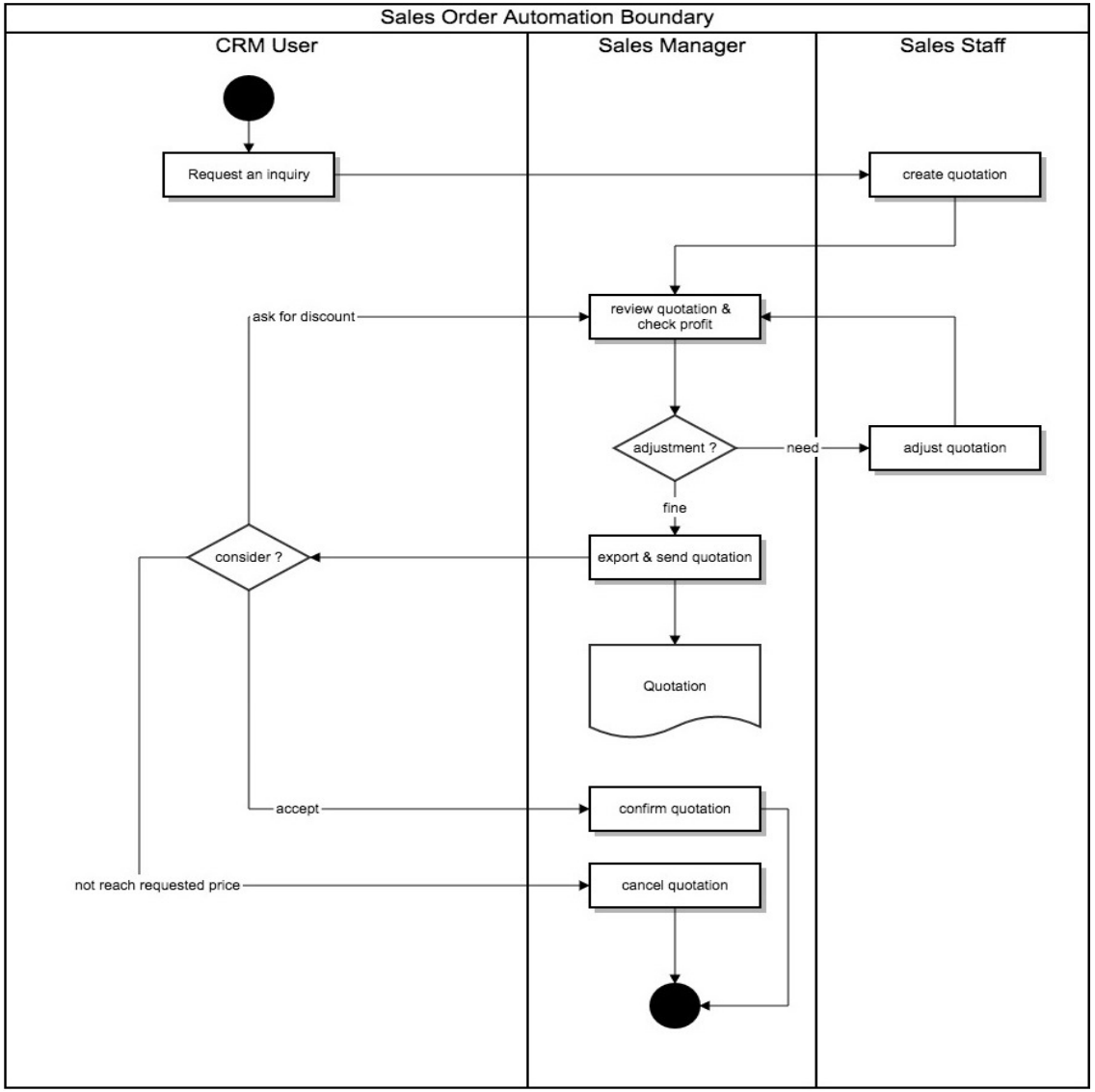
- Automation boundary - separates the automated part of the IS (where work is done by computers) from the manual part (where work is done by the people).



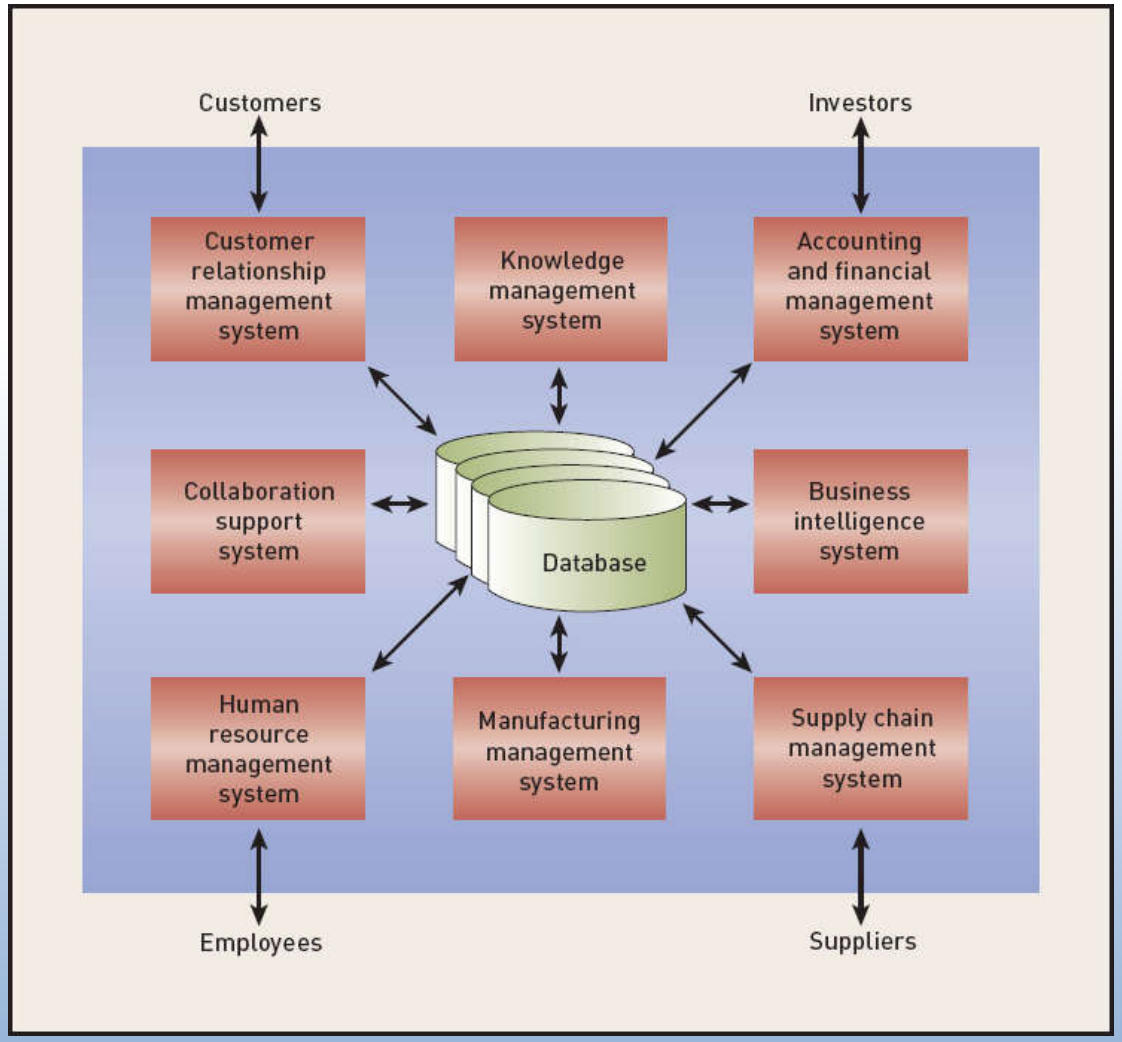
Example _ System Boundary



Example _ Automation Boundary



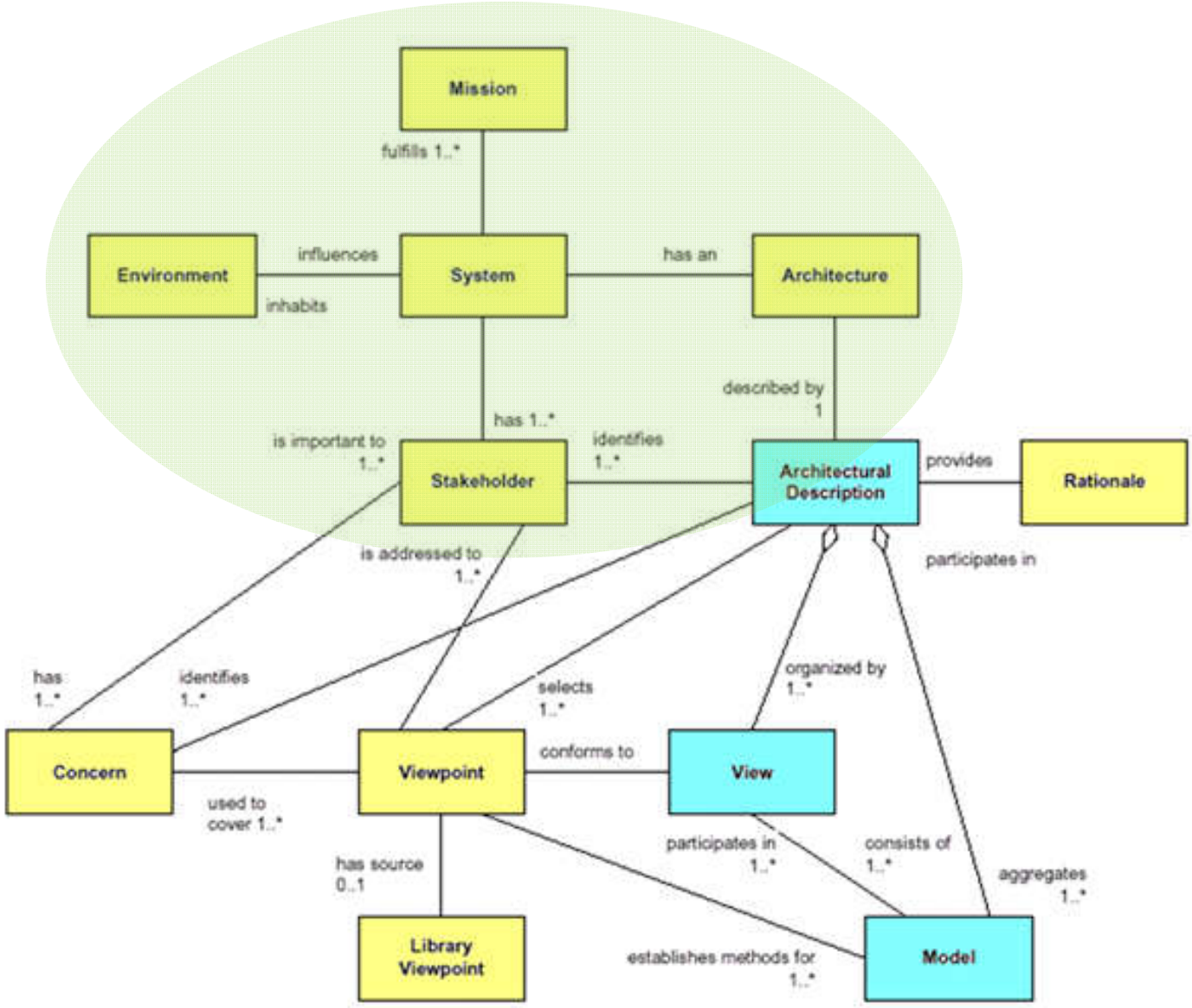
Types of Information Systems



Part B

REQUIREMENT ANALYSIS

Conceptual Model for SW Architectures



Requirements Analysis

Traditional SE suggests requirements analysis should remain unsullied by any consideration for a design

However, without reference to existing architectures it becomes difficult to assess practicality, schedules, or costs

In engineering, new products come from the observation of existing solution and their limitations

New Perspective on Requirements Analysis

Existing designs and architectures provide the solution vocabulary

Our understanding of what works now, and how it works, affects our wants and perceived needs

The insights from our experiences with existing systems helps us imagine what might work and enables us to assess development time and costs

→ Requirements analysis and consideration of design must be pursued at the same time

Software Architecture's Elements

A software system's architecture typically is not (and should not be) a uniform monolith

A software system's architecture should be a composition and interplay of different elements:

- Processing
- Data, also referred as information or state
- Interaction

Deployment

- A software system cannot fulfill its purpose until it is *deployed*
  - Executable modules are physically placed on the hardware devices on which they are supposed to run
- The deployment view of an architecture can be critical in assessing whether the system will be able to satisfy its requirements
- Possible assessment dimensions
  - Available memory
  - Power consumption
  - Required network bandwidth

Components

Elements that encapsulate processing and data in a system's architecture are referred to as *software components*

Definition

A *software component* is an architectural entity that

- encapsulates a subset of the system's functionality and/or data
- restricts access to that subset via an explicitly defined interface
- has explicitly defined dependencies on its required execution context

Components typically provide application-specific services

Connectors

In complex systems *interaction* may become more important and challenging than the functionality of the individual components

Definition

A *software connector* is an architectural building block tasked with effecting and regulating interactions among components

In many software systems connectors are usually simple procedure calls or shared data accesses

Connectors typically provide application-independent interaction facilities

Some types of Connectors

Procedure call connectors

Shared memory connectors

Message passing connectors

Streaming connectors

Distribution connectors

Wrapper/adaptor connectors

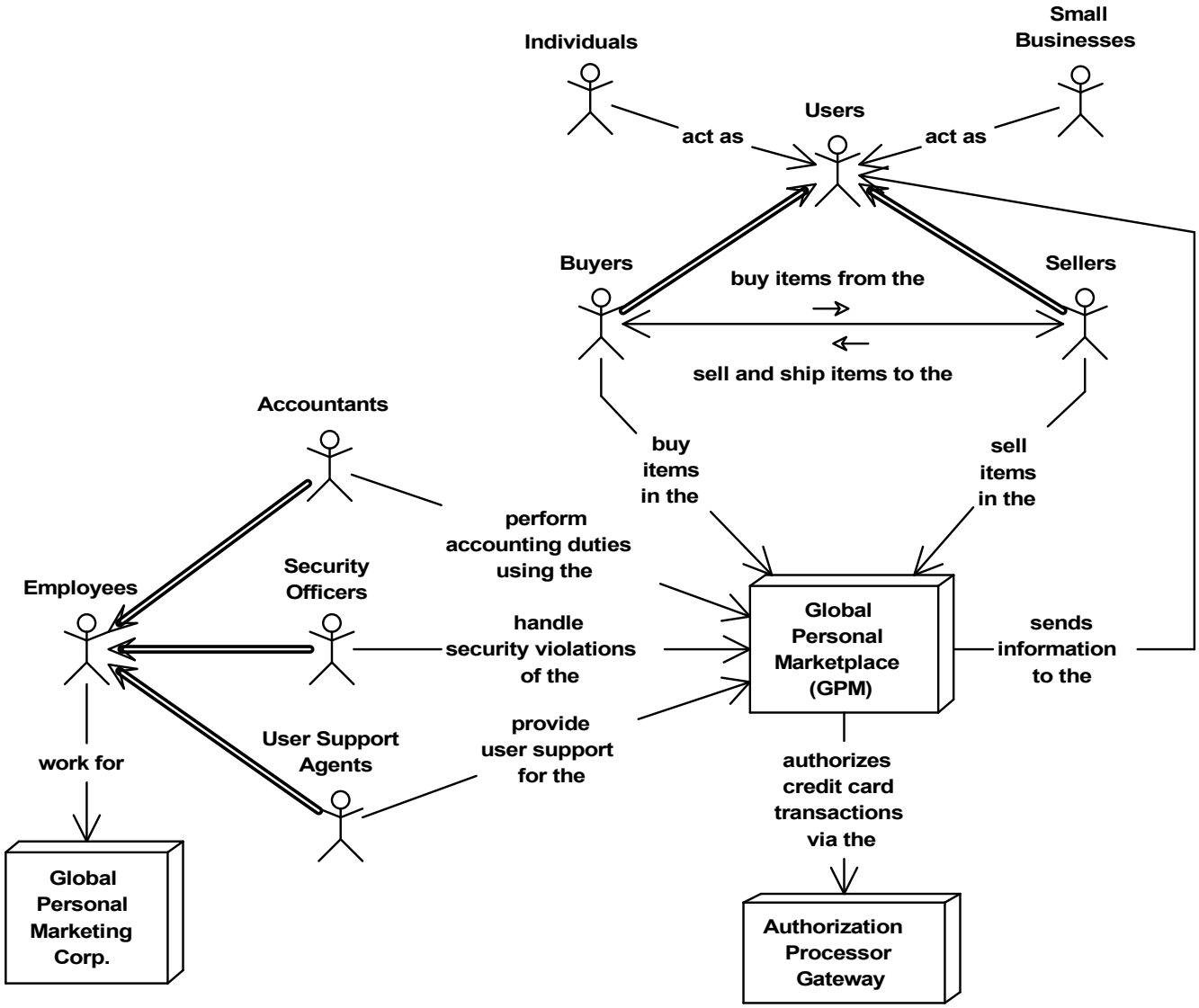
Configurations

Components and connectors are composed in a specific way in a given system's architecture to accomplish that system's objective

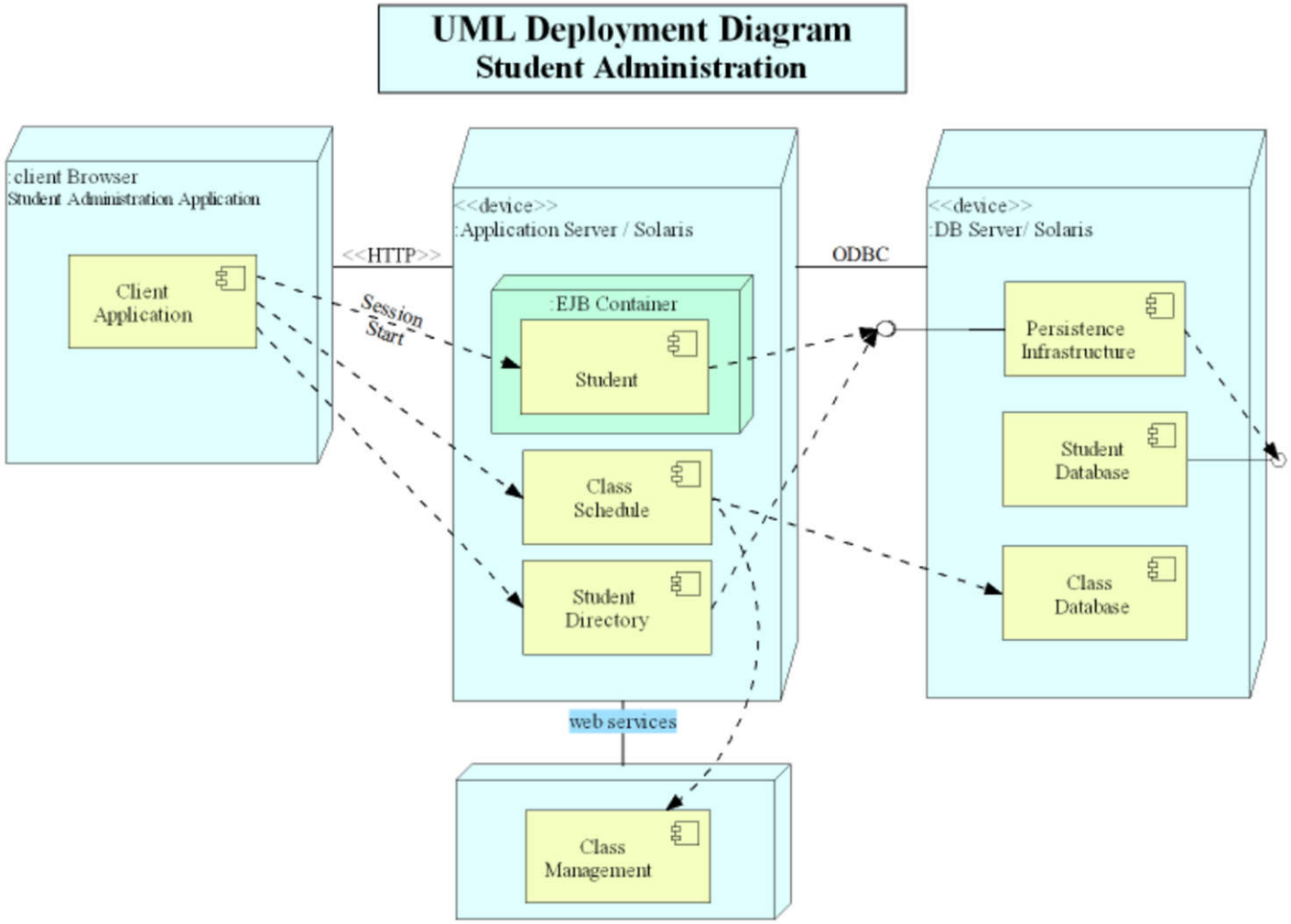
Definition

An *architectural configuration*, or topology, is a set of specific associations between the components and connectors of a software system's architecture

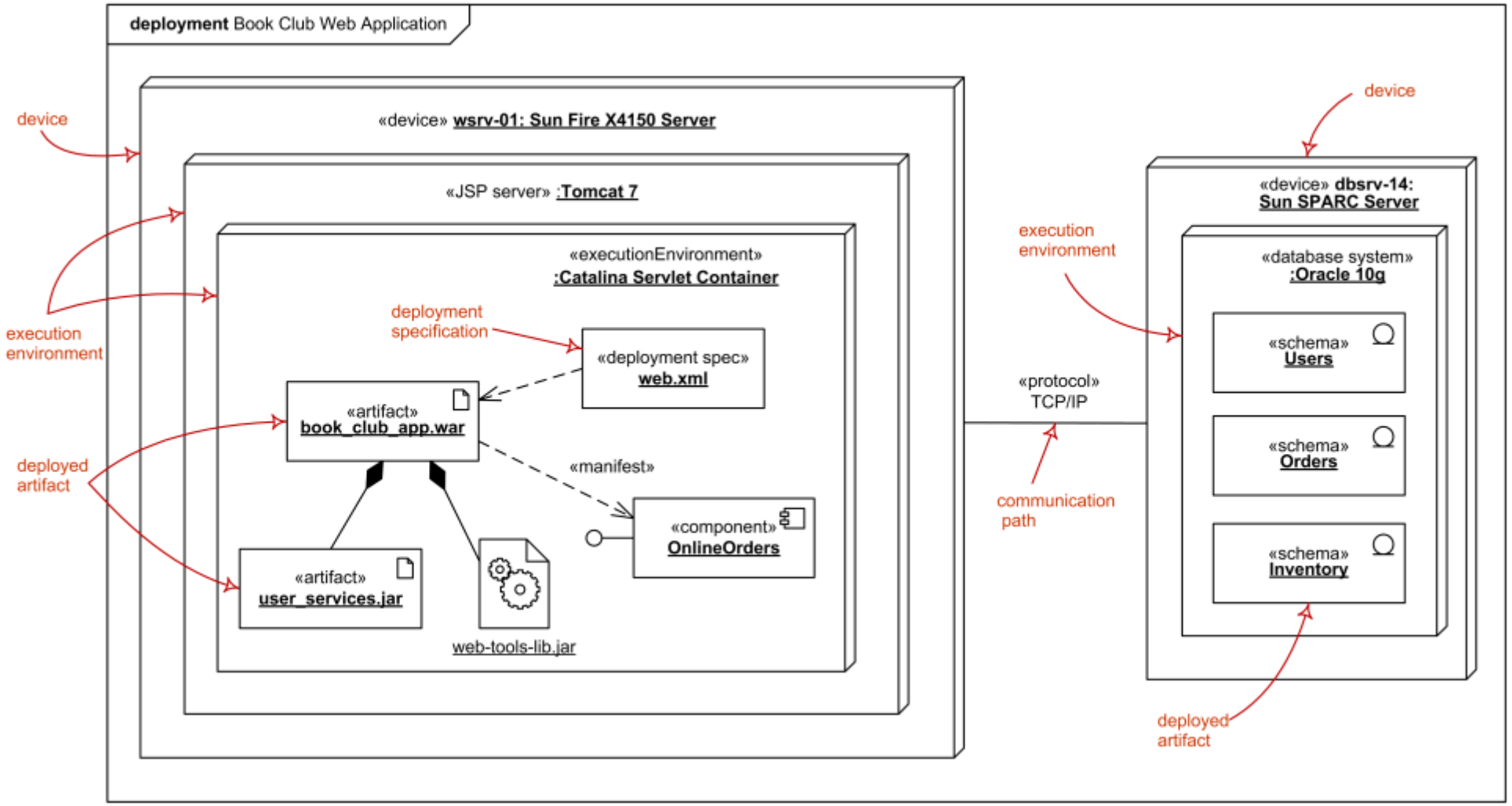
Example for Context Diagram



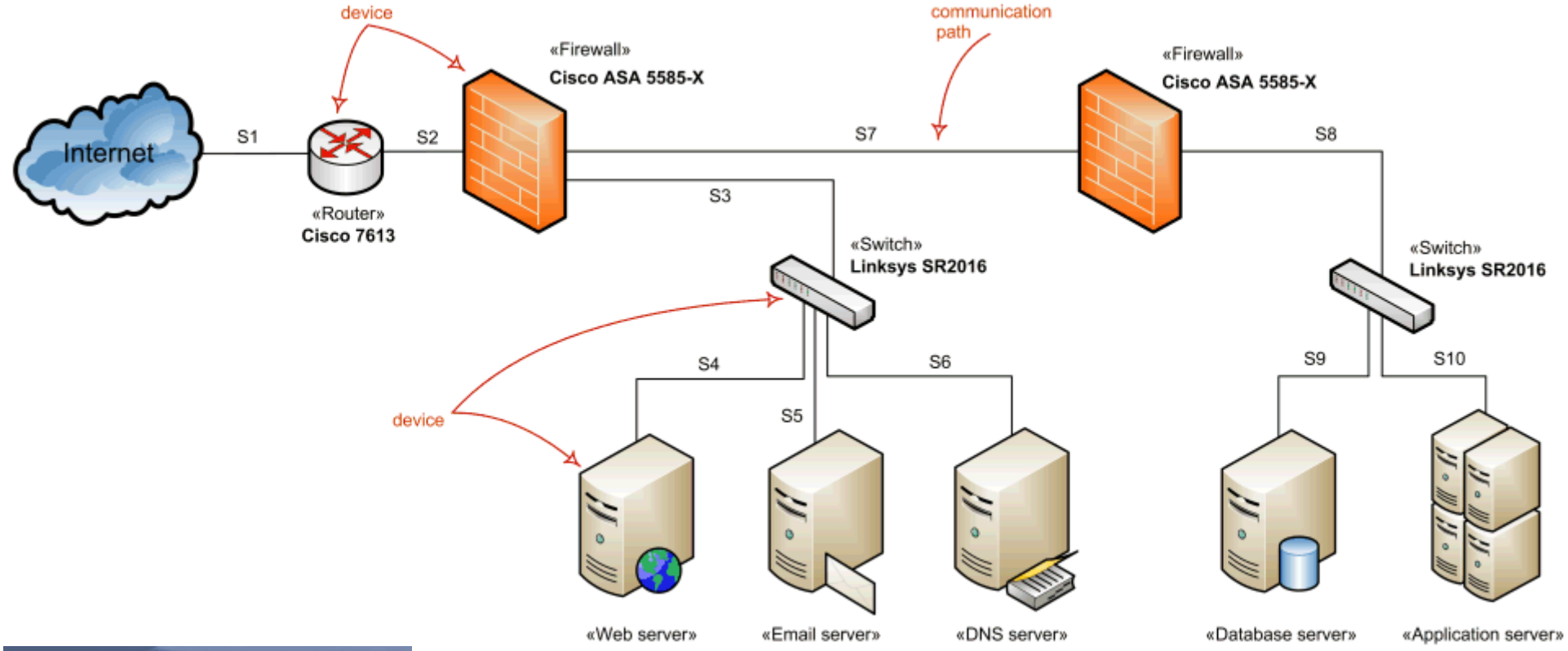
Example : Deployment Diagram



Deployment Diagram



Network Architecture Diagram



Architectural Models, Views, and Visualizations

Architecture Model

An artifact documenting some or all of the architectural design decisions about a system

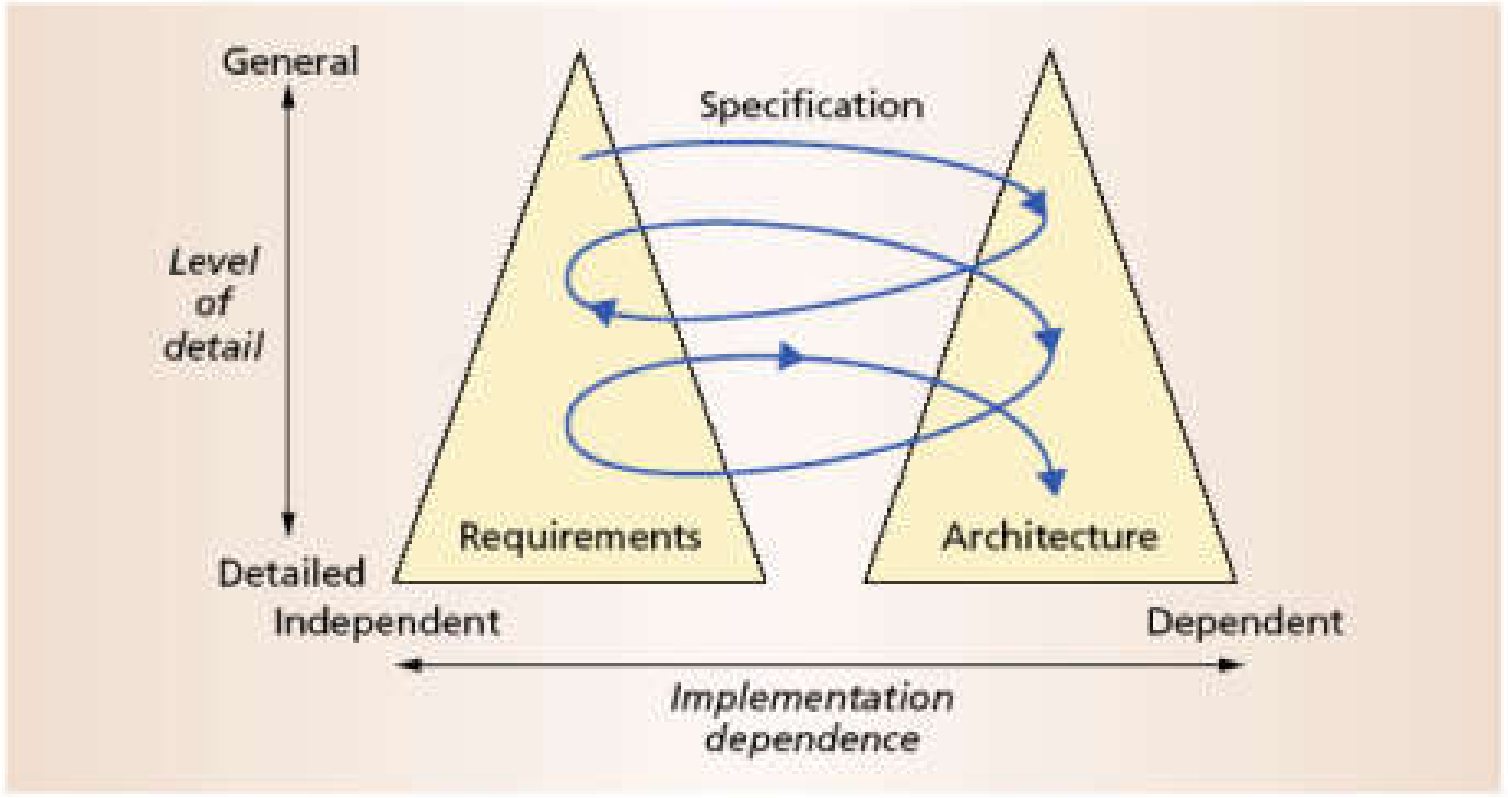
Architecture Visualization

A way of depicting some or all of the architectural design decisions about a system to a stakeholder

Architecture View

A subset of related architectural design decisions

The Twin Peaks Model



Part C

SOFTWARE REQUIREMENTS SPECIFICATIONS _SRS

Recommended document structure

1. Introduction
1.1 Purpose
1.2 Scope
1.3 Definitions, acronyms, and abbreviations
1.4 References
1.5 Overview
2. Overall description
2.1 Product perspective
2.2 Product functions
2.3 User characteristics
2.4 Constraints
2.5 Assumptions and dependencies
3. Specific requirements
Appendixes
Index

1. Introduction

Review business needs

- Use strategic plan documents
- Consult key users
- Develop list of expected business benefits (i.e., results organization expects to achieve from the new IS)

- Identify expected system capabilities (at a general level)
- Define scope in terms of requirements

Create system scope document (3 components: problem description, business benefits, system capabilities)

Build proof of concept prototype (if new technology or new solutions) to show they are feasible and possible

Create context diagram (scope of the IS): an IS, external entities and input/output information flows

Example of purpose and scope

1.1 Purpose

This document specifies the Software Requirements Specification (SRS) for the Project Management System (PMS). It describes scope of the system, both functional and non-functional requirements for the software, design constraints and system interfaces.

1.2 Scope

The Project Management System addresses the management of software projects. It provides the framework for organizing and managing resources in such a way that these resources deliver all the work required to complete a software project within defined scope, time and cost constraints.

The system applies only to the management of software projects and is a tool that facilitates decision making; the PMS does not make decisions.

This SRS describes only required functionality of PMS, not the functionality of external systems like data storage, change management or version control systems.

This document does not divide the PMS into subsystems; it describes only requirements for the whole system functionality which is defined in the use case model.

Definitions, acronyms, abbreviations

Define all terms, acronyms, and abbreviations required to properly interpret the SRS

Example of definitions

1.3 Definitions, Acronyms and Abbreviations

The following table explains the terms and abbreviations used in the document.

Term/Abbreviation	Explanation
PMS	Project Management System
CMS	Change Management System (Bug tracking tool)
CVS	Concurrent Versions System
VSS	Microsoft Visual SourceSafe
PERT	Program Evaluation and Review Technique
GUI	Graphical User Interface
LAMP	A server that is running Linux, Apache, My-SQL and PHP
DBMS	Database Management System
DSS	Data Storage System
RBAC	Role Based Access Control

1.4 Glossary

The glossary defines the key terms and concepts mentioned and used in this SRS.

Word	Explanation
Project Management System	The subject of this document. Represents the whole solution as aggregate of all subsystems and interfaces.
Host System	The main part of the system that resides on the server and where the business logic runs. Maintains physical connections to all external systems (data storage system, version control and change management systems)

2. Overall description

Product perspective:

Describe relation with other products if any.

Examples:

- ☐System interfaces
- ☐User interfaces
- ☐Hardware interfaces
- ☐Software interfaces
- ☐Communications interfaces
- ☐Memory
- ☐Operations
- ☐Site adaptation requirements

Example of product perspective

2.2 Product perspective

PMS it a standalone system that provides functionality described in the Product functions section. It includes all subsystems needed to fulfil these software requirements. In addition, the PMS has interfaces to the external systems, such Version Control System, Change Management and Bug Tracking System and Payroll System. These interfaces shall be implemented according to available industry standards and shall be independent from a specific external system.
Any detailed definition of an external system is out of scope of this document.
The figure 1 shows the decomposition of PMS on the functionality areas and the supported external systems.
We have to distinguish a Data Storage System (DSS) from all other external systems in that way, that Data Storage System enables normal functioning of PMS and is therefore essential. PMS stores all its data in the DSS and hence has to maintain the connection to it. PMS shall access the data storage system through standard interface (JDBC, ODBS, ADO etc). See Data storage system section for more information.

Constraints

Describe any properties that will limit the developers' options

Examples:

- ☐Regulatory policies
- ☐Hardware limitations (e.g., signal timing requirements)
- ☐Interfaces to other applications
- ☐Parallel operation
- ☐Audit functions
- ☐Control functions
- ☐Higher-order language requirements
- ☐Reliability requirements
- ☐Criticality of the application
- ☐Safety and security considerations

3. Specific requirements

This section brings requirements to a level of detail making them usable by designers and testers.

Examples:

- ☐Details on external interfaces
- ☐Precise specification of each function
- ☐Responses to abnormal situations
- ☐Detailed performance requirements
- ☐Database requirements
- ☐Design constraints
- ☐Specific attributes such as reliability, availability, security, portability

Possible section 3 structure

3. Specific requirements
3.1 External interfaces
    3.1.1 User interfaces
    3.1.2 Hardware interfaces
    3.1.3 Software interfaces
    3.1.4 Communication interfaces
3.2 Functional requirements
…
3.3 Performance requirements
…
3.4 Design constraints
…
3.5 Quality requirements
…
3.6 Other requirements
…

Example of functional requirements

Requirement ID	R1.01.01
Title	Main Functionality\Users
Description	The system shall support the concept of **user**. Every user of the system has a name and a password. The name must be unique within the installed instance of the system. In addition, every user has a set of properties: *Full Name*, *Full Business Title* (Company Name, Position), *E-Mail Address, Phone, Working Address, Alternative Phone, and Alternative Working Address*. Each user is uniquely identified by its name within the system.
Priority	1
Source	
Risk	C
References	

Requirement ID	R1.01.04
Group	Main Functionality\User Roles\Predefined Roles
Description	The default installation of the system shall provide at least the following preconfigured user roles: "*Manager*", "*Team Leader*", "*Team Member*", "*Administrator*". The Table 3 lists the default rights of each role. The system administrator (user with the right to edit user roles) can configure permissions of the roles.
Priority	2
Source	
Risk	M
References	

Priority:
1: first version
2: final version
>3: optional

Risk:
C: critical
H: high impact
M: medium imp.
L: low impact

Example of non-functional requirements

Requirement ID	R5.01.03
Group	Performance\Start-up time
Description	Under the condition that the host system fulfils the hardware requirement R13.01.01, the time between initiation of the system startup and availability of full system functionality must be not longer 10 minutes.
Priority	1
Source	
References	R13.01.01

Requirement ID	R6.02.01
Group	Deployment\Upgrade
Description	The upgrade of the system must be a particular case of the installation and fulfill the same requirements. The upgrade shall preserve all user data: projects, tasks, resources, project portfolios.
Priority	1
Source	
References	R6.01.01

Requirement ID	R13.01.02
Group	Hardware\Client system
Description	The client part of the PMS shall be able to run and fulfill the performance requirements on: Single Pentium 1.8 GHz, 1 GB RAM, 1 GB disk space. LAN bandwidth: 1 Gbps; WAN bandwidth: 2 Mbps; minumum screen resolution 1024x768
Priority	1
Source	
References	3.6 Performance

Some recipes for good requirements

Managerial aspects:

- ☐Involve all stakeholders
- ☐Establish procedures for controlled change
- ☐Establish mechanisms for traceability
- ☐Treat requirements document as one of the major assets of the project; focus on clarity, precision, completeness

Technical aspects: how to be precise?

- ☐Formal methods?
- ☐Design by Contract

Checklist

Premature design?

Combined requirements?

Unnecessary requirements?

Conformance with business goals

Ambiguity

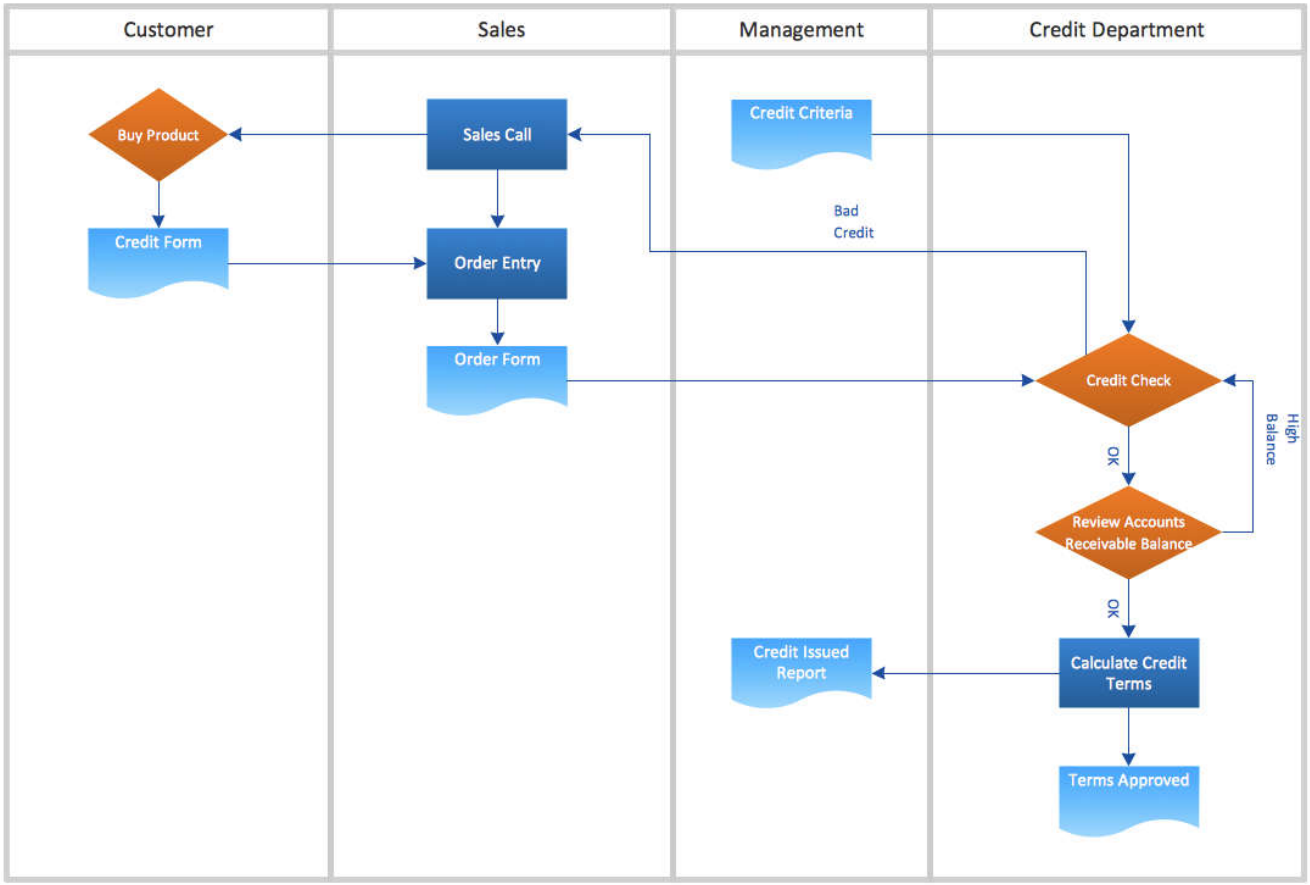
Realism

Testability

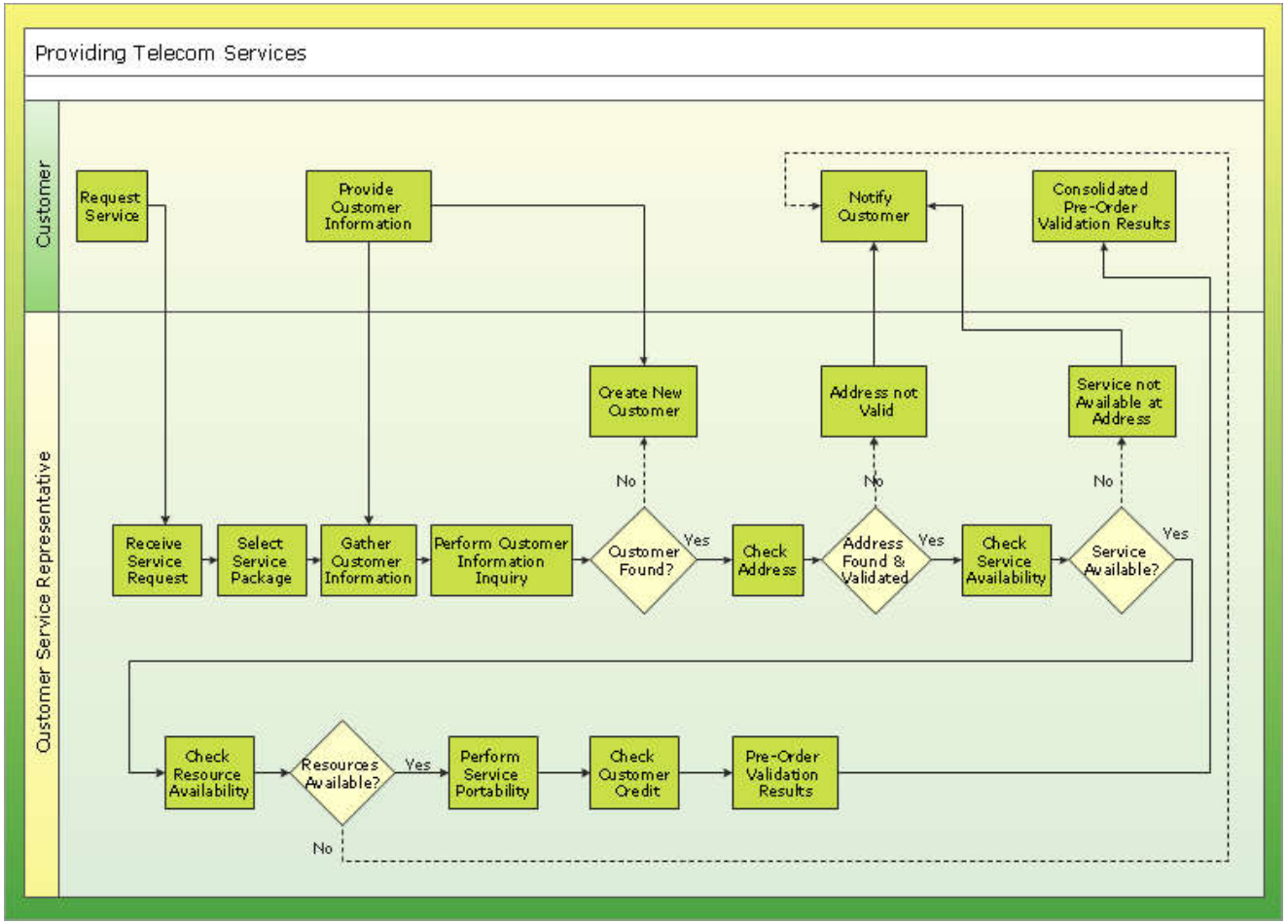
Any Questions?

✉ hvusynh@hcmiu.edu.vn

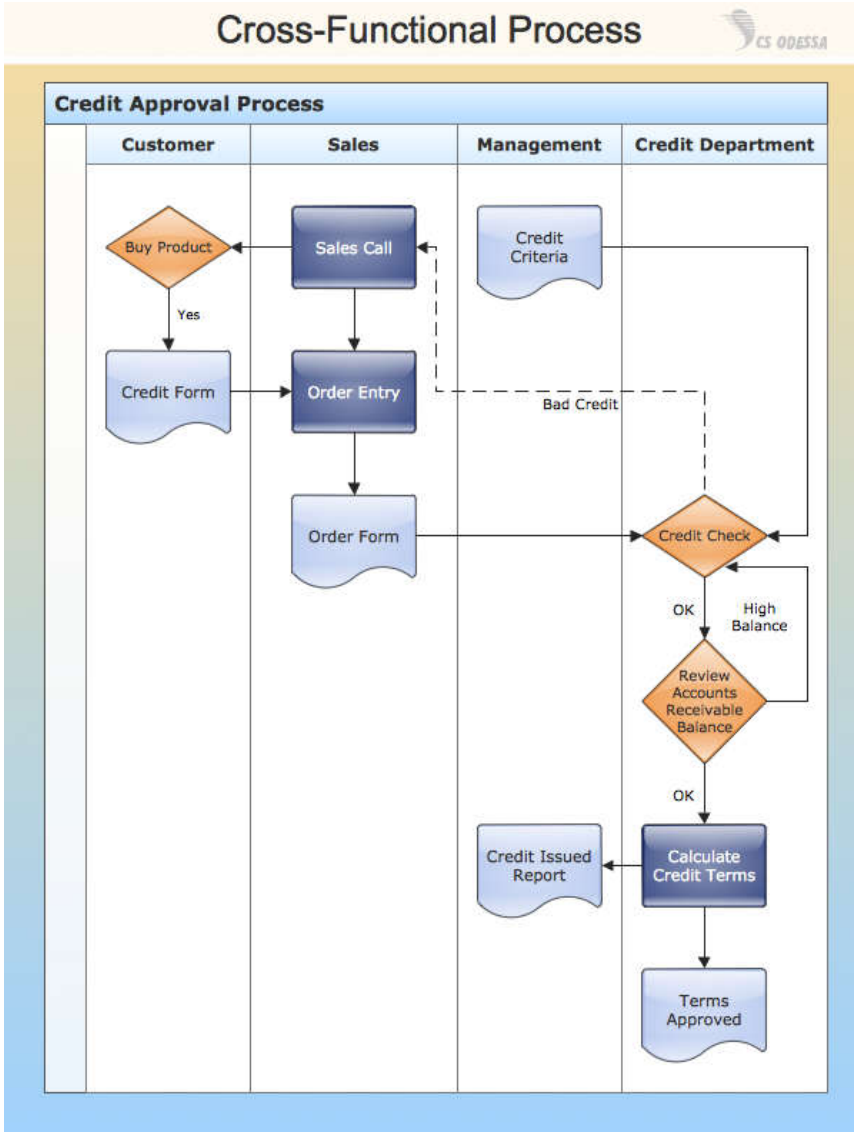
Exercise #1


Customer
Sales
Management
Credit Department
Buy Product
Sales Call
Credit Criteria
Credit Form
Order Entry
Bad
Credit
Order Form
Credit Check
High
Balance
OK
Review Accounts
Receivable Balance
OK
Credit Issued
Report
Calculate Credit
Terms
Terms Approved

Exercise #2


Providing Telecom Services
Customer
Customer Service Representative
Request Service
Provide Customer Information
Notify Customer
Consolidated Pre-Order Validation Results
Create New Customer
Address not Valid
Service not Available at Address
No
No
No
Receive Service Request
Select Service Package
Gather Customer Information
Perform Customer Information Inquiry
Customer Found?
Yes
Check Address
Address Found & Validated
Yes
Check Service Availability
Service Available?
Yes
Check Resource Availability
Resources Available?
Yes
Perform Service Portability
Check Customer Credit
Pre-Order Validation Results
No

Exercise #3


Cross-Functional Process
CS ODESSA
Credit Approval Process
Customer
Sales
Management
Credit Department
Buy Product
Sales Call
Credit Criteria
Yes
Credit Form
Order Entry
Bad Credit
Order Form
Credit Check
OK
High Balance
Review Accounts Receivable Balance
OK
Credit Issued Report
Calculate Credit Terms
Terms Approved