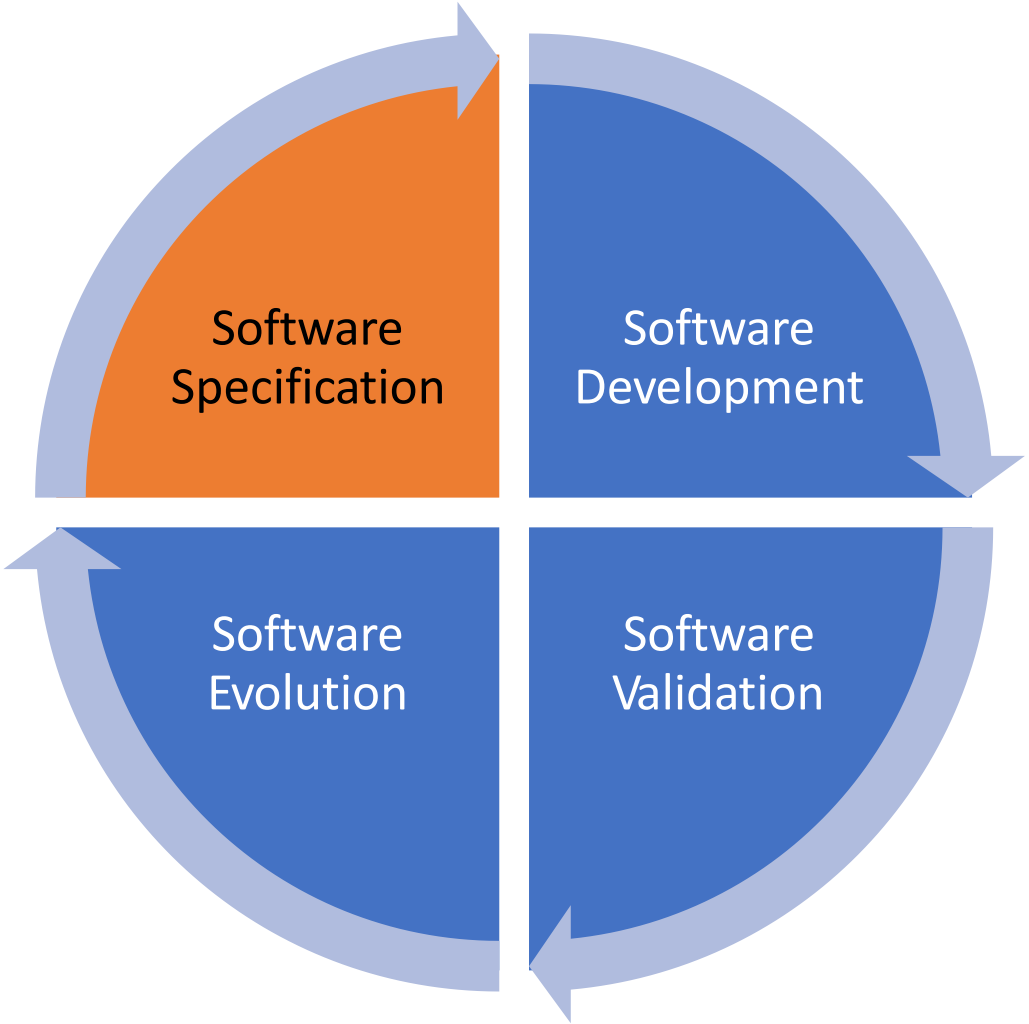
CS335FZ,

Requirements Engineering

Dr. Lanlan Gao

Objectives

- Understand the importance of the requirements engineering
- Get familiar with the concepts and processes of requirements engineering
- Be able to apply the main requirements engineering techniques to simple software system projects



The four fundamental activities that are common to all software processes

Definition

”The process of finding out, analysing, documenting and checking the services and constraints of a software system is called requirements engineering (RE).”

--Sommerville, I., 2011. Software engineering 9th Edition. ISBN-10, 137035152.

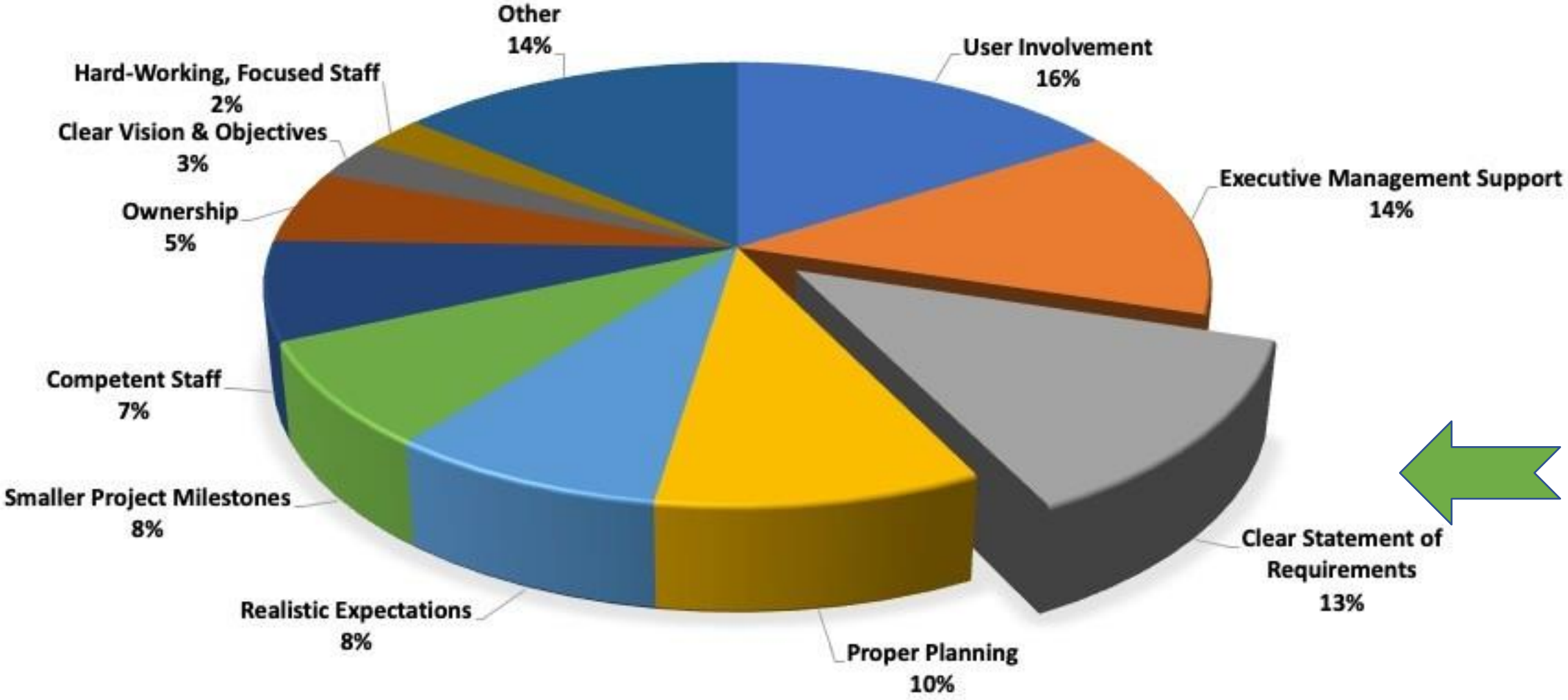
”A statement that identifies a system, product or process characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability.”

--INCOSE Systems Engineering Handbook., 2015. A Guide for System Life Cycle Processes and Activities.

”The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering. It establishes a solid base for design and construction.”

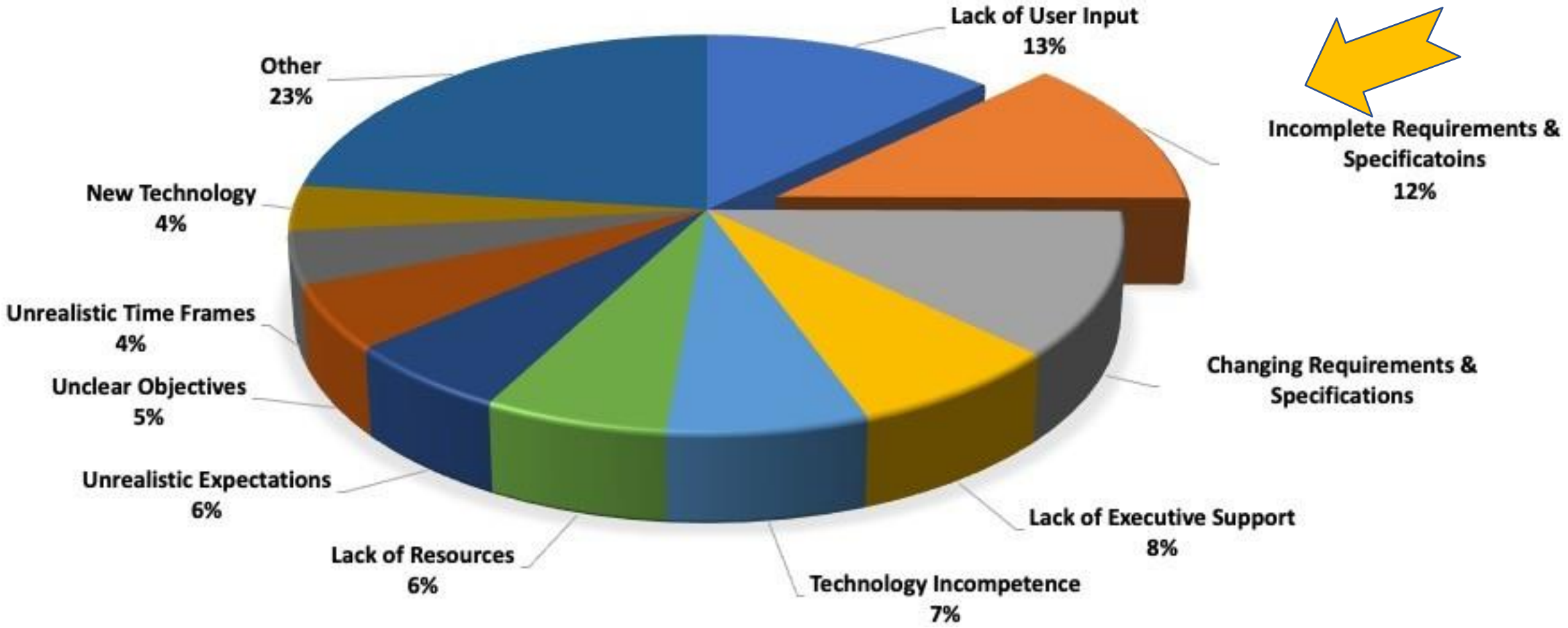
--Pressman, R.S., 2015. Software engineering: a practitioner's approach. McGraw-Hill Education.

The Importance of Requirements Engineering – Project Success Factors



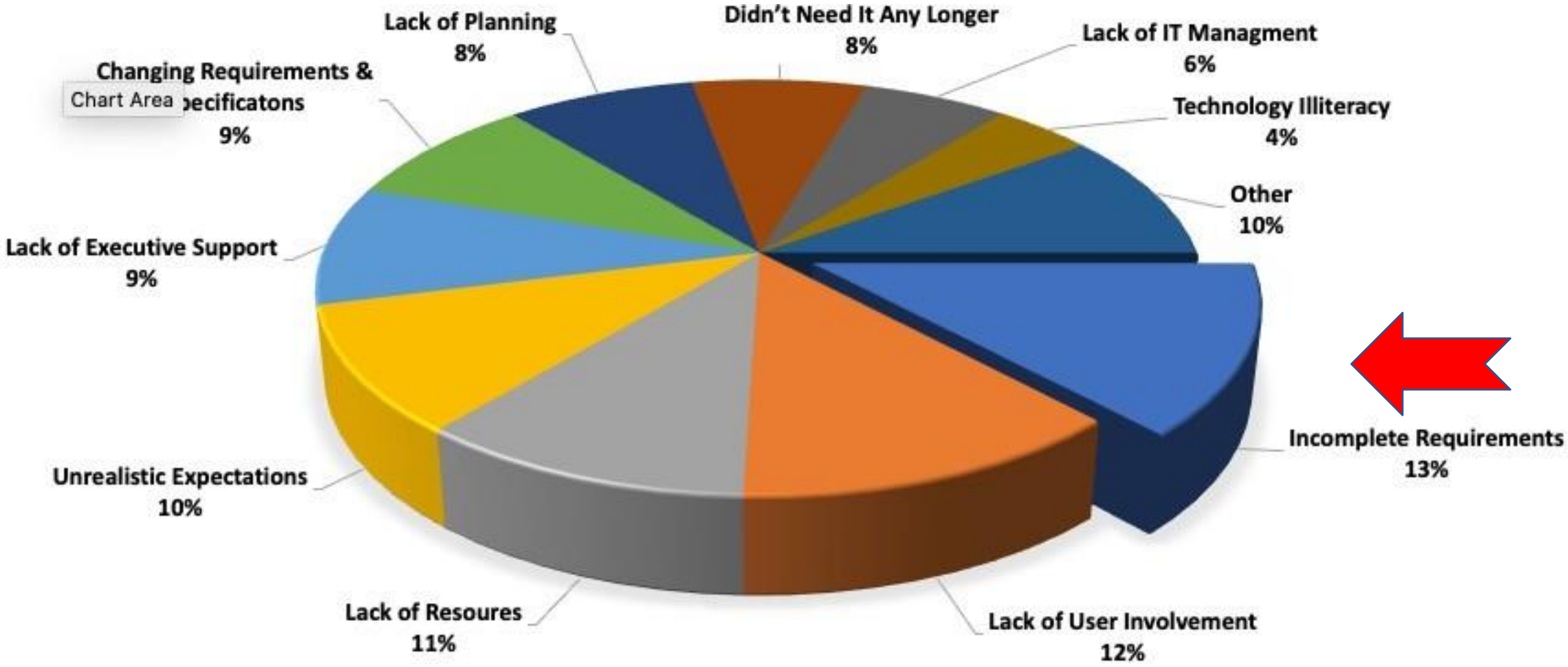
***Reference**: CHAOS, The Standish Group Report, Project Smart, 2014*

The Importance of Requirements Engineering – Project Challenged Factors



***Reference**: CHAOS, The Standish Group Report, Project Smart, 2014*

The Importance of Requirements Engineering – Project Impaired Factors



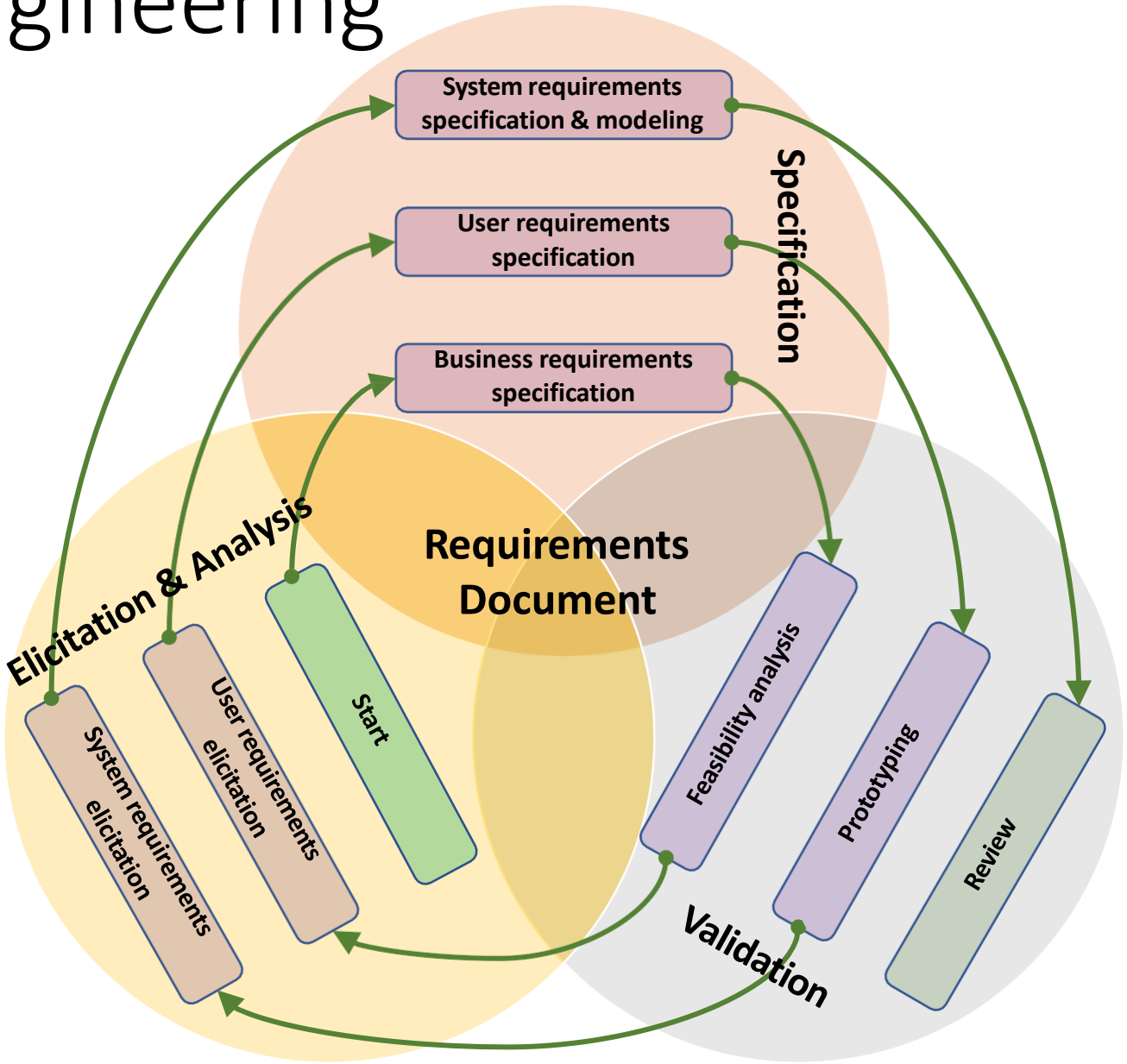
***Reference**: CHAOS, The Standish Group Report, Project Smart, 2014*

The Requirements Engineering Processes

- Requirements engineering is an iterative process
  - activities are interleaved
  - requirements documents are produced for project ***stakeholders***

- Three key activities:

1. Elicitation and analysis: interact with stakeholders, discovering the user and system requirements.
2. Specification: convert requirements into standard form
3. Validation: assess feasibility of the project; building prototype if needed; review requirements



Terminology: Stakeholder

"an individual, group, or organization, who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project"

--Project Management Institute (PMI), 2013

"A stakeholder is anyone who has a stake in the success of the system: the customer, the end users, the developers, the project manager, the maintainers, and even those who markets the system for example."

--Bass, L., Clements, P. and Kazman, R., 2003. Software architecture in practice. Addison-Wesley Professional.

Project Scenario:

The Mid-Scotland regional health authority wishes to procure an information system to help manage the care of patients suffering from mental health problems. The overall goals of the system are twofold:

1. To provide better management information about mental healthcare in the region.
2. To provide an improved records system for clinical staff involved in diagnosis and treatment.

Mid-Scotland regional health authority

Manager

Patients

Clinical staffs

System designer

Software Manager

Software Developer

……

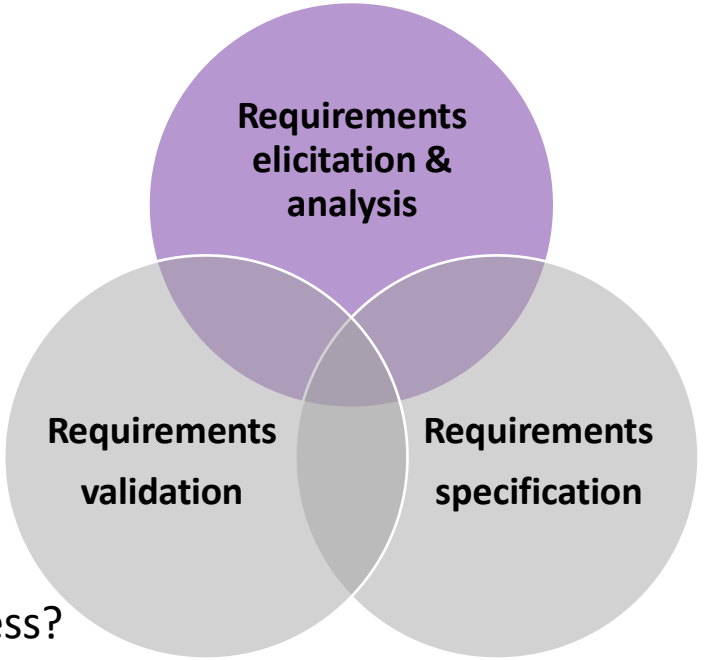
Exercise: Identify Potential Stakeholders

Albatel Ltd is a start-up SME providing HR consultancy specialising in payroll, recruitment and coaching services. The company currently provides services to their customers in a traditional face-to-face approach. The only IT infrastructure owned by the company is the company's website. The construction and management of the website were outsourced. The company's website is based on the *WordPress* platform and hosted by a third-party hosting company. Permissions for configuring the hosting environment to accommodate other customized web applications are very limited.

The company has a long-term plan, aiming to provide their HR related services through an integrated software platform. For example, allowing multiple employee applications feed information (e.g., time, attendance, payslips and pension, etc.) into one unified platform for comprehensive analysis and management. At this initial phase of this transition, the company is exploring some web-based online questionnaire and automated HR health check report generation services. This exploratory service should be easily integrated with the company's existing website without affecting existing functions or incurring major modifications.

The objective of this project is to provide an interactive web-service that allows the company's customers to take online questionaries, and based on their responses received, an analysis report will automatically be generated for the customer to download.

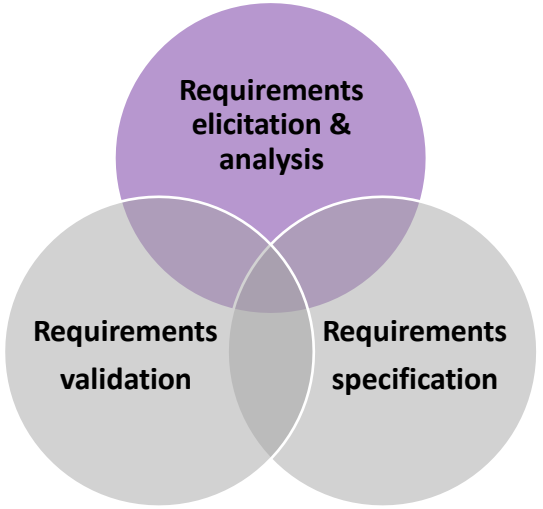
Requirements Elicitation

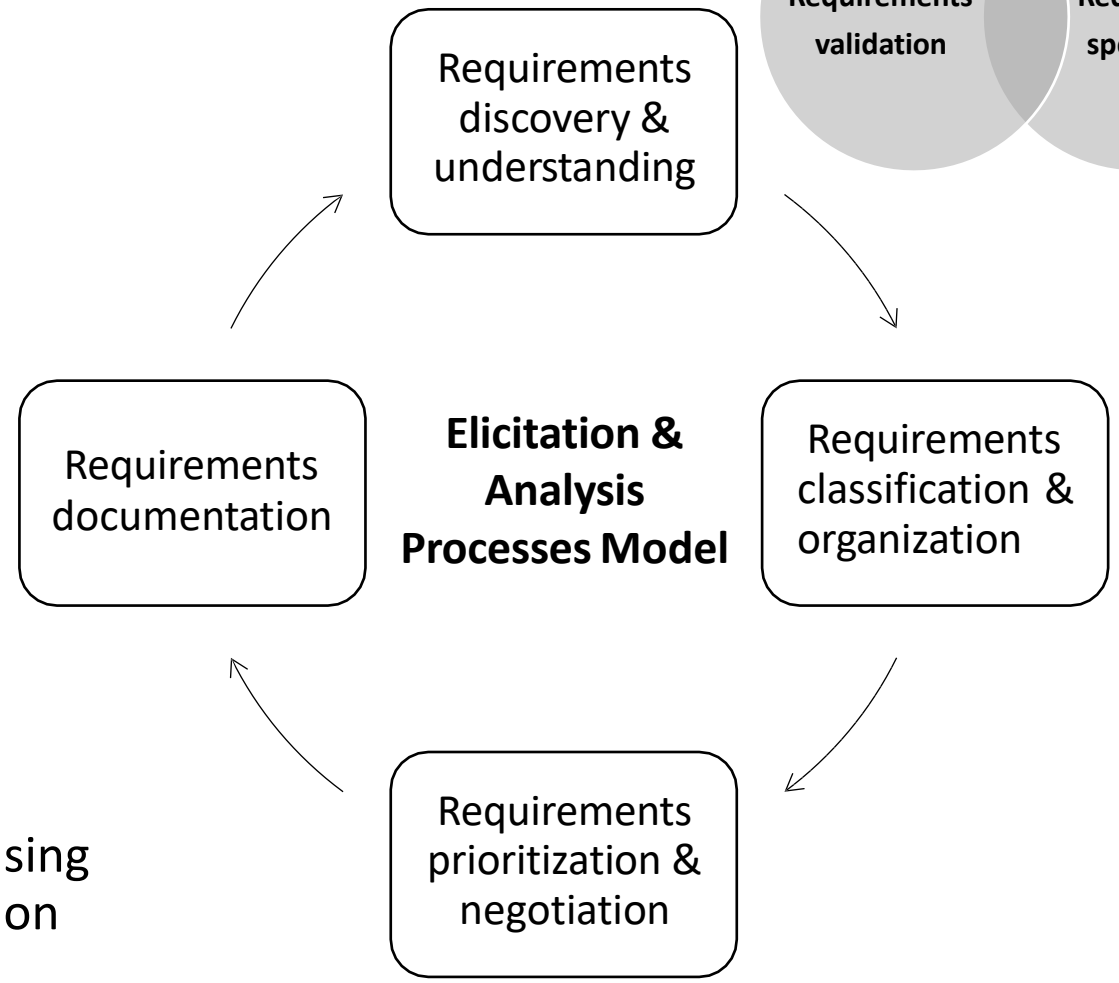


- The objectives of requirements elicitation:
  - To understand the application domain
    - E.g., what are the standard procedures used in a HR payroll system?
  - To identify relevant work activities
    - E.g., what are the activities/steps involved in a staff overtime approval process?
  - To identify service components
    - E.g., the system shall provide a pay slip generation functional component.
  - To identify system features
    - E.g., the system shall be highly available so that it can survives from a database crash.
  - To identify constraints and environment of the system
    - E.g., the system's user interface should be compatible with Firefox Browser (v.20.0 or above) and Safari Browser (v14.0 or above).
  - How the new system is going to be used
    - E.g., the system shall be accessed through web browsers and integrated into the customer's existing IT infrastructure.
  - How the new system is going to help the users
    - E.g., the system shall be provided to improve current work efficiency.

A Process Model of Elicitation and Analysis Process

- Discovery and understanding
  - Discover requirements and **domain requirements**
- Classification and organization
  - Take the unstructured collection of requirements, groups related requirements
- Prioritization and negotiation
  - Prioritizing requirements, identify duplicated requirements, and resolving requirements conflicts
- Documentation
  - Requirements maybe formally documented using standard templates or informally maintained on whiteboards, wikis, or other shared spaces

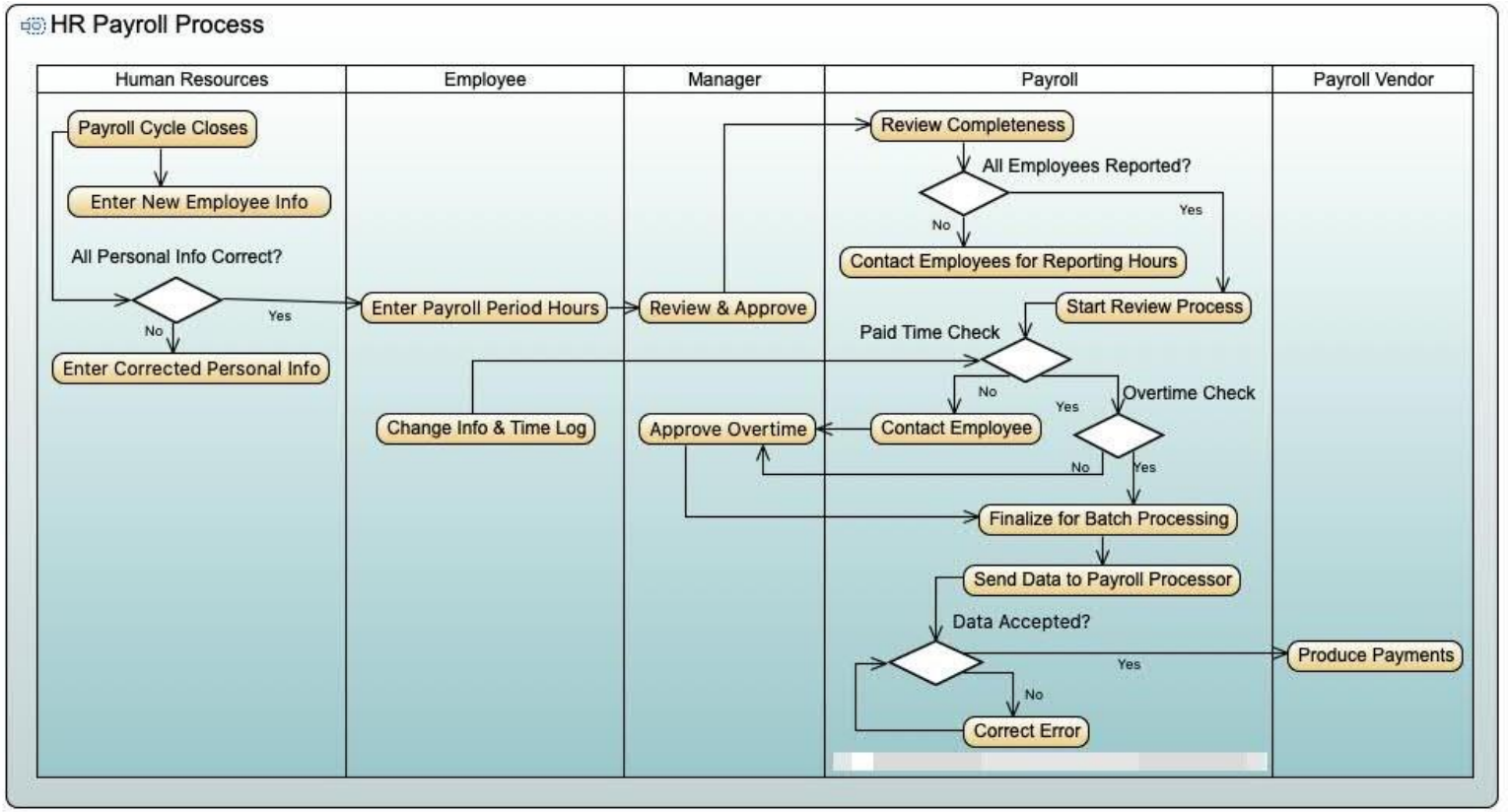




Terminology: Domain Requirements

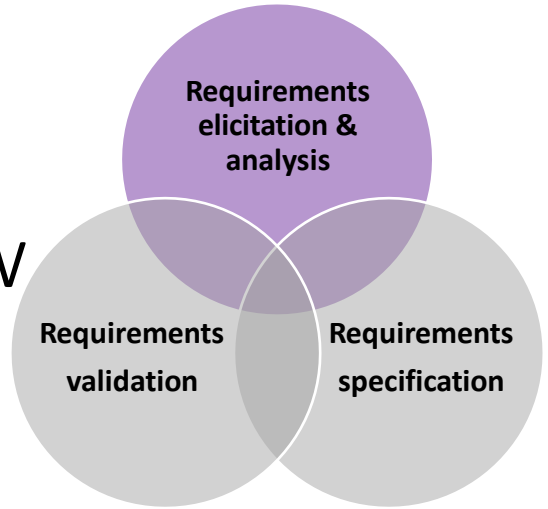
> *"Domain requirements are derived from the application domain of the system rather than from the specific needs of system users."*
>
> *--Sommerville, I., 2011. Software engineering 9th Edition. ISBN-10, 137035152.*

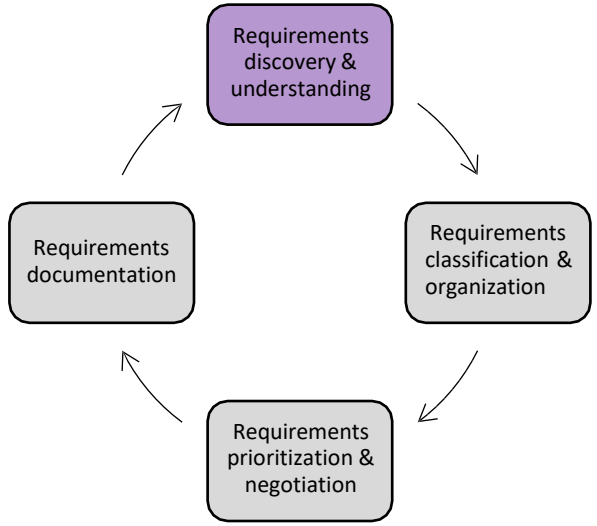
- Stakeholders often don't clearly know what they want from a computer system
- Stakeholders often naturally express requirements in their own terms that we don't understand
- Different stakeholders of the same project may express their requirements in different ways
- Political and/or regulatory factors may influence the requirements of the system
- For example*, the system safety shall be assured according to standard IEC 60601-1:Medical Electrical Equipment – Part 1:General Requirements for Basic Safety and Essential Performance*



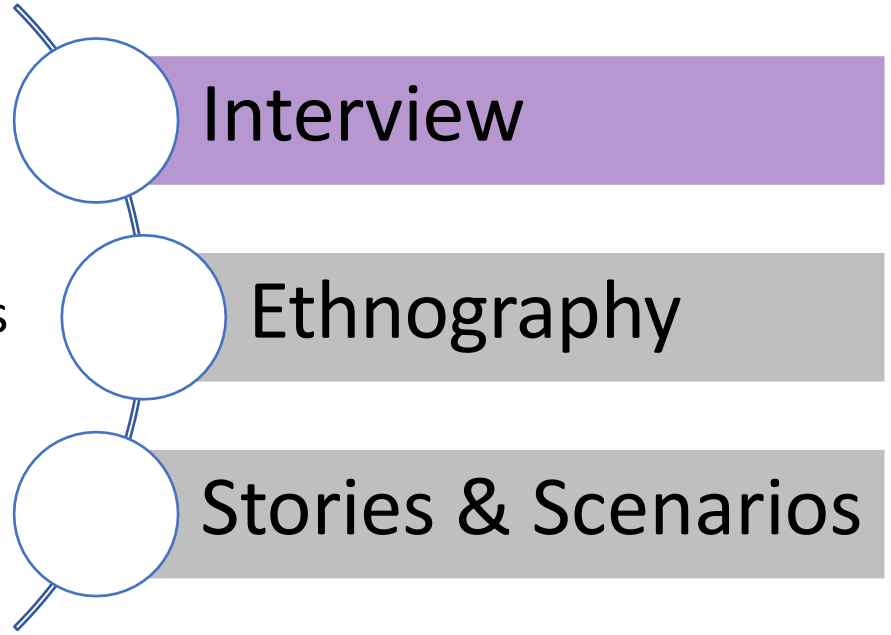
Activity Diagram

Techniques for Requirements Elicitation & Analysis – Interview





- Closed Interviews
  - Stakeholders answers a predefined set of questions
    - E.g., what is the current workflow?
    - How do you want to improve the current workflow using the new system?
    - What worries you about this project?
    - Does the current system rely on particular technologies?
- Open Interviews
  - No predefined set of questions, free form discussions with stakeholders
- Aim to get an overall understanding of what users do, and how they might interact with the new system

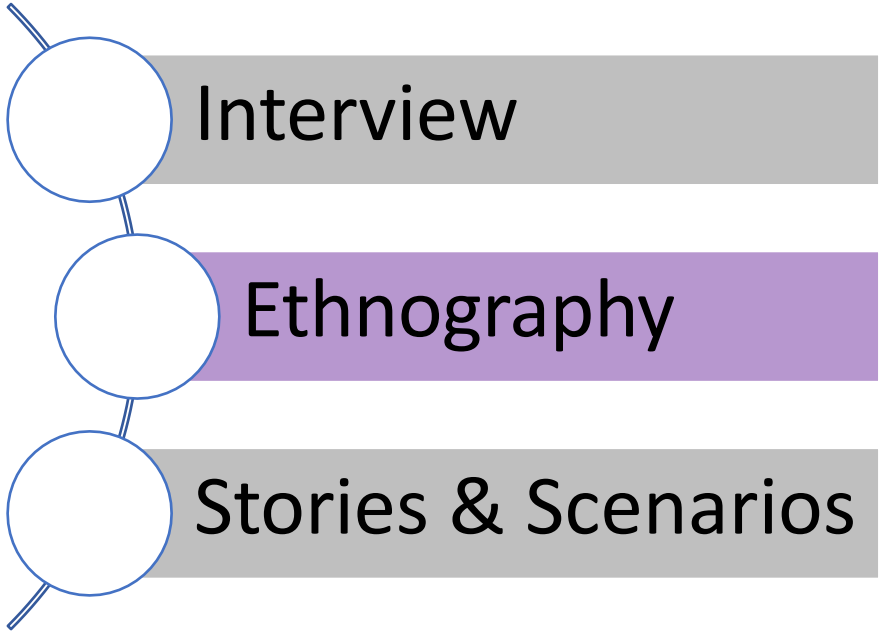


Techniques for Requirements Elicitation & Analysis – Ethnography

"Ethnography is an observational technique that can be used to understand operational processes and help derive requirements for software to support these processes."

--Sommerville, I., 2011. Software engineering 9th Edition. ISBN-10, 137035152.

1. Immerse yourself in the working environment
2. Observe day-to-day work and taking notes
3. Discover how people actually work, rather than the pre-defined (standard) business processes
4. Discover implicit requirements of the system



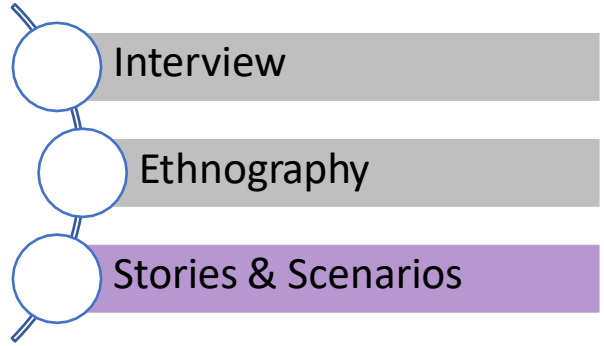
Reading (Ethnography): Viller, S. and Sommerville, I., 2000. Ethnographically informed analysis for software engineers. *International Journal of Human-Computer Studies*, *53*(1), pp.169-196.

The Importance of the Observation Activity

1. Actual work practices are far richer, more complex, and more dynamic than the models assumed
2. People often find it very difficulty to clearly articulate details of their work
3. People may understand their own work, but may not understand its relationship to other work in the organization

Date	Project Name	Type of System	Country	Problem	Cost	Result
1997 - - 2000	Bolit	Customer service, finance and administration system	Sweden	Bad functioning, too complicated, cost overrun	$35 M	Scrapped, **NEVER USED**
1999 - - 2006	CSIO Portal	Common technological platform for brokers and insurers.	Canada	Incompatible functions between users, time overrun, cost overrun.	$15 M	Abandoned, **LOW USER ADOPTION**

Techniques for Requirements Elicitation & Analysis – Stories and Scenarios



Stories

- Written as narrative text
- Present a high-level description of system use
- Effective in setting out the "*big picture*"

In common between Stories and Scenarios

- Describe how the system can be use for some particular task
- Describe what people do and how they interact with the system
- Describe what information they use and produce
- Describe what systems they may use in this process

Scenarios

- Structured with specific information, such as *inputs* and *outputs*
- Parts of stories can be developed in more detail and represented as scenarios

Stories

An Example Story (User Story Index Card Management System):

When a user starts the program, a graphical user interface should be displayed. The user must load a valid project file from local disk or network storage. After loading the project file, all existing index cards should be displayed in one of the columns (‘*Todo*’, ’*In Progress*’ or ‘*Done*’), according to the status of the index cards. If the user decided to create a new *User Story*, she/he must select the ‘*create user story*’ option in the menu of the program. The action should results in a pop-up window displayed on top of the main window of the program.

During the editing process, the background saving process must periodically store the uncompleted user story to the project file with a special status ‘*incomplete*’, so that in the event of system crash, all information could be retrieved. When the user clicks on the ‘*confirm*’ button in the pop-up window, all information should be persisted in the project file and the pop-up window should now be hidden.

A high-level description of :

1. how the system can be used
2. what people do
3. what information to use and to produce
4. what other systems that may be used
5. the context of the system

Scenarios

Initial assumption: what the system and users expect when the scenario start.

Normal: the normal flow of events in the scenario.

What can go wrong: what can go wrong & how resulting problems can be handled.

Other Activities: information about other activities that might be going on at the same time.

System state on completion: the system state when the scenario ends.

Template

Initial assumption: the project file has already been created and loaded into the program

Normal: a user selects the '*create user story*' option from the menu of the program, a pop-up window shall display, which allows the user to input all information of the user story. When the user clicks on the '*confirm*' button, the index card shall be stored and subsequently displayed in the '*ToDo*' column in the main frame of the program

What can go wrong: storage system run out space; the maximum number of index card allowance has reached; the project file has been opened by another program.

Other Activities: index card integrity checker may be running at the same time.

System state on completion: all index cards are stored persistently.

Example: Saving index cards

Brainstorming is often needed when creating stories and scenarios.

Summary: Requirements Elicitation and Analysis

Depict the *big picture* of the system and the application domain.

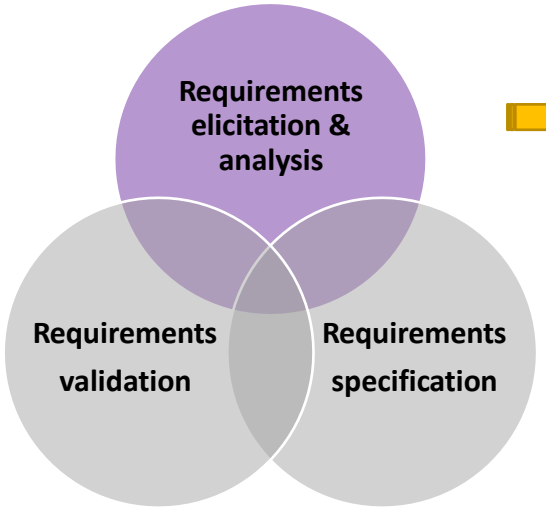
Discover the implicit requirements and the *actual* workflow

Describe the user and system expectations, & workflows, etc. Brainstorming+Innovations

Interview

Ethnography

Stories & Scenarios



Requirements discovery & understanding

Requirements classification & organization

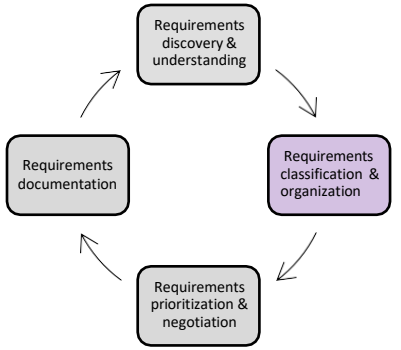
Requirements prioritization & negotiation

Requirements documentation

Pedigree Attribute eLicitation Method (PALM)

Quality Attribute Workshop (QWA)

User Requirements vs. System Requirements



- User Requirements
  - Often written in natural langue with diagrams
  - High-level, abstract statement of a service that a system should provide or a constraint on a system
  - Usually written for non-technical people
- System Requirements
  - Detailed description of software system's services and operational constraints
  - Define exactly what is to be implemented
  - It may be part of the contract between system investors and software developers
  - Usually written for technical people

- Client managers
- Contractor managers
- System end-users
- Client engineers
- …

- System architects
- Software developers
- Product Managers
- Project managers
- …

Reading (The inconsistent use of the term 'requirement'): Davis, A.M., 1993. *Software requirements: objects, functions, and states*. Prentice-Hall, Inc..

The Granularity of Requirement Statements

Version 1:
The User Story Index Card Management System shall automatically and periodically store information to a file on the local hard drive.

Version 2:
1. The system shall automatically store all information to a file on the local disk every 5 minutes.
2. All user stories and their associated status shall be stored as serializable objects, so that they can be transferred to another running instance of the User Story Index Card Management System over the network.
3. The file shall have an extension name '.sesp'.
4. The autosaving component shall be implemented as a background service.

A User Requirement

System Requirements

Functional Requirements and Non-functional Requirements

- Functional Requirements
  - The statements of services that the system should provide
  - How the system should react to particular inputs
  - How the system should behave in a particular situation
  - *Example 1:* The User Story Index Card Management system shall provide an export function allowing users to save all index cards in a PDF file
  - *Example 2:* Data scientists and analysts shall be able to perform ad-hoc data analysis through SQL-like queries to find specific data patterns and correlations to improve infrastructure capacity planning
- Non-functional Requirements
  - The constraints on the services, e.g., timing constraints, resource constraints, constraints imposed by standards
  - *Example 1:* When the system is under peak load, no emergency messages received from the mobile clients shall be dropped, and the cached messages shall be processed as soon as possible and be multi-casted to preselected receivers with no further delay.
  - *Example 2:* The system shall collect up to 15,000 even/sec from approximately 300 web servers.

PDF (Portable Document Format): a cross-platform file format developed by Adobe.

SQL (Structured Query Language, An Interesting fact): Chamberlin, D. D. (Donald Dean). (2001). Oral history interview with Donald D. Chamberlin. Charles Babbage Institute. Retrieved from the University of Minnesota Digital Conservancy, https://hdl.handle.net/11299/107215.

Functional Requirements

- Use natural language to write functional requirements
  - E.g., the big data analytic system shall provide a web-based portal for users to interact with the system
- Functional requirements traditionally focused on what the system should do
  - E.g., for the User Story Index Card Management system, a user shall be able to search index cards via user story ID.
- Depending on the nature of the system to be developed, the focus may be shifted to other aspects of the system
  - E.g., in the HR Online Report Management project, if the organization decided to use a 3rd -party off-the-shelf product, there will be a little to say about the system requirements, but the focus will be on how to format the inputs for the system.

Non-Functional Requirements

- Specify characteristics of the system as a whole.
  - E.g., reliability, performance, security, etc.

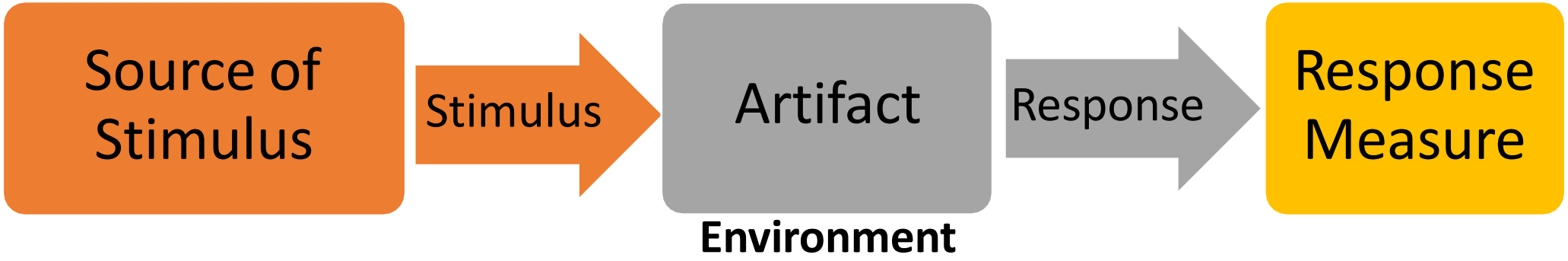
- Non-functional requirements are more critical than individual functional requirements.
  - E.g., if emergency messages received from the mobile clients that couldn’t be processed immediately during system peak load, then the receivers wouldn’t be able to receive messages in time, the system became less useful.

- Non-functional requirements are difficult to implement.
  - E.g., implementing a non-functional requirement may affect the overall architecture of the system.

Measurable Non-Functional Requirements

System Reliability:
When the system is under peak load, no emergency messages received from the mobile clients shall be dropped, and the cached messages shall be processed within 10 seconds and be multi-casted to preselected receivers with no further delay.

Narrative



Quality Attribute Scenario

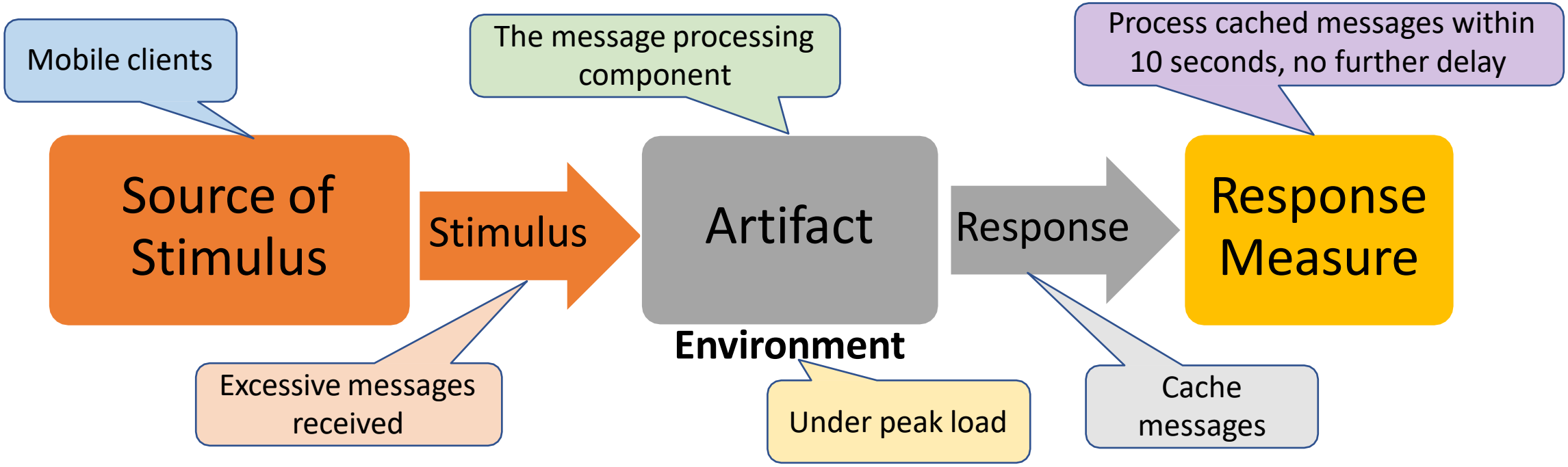
Quality Attribute Scenarios

1. **Source of stimulus:** entities (a human, a computer system, or any other actuator) that generated the stimulus
2. **Stimulus:** a condition that requires a response when it arrives at a system
3. **Environment:** the stimulus occurs under certain conditions. For example, the system is under an overloaded condition or in a normal operation, or some other relevant state
4. **Artifact:** a collection of systems, the whole system, or some pieces of it.
5. **Response:** the activity undertaken as the result of the arrival of the stimulus
6. **Response measure:** when the response occurs, it should be measurable in some ways so that the requirements can be tested.

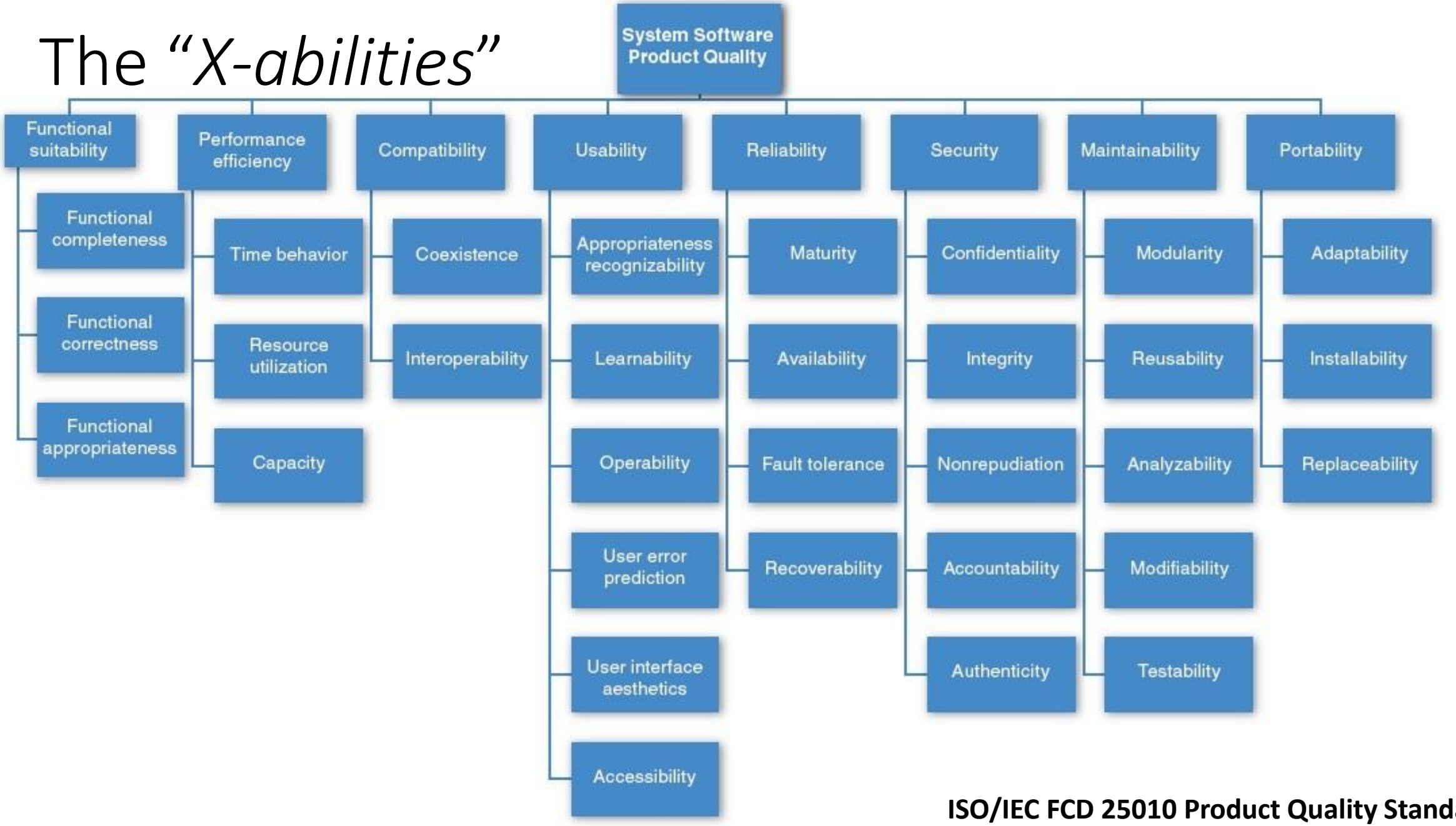
Example: Quality Attribute Scenario for System Reliability

Non-functional Requirements must be testable.

When the system is under peak load, no emergency messages received from the mobile clients shall be dropped, and the cached messages shall be processed within 10 seconds and be multicasted to preselected receivers with no further delay.



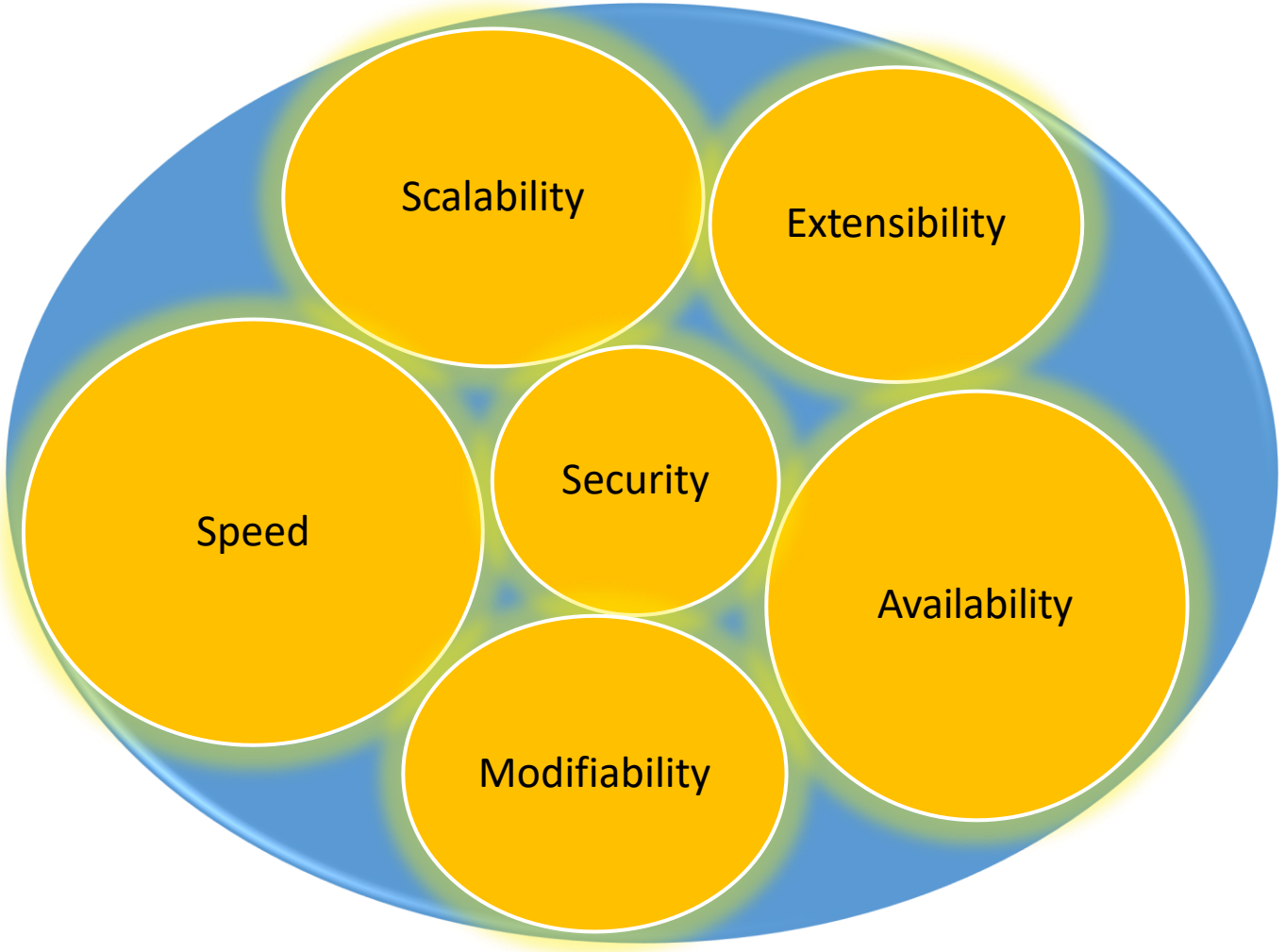
The *"X-abilities"*



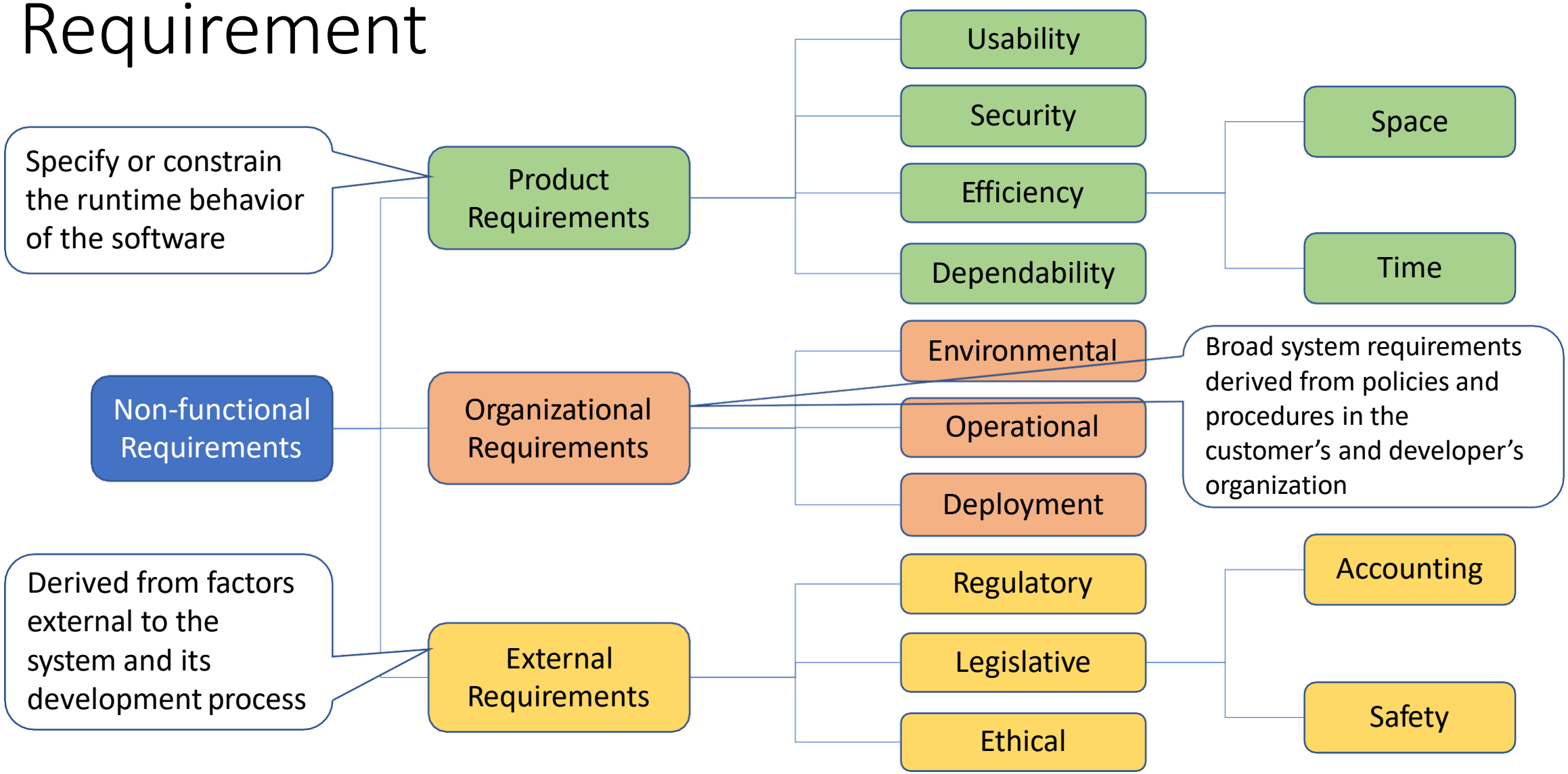
ISO/IEC FCD 25010 Product Quality Standard

Issues with too Many Non-Functional Requirements

- Many non-functional requirements conflict with others, e.g., speed/security, scalability/availability.
- Realizing a non-functional requirement may require staffs to have specialized skills
- Extra resources may be needed, for example, specially configured testbed, more time required for testing, extra budget to spend on hiring external specialists, etc.

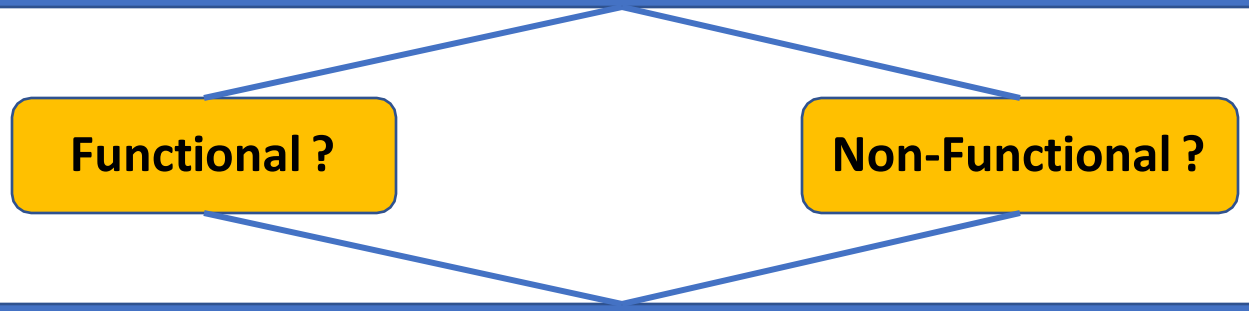


The Categories of Non-Functional Requirement



The Boundaries between Functional and Non-Functional Requirements

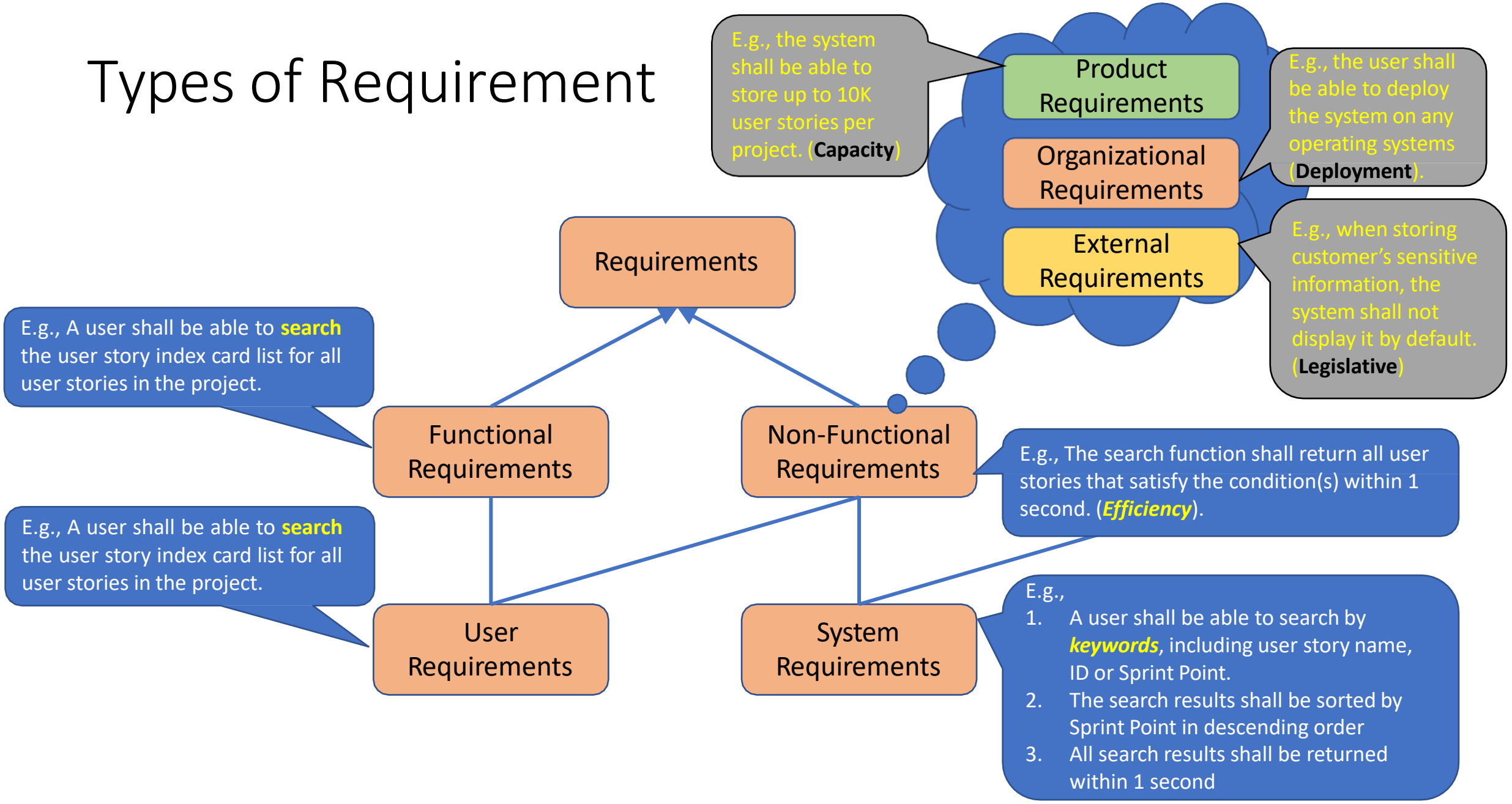
Requirement Statement:
The system shall maintain information securely. Only authorised users may have access to the user story index cards.



Requirement Statement:
The system shall provide a login mechanism so that only authorised users may have access to the user story index cards.

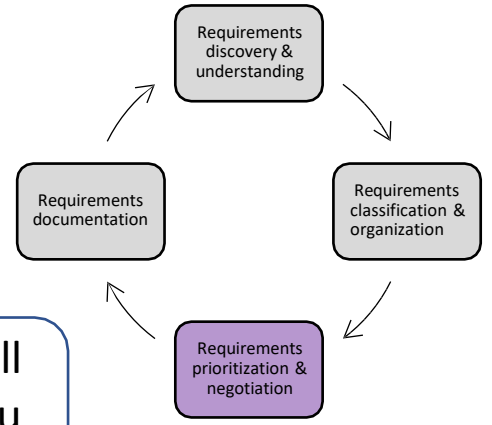
When documenting requirements, we often need to specify the relationships between functional and non-functional requirements.

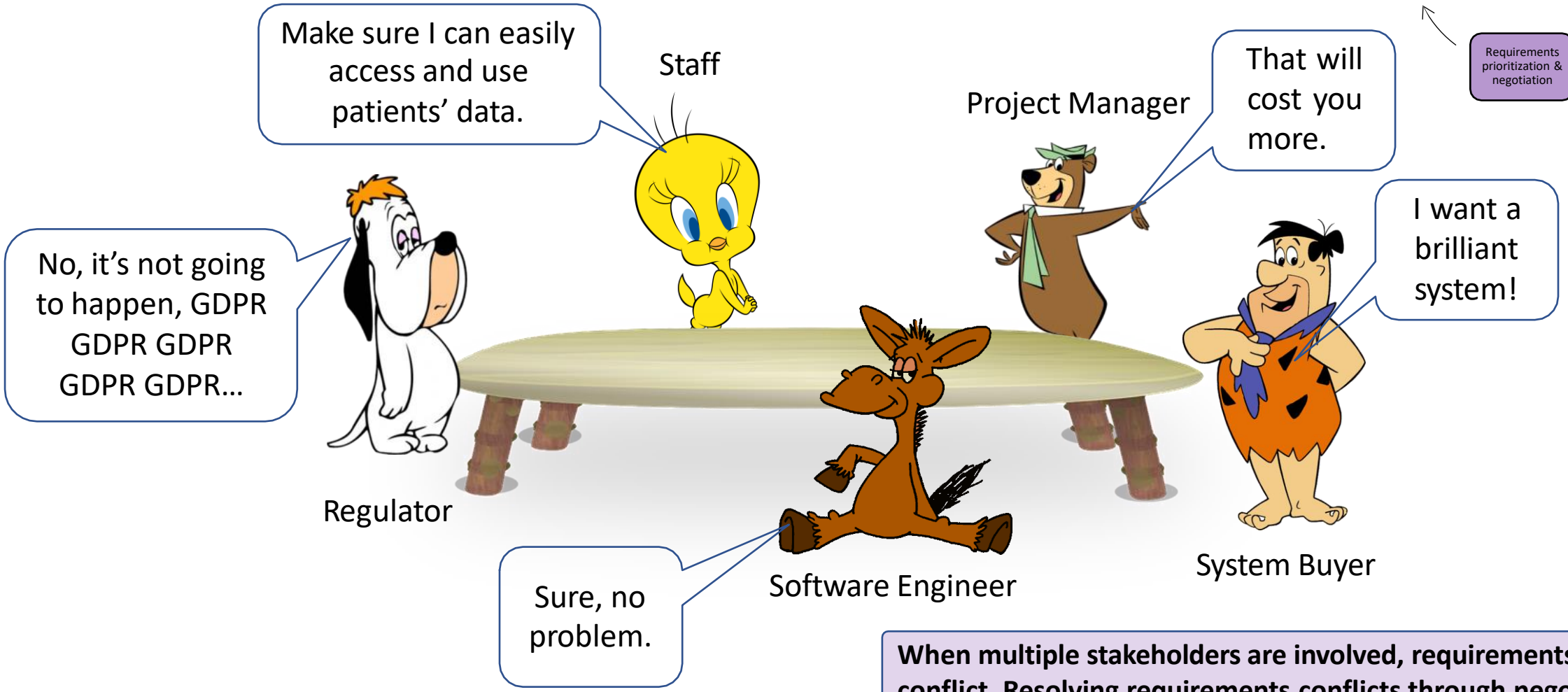
Types of Requirement



Explain why a software system should not have too many non-functional requirements?

Prioritization and Negotiation





When multiple stakeholders are involved, requirements will conflict. Resolving requirements conflicts through negotiation.

GDPR: *General Data Protection Regulation 2016/679 is a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA). It also addresses the transfer of personal data outside the EU and EEA areas.*

Prioritizing Non-Functional Requirements Using Utility Tree

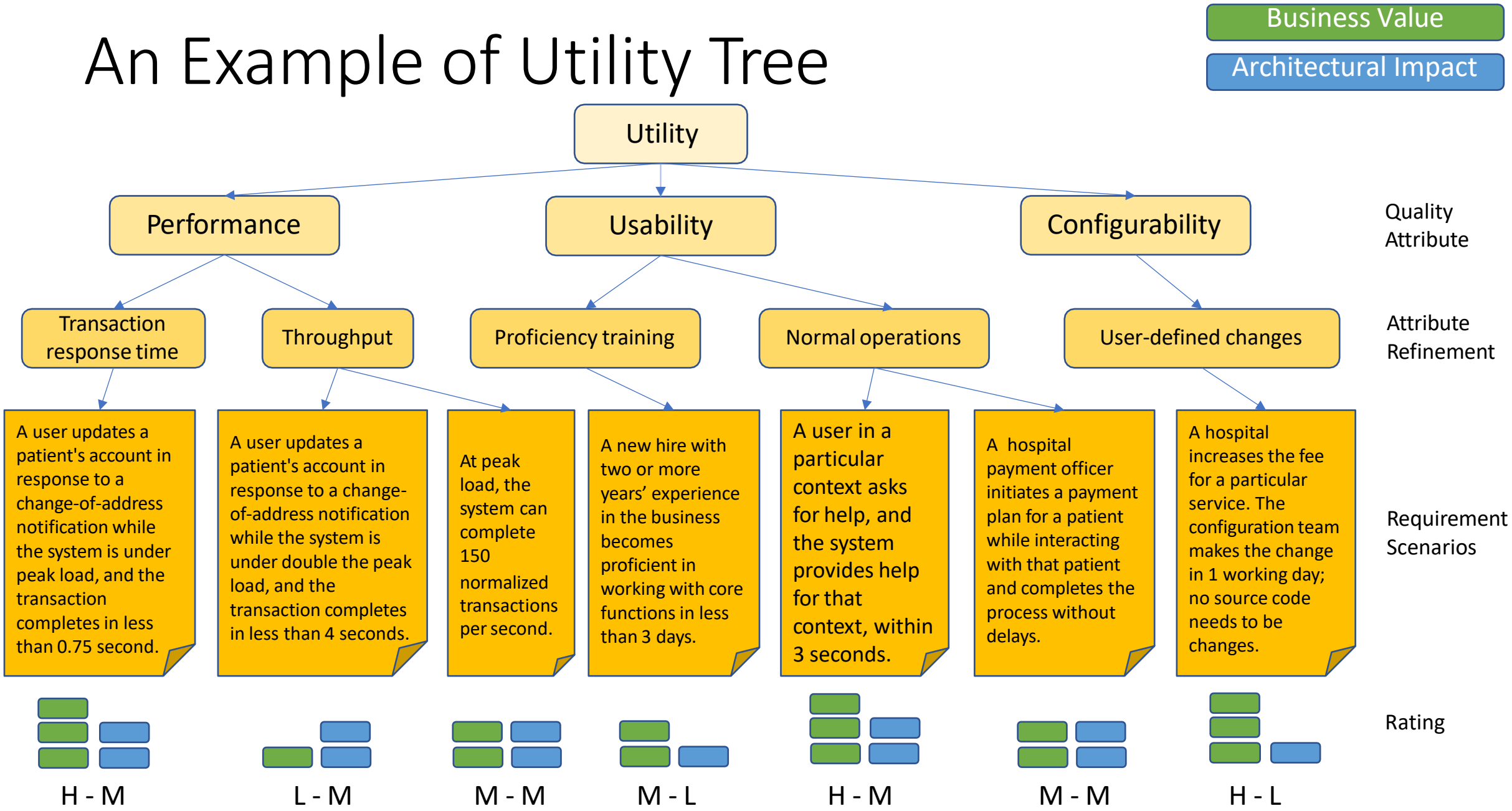
- A way to record non-functional requirements all in one place
- Establishes priority of each requirement in terms of
  - impact on software architecture
  - impact on business value
- Stakeholders can review the utility tree to make sure their concerns have been addressed

Constructing a Utility Tree

- The ***root*** of a utility tree is a placeholder node, labelled as "***Utility***"
- The second level of the tree contains broad quality attribute categories.
- The third level of the tree refines those categories.
- Requirements are captured as scenarios.

- Each scenario is rated by both **system buyers** and **architects**, at level of Low (L), Medium (M) or High (H).
  - Scenarios with (H,H) rating are the ones that deserve the most attention.
  - Having too many (H, H) might be a cause for concern: Is the system achievable?

An Example of Utility Tree



Feasibility Study

Business

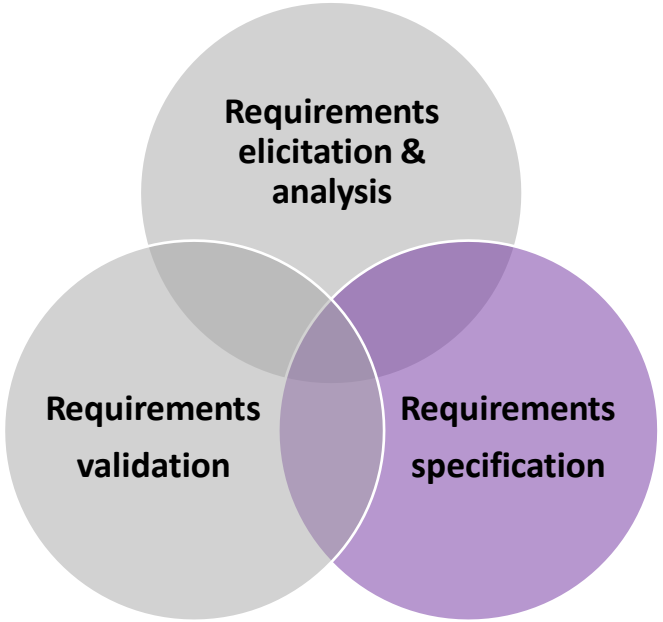
- Operational feasibility
  - How the proposed system will affect organizational structures, working procedures and people
- Economic feasibility
  - Assesses the costs and benefits of the project
  - Also known as *cost-benefit analysis*

Technical

- Technical feasibility
  - Assesses the practicality of the proposed technical solution and the availability of technical skills, expertise and resources
- Schedule feasibility
  - Assesses the project timetable

Requirements Specification

- Structured natural language
  - Requirements are written in natural language in standard form or template
  - Use language consistently to distinguish between mandatory and desirable requirements.
  - Do NOT assume that readers understand technical or software engineering language
- Graphical notations
  - Unified Modeling Language (UML) diagrams
    - Use case diagrams
    - Sequence diagrams
- Mathematical specification
  - Using notations based on mathematical concepts
    - E.g., Finite-state machines or Formal Methods



Structured Natural Language – Using Standard Form

Function:
Description:

Inputs:
Source:

Outputs:
Destination:

Action:

Requires:
Precondition:
Postcondition:

Side-effects:

Template

Function: compute insulin does.
Description: computes the does of insulin to be delivered when the current measured sugar level is in the safe zone between 7 and 7 units.
Inputs: current sugar reading (r2), the previous two readings (r0 & r1).
Source: current sugar reading from sensor & other reading from memory.
Outputs: dose in insulin to be delivered.
Destination: main control loop.
Action: dose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then dose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result.
Requires: two previous readings
Precondition: the insulin reservoir contains at least the maximum allowed single does of insulin.
Postcondition: r0 is replaced by r1 then r1 is replaced by r2.
Side-effects: none.

An Example (Software Controlled Insulin Pump)

Structured Natural Language – Consistency

A Functional Requirement Statement:
The system shall provide a login ***service***.

Software Engineer

= =

!=

System Buyer

A Non-Functional Requirement Statement:
The system shall be secure by having a login ***functional component***.

Structured Natural Language – Ambiguity

A Functional Requirement Statement:
The system shall allow staffs to upload ***large image files*** to the database.

Software Engineer

<50MB

~2GB

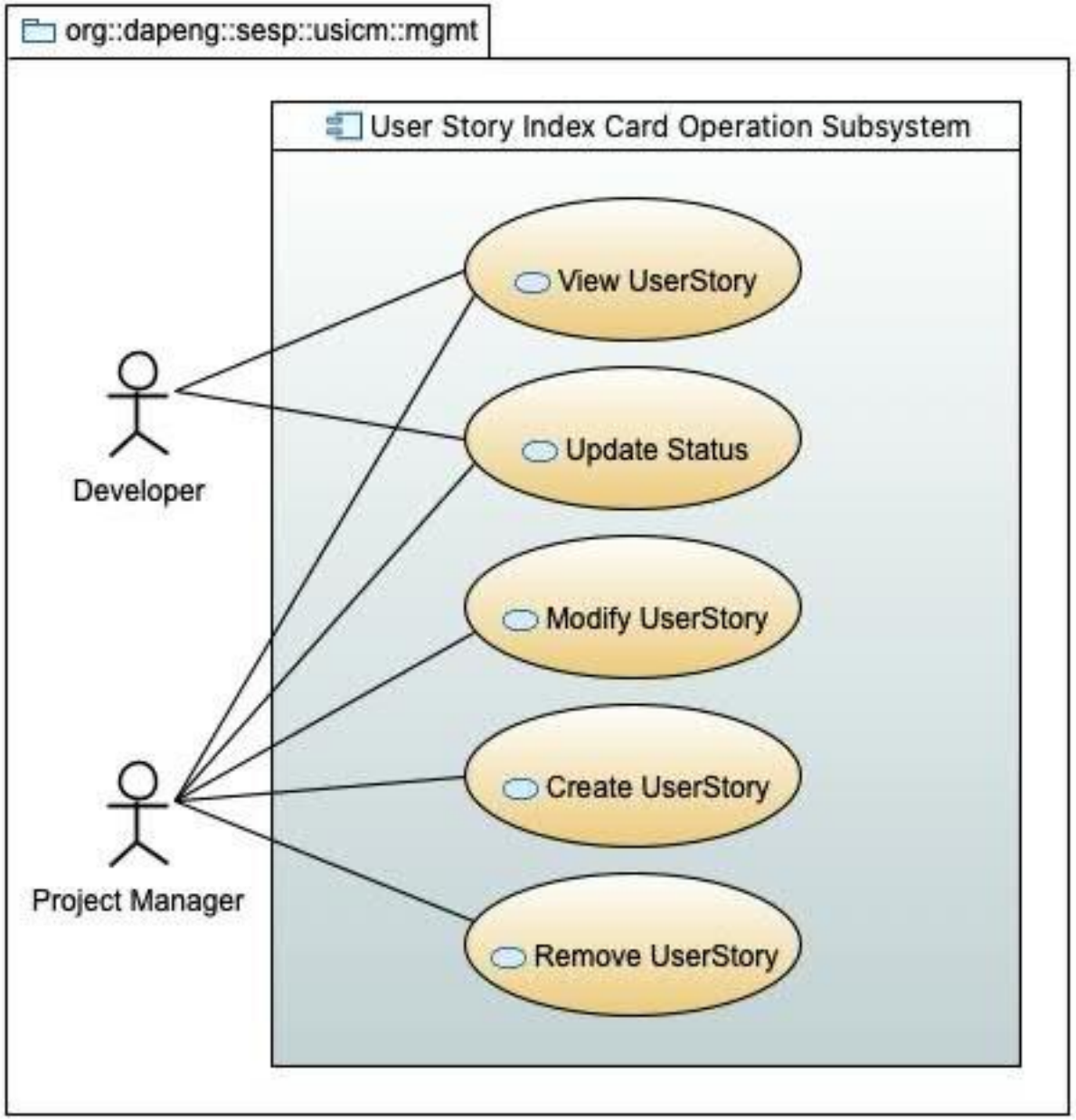
Staff

A Non-Functional Requirement Statement:
Each staff member of the system shall be uniquely identified by his or her eight-digit staff number before using the services provided by the system.

Graphical Notation -- UML Use Case Diagram

*"When a user uses the system, she or he will perform a behaviourally related sequence of transactions in a dialogue with the system. We call such a special sequence a **use case**"*

--Jacobson, I., 1993. Object-oriented software engineering: a use case driven approach. Pearson Education.



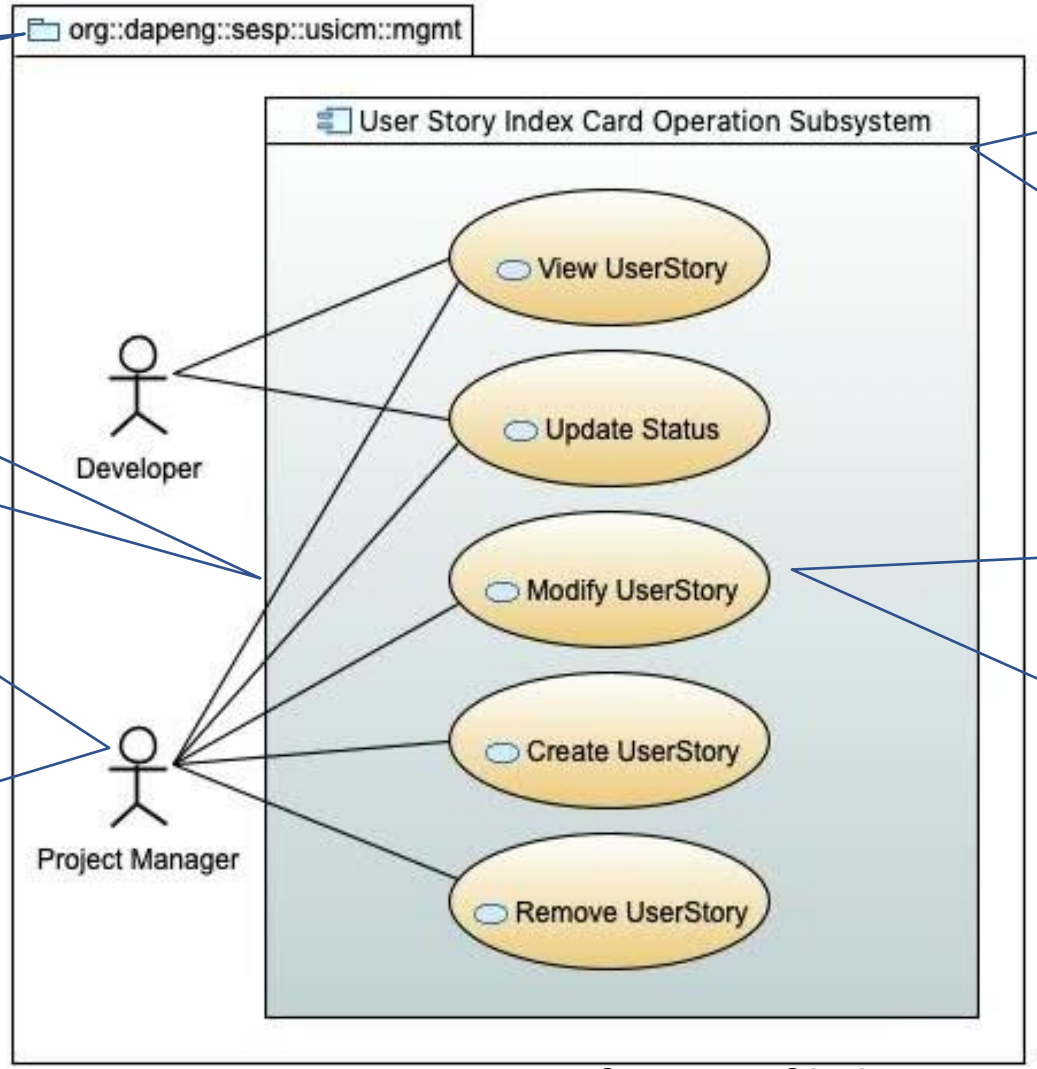
UML (Unified Modelling Language): is a modeling language that helps to specify, visualize, and document models of software systems, including their structure and design. (www.uml.org)

UML Use Case Diagram Notation

Package:
- owns the set of ***UseCases***, ***Actors*** and ***subject***

Association:
- associates an ***Actor*** with corresponding ***UseCases***
- is represented as a solid line

Actor:
- Users and other systems that may interact with a ***subject*** are represented as ***Actors***.
- is shown as a stick-man
- specifies a role played by a user or a system



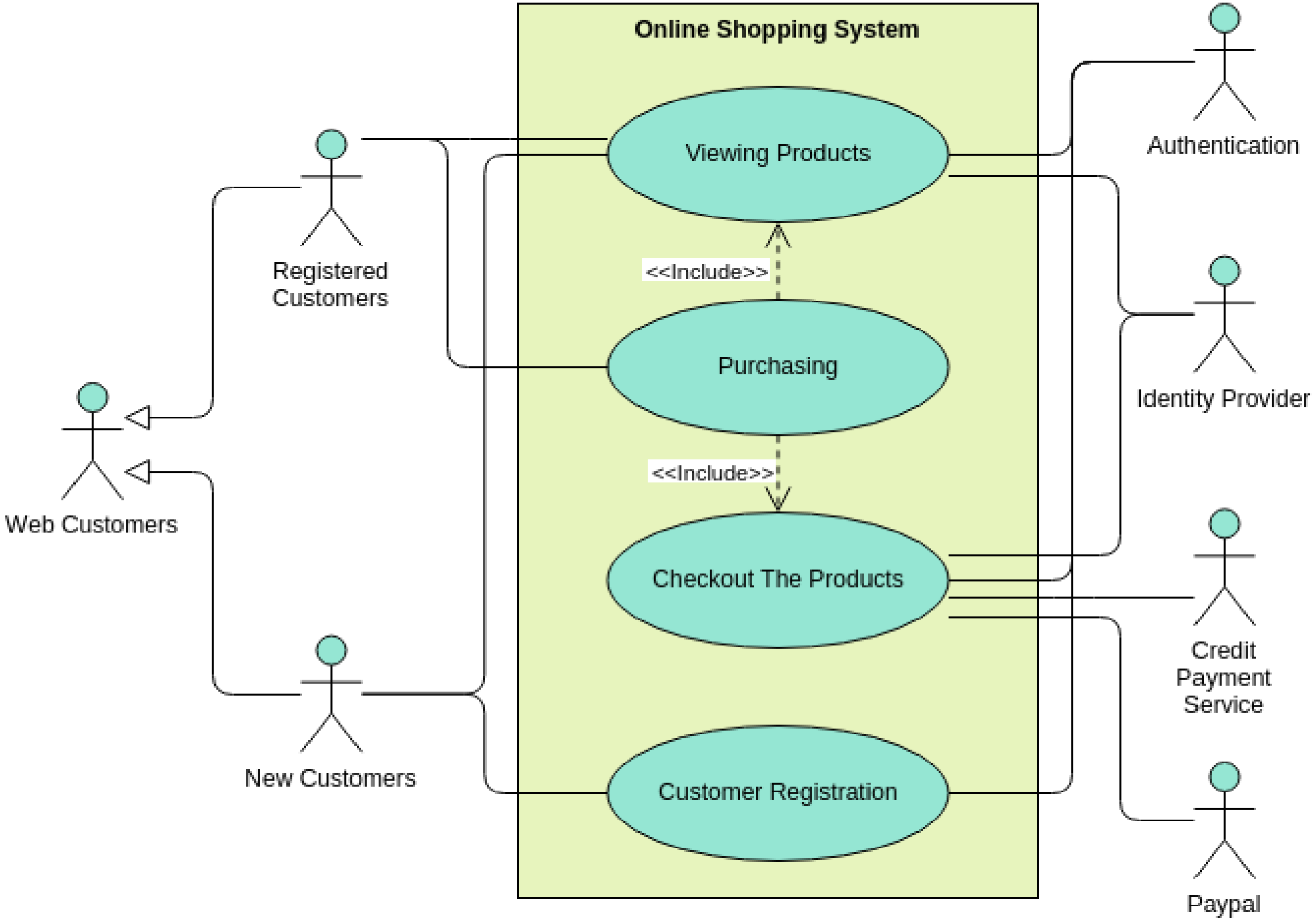
subject:
- represents a system under consideration
- is shown as a rectangle with its name in the top-left corner
- is also known as *system boundary*

UseCase:
- Specifies some behavior that a ***subject*** can perform with one or more ***Actors***.
- is shown as an oval shape
- defines behaviors of the ***subject*** without reference to its internal structure
- Each ***UseCase*** specifies a unit of useful functionality that the subject provides to its users

A UseCase is a specification of behavior.

Practice:

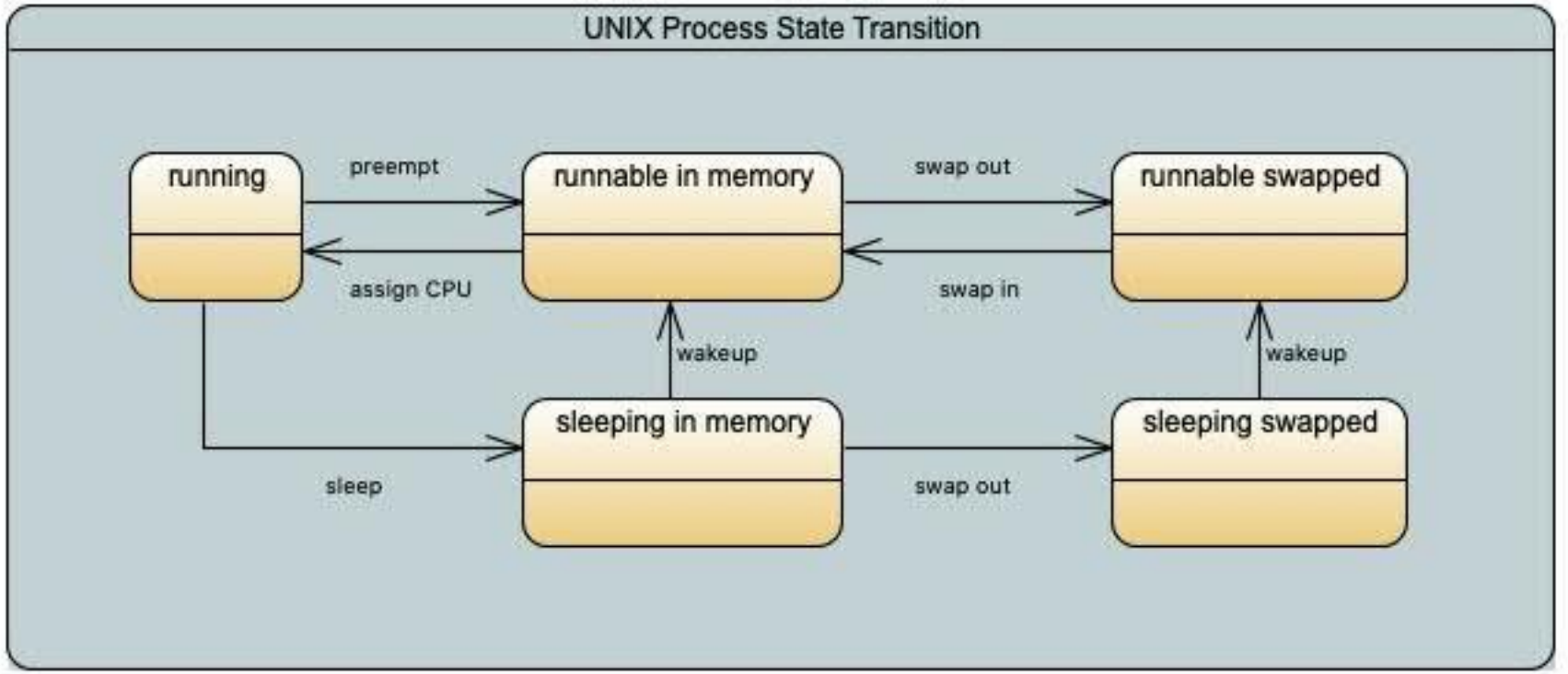
Create a use case diagram for an online shopping platform that supports user registration, product browsing, cart management, and checkout operations.


Online Shopping System
Viewing Products
Purchasing
Checkout The Products
Customer Registration
<<Include>>
<<Include>>
Registered Customers
Web Customers
New Customers
Authentication
Identity Provider
Credit Payment Service
Paypal

Mathematical Specification -- Finite State Machine

State

Transition



Requirements Documentation Standards

The IEEE standard for requirements documents (IEEE 1998).

Preface: Readership, version history, summary of changes.

Introduction: The need for the system, system's functions, basic workflow, how the system meet the business or strategic objectives.

Glossary: Technical terms.

User Requirements Definition: Functional requirements, non-functional requirements.

System Architecture: High-level overview of the system and architectural components.
System Requirements Specification: Detailed functional/non-functional requirements.
System Models: Relationship between system components, the system, and its operational environment.
System Evolution: Assumptions on which the system is based, anticipated changes.

Appendices: Detailed, specific information that is related to the software system.
Index: index of terms, tables, diagrams, and so on.

IEEE(The Institute of Electrical and Electronics Engineers): a professional association for electronic engineering and electrical engineering. (www.ieee.org)

Requirements Engineering Tools

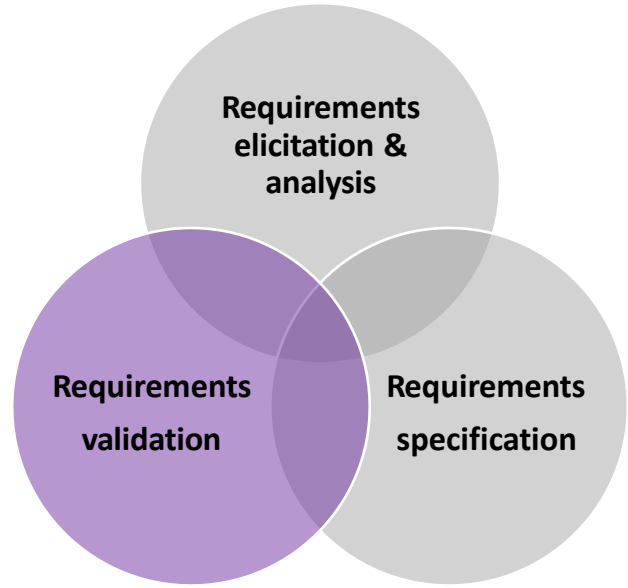
RE-Tools

Confluence

ReqView

rmToo

Requirements Validation

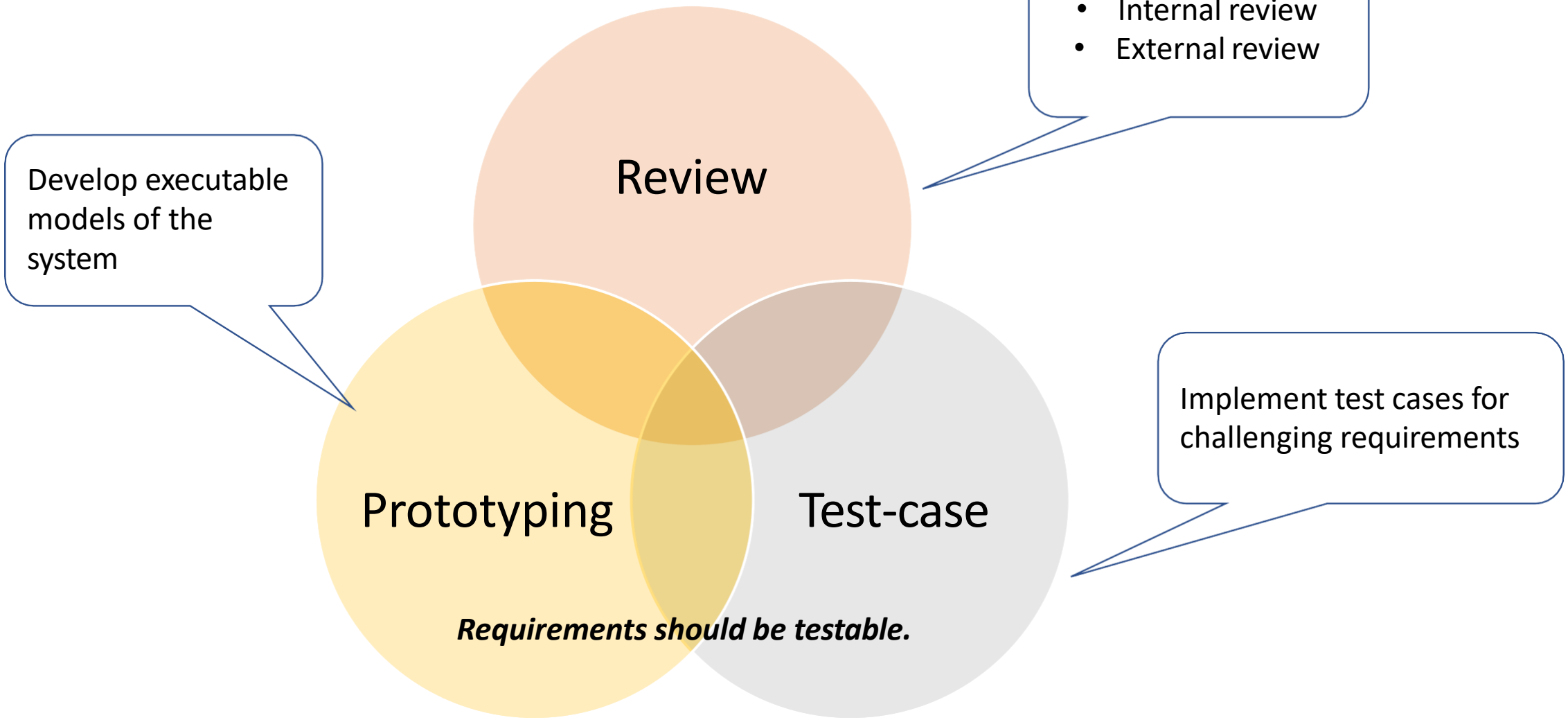


- Validity checks
  - Whether the requirements reflect the real needs of system users
- Consistency checks
  - Identify conflict & confusion requirements
- Completeness checks
  - Whether the documented requirements define all functions and constraints
- Realism checks
  - Whether the software system can be implemented within the proposed budget or supported by existing technologies
- Verifiability
  - Whether functions and quality attributes are verifiable

"The cost of fixing a requirements problem by making a system change is usually much greater than repairing design or coding errors."

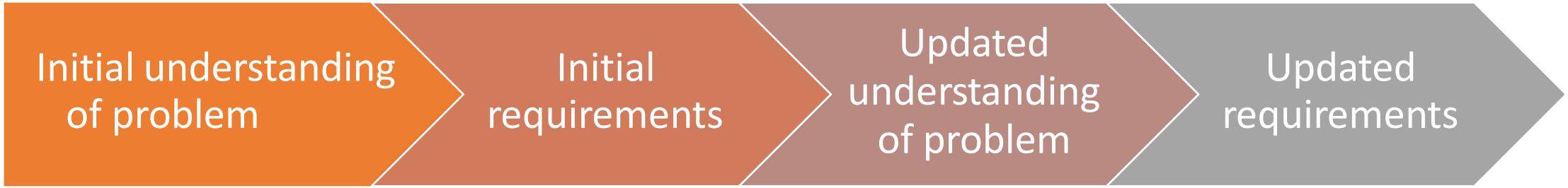
--Sommerville, I., 2011. Software engineering 9th Edition. ISBN-10, 137035152.

Validation Methods



If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered.

Requirements Evolution and Change Management



Requirements Evolution

Requirements Change Management

Do NOT invent requirements!