PACE UNIVERSITY
Seidenberg School of Computer Science and Information Systems

CS631G Software Verification

CHAPTER 4: INTRODUCTION TO REQUIREMENTS ENGINEERING (W3)

Class instructor: Yuri Chernak, PhD

Outline

Class Objectives

The objective of this class is to introduce software requirements and to explain the processes involved in discovering these requirements.

At the end of this class you will:

- understand the basic taxonomy of requirements, i.e., the concepts of user and system requirements and how they are different;
- understand the difference between functional and non-functional software requirements;
- understand the requirements engineering process phases and tasks;
- understand why requirements management is necessary and how it supports other requirements engineering activities.

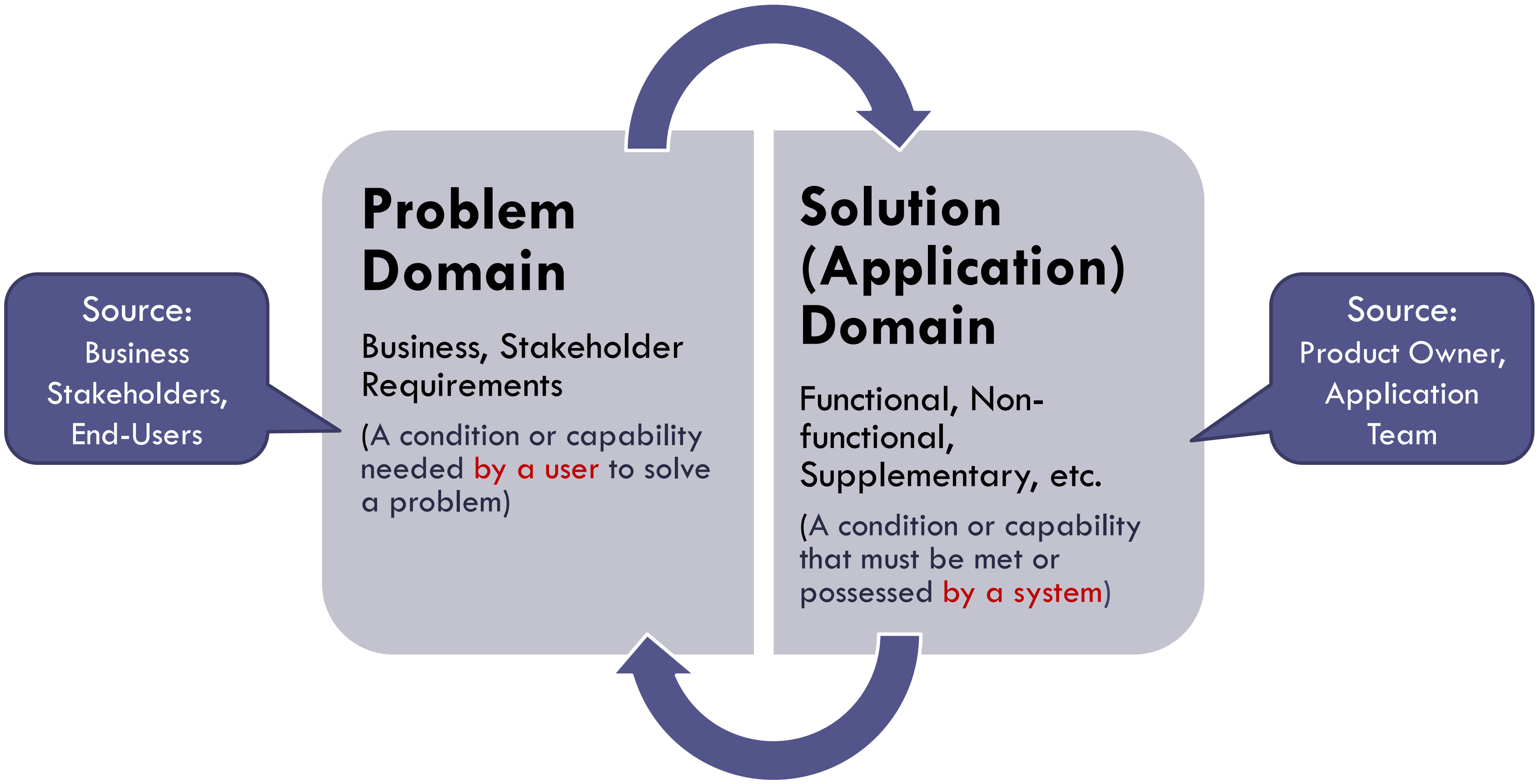
Requirements FAQs

Question	Answer
What are requirements?	IEEE Std. 610 Standard Glossary (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).
What is requirements engineering?	The process of developing and managing requirements that includes defined process phases and practices.
What is the cost of developing requirements?	About 15% of the project cost.
What happens when requirements are wrong?	Projects are late, systems are unreliable and do not meet users' expectations.
Is there an ideal requirements engineering process?	No, the process needs to be tailored to a given project's context.

Requirements FAQs

Question	Answer
What is requirements analysis?	IEEE Std. 610 Standard Glossary (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements. (2) The process of studying and refining system, hardware, or software requirements.
What is a requirements specification?	IEEE Std. 610 Standard Glossary A document that specifies the requirements for a system or component. Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards.
Who are system stakeholders?	BABOK v2 A group or person who has interests that may be affected by an initiative or influence over it.
What is requirements management?	BABOK v2 The activities that control requirements development, including requirements change control, requirements attributes definition, and requirements traceability.

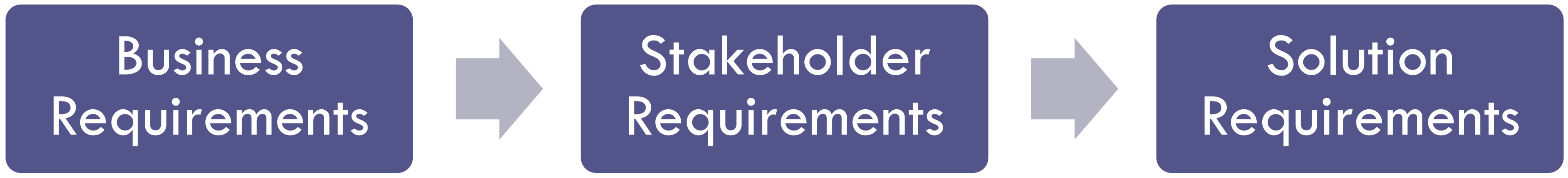
Requirements Conceptual Classification


Source:
Business Stakeholders, End-Users
Problem Domain
Business, Stakeholder Requirements
(A condition or capability needed by a user to solve a problem)
Solution (Application) Domain
Functional, Non-functional, Supplementary, etc.
(A condition or capability that must be met or possessed by a system)
Source:
Product Owner, Application Team

The conceptual separation between **Problem vs. Solution** domains is fundamental and applies to any development methodology.

BABOK Requirements Classification

BABOK - Business Analysis Body of Knowledge



Business Requirements

- High-level statements of the goals, objectives, or needs of the enterprise.
- They are developed and defined through *enterprise analysis.*

Stakeholder Requirements

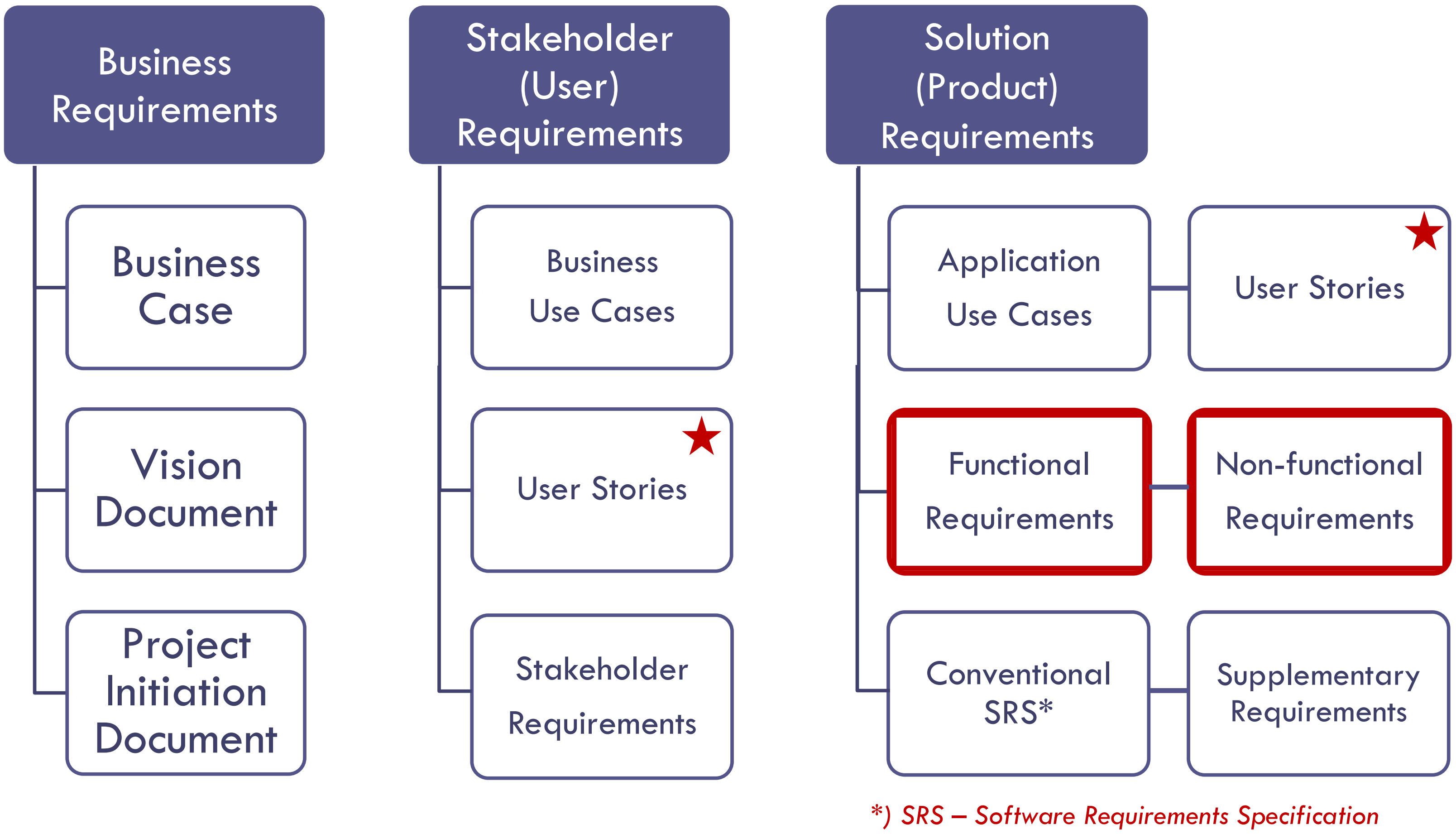
- a.k.a. User Requirements.
- Statements of the needs of a particular stakeholder or class of stakeholders.
- They are developed and defined through *requirements analysis.*

Solution Requirements

- a.k.a. System or Product Requirements.
- Describe the characteristics of a solution that meet business requirements and stakeholder requirements.
- They are developed and defined through *requirements analysis.*

In practice, the terms Business Requirements and User Requirements are commonly used interchangeably.

Types of Requirements



*) *SRS – Software Requirements Specification*

★ Agile user stories can represent both - user needs and product features.

Project Initiation Document (PID) Explained

The **Project Initiation Documentation** (**PID**), a.k.a. Project Charter, is one of the most significant artifacts in project management, which provides the foundation for approving and funding a new software project.

PID Purposes

Project Initiation Documentation (PID) has three primary purposes:

- Ensure that the project is well understood before asking the Project Board to make any major financial commitment, i.e., funding to the project to produce the products.
- Act as a baseline document that the Project Board and Project Manager can then use to assess progress, issues and ongoing viability questions; e.g., *Is this project still worth doing?*
- Provide a single reference for the project in one place, so that people joining the project can easily find out what the project is about, the reasons for the project, the business justification, major risks, how it is being managed and controlled.

Stakeholder Requirements Explained

Business Requirements, also known as *Stakeholder Requirements*, describe the characteristics of a proposed system from the viewpoint of the system's end users. In contrast, Product, System, or software requirements are ways of *how* to deliver, satisfy, or meet Business Requirements.

Business Requirements often include:

- Business context, scope, and background, including reasons for change;
- Key business stakeholders that have requirements;
- Success factors for a future/target state;
- Constraints imposed by the business or other systems;
- Business process models and analysis, often using flowchart notations to depict either 'as-is' and 'to-be' business processes;
- Data dictionary, glossary of business terms;

Product Requirements Classification

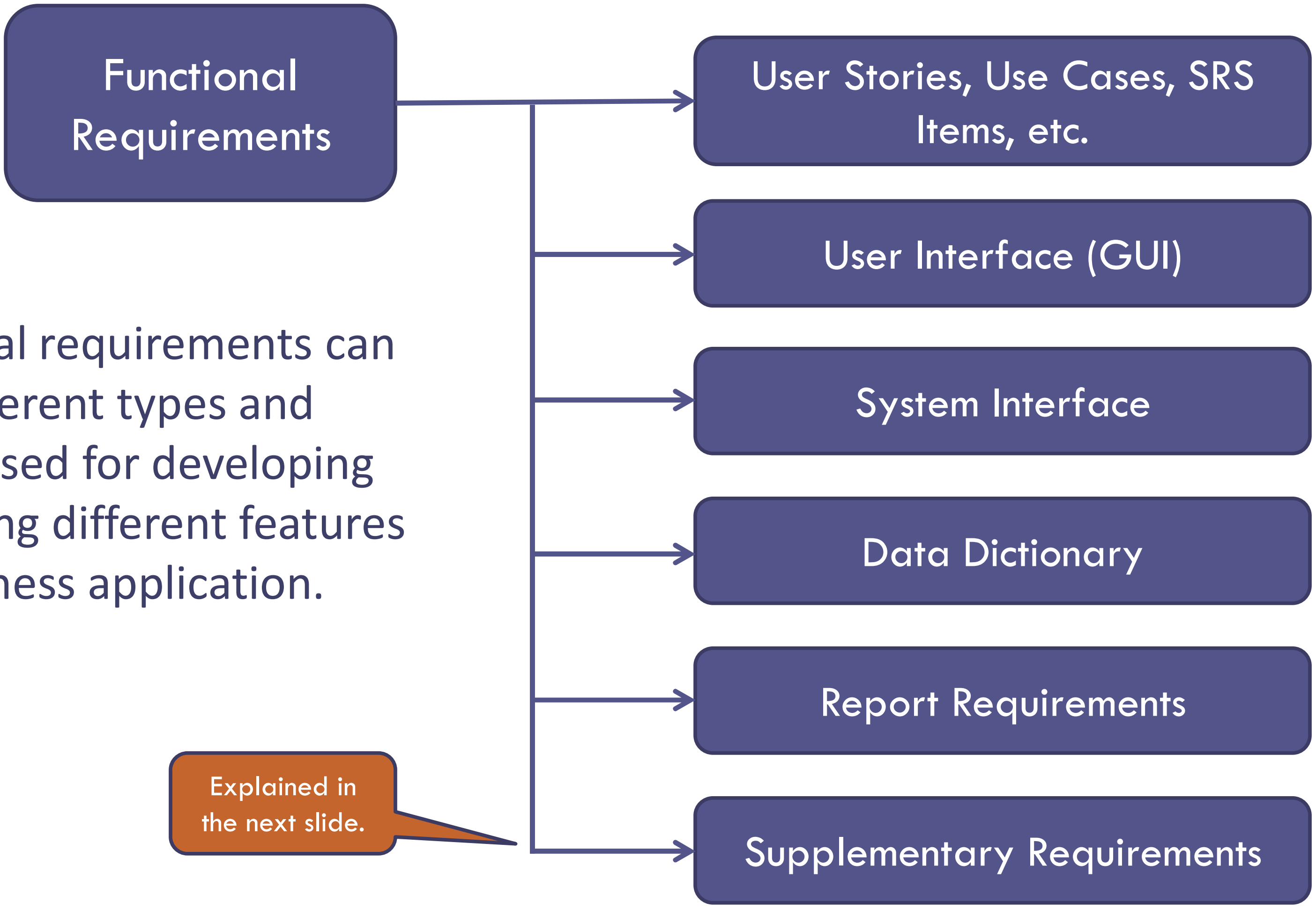
Functional Requirements

- Functional Requirements describe the behavior and information that the solution will manage.
- They describe capabilities the system will be able to perform in terms of behaviors or operations, i.e., application features.
- They depend on the type of software, expected users , and the type of system where the software is used.

Non-Functional Requirements

- Non-functional Requirements capture conditions that do not directly relate to the behavior or functionality of the solution, but rather describe environmental conditions under which the solution must remain effective or qualities that the systems must have.
- They are also known as “quality attributes” requirements. These can include requirements related to capacity, performance, security, availability, architecture, etc.

Functional Requirements Classification


Functional Requirements
User Stories, Use Cases, SRS Items, etc.
User Interface (GUI)
System Interface
Data Dictionary
Report Requirements
Supplementary Requirements
Explained in the next slide.

Functional requirements can be of different types and may be used for developing and testing different features of a business application.

Supplementary Requirements Explained

Definition

Supplementary term definition in the Marriam-Webster Dictionary:

"something available to supply extra when needed"

Example – Instructor's lecture notes come with many supplementary materials.

In Software Engineering – **Supplementary Requirements** capture those system details that we do not include in primary (or core) requirements for the following reasons:

- Supplementary requirements do not relate to a specific feature, rather can be scattered across multiple features, or they relate to the entire application.

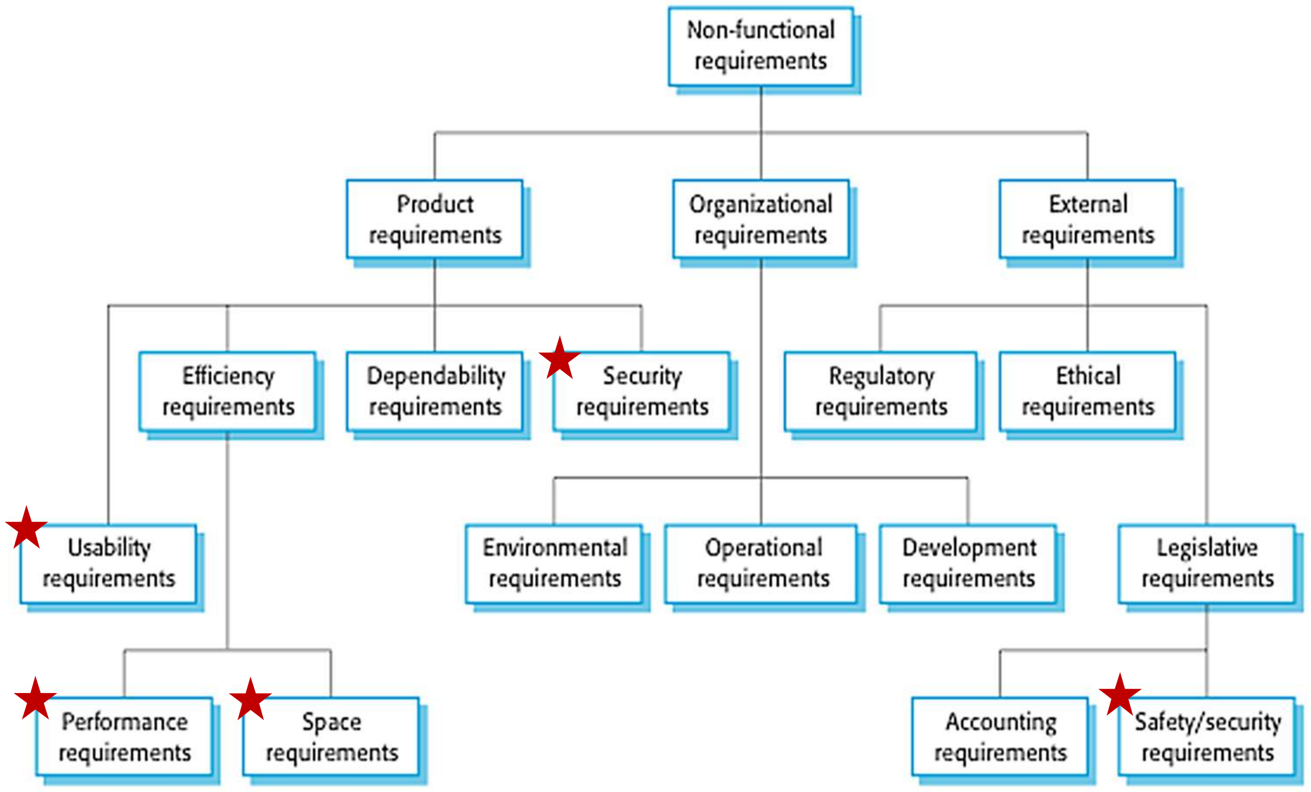
Benefits of keeping supplementary requirements separate from the core requirements:

- It improves clarity of core requirements.
- It improves maintainability of supplementary requirements.
- It improves change impact analysis for new releases.

Scattered supplementary requirements, called *crosscutting concerns*, we discuss in the following lecture.

Non-Functional Requirements Classification

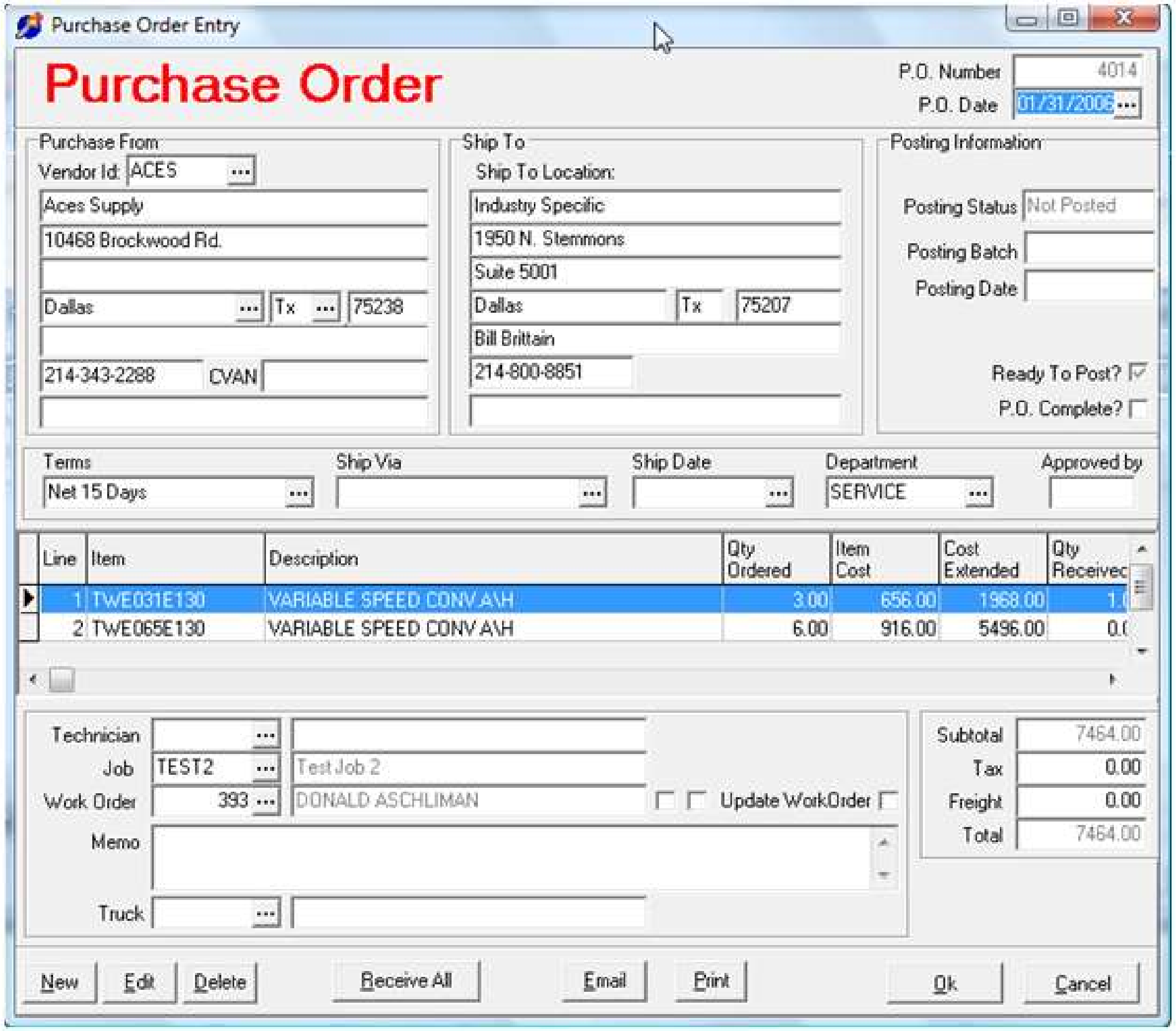
Some non-functional requirements (marked below with ★) can be critical to the project success. If they are not met, the system may be useless.



Example – Purchase Order Entry Screen

The functionality of this screen was designed based on various requirements:

- New Order use case,
- GUI specification,
- Data dictionary for field edits, data dependency;
- Supplementary requirements for user entitlements, calculation, connectivity, concurrency;
- System interface for data coming in from and going to external systems, etc.



Purchase Order Entry

Purchase Order

P.O. Number 4014
P.O. Date 01/31/2006

Purchase From
Vendor Id: ACES
Aces Supply
10468 Brockwood Rd.
Dallas, Tx 75238
214-343-2288 CVAN

Ship To
Ship To Location:
Industry Specific
1950 N. Stemmons
Suite 5001
Dallas, Tx 75207
Bill Brittain
214-800-8851

Posting Information
Posting Status: Not Posted
Posting Batch:
Posting Date:
Ready To Post? ✓
P.O. Complete?

Terms: Net 15 Days
Ship Via:
Ship Date:
Department: SERVICE
Approved by:

Line	Item	Description	Qty Ordered	Item Cost	Cost Extended	Qty Receivec
1	TWE031E130	VARIABLE SPEED CONV A\H	3.00	656.00	1968.00	1.(
2	TWE065E130	VARIABLE SPEED CONV A\H	6.00	916.00	5496.00	0.(

Technician:
Job: TEST2 Test Job 2
Work Order: 393 DONALD ASCHLIMAN
Update WorkOrder
Memo:
Truck:

Subtotal 7464.00
Tax 0.00
Freight 0.00
Total 7464.00

Characteristics of Good Requirements

The **IEEE Std. 830-1998** "Recommended Practice for Software Requirements" defines characteristics of good requirements:

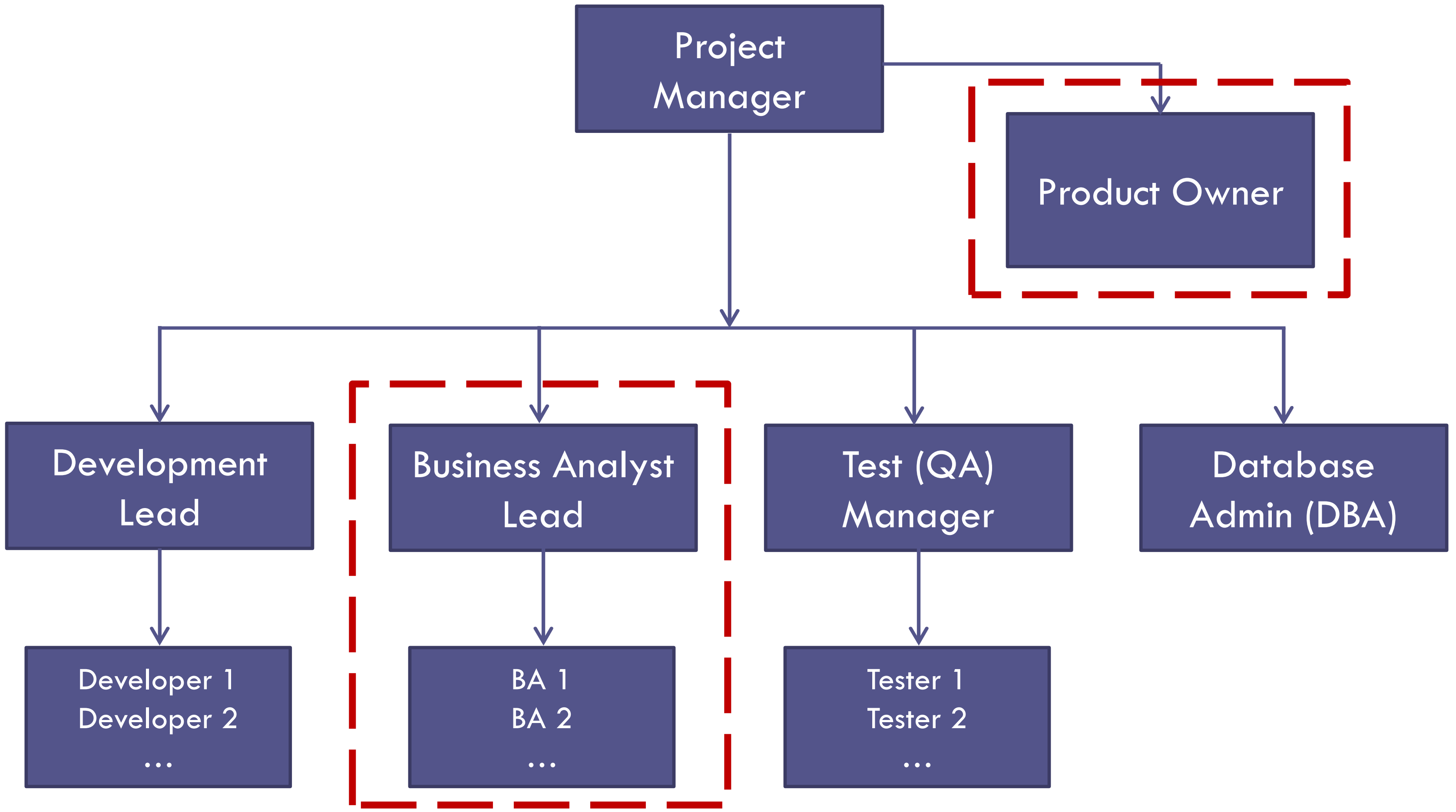
- Correct
- Unambiguous
- Traceable
- Consistent
- Ranked for importance
- Verifiable (i.e., Testable)
- **Modifiable** (i.e., Maintainable)
- **Complete**

The most challenging characteristics to implement

Improving these characteristics is the focus of Aspect-Oriented Requirements Engineering, discussed in the next lecture.

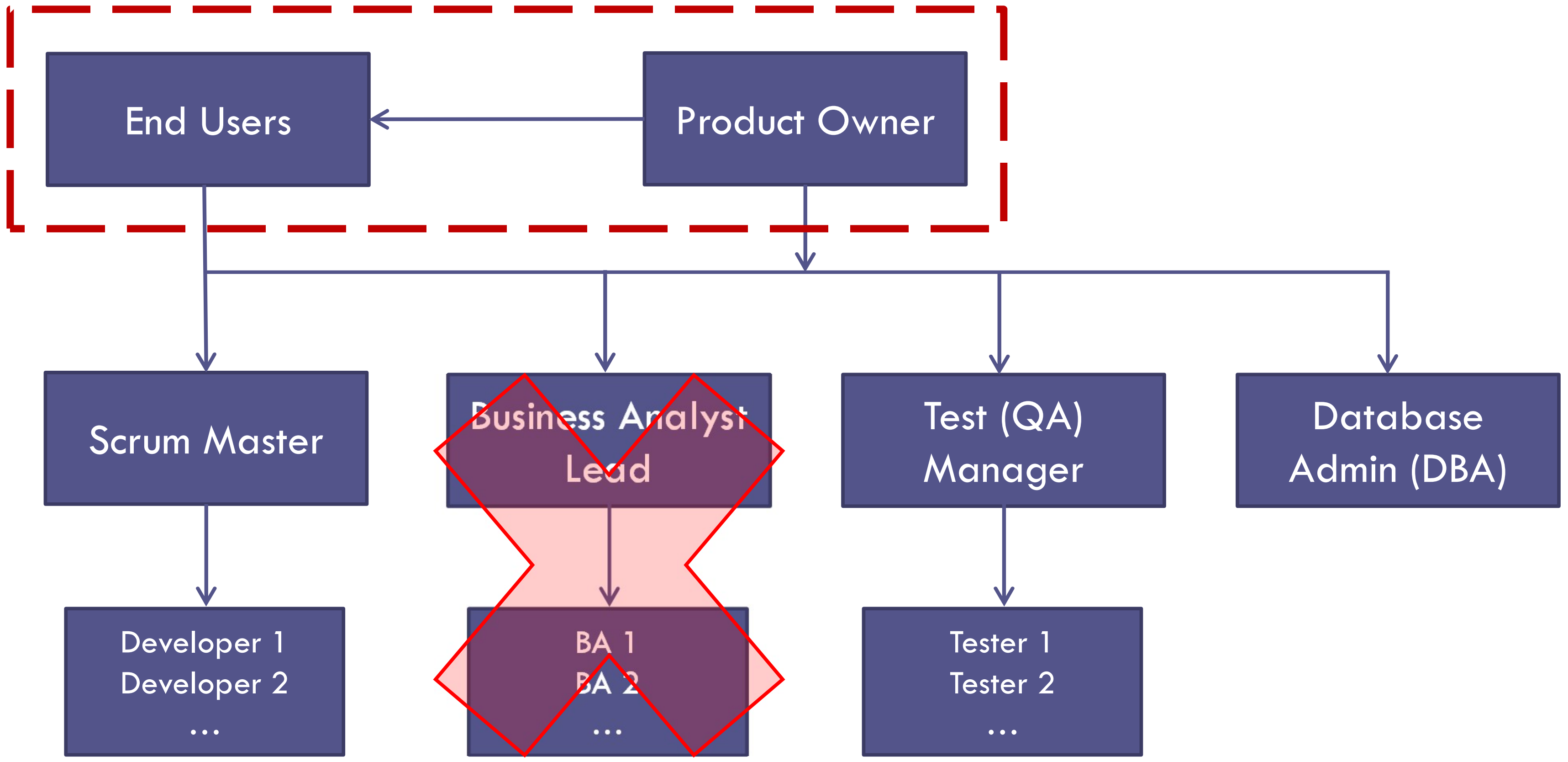
Who Develops Requirements?

On traditional projects, business (user) requirements can be developed by a Product Owner and product requirements are developed by Business Analysts.



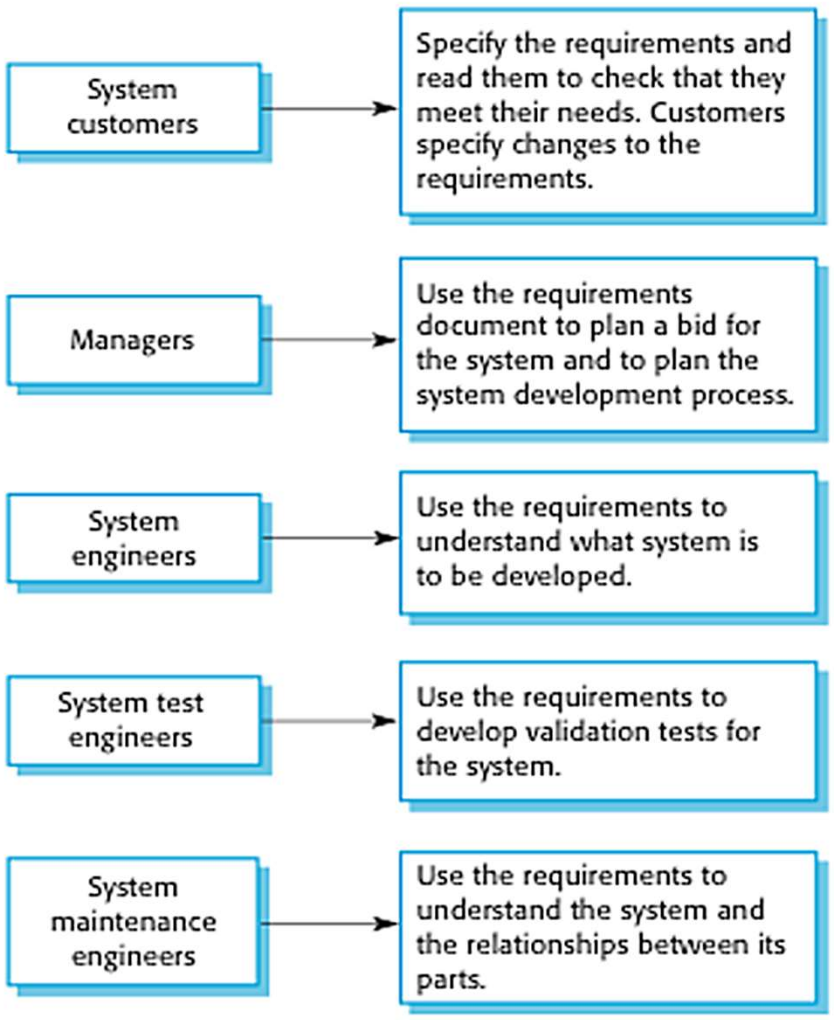
Who Develops Requirements in Agile Teams?

According to the original XP methodology, end-users and a Product Owner are responsible for the development of user stories.



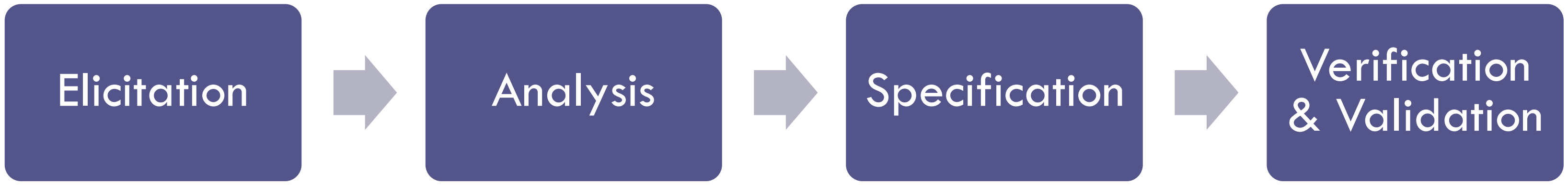
Who Uses Requirements?

Requirements are used by various project team members and provide a basis for these parties to produce their own work products (project deliverables).


System customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers
Use the requirements to understand what system is to be developed.
System test engineers
Use the requirements to develop validation tests for the system.
System maintenance engineers
Use the requirements to understand the system and the relationships between its parts.

Requirements Engineering (RE) Process

A conventional RE process includes four phases:



Elicitation

- Document a business case (or Vision document) for developing the new product.
- Identify and analyze project stakeholders.
- Conduct regular meetings with end-users to gather and understand user needs.

Analysis

- Analyze stakeholder requests.
- Analyze the business domain.
- Develop stakeholder requirements.
- Prioritize stakeholder requirements.
- Manage the development of stakeholder requirements.

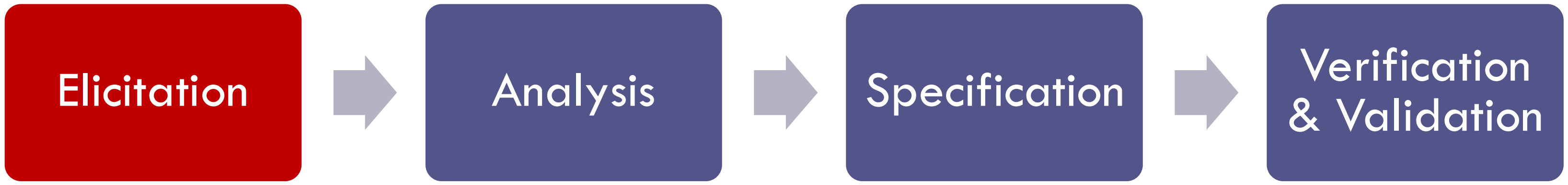
Specification

- Develop product requirements.
- Manage the development of product requirements.
- Maintain versions of requirements artifacts.

Verification & Validation

- Conduct requirements reviews, formal inspections.
- Validate requirements with prototypes.
- Plan and conduct user acceptance testing to validate the implementation of requirements.

Requirements Elicitation Phase



The next slides will discuss the requirements elicitation practices.

Requirements Elicitation Practices

Objectives of requirements elicitation are to understand the work that **stakeholders** do and how they might be using a new system to help support their business needs. The main deliverables of this phase are the Business Case or Vision documents and the inventory of User Requirements.

Requirements elicitation tasks include:

- documenting a business case, a.k.a., vision and scope document, for developing the new product;
- understanding and documenting a business domain model (*see next slide*);
- identifying and analyzing project stakeholders (*see next slides*);
- assigning stakeholders' responsibilities and establishing their commitments;
- defining a procedure to elicit stakeholder needs;
- conducting regular meetings and workshops with end-users to document user requirements;
- uniquely identifying and managing user needs (user requirements);
- agreeing on priorities of user requirements.

Business Domain Model Explained

Business Domain Model

- A conceptual view of all or part of an *enterprise* focusing on *products*, *deliverables* and *events* that are important to the mission of the *organization*.
- The domain model is useful to validate the solution scope with the business and technical *stakeholders*.

A need for a Domain Model

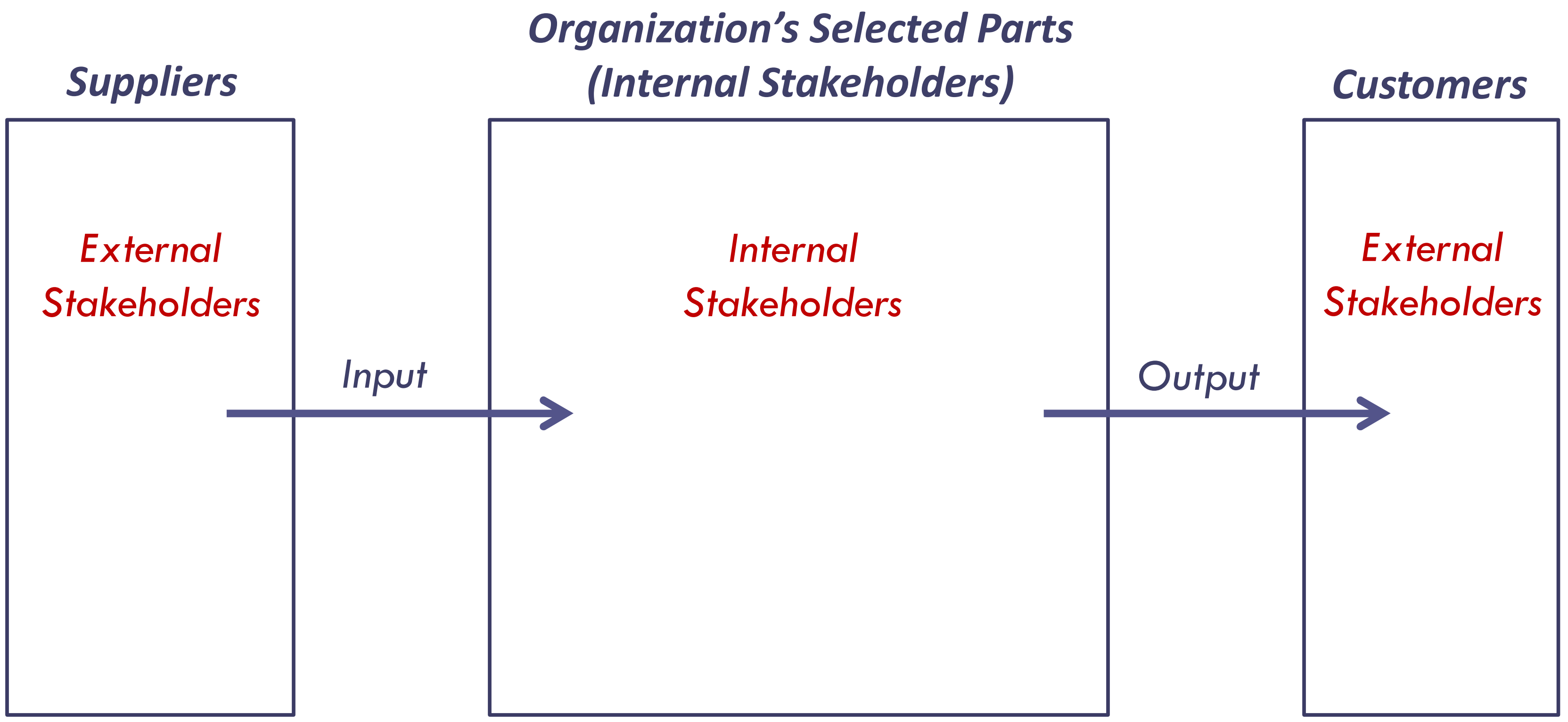
- In order to create good software, you first have to know what that software is all about, i.e., one must understand the business domain for which software will be used for.
- The entire purpose of the software is to provide a solution to a specific problem domain. In the beginning we need to create an abstraction of that domain, i.e., a model of the business domain.
- During the software design process, we make lots of references to the domain model. We need it in order to be able to deal with complexity and to communicate with other developers and business stakeholders.
- There are different ways to document a domain model, but most common way is producing diagrams. One of them – Business Relationship Map (BRM) we discuss next.

Stakeholder Analysis with Business Relationship Map

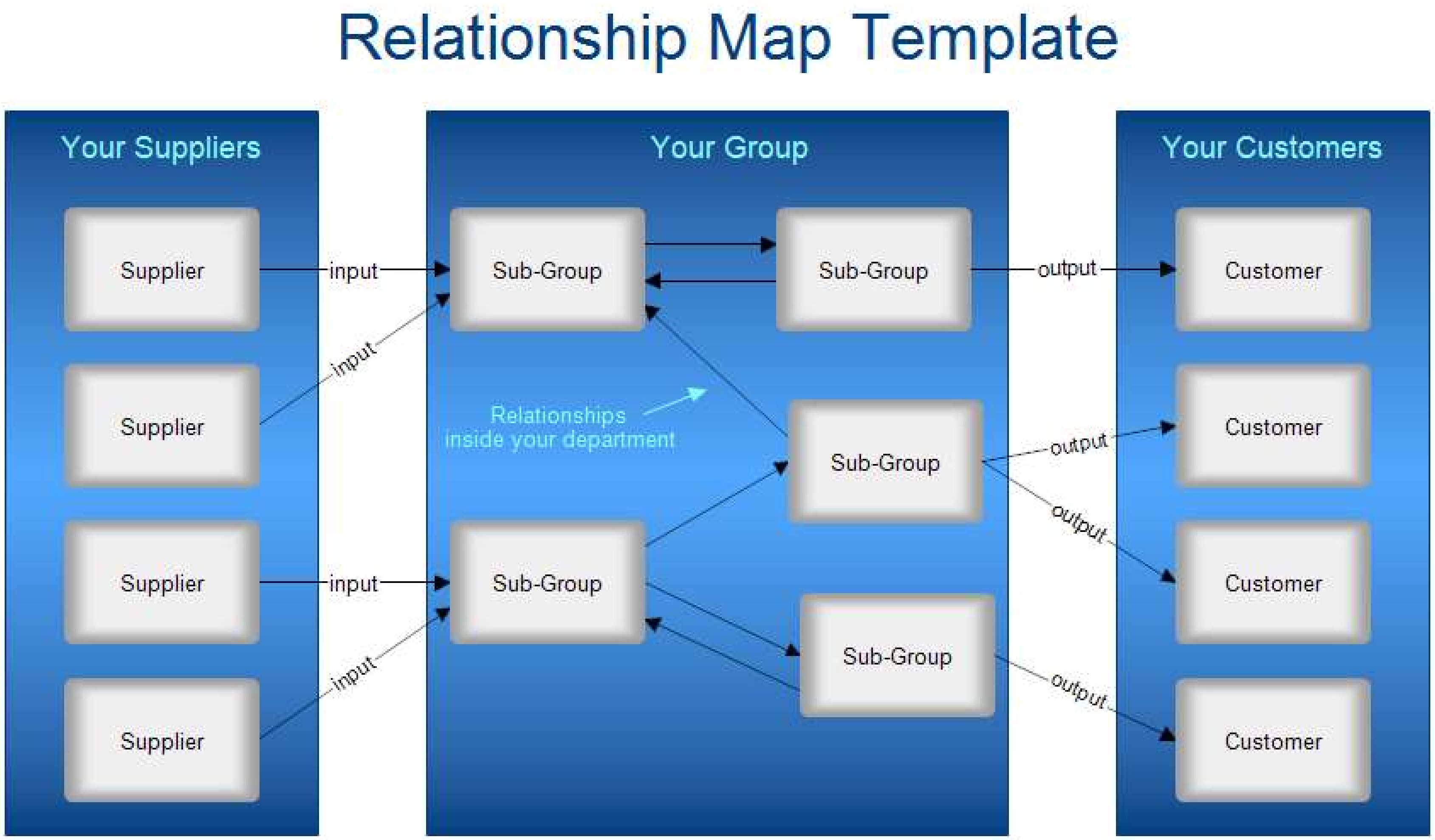
- BABOK Definition – Process Map:
  - *A business model that shows a business process in terms of the steps and input and output flows across multiple functions, organizations, or job roles.*
- Business Relationship Map (BRM) is a kind of a business process map that:
  - elaborates the information already captured in the Business Case;
  - is an effective technique to analyze a project’s stakeholders (internal and external), their roles, and capture the information they exchange;
  - depicts the internal parts of an organization and the external supplier-customer relationships among those parts;
  - follows a general left-to-right sequence of the business process flow;
  - helps you view work at the organizational level and it can be a basis for developing a detailed business domain model and business requirements.

Business Relationship Map Template

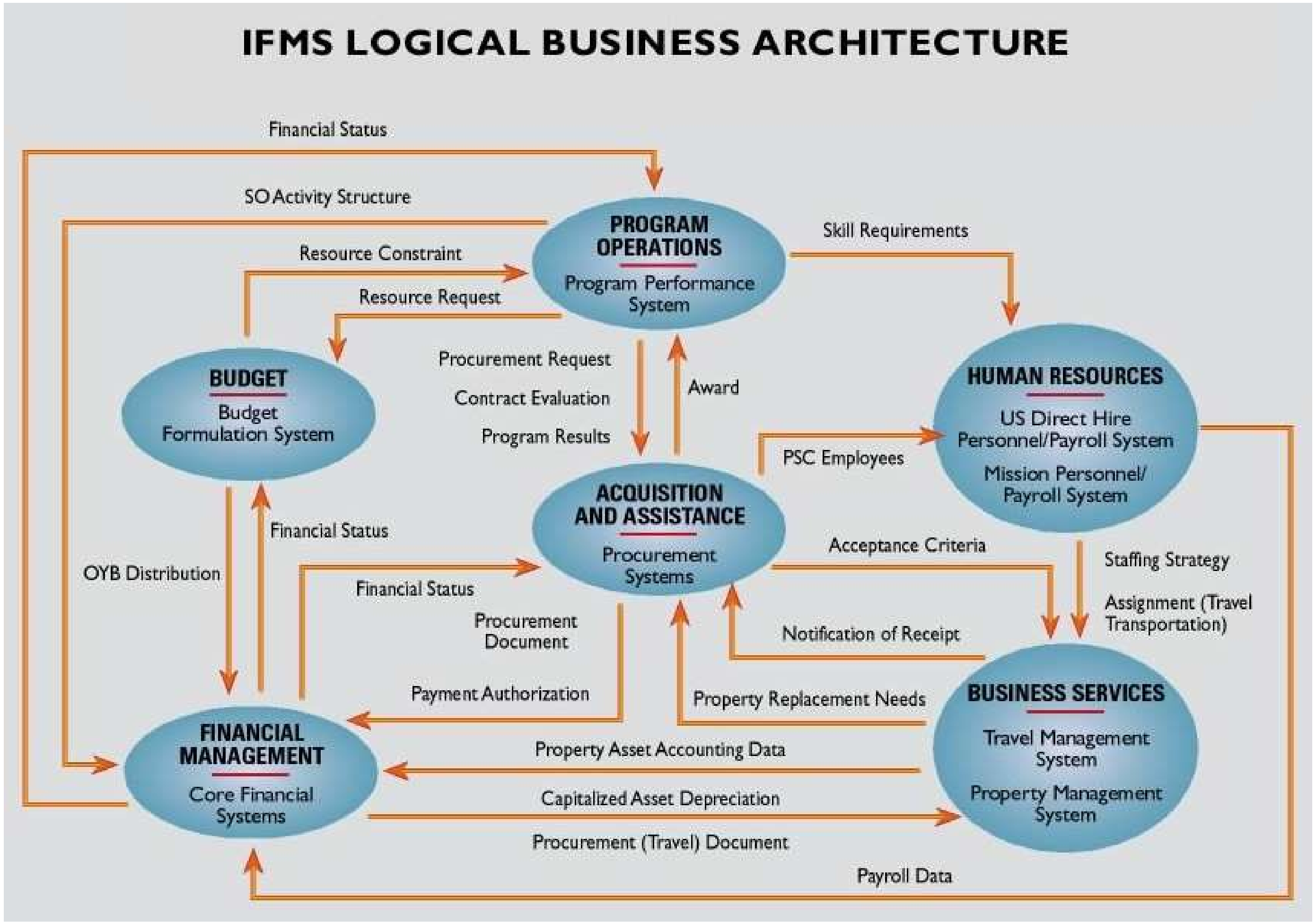
The Business Relationship Map template includes three parts:



Business Relationship Map Template

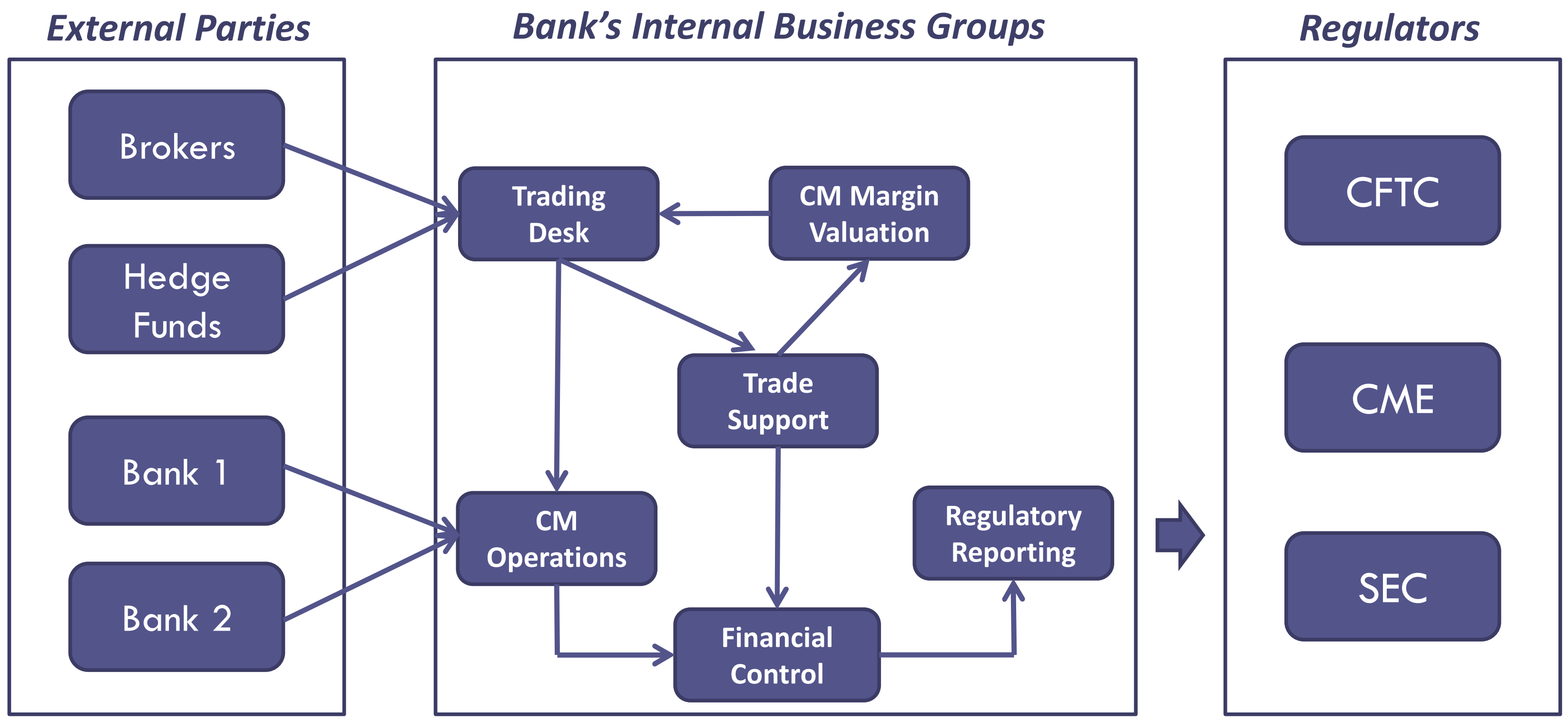


Elaboration of the BRM’s Central Part



Business Relationship Map Example

- Below is an example of the business relationship map developed for the Collateral Management business unit in investment banking. See other examples on Pace/Classes Portal.
- Once stakeholder groups are identified, we then identify the stakeholder roles within each of the business groups.



SIPOC Diagram Explained

What is a SIPOC Diagram?

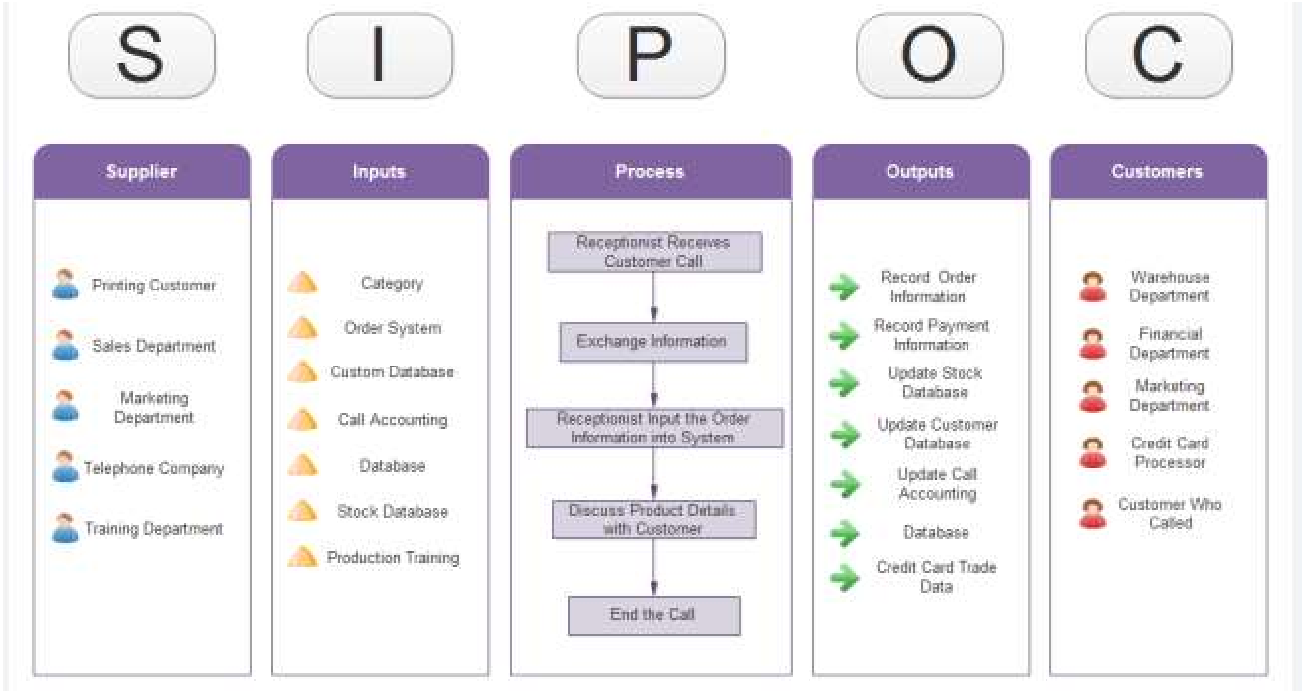
- **SIPOC diagram** is one of the most useful techniques of process management and process improvement.
- The name comes from the five elements of the diagram:
  Supplier – the person or group providing key information, materials, or other resource to the process
  Input – the “thing” provided
  Process – the set of steps that adds value to the Input
  Output – the final product of the process
  Customer – the person or group that receives the Output

- It is used to define a business process from beginning to end before work begins.
- It was in use at least as early as the total quality management programs of the late 1980s and continues to be used today in Six Sigma, lean manufacturing, and business process management.

SIPOC Diagram vs. BRM Diagram

How does the SIPOC Diagram relates to the BRM Diagram?

- The **SIPOC** diagram (see an example below) elaborates details of the **BRM** diagram.
- The **BRM** diagram shows a static view of the organization's business groups, whereas the SIPOC diagram shows a process flow how these groups operate and conduct their business.

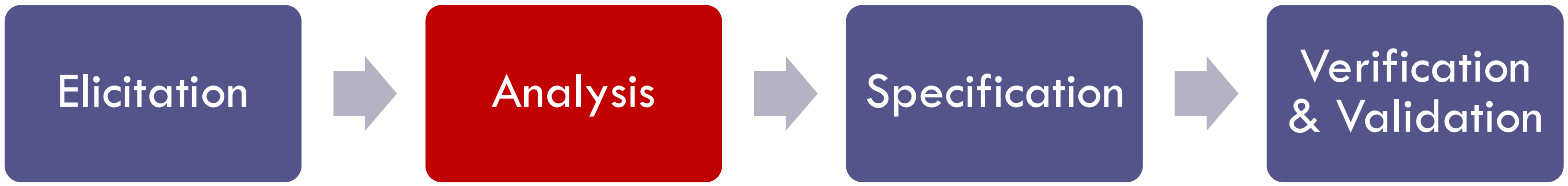


Requirements Elicitation Challenges

Requirements elicitation can be a challenging process for the following reasons:

- Stakeholders might be reluctant to be engaged in requirements gathering sessions that take time away from the performance of their daily job responsibilities.
- In some cases, stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organizational and political factors may influence the system requirements.
- The stakeholder requirements can change during the analysis phase.
- New stakeholders may emerge that can change the business context.

Requirements Analysis Phase



The next slides will discuss the requirements analysis practices.

Requirements Analysis Practices

Requirements analysis and negotiation closely relate to requirements elicitation and focus on both – **stakeholder** and **product** requirements:

Problem Domain Analysis → Solution Domain Analysis

Stakeholder Needs Analysis	**Product Requirements Analysis**
Analyze stakeholder needs.	Analyze stakeholder/user requirements.
Analyze the impact of the proposed solution on external parties.	Define the application's decomposition.
Develop a domain model and glossary of terms.	Define a taxonomy (types) of product requirements.
Analyze interfaces to external applications.	Identify an inventory of product requirements within each type.
Develop stakeholder (business) requirements.	Analyze the mutual impact of product requirements.
Prioritize stakeholder requirements.	Manage analysis artifacts.
Approve and baseline stakeholder requirements.	

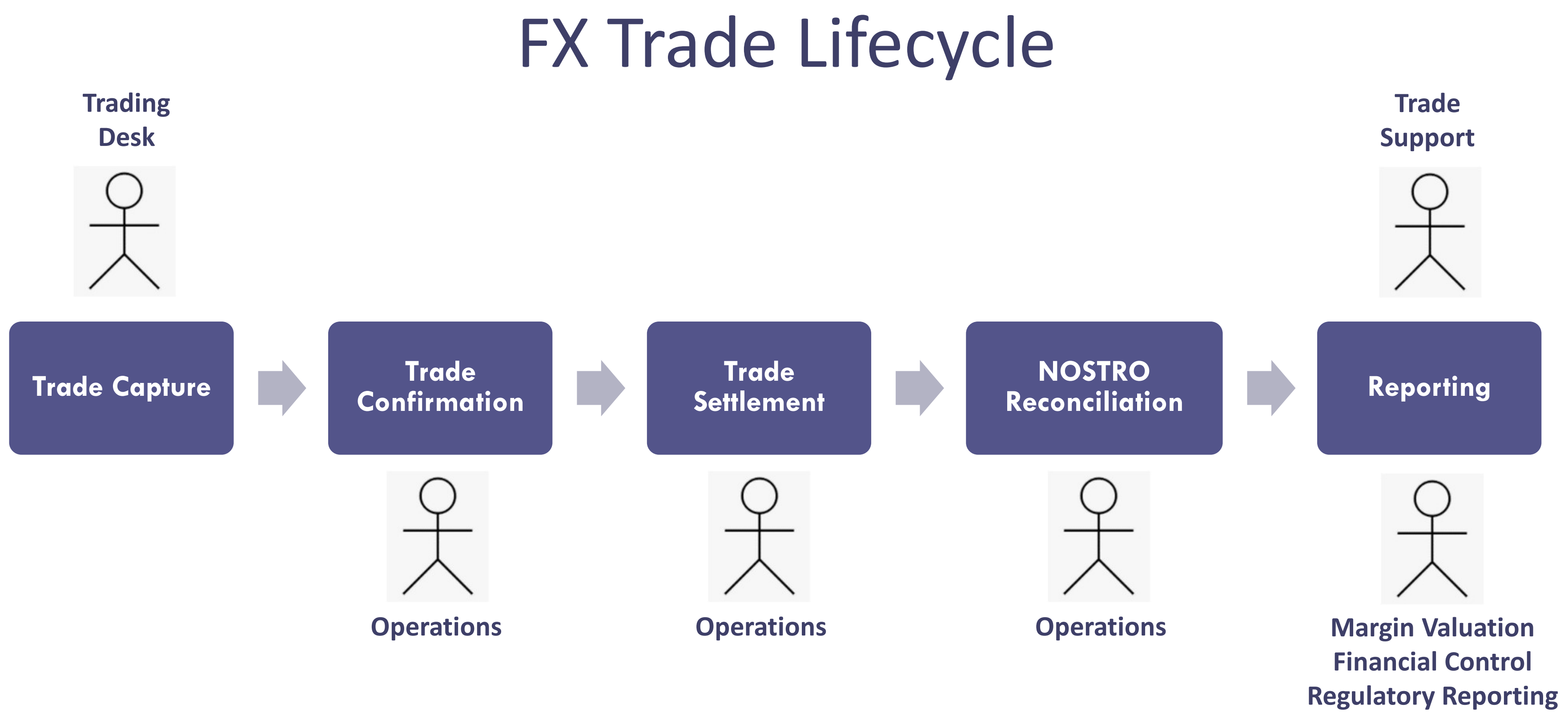
Requirements Analysis Diagrams

Requirements analysis is commonly performed with visual models or diagrams.

The most common diagrams include:

- Process Flow diagram – useful for analyzing activities of business users.
- Functional Decomposition diagram – useful for allocating product requirements to application modules (a.k.a. functional areas).
- Fishbone (Ishikawa) diagram – commonly used for product design, root-cause analysis of production incidents, etc.
- Context diagram – useful for analyzing a system's boundaries, interfaces to external systems, a.k.a. Level 0 data flow diagram.
- Data Flow diagram – useful for analyzing data exchange among related application features, a.k.a. Level 1 data flow diagram (discussed in W6 lecture).
- Use Case diagram – useful for analyzing a group of related use cases and their relationship with Actors.

Process Flow Diagram Example



A process flow diagram can be used to represent a flow of events and who is performing business activities.

Functional Decomposition Explained

IEEE Std.610 Definition

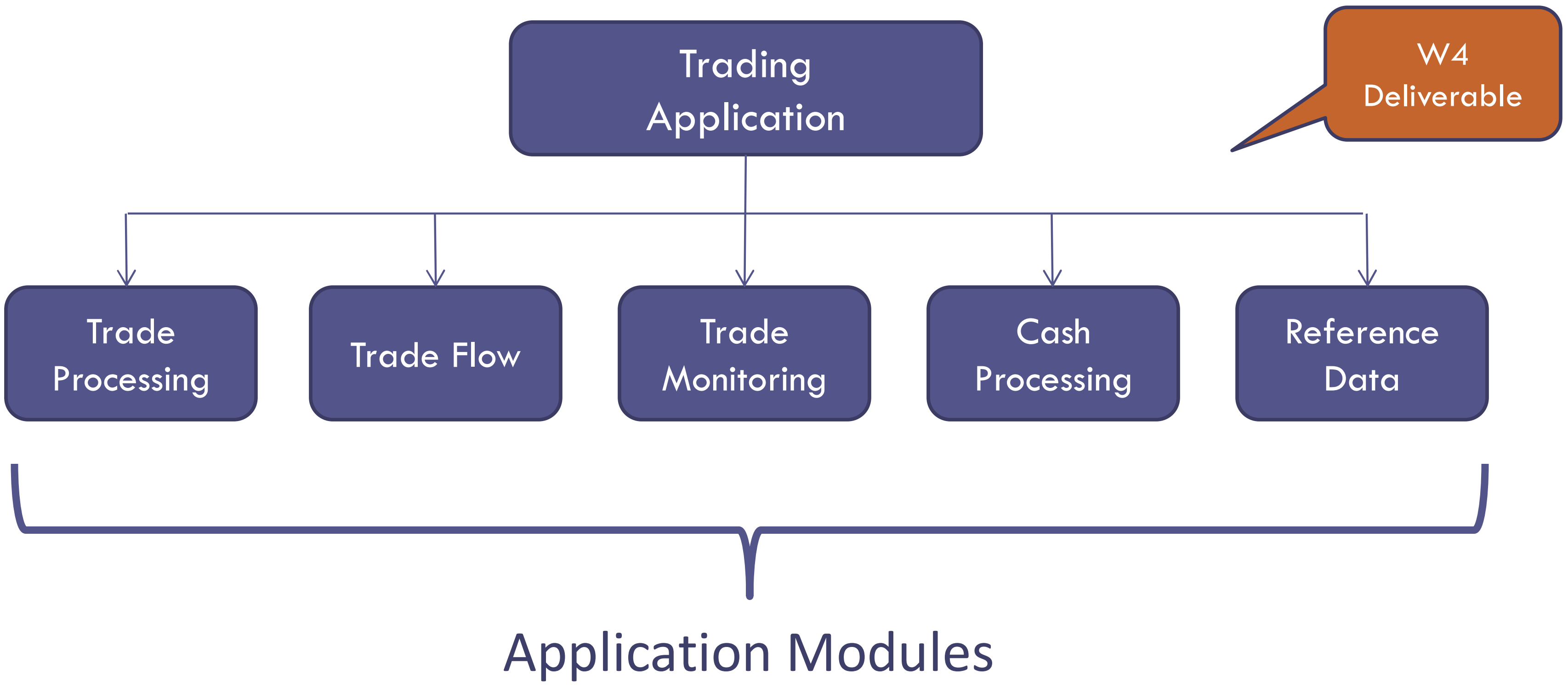
Functional Decomposition - A type of modular decomposition in which a system is broken down into components that correspond to system functions and subfunctions.

Explaining Modules

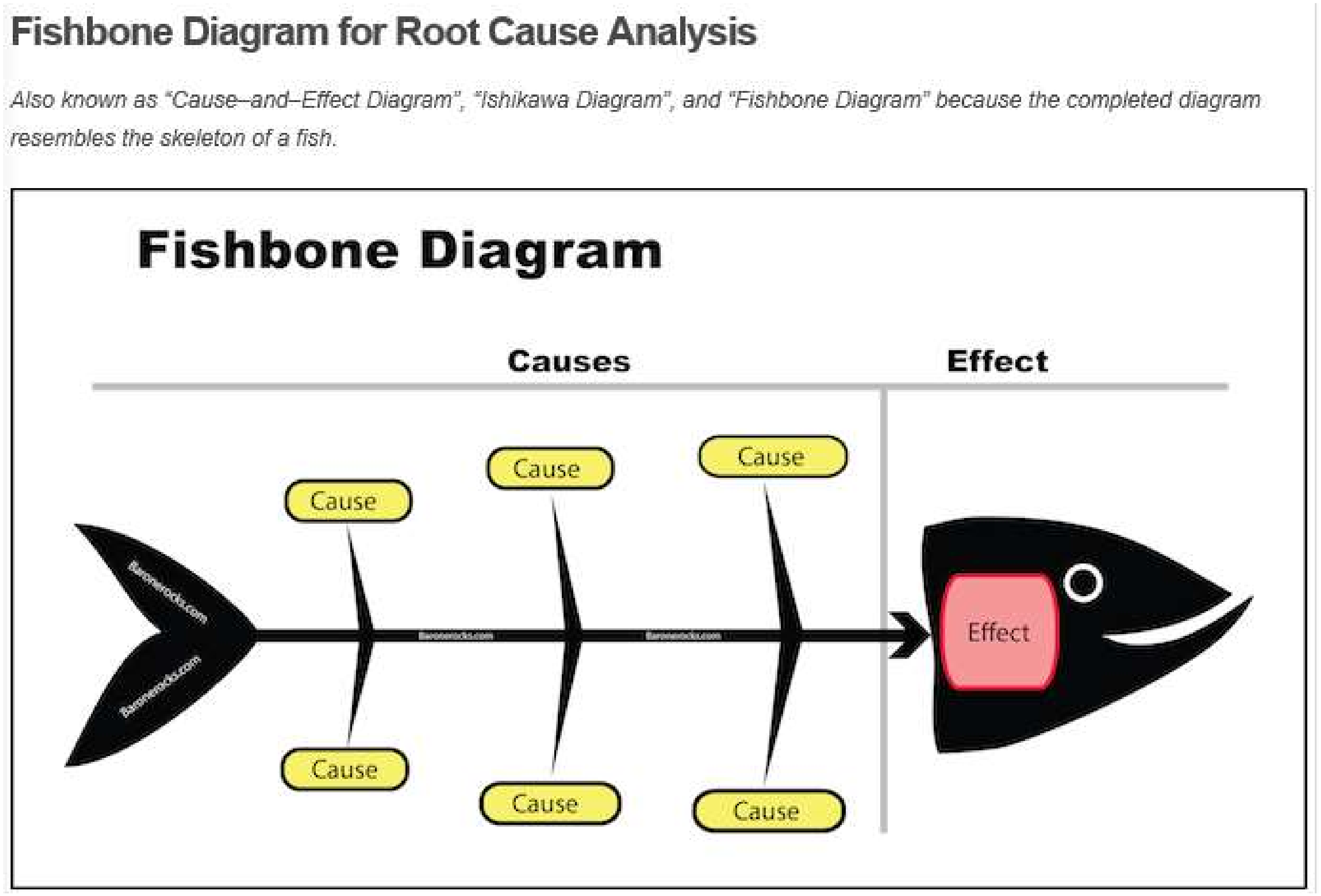
- Modules are used as a method of organizing related application features in order to reduce complexity.
- Give the Modules names that become part of a common language within the team.
- Modules and their names should reflect insight into a business domain.
- A conventional way to document functional decomposition is producing a Functional Decomposition Diagram (FDD). *See an example on the next slide.*

Functional Decomposition Diagram Example

- A Functional Decomposition Diagram (FDD) shows an application's functionality breakdown into smaller and manageable parts (a.k.a. modules).
- FDD provides a basis for the menu structure of your application's Home Page.

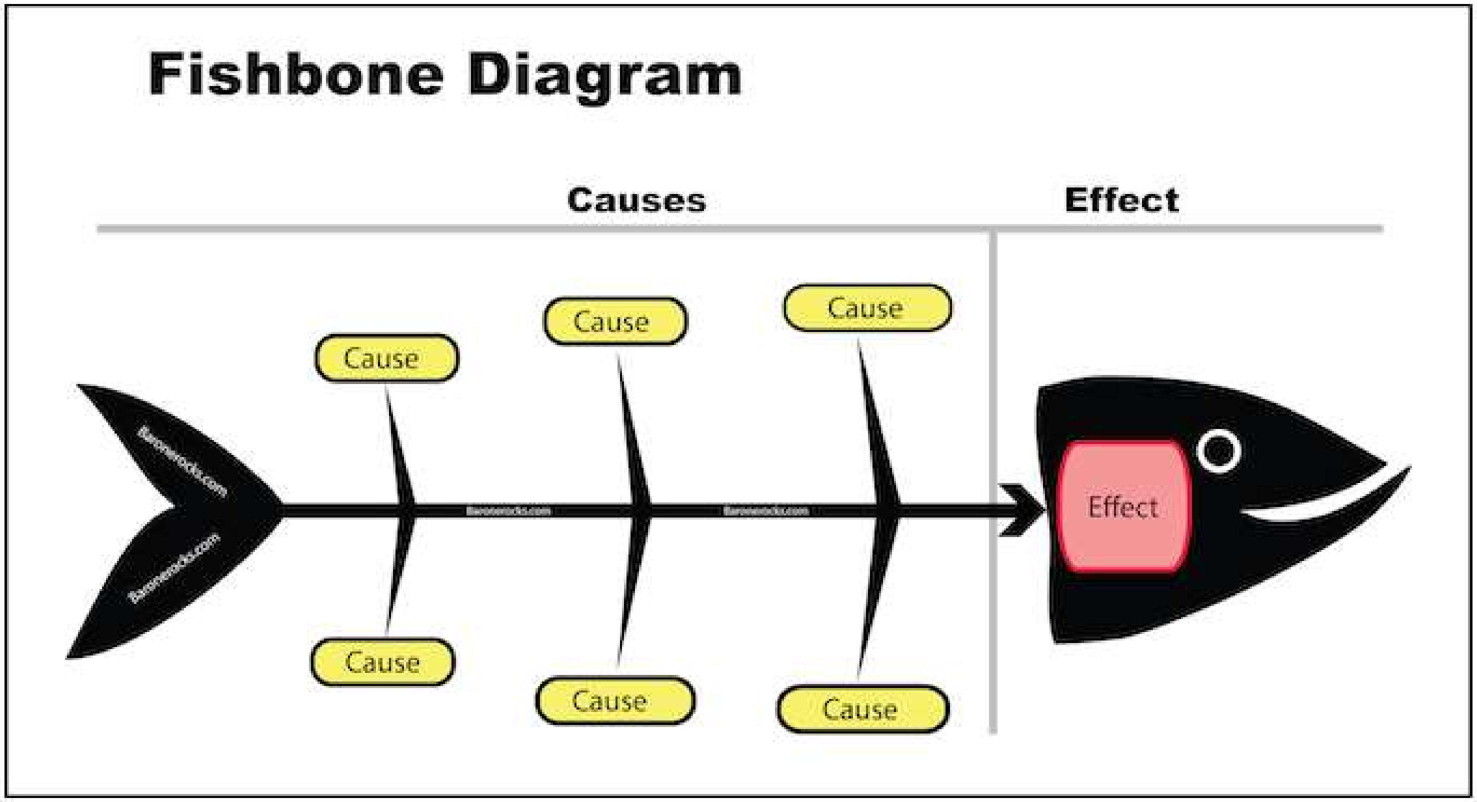


Fishbone (Ishikawa) Diagram Explained



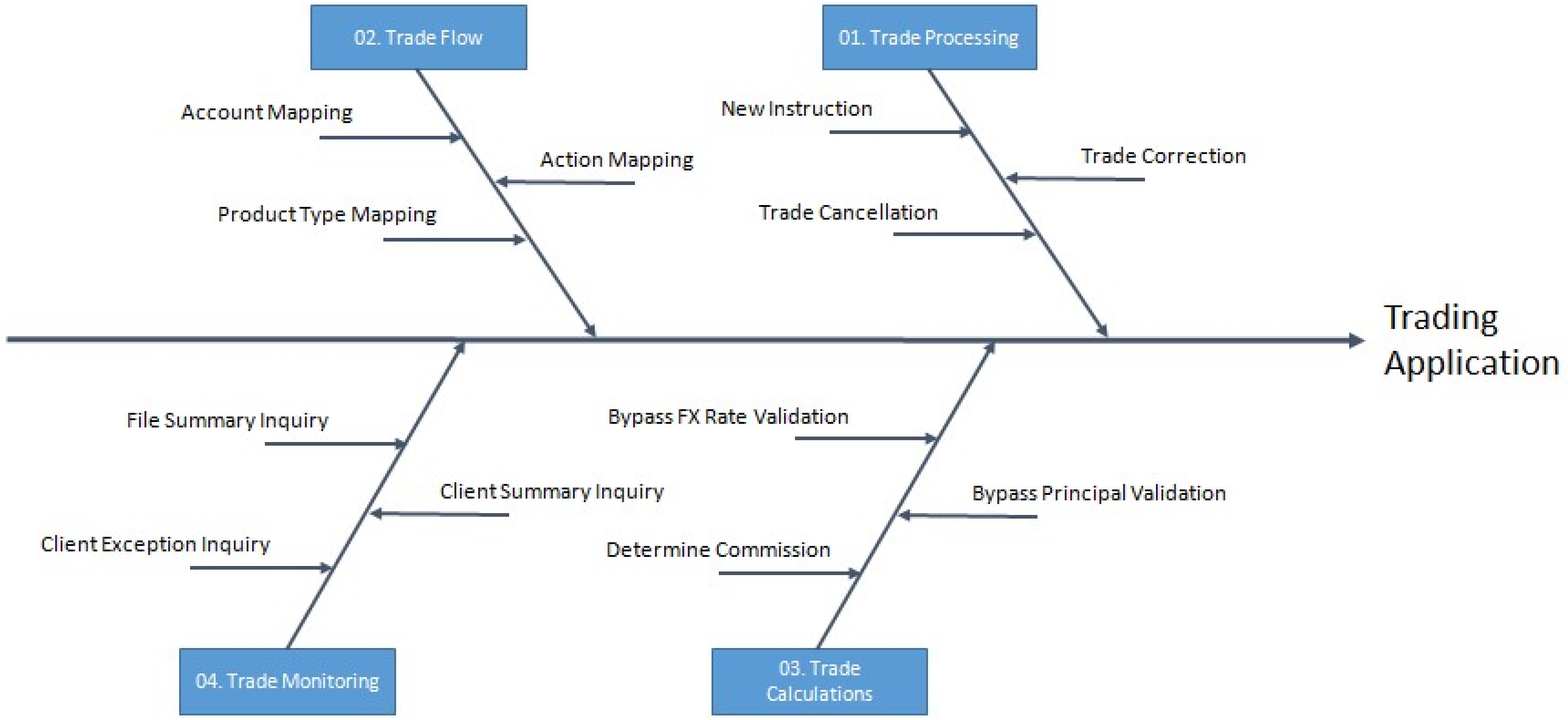
Fishbone Diagram for Root Cause Analysis

Also known as "Cause–and–Effect Diagram", "Ishikawa Diagram", and "Fishbone Diagram" because the completed diagram resembles the skeleton of a fish.



Fishbone (Ishikawa) Diagram Example 1

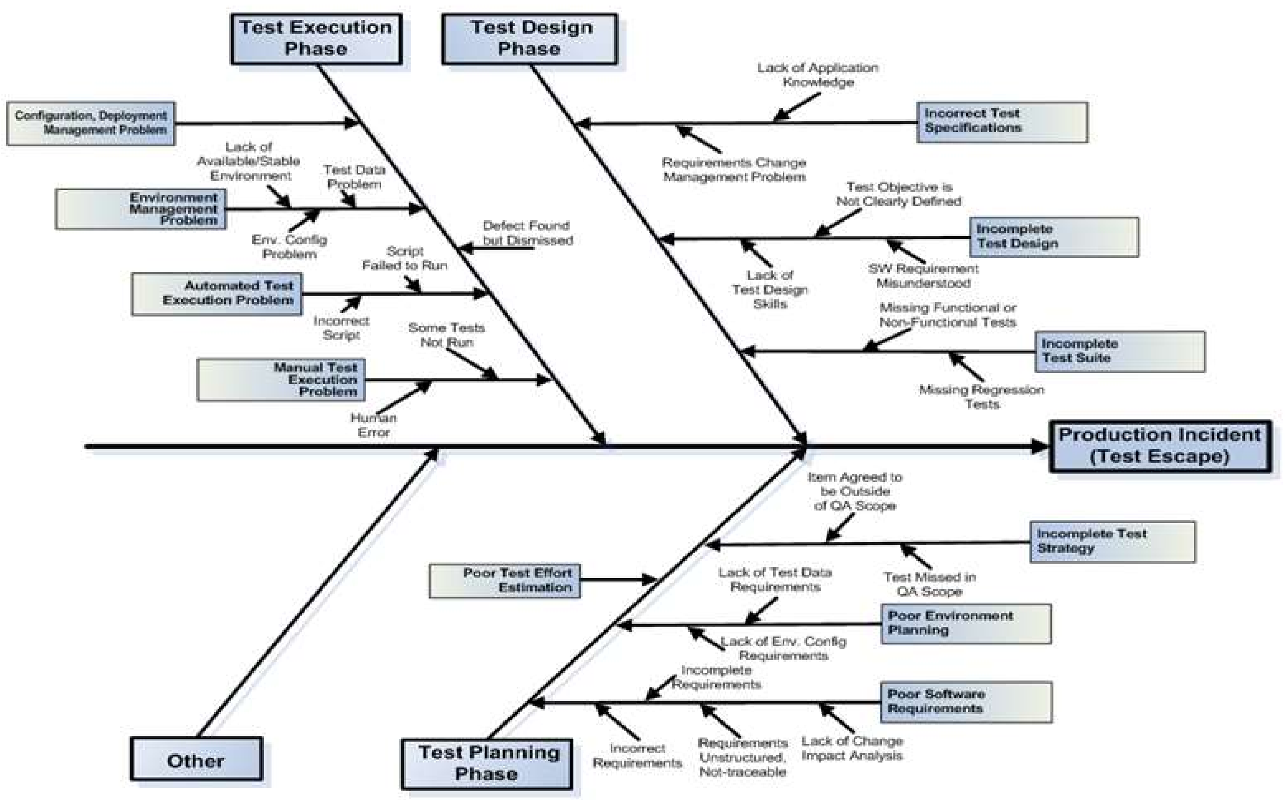
In this example, the Fishbone diagram shows application modules from the functional decomposition and the initial set of core features.



Fishbone Diagram Example 2

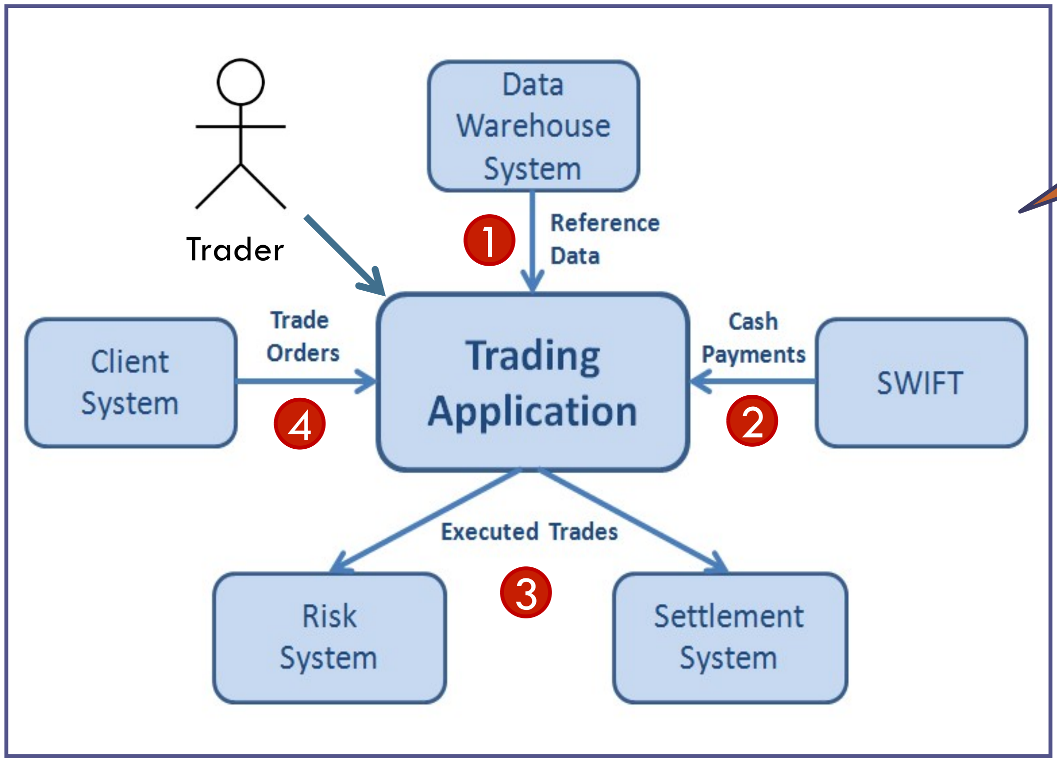
In this example, the Fishbone diagram represents a root-cause & effect model used to investigate causes of production incidents.

Testing-Related Causes of Production Incidents


Test Execution Phase
Test Design Phase
Configuration, Deployment Management Problem
Lack of Available/Stable Environment
Test Data Problem
Environment Management Problem
Env. Config Problem
Defect Found but Dismissed
Script Failed to Run
Automated Test Execution Problem
Incorrect Script
Some Tests Not Run
Manual Test Execution Problem
Human Error
Lack of Application Knowledge
Incorrect Test Specifications
Requirements Change Management Problem
Test Objective is Not Clearly Defined
Incomplete Test Design
Lack of Test Design Skills
SW Requirement Misunderstood
Missing Functional or Non-Functional Tests
Incomplete Test Suite
Missing Regression Tests
Production Incident (Test Escape)
Item Agreed to be Outside of QA Scope
Incomplete Test Strategy
Poor Test Effort Estimation
Lack of Test Data Requirements
Test Missed in QA Scope
Poor Environment Planning
Lack of Env. Config Requirements
Incomplete Requirements
Poor Software Requirements
Incorrect Requirements
Requirements Unstructured, Not-traceable
Lack of Change Impact Analysis
Other
Test Planning Phase

Context Diagram (DFD Level 0) Example

- A System Context Diagram (SCD) in software engineering is a diagram that defines the boundary between the system being developed and its environment, showing the main users and external systems that interact with the system.
- All interfaces are enumerated for specification purposes (see next slide).



W4 Deliverable

System Interface Table

- The System Interface Table is a specification that details the transaction flow between two systems (Source -> Target) from a business stakeholder's point of view.
- The System Interface Table details the information that must be transferred from a **Source** system to a **Target** system.

System Interface Table Template

No	Description	Source	Target	Frequency	Validation
1	Reference Data	DWS	Trading Application	End-of-Day	
2	Cash Payments	SWIFT	Trading Application	Systemic Feed	
3	Executed Trades	Trading Application	Risk System Settlement System	Systemic Feed	
4	Trade Orders	Client System	Trading Application	Systemic Feed	

Data Flow Diagram (Level 1) Explained

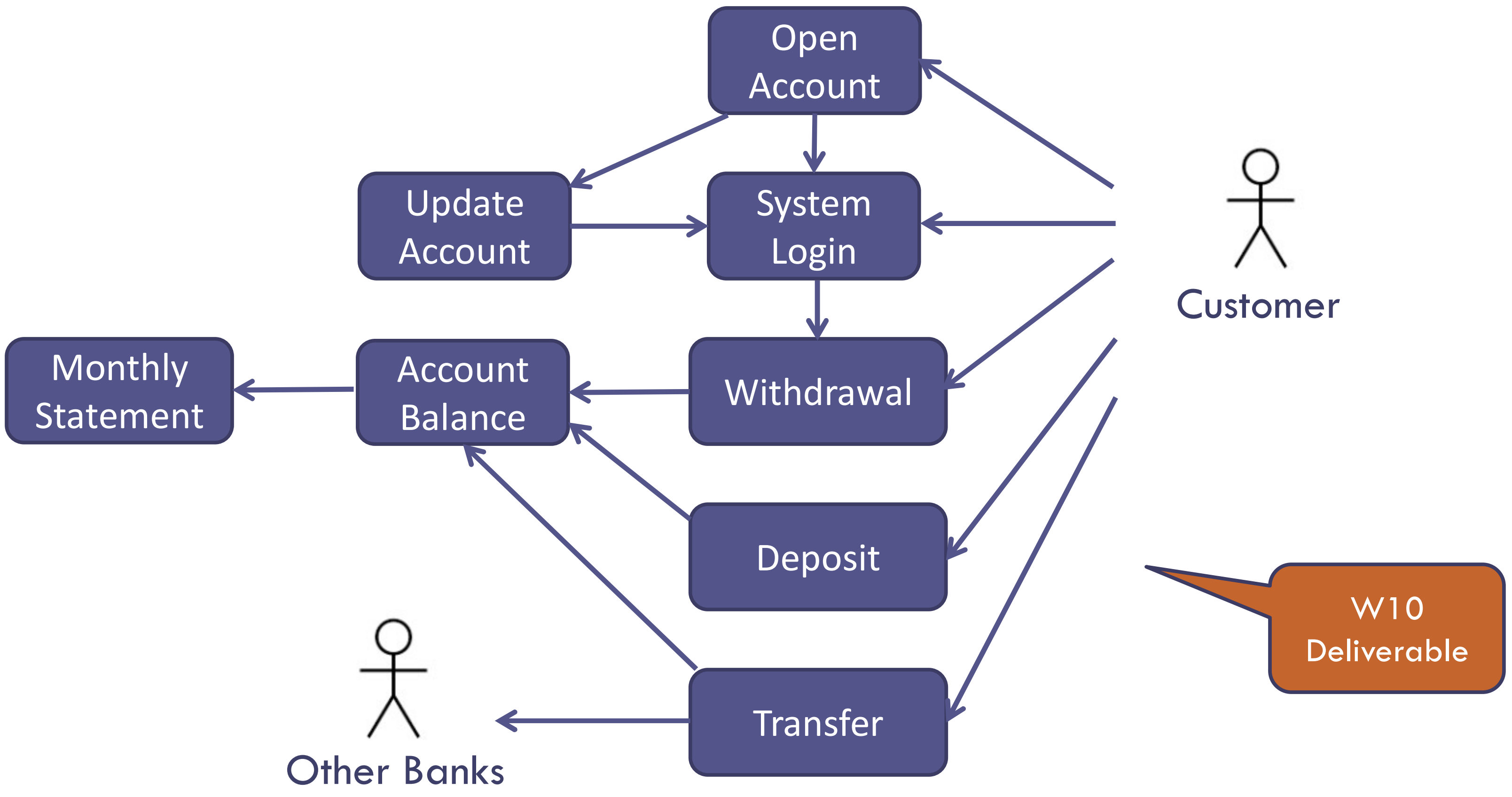
- Larry Constantine introduced data flow diagrams (DFD) in the 1970s as part of the Structured Analysis and Design methodology.
- A data flow diagram is an effective technique to analyze data exchange within a small group of related features.
- DFD can be logical or physical, depending on the level of detail:
  - **Logical DFD** is intended for communication to business users;
  - **Physical DFD** is intended for designers and testers, it shows how the system will be implemented, and which data tables are used;

Guidelines to create a DFD

- A DFD should have at least one process (feature) and one actor;
- A process should receive at least one data flow coming in or going out;
- A data store should be connected to at least one process;
- External entities, Actors should not be connected to each other;

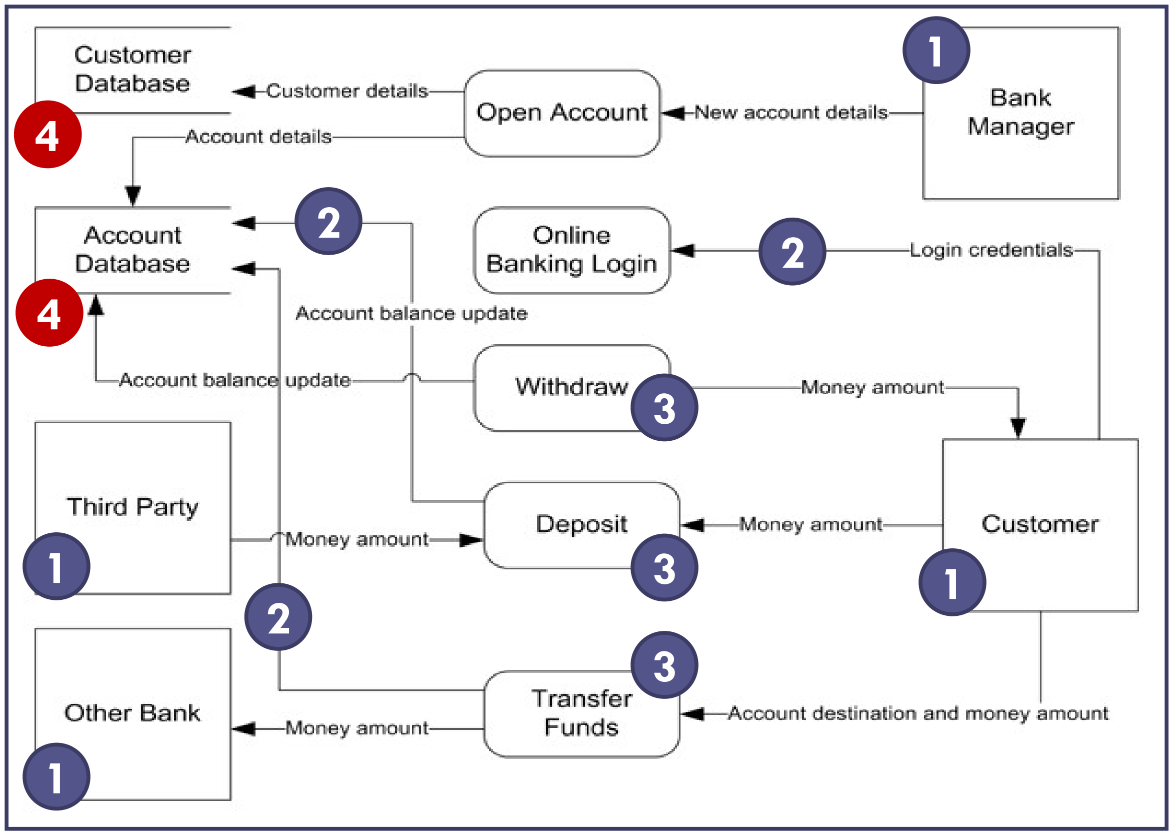
Logical DFD Example

A **logical DFD** describes business functions and the data produced by or supplied to other functions.



Physical DFD Example

The **physical DFD** shows how the system will be implemented, including data stores (tables) needed to implement each function or feature.



1 Source and destination of data

2 Flow of data

3 Process/Feature

4 Data stores

What Are Use Cases?

OMG UML Specification (v.1.5, March 2003) –

“The use case construct is used to define the behavior of a system without revealing its internal structure. Each use case specifies a sequence of actions that the system can perform interacting with its actors.”

A. Cockburn, “Writing Effective Use Cases”, 2001 –

“A use case captures a contract between the stakeholders of a system about its behavior. The use case describes the system’s behavior under various conditions as the system responds to a request from a stakeholder, called primary actor.”

K. Bittner, I. Spencer “Use Case Modeling”, 2003 –

“Use cases represent the things of value that the system performs for its actors.”

Use Case Types and Relationships

Perspective #1 – Problem Domain

- Business Use Cases (*capture business flow and operations*)
- System Use Cases (*capture software requirements to build a system*)
    - Application Use Cases *(address functional requirements)*
    - Infrastructure Use Cases *(address non-functional requirements)*

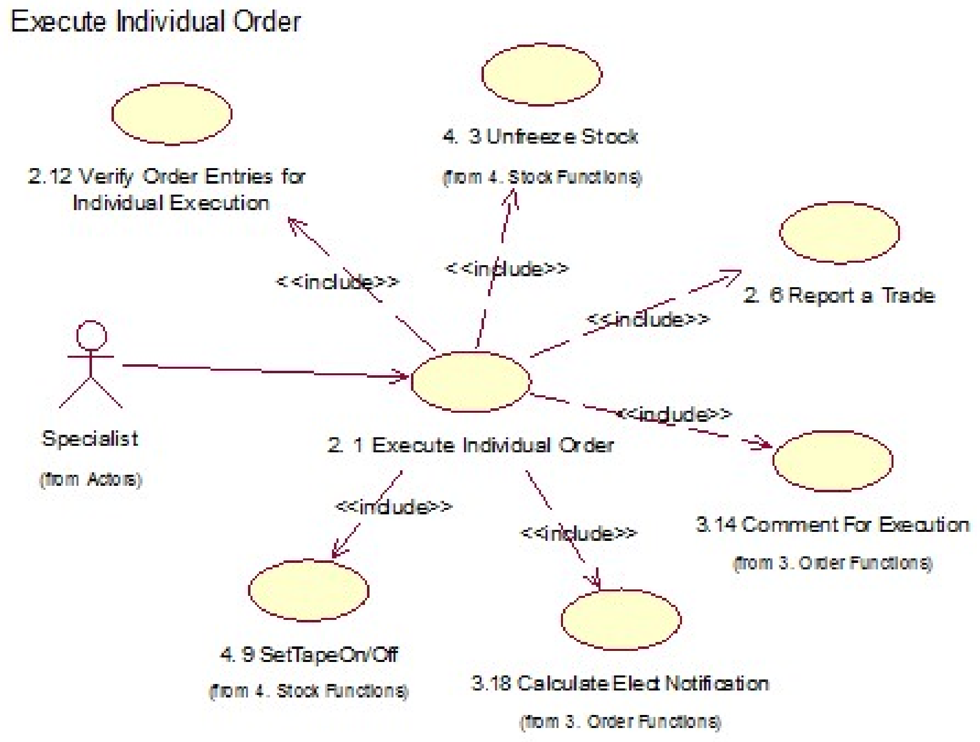
Perspective #2 – Use-Case Relationships

- Generalization Use Cases (*capture a generalized behavior*)
- Specialization Use Cases (*capture a specialized behavior*)
- Base Use Cases (*can have one or more inclusions or extensions*)
- Inclusion Use Cases (*capture functionality included in and reused by other base use cases*)
- Extension Use Cases (*extend functionality of a base use case*)

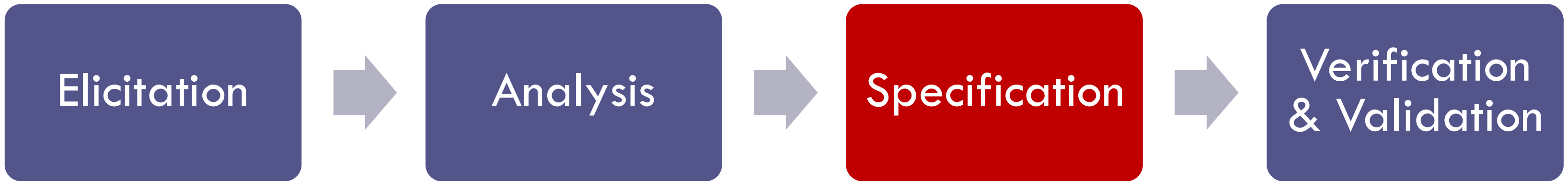
Use Case is not a type of requirements. It is rather a style to document requirements that can be applied to both **Problem and Solution** domains.

Use Case Diagram Example

- A use case diagram shows an **Actor** interacting with a **Use Case**.
- In addition, the diagram can show multiple use cases related via the UML relationships <<include>> or <<extend>>.


Execute Individual Order
2.12 Verify Order Entries for Individual Execution
4. 3 Unfreeze Stock
(from 4. Stock Functions)
2. 6 Report a Trade
Specialist
(from Actors)
2. 1 Execute Individual Order
3.14 Comment For Execution
(from 3. Order Functions)
4. 9 SetTapeOn/Off
(from 4. Stock Functions)
3.18 Calculate Elect Notification
(from 3. Order Functions)
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>

Requirements Specification Phase



The next slides will discuss the requirements specification practices.

Requirements Specification Practices

- Requirements Specification is the process of capturing requirements in a formal specification document.
- This process phase includes both requirements specification and requirements management practices. Examples of common practices:

Requirements Specification	**Requirements Management**
Develop product requirements.	Adopt templates for requirements.
Verify completeness of product requirements.	Manage the development of product requirements.
Identify product requirements for reuse.	Establish and maintain traceability of product requirements. *(W6 deliverable)*
Reuse, modify existing product requirements.	Manage requirements changes and maintain versions of requirements artifacts.
	Baseline, approve product requirements for releases.

Common Forms of Requirements Documentation

- Product requirements are commonly captured in a natural language.
- Common forms of documenting requirements include:
  - IEEE Std.830 “Recommended Practice for Software Requirements Specifications”, a.k.a., an SRS style.
  - Company-specific template, Functional Specification Document (FSD).
  - Use Case specification
  - User Story
- Product requirements can also be captured using formal techniques, e.g., Decision Table, State-Transition Table, etc.
- *Templates for requirements specifications are available in Classes portal, “Supplementary Materials” module.*

IEEE Std.830 – SRS Template

Table of Contents

SRS – Software Requirements Specification

Company-specific BRD Template Example

BRD – Business Requirements Document

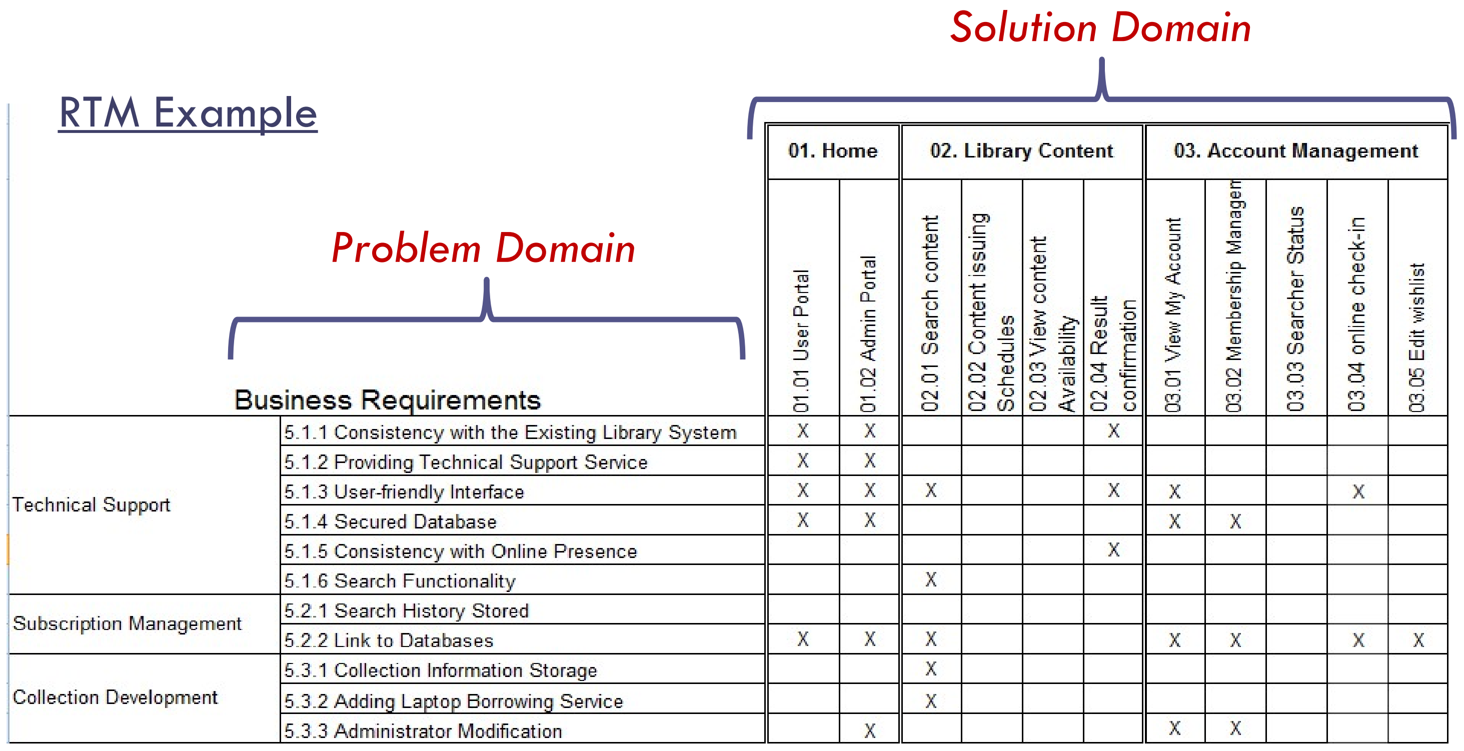
Table of Contents

1. Document Information
   a) Audience
   b) Naming Convention
   c) Additional Instructions
   d) Related Documents
2. Business Opportunity
   a) Background
   b) Current State Analysis
   c) Future State Objectives
3. Business Requirements
   a) Business Scope
   b) Detailed Business Requirements
   c) External Data Feeds (interfaces)
   d) Non-Functional Requirements
   e) Business Risks

Requirements Traceability Matrix (RTM)

- A **traceability matrix** is a document, usually in the form of a table, used to assist in determining the **completeness** of a relationship by correlating any two documents using a many-to-many type of relationship.
- It is often used for demonstrating complete coverage of high-level business requirements by detailed functional requirements of the product.

RTM Example


Solution Domain
Problem Domain

Business Requirements		01. Home		02. Library Content				03. Account Management				
		01.01 User Portal	01.02 Admin Portal	02.01 Search content	02.02 Content issuing Schedules	02.03 View content Availability	02.04 Result confirmation	03.01 View My Account	03.02 Membership Managem	03.03 Searcher Status	03.04 online check-in	03.05 Edit wishlist
Technical Support	5.1.1 Consistency with the Existing Library System	X	X				X					
	5.1.2 Providing Technical Support Service	X	X									
	5.1.3 User-friendly Interface	X	X	X			X	X			X	
	5.1.4 Secured Database	X	X					X	X			
	5.1.5 Consistency with Online Presence						X					
	5.1.6 Search Functionality			X								
Subscription Management	5.2.1 Search History Stored											
	5.2.2 Link to Databases	X	X	X				X	X		X	X
Collection Development	5.3.1 Collection Information Storage			X								
	5.3.2 Adding Laptop Borrowing Service			X								
	5.3.3 Administrator Modification		X					X	X			

Company-specific FSD Template Example

FSD – Functional Specification Document

Table of Contents

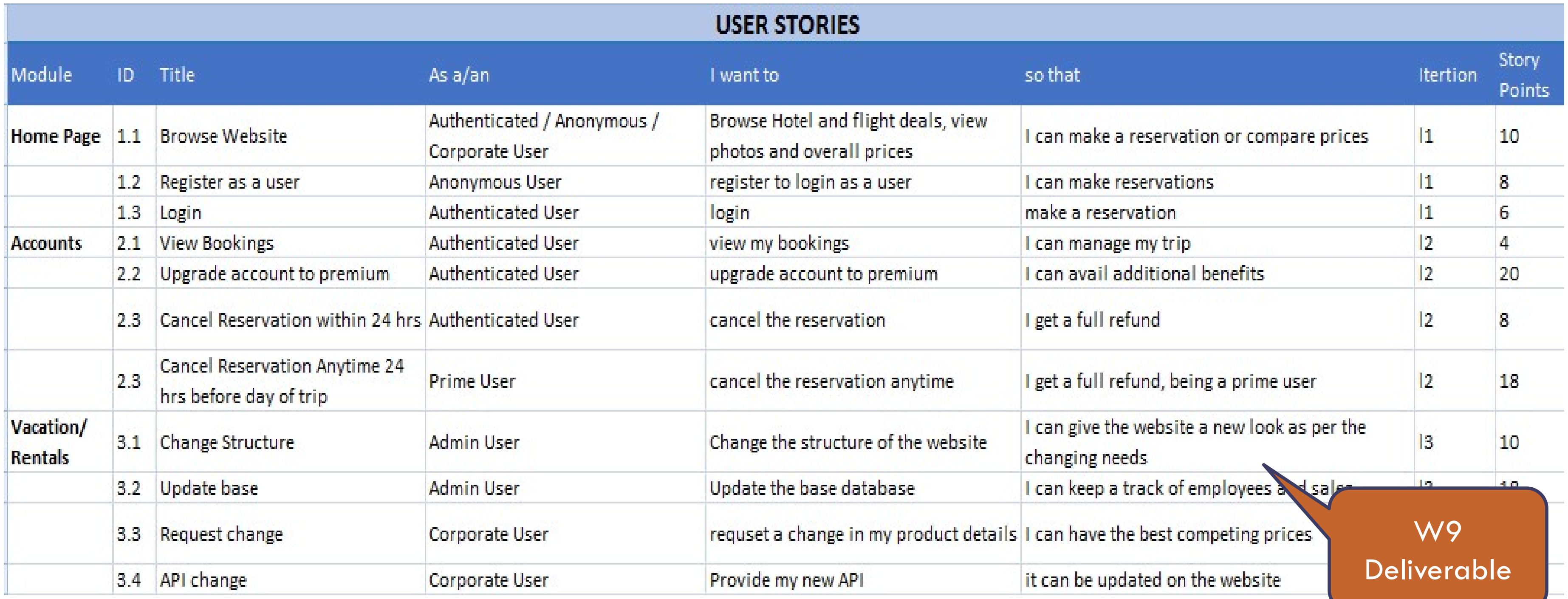
1. Document Information
    a) Audience
    b) Naming Convention
    c) Additional Instructions
    d) Related Documents
2. Background
3. Current State Analysis
4. Gap Analysis
5. Key Constraints
6. Key Assumptions
7. Functional Requirements

This section can be structured by application modules.

Agile User Story Specifications

- A **user story** is an informal, natural language description of one or more features of a software system.
- User stories are often written from the perspective of end-users of a system.
- They are commonly captured in a "user-voice" form shown below:

As a <user role>, I want to <activity> so that <business value>



USER STORIES

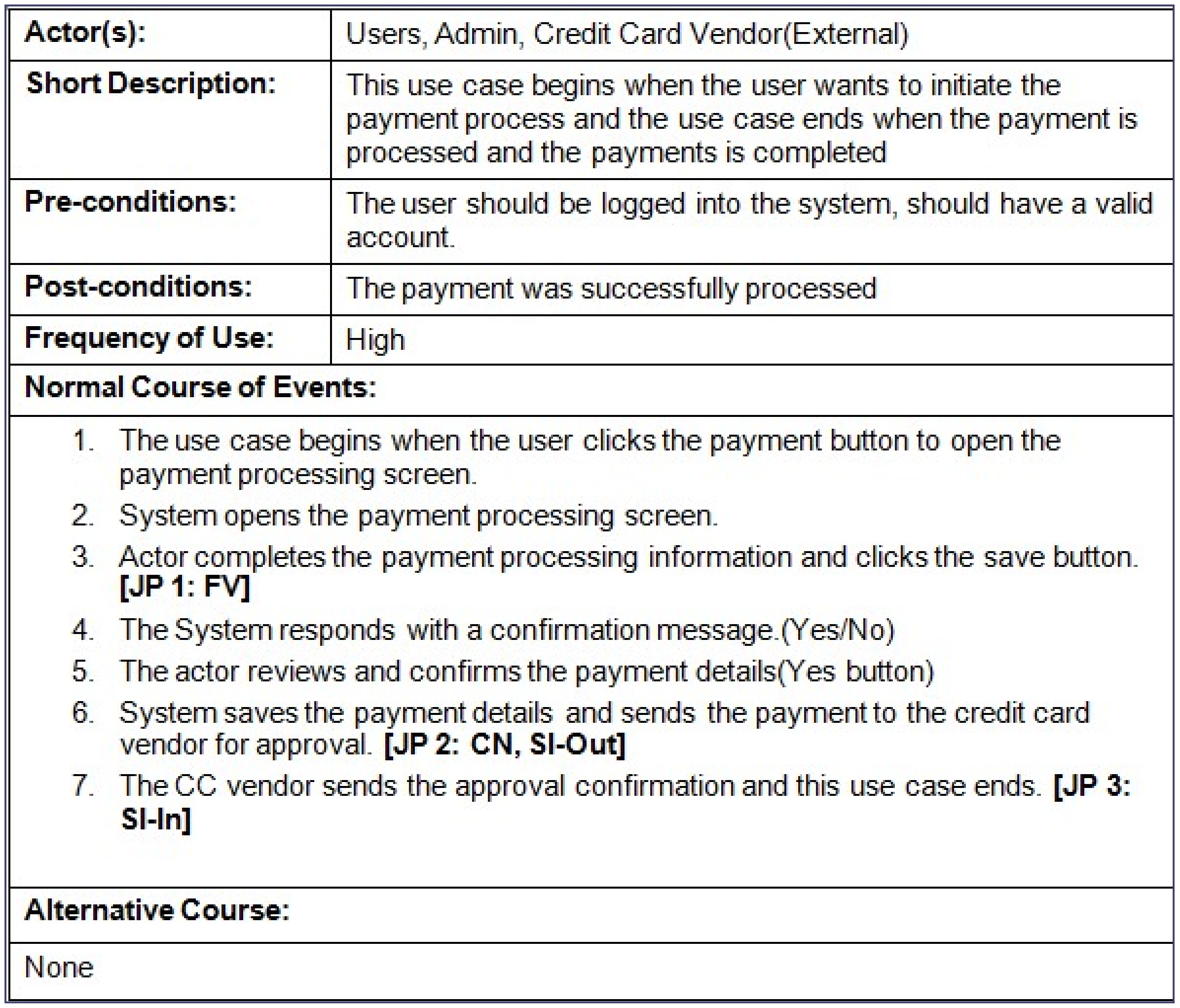
Module	ID	Title	As a/an	I want to	so that	Itertion	Story Points
Home Page	1.1	Browse Website	Authenticated / Anonymous / Corporate User	Browse Hotel and flight deals, view photos and overall prices	I can make a reservation or compare prices	I1	10
	1.2	Register as a user	Anonymous User	register to login as a user	I can make reservations	I1	8
	1.3	Login	Authenticated User	login	make a reservation	I1	6
Accounts	2.1	View Bookings	Authenticated User	view my bookings	I can manage my trip	I2	4
	2.2	Upgrade account to premium	Authenticated User	upgrade account to premium	I can avail additional benefits	I2	20
	2.3	Cancel Reservation within 24 hrs	Authenticated User	cancel the reservation	I get a full refund	I2	8
	2.3	Cancel Reservation Anytime 24 hrs before day of trip	Prime User	cancel the reservation anytime	I get a full refund, being a prime user	I2	18
Vacation/ Rentals	3.1	Change Structure	Admin User	Change the structure of the website	I can give the website a new look as per the changing needs	I3	10
	3.2	Update base	Admin User	Update the base database	I can keep a track of employees a d sale	[illegible]	[illegible]
	3.3	Request change	Corporate User	requset a change in my product details	I can have the best competing prices	[illegible]	[illegible]
	3.4	API change	Corporate User	Provide my new API	it can be updated on the website	[illegible]	[illegible]

W9
Deliverable

Use-Case Template Example

Document Section	Description
Use Case Name (Title)	Enter the goal of the use case – preferably as a short, active verb phrase, serves as a Title of a use case.
Short Description	Describe the goal and context of this use case. This is usually an expanded version of what you entered in the “Title” field.
Actor(s)	A person or a software/hardware system that interacts with your system to achieve the goal of this use case.
Precondition	Describe the state the system is in before the first event in this use case.
Postcondition	Describe the state the system is in after all the events in this use case have taken place.
Normal Flow of Events (Main scenario)	Describe the flow of events from pre-conditions to post-conditions, when nothing goes wrong.
Sub- and Alternative Flows, Exceptions	Describe an alternative flow of evens or exceptions from a specific step in the Main scenario.
Relationships	Reference other use cases that have relationships (<<include>>, <<extend>>) with this use case.

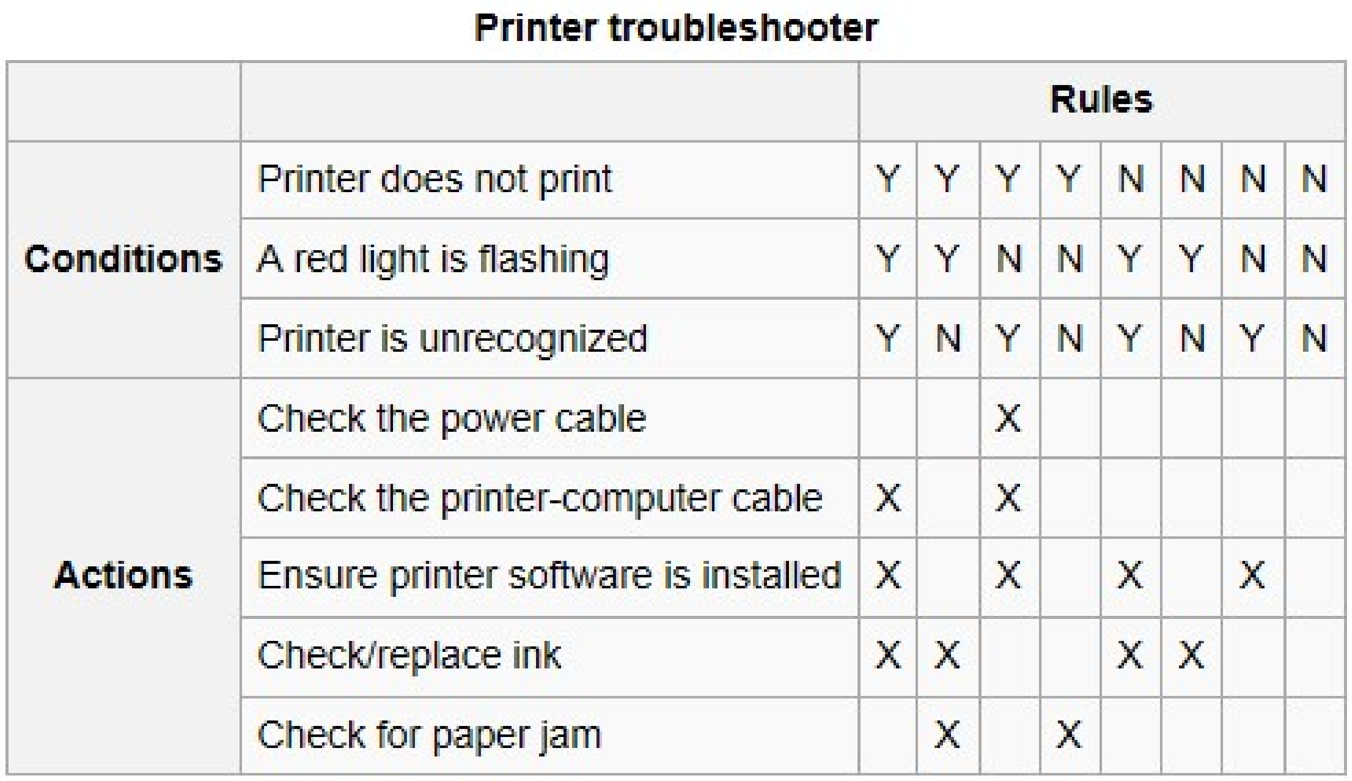
Use Case Specification Example



Actor(s):	Users, Admin, Credit Card Vendor(External)
Short Description:	This use case begins when the user wants to initiate the payment process and the use case ends when the payment is processed and the payments is completed
Pre-conditions:	The user should be logged into the system, should have a valid account.
Post-conditions:	The payment was successfully processed
Frequency of Use:	High
Normal Course of Events:	
1. The use case begins when the user clicks the payment button to open the payment processing screen. 2. System opens the payment processing screen. 3. Actor completes the payment processing information and clicks the save button. **[JP 1: FV]** 4. The System responds with a confirmation message.(Yes/No) 5. The actor reviews and confirms the payment details(Yes button) 6. System saves the payment details and sends the payment to the credit card vendor for approval. **[JP 2: CN, SI-Out]** 7. The CC vendor sends the approval confirmation and this use case ends. **[JP 3: SI-In]**	
Alternative Course:	
None	

Decision Table Example

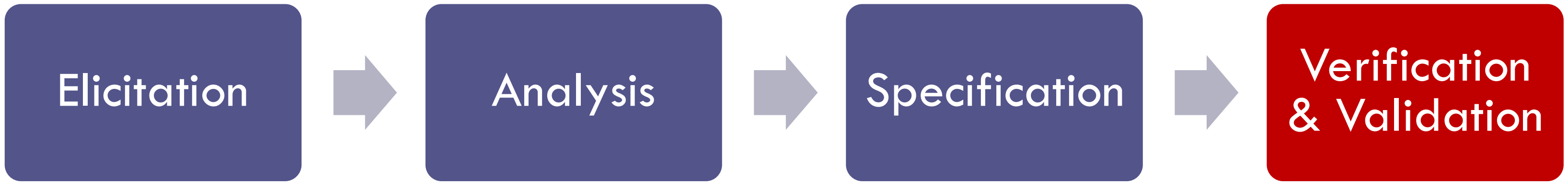
Decision tables are a precise yet compact way to model complex business rules as a combination of conditions and their corresponding actions.



Printer troubleshooter

		Rules							
Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Check the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

Requirements V & V Phase



The next slides will discuss the requirements V & V practices.

Requirements Verification and Validation

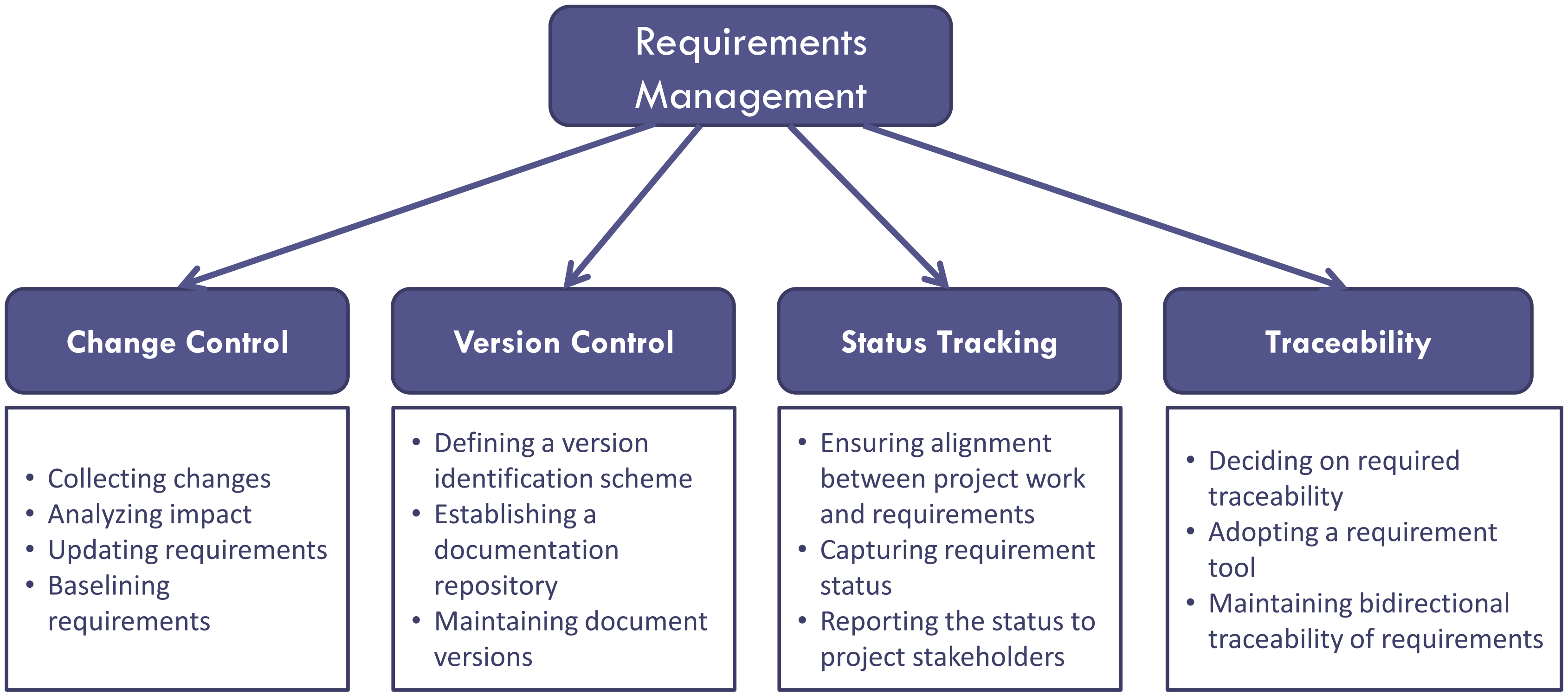
Activities Requirements **Verification** and **Validation** (V&V) refer to checking requirements quality:

- Verification – this activity is concerned with checking that a requirements document is consistent with an approved template and requirements descriptions are complete and correct.
- Validation – this activity is concerned with demonstrating that requirements capture information that the customer really wants.

Verification Practices	**Validation Practices**
Define guidelines and standards for developing requirements.	Develop system prototypes to validate requirements.
Conduct requirements reviews or formal inspections.	Review requirements with end-users.
Document and analyze the review data.	Plan and conduct User Acceptance Testing.

Summary of Requirements Management

- On software projects, requirements represent critical assets and should be managed effectively.
- The Requirements Management process includes the following practices:



Change Control
- Collecting changes
- Analyzing impact
- Updating requirements
- Baselining requirements

Version Control
- Defining a version identification scheme
- Establishing a documentation repository
- Maintaining document versions

Status Tracking
- Ensuring alignment between project work and requirements
- Capturing requirement status
- Reporting the status to project stakeholders

Traceability
- Deciding on required traceability
- Adopting a requirement tool
- Maintaining bidirectional traceability of requirements

Key Points

- Requirements for a software system specify what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of application features.
- Non-functional requirements, a.k.a., quality attributes, often constrain the system being developed.
- The requirements engineering process includes four phases - requirements elicitation, requirements analysis, requirements specification, and requirements verification and validation.
- Change impact analysis is a process of identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change.
- Business, organizational, and technical changes lead to changes to the existing requirements. Requirements management is the process of managing and controlling these changes.

Exercises

1. Explain the phases of the conventional requirements engineering process.
2. Identify and describe types of requirements that can be used for your project.
3. Give examples of functional requirements.
4. Give examples of non-functional requirements.
5. Compare who develops requirements on a conventional vs. agile projects.
6. What diagram can help us identify project stakeholders and support requirements elicitation?
7. Name the three standard parts of a business relationship map template.
8. Name two areas that requirements analysis is concerned with.
9. Name some diagrams that can be produced in requirements analysis.
10. What is a use case? What types of use cases can be produced?
11. Describe a use case specification template.
12. What is the goal of change impact analysis (CIA)?
13. Describe how you would plan and conduct a CIA process.
14. Explain, what requirements management is concerned with.
15. Explain the difference between requirements verification and validation.
16. Explain the benefits of creating and maintaining a requirements traceability matrix.

Appendix A: CMMI Requirements Practices

Process Area	Process Goals	Specific Practices
Requirements Development	SG 1 Develop Customer Requirements	SP 1.1 Elicit Needs
		SP 1.2 Transform Stakeholder Needs into Customer Requirements
	SG 2 Develop Product Requirements	SP 2.1 Establish Product and Product Component Requirements
		SP 2.2 Allocate Product Component Requirements
		SP 2.3 Identify Interface Requirements
	SG 3 Analyze and Validate Requirements	SP 3.1 Establish Operational Concepts and Scenarios
		SP 3.2 Establish a Definition of Required Functionality and Quality Attributes
		SP 3.3 Analyze Requirements
		SP 3.4 Analyze Requirements to Achieve Balance
		SP 3.5 Validate Requirements

Appendix A: CMMI Requirements Practices

Process Area	Process Goals	Specific Practices
Requirements Management	SG 1 Manage Requirements	SP 1.1 Understand Requirements
		SP 1.2 Obtain Commitment to Requirements
		SP 1.3 Manage Requirements Changes
		SP 1.4 Maintain Bidirectional Traceability of Requirements
		SP 1.5 Ensure Alignment Between Project Work and Requirements