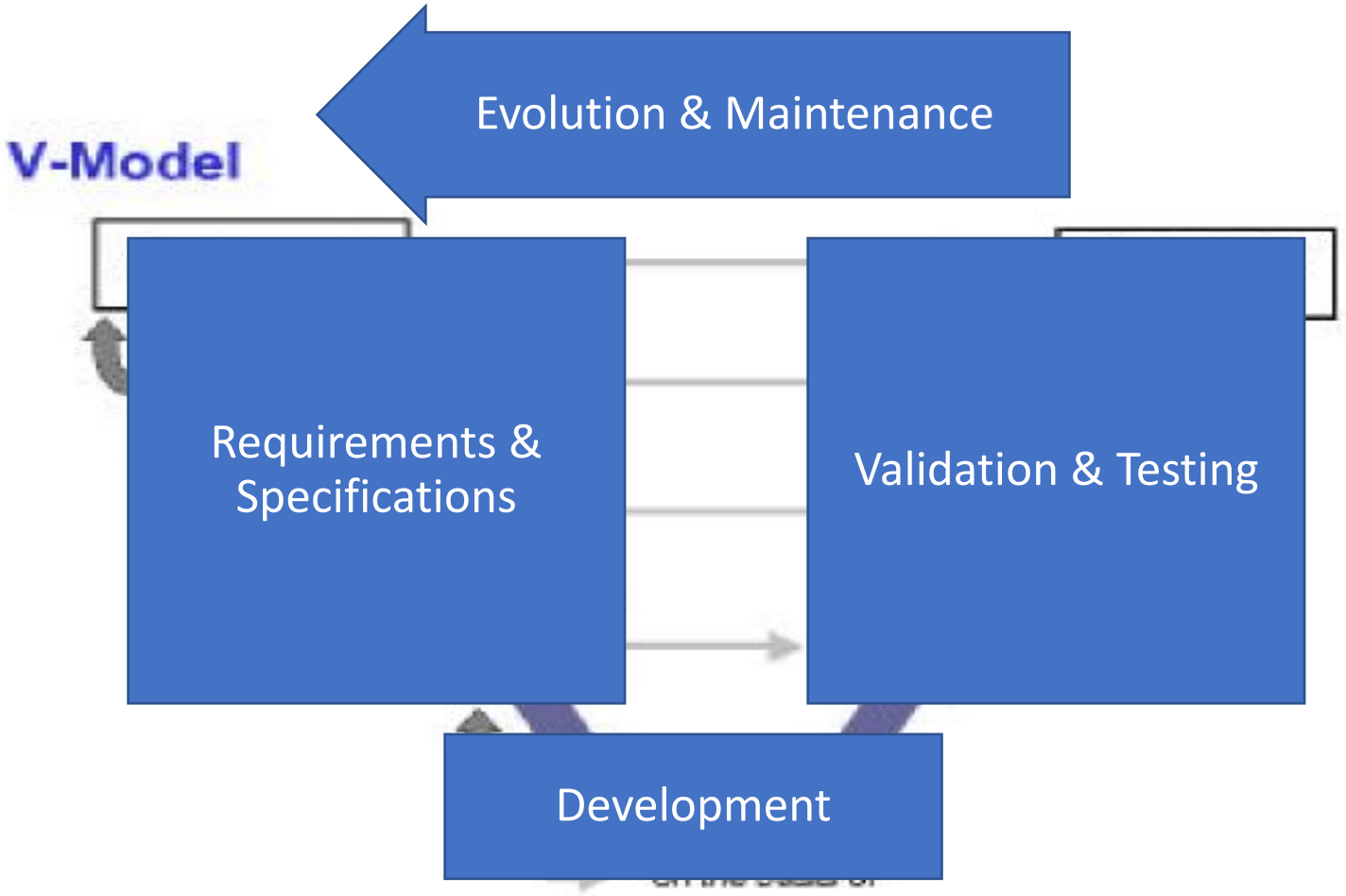
Software Engineering COMP1035

Lecture 03

Requirements Engineering

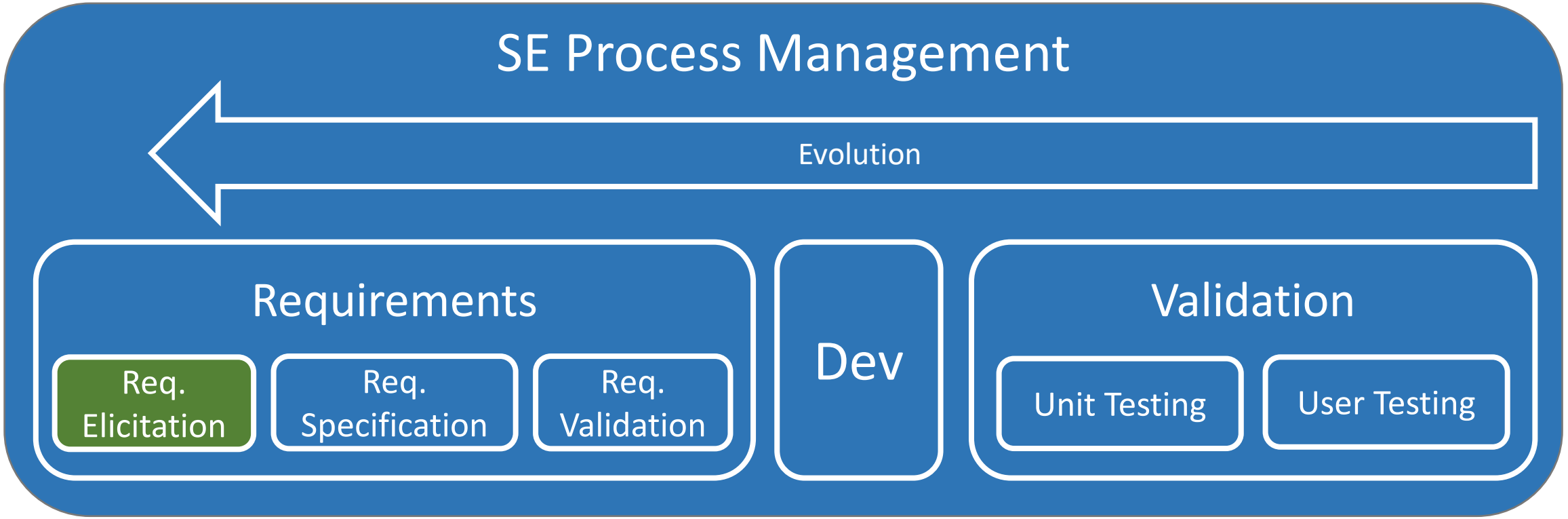
Keeping Track of SE Module



Core SE Processes

- **Requirements & Specifications** – Designing the system for what the customer wants.
- **Development** – Production of the software system.
- **Validation & Testing** – Checking that the software is what the customer wanted.
- **Evolution & Maintenance** – Changing code in response to new requirements.

Keeping Track of SE Module



Why Requirements are Important

Ongoing SE Project Failures

- The global cost of IT failure was estimated at +6 trillion dollars in 2009. (source: Roger Sessions, The IT Complexity Crisis: Danger and Opportunity)
- Only 32% of software projects were “successfully completed” (i.e., on time, on cost, and with expected functionality) in 2009. (Source: Standish CHAOS 2009 Update)
- Only 16% of software projects were successfully completed in the UK in 2009. (Source: British Computer Society)
- “A failing industry …
  - If building engineers build buildings with the same care as software engineers build systems, the first woodpecker to come along would be the end of civilization as we know it.” (Source: Paul Dorsey, Top 10 Reasons Why System Projects Fail)

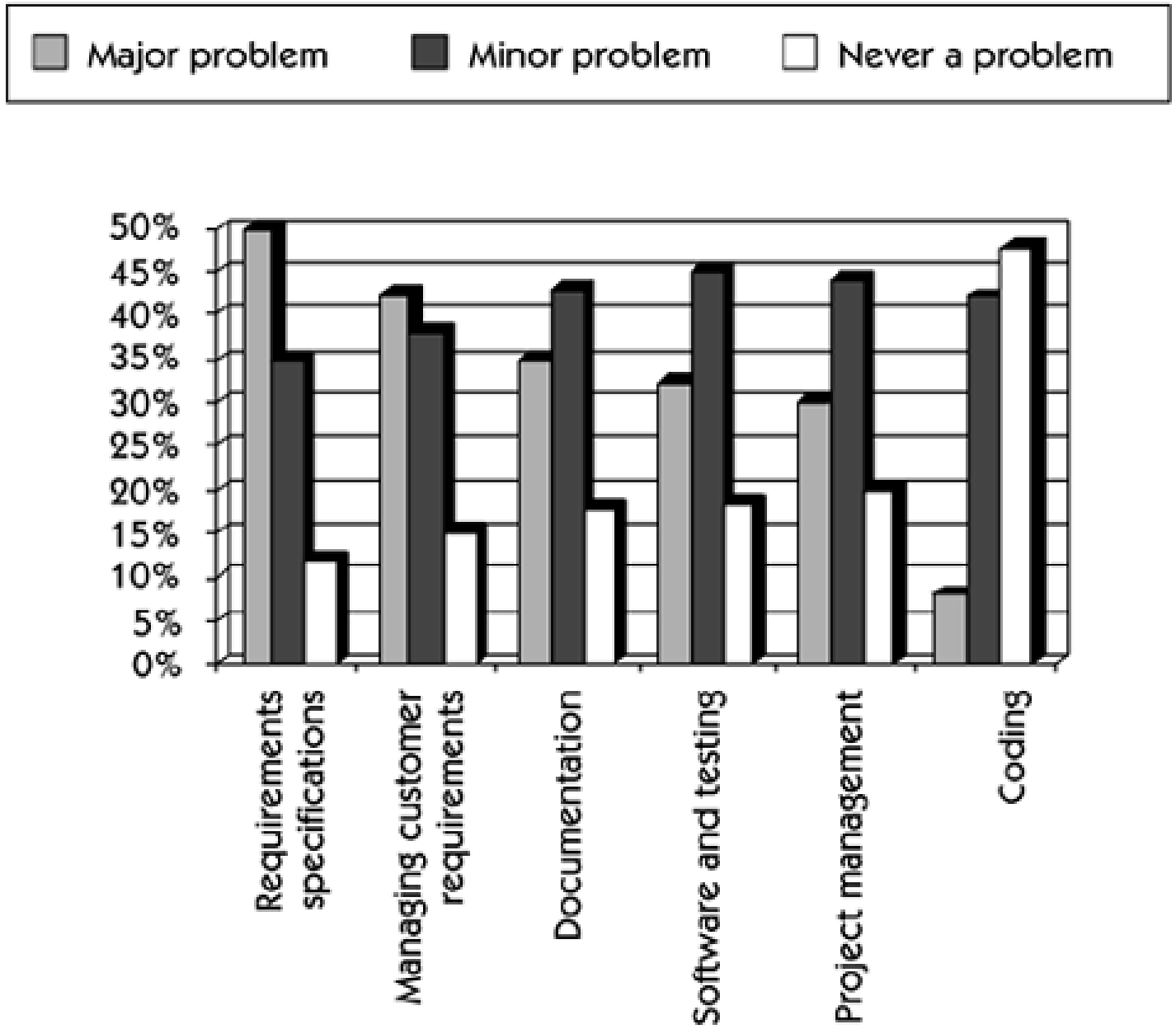
What Causes Failure?

- **Incomplete requirements** and **lack of user involvement** rank top of the list of causes of project failure. (Source: Standish CHAOS 1995 report)

- Understanding product requirement is **the major problem** in software development.

- Coding or programming is not a major problem.

What Causes Failure?



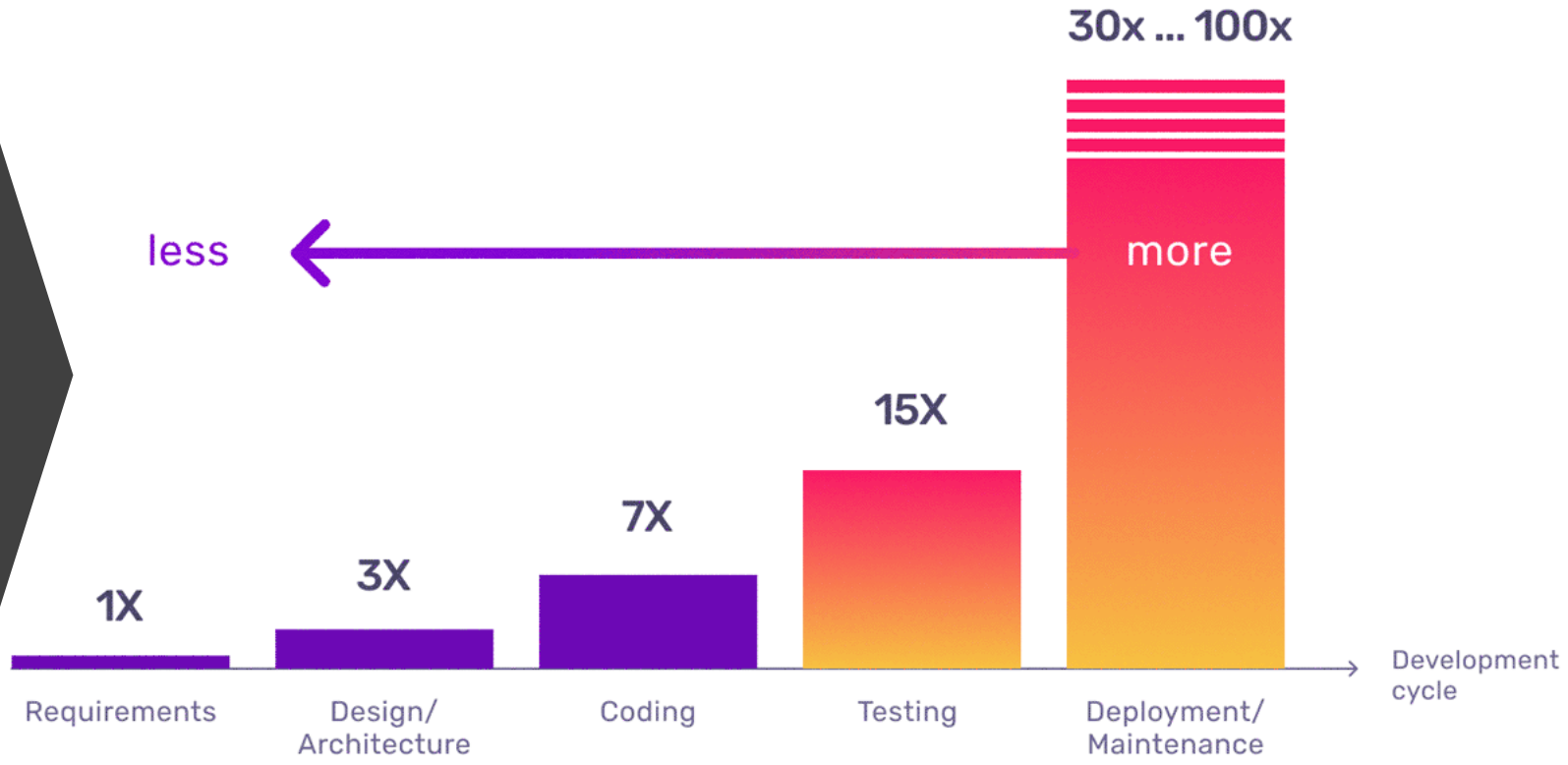
What Causes Failure?

- "**The hardest single part of building a software system is deciding precisely what to build** … No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to recify later." (Source: Fredrick Brooks, No Silver Bullet: Essence and Accidents of Software Engineering)

- As much as a **200:1 cost savings** results from finding errors in the requirements stage versus finding errors in the maintenance stage of the software lifecycle.

Cost of Defects

What Causes Failure?



The more time we save your team, the more time they have to find bugs sooner.

That Saves Money

The Requirements Problem

The Requirements Problem

- You all want to build software.
  - However, there is a **great deal more** to doing that successfully than learning to cut code.
  - Understanding “stakeholder” requirements (i.e., client, customer, user and other interested parties needs) is a key to effective systems development.
  - That means you are going to learn **how to identify their requirements**.
  - Which means you are going to have to learn to work with human beings as well as machines …

The Requirements Problem

- “I would go a step further and assert that is really impossible for clients, even those working with software engineers, **to specify completely, precisely, and correctly, the exact requirements of a modern software product**.” (Source: Fredrick Brooks, No Silver Bullet: Essence and Accidents of Software Engineering)
- We need to be able to **determine them**, by **working with clients**.
- Learning how to do requirements engineering:
  - In more technical terms its about “requirements engineering” or figuring out what **stakeholder needs** and what **services** a product will need to provide **to meet those needs**.

The Requirements Problem

- "**Requirements are your project's fundation**. They define the level of quality you need, facilitate decision making, provide a basis for tracking progess, and serve as the basis for testing."
- "If you let them remain unstated, you have no opportunity to examine and negotiate them with your customer and no way to tell when your project has met its objectives."
  (Source: Brian Lawrence, Top Risks of Requirements Engineering)

Types of Requirements

User vs System Requirements

- **User Requirements**
  - Consist of statements, conveyed in natural language accompanied by diagrams.
  - Include the services anticipated from the system's users and the operating constraints it must adhere to.
  - "**What a user needs to be able to do.**"
- **System Requirements (some refer to as System Specifications)**
  - Provide more elaborate descriptions of the software system's functions, services, and operational limitations.
  - Outlining system requirements, often referred to as a functional specification, serves to precisely define what is to be implemented.
  - It might constitute a segment of the agreement between the client and the software developers.
  - "**What the software must do to meet the user requirements.**"

FOUR Types of Requirements

- **Business Requirements**
  - Outline the project's **purpose** and the company's anticipated gains from it.
- **Stakeholder Requirements**
  - Encapsulate the **needs and anticipations of stakeholders**.
- **Solution Requirements**
  - Include technical specifications, including **functional** aspects detailing system functionality and **non-functional** aspects describing system qualities.
- **Transition Requirements**
  - Define steps needed to transition an organization from its existing state to the desired state.

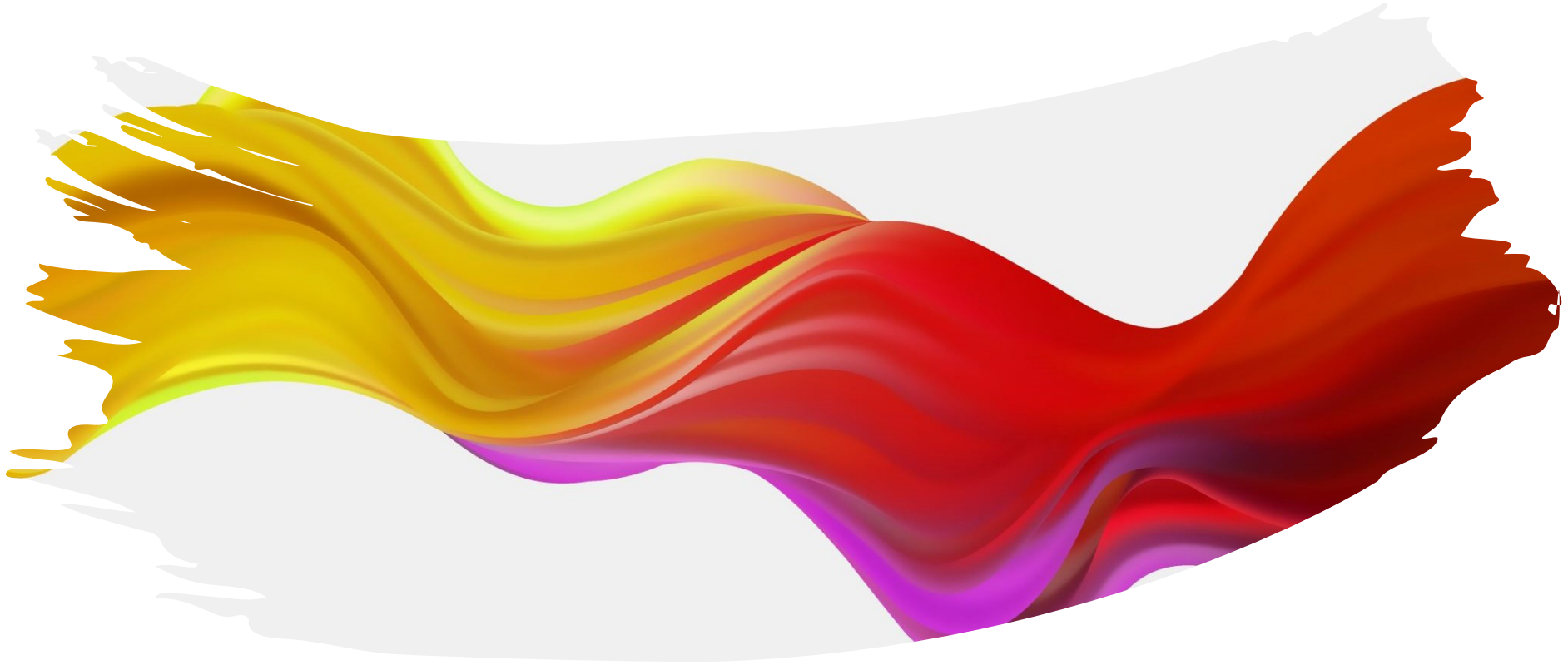
Stakeholder Requirements

Stakeholder Requirements

- **Identify all individuals directly or indirectly involved** in the system, including users, funders, etc.
- Involves a combination of:
  - Engaging with initial briefs
  - Problem analysis
  - Interviews/discussions
- Essential to identify:
  - Primary stakeholders
  - Secondary stakeholders
  - Tertiary stakeholders

Stakeholder Requirements

- Primary Stakeholders (Direct Users)
  - Beneficiary: Hands-on daily users.
  - Target: Departments or organisations that stand to gain or lose as a whole.
  - Usually referred to as "**key stakeholders**".
- Secondary Stakeholders (Indirect Users)
  - Project or decisions that affect people or groups.
  - E.g., if a park is being built in a neighbourhood, local construction companies and conservation groups might be secondary stakeholders.
- Tertiary Stakeholders (Indirect)
  - People or groups affected more indirectly than secondary stakeholders.
  - Such as business owner, public and sometimes government agencies.
  - Often referred to as "**external**" – play as an advisory or advocacy role.



Solution Requirements

Functional and Non-Functional Requirements

Functional vs Non-Functional Requirements

- System requirements are commonly categorized as **functional** or **non-functional** requirements:

- **Functional Requirements**:
  - The **services the system is expected to provide**, how it should respond to specific inputs, and its behavior in various situations.
  - Functional requirements may also explicitly specify what actions the system should avoid.

- **Non-Functional Requirements**:
  - Impose **limitations** on the services or functionalities offered by the system.
  - Including timing restrictions, constraints related to the development process, and standards-imposed restrictions.
  - Usually related to the system in its entirety rather than individual features or services.

Functional vs Non-Functional Requirements

	Functional requirements	Nonfunctional requirements
Objective	Describe what the product does	Describe how the product works
End result	Define product features	Define product properties
Focus	Focus on user requirements	Focus on user expectations
Essentiality	They are mandatory	They are not mandatory but desirable
Origin type	Usually defined by the user	Usually defined by developers or other tech experts
Testing	Component, API, UI testing, etc. Tested before nonfunctional testing	Performance, usability, security testing, etc. Tested after functional testing
Types	Authentication, authorization levels, data processing, reporting, etc.	Usability, reliability, scalability, performance, etc.

Functional vs Non-Functional Requirements

	Functional	**Non-Functional**
(User) Req.	Functions the user needs to achieve (as requirement). Functions the software will include (as specifications). E.g., allow login, provide creation, provide password change, display profile information, edit profile information, close account, delete posts.	Constraints on what the user needs to do (as requirements). Constraints on how well the system performs (as specifications). Up time, security standards, number of concurrent users, failure safety, must ue university authentication etc.
(System) Spec.		

Functional vs Non-Functional Requirements

	Functional	
(User) Req.	1. A module convenor will need to see and check if a student has the pre-requisite knowledge to take their module.	
(System) Spec.	1. A module convenor shall be able to see a list of students who want to take their module. 2. A module convenor shall be able to see the modules/grades of a student. 3. A module convenor shall be able to see additional comments put by students. 4. A module convenor shall be able to accept/reject a student. 5. A module convenor shall be able to contact/ask the student to clarify.	

Functional vs Non-Functional Requirements.

	Functional	
(User) Req.	1. A module conve … f a student has the pre-requisite knowl …	
(System) Spec.	1. A module convenor shall be able to see a list of students who want to take their module. 2. A module conv … odules/grades of a student. 3. A module conv … nal comments put by students. 4. A module convenor shall be able to accept/reject a student. 5. A module convenor shall be able to contact/ask the student to clarify.	

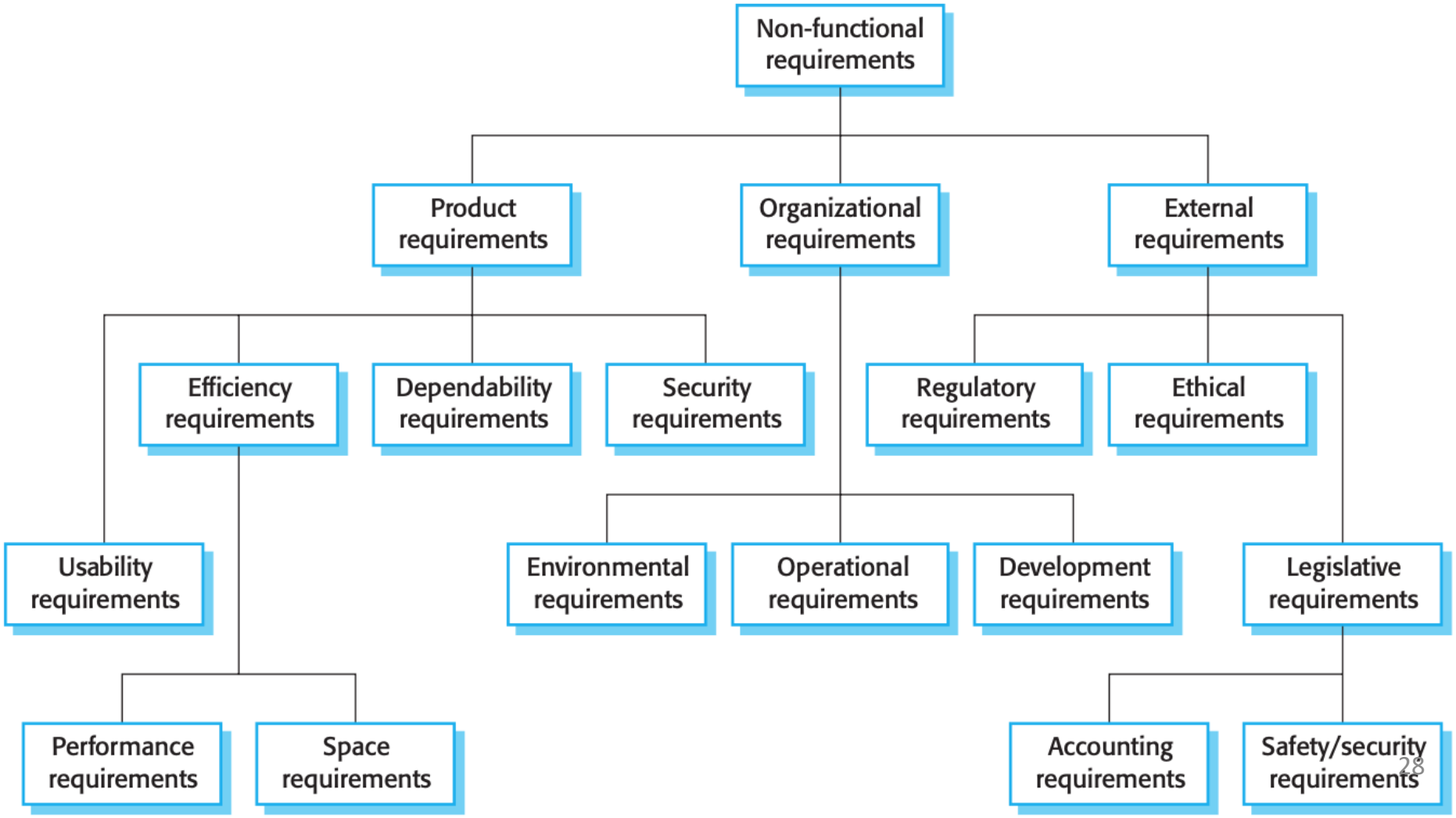
What a stakeholder needs to be able to do.

What the software must do to meet the requirement above.

Non-Functional Requirements

- Product NF Requirements
  - *90% of this job happens during the first 2 weeks of each semester and we have extra staff 8 am to 8 pm.*
- Organisational Requirements
  - *We want to login to use university authentication service that uses their university IDs, e.g., scyxxx*
- External Requirements
  - *The software should meet standard web accessibility guidelines, because we want it to work well for students with disabilities.*

Types of Non-Functional Requirements



Self Test: Functional or Non-Functional

A. A user should be able to post a new image.
B. Ideally it should be able to handle all major image types.
C. Users should be able to post big high-definition images.
D. We want animated images (e.g., gifs) to auto play.
E. A user should be able to repost images they like.
F. A user should be able to ‘like’ images they like.
G. A user should be able to see who else likes the image.
H. But only people that they are friends with.

Requirements Engineering Processes

Requirements Engineering

- First SE process!

Short-term, relatively cheap studies that inform the decision of whether go ahead with a more detailed analysis.

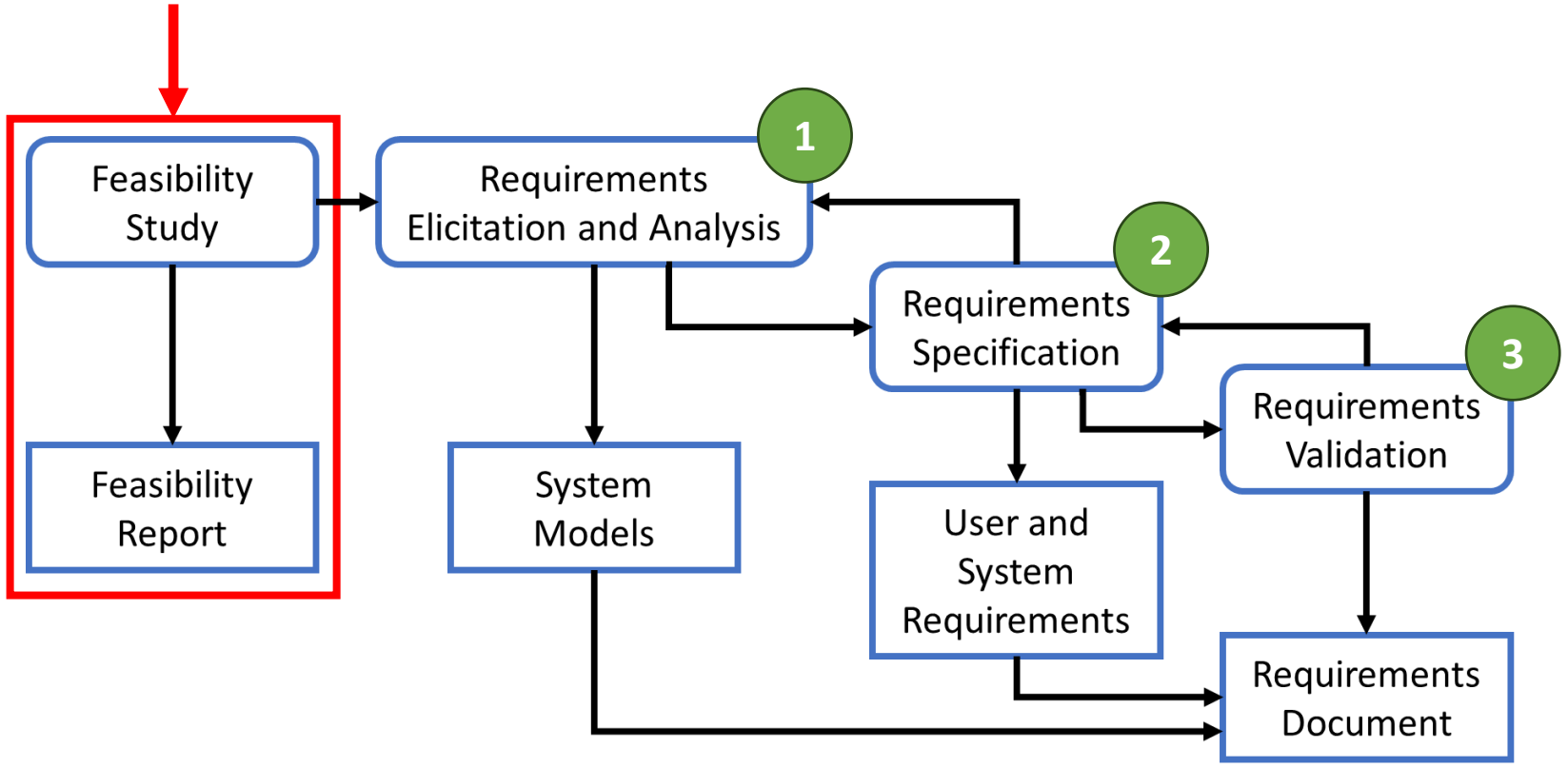


Fig: The Requirements Engineering Process

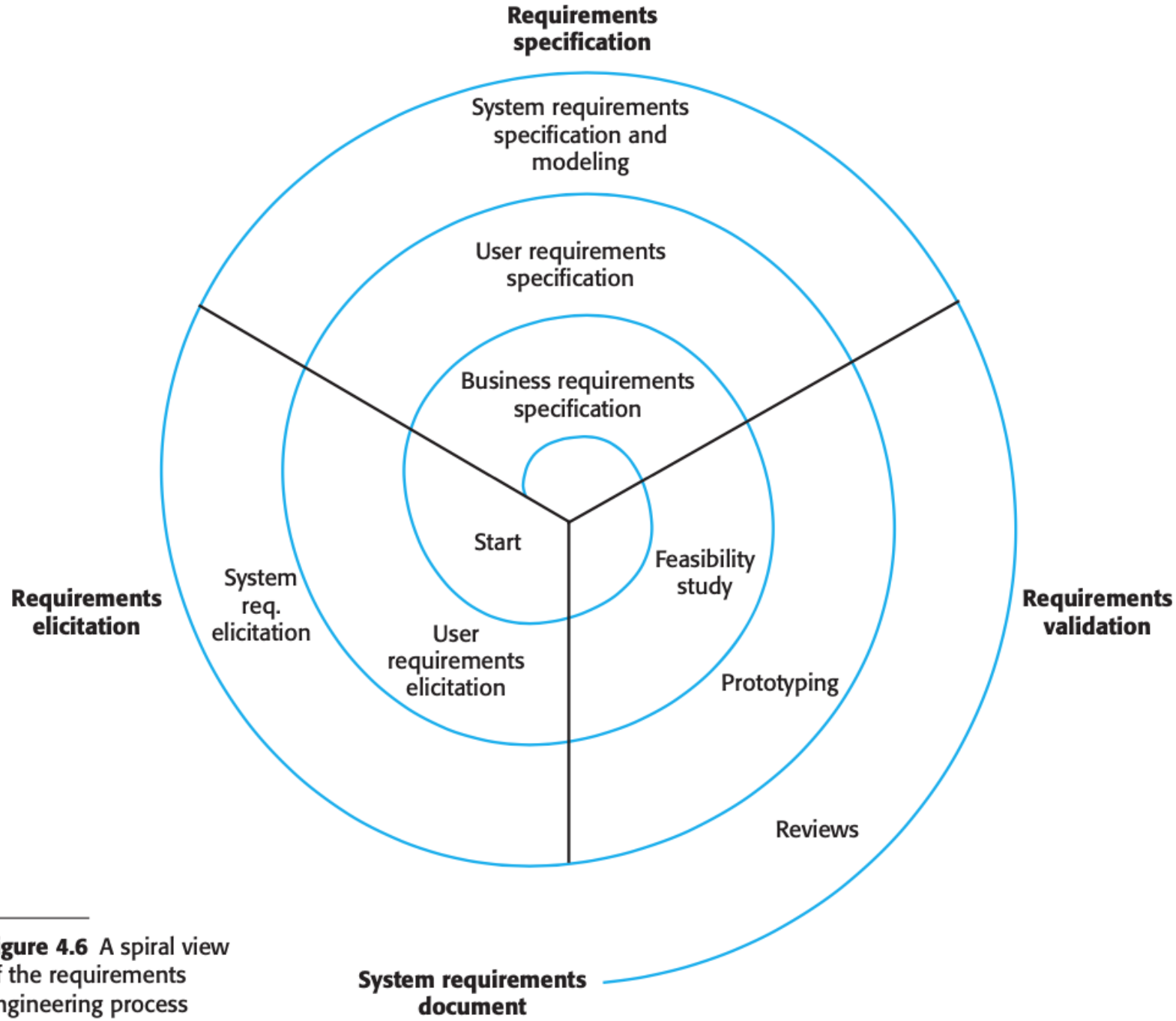
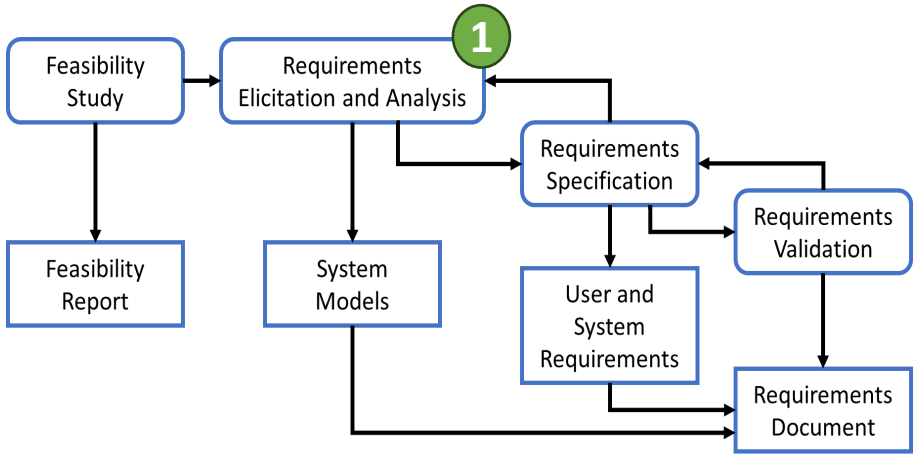


Figure 4.6 A spiral view of the requirements engineering process



Requirements Elicitation

Requirements Elicitation

- Requirements elicitation involves the **process of investigating and identifying the needs of a system from users, customers, and other stakeholders**.
  - This practice is also known as "requirement gathering".

- Include gathering and identifying the requirements for a software system.

- The aim of requirements elicitation is to **guarantee** that the software development process is founded upon a precise and thorough understanding of the customer's needs and requirements.

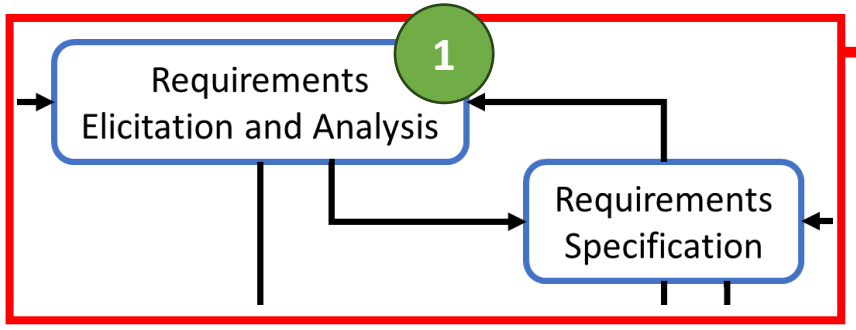
Requirements Elicitation

- Stands out as **one of the most challenging, error-prone, and communication-intensive** aspects of software development.
  - Its success hinges on the **effectiveness of the partnership between customers and developers**, as it is essential to understand the users' needs.
  - This process includes the **identification, gathering, analysis, and refinement of requirements** for a software system.
  - It plays a pivotal role in the software development life cycle, typically **initiating the project** (beginning of the project).
  - **Stakeholders** from various sectors of the organization, such as business owners, end-users, and technical experts, participate in requirements elicitation.
  - The outcome is a set of well-defined requirements, clear and concise, laying the groundwork for the design and development of the software system.

Five Reasons to Elicit Requirements

- Define a **budget and clear scope** of work.
  - To establish cost and timeline parameters
- **Mitigating confusion** during development.
  - Ensure a shared understanding of project goals, tasks, and deadlines.
- To **add business value**.
  - Discuss both the development team's responsibilities to meet client needs comprehensively.
- **Uncover hidden and assumed requirements**.
  - Help developers to justify the requirements.
- Comprehensive understanding of the solution to **develop only relevant functionality**.
  - Assisting clients in discarding inefficient features and selecting optimal technologies.

Requirements Elicitation



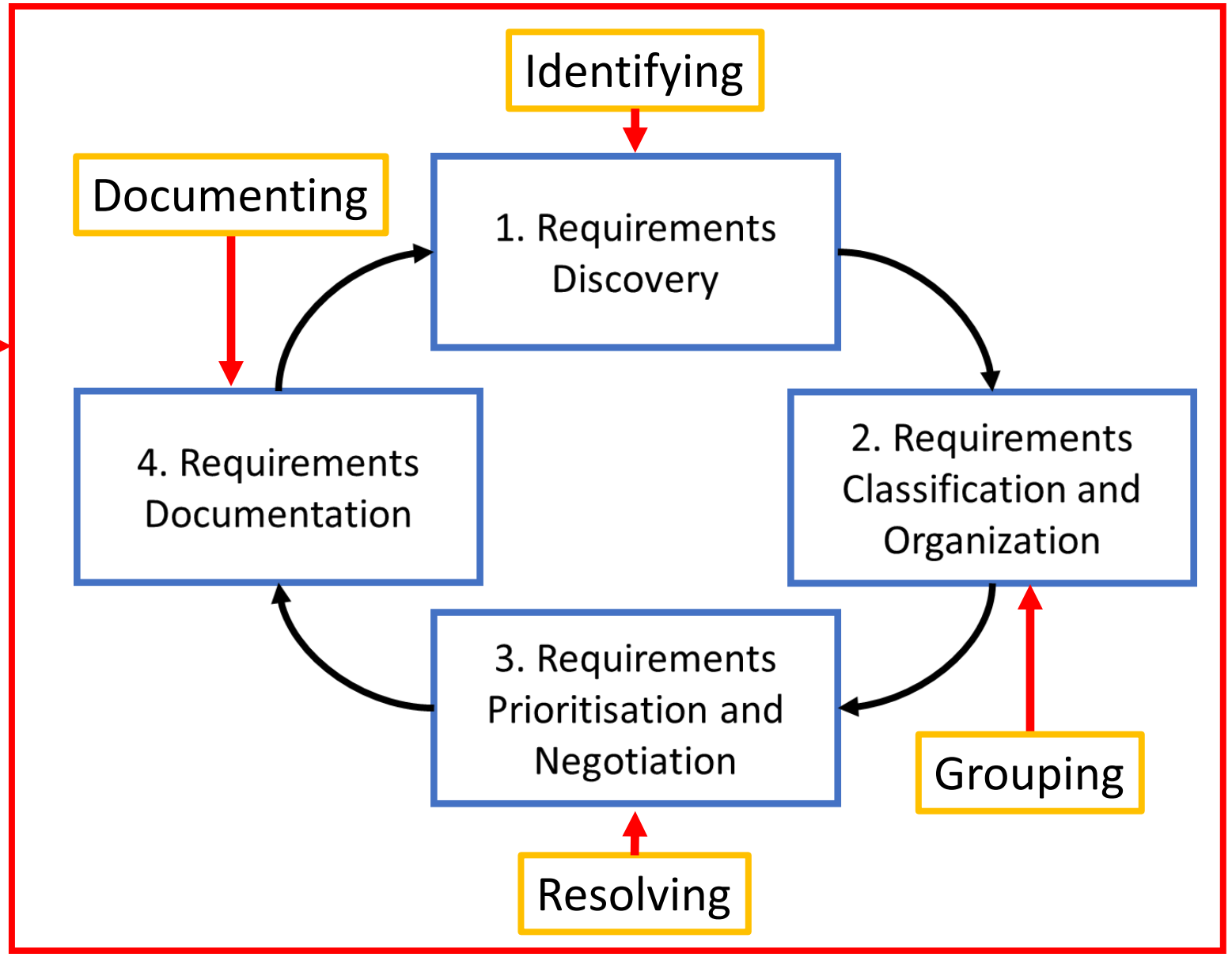
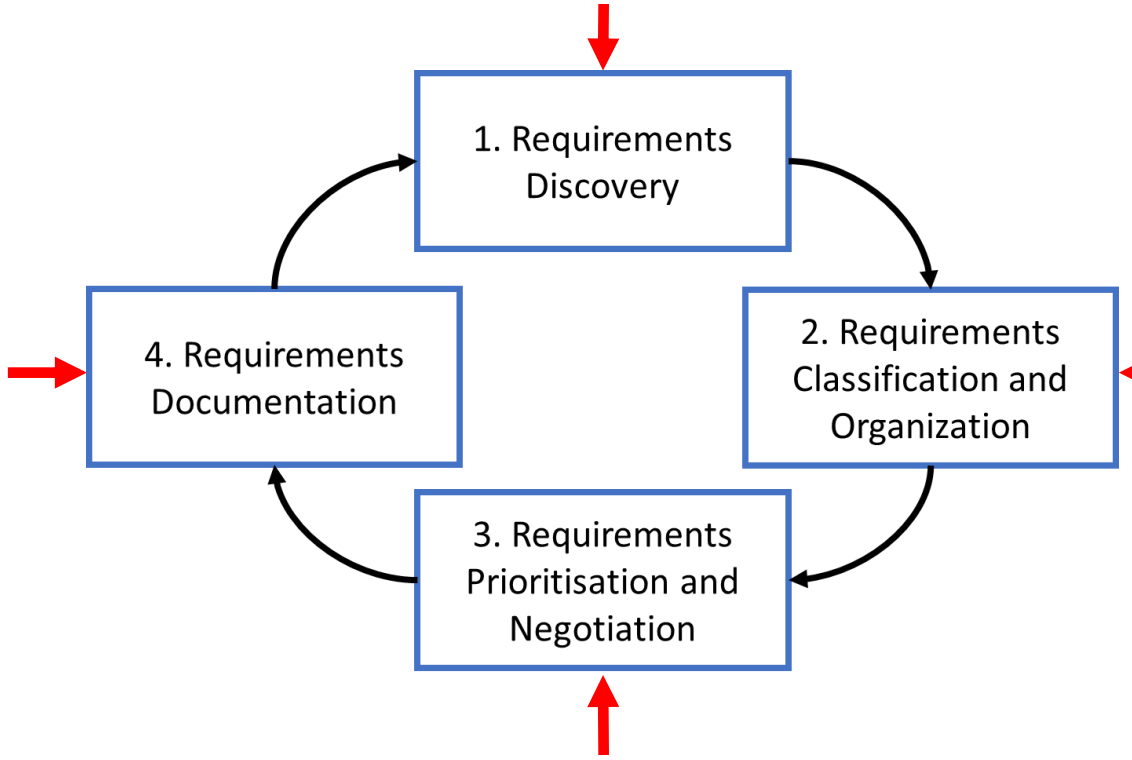


Fig: The Requirements Elicitation and Analysis Process

Requirements Elicitation and Analysis Process

Involve **identifying requirements from stakeholders** and collating documentation during this process.



Initial draft of the software **requirements document** may be designed during this stage, or the requirements may be informally maintained on platforms like whiteboards, wikis, or shared spaces.

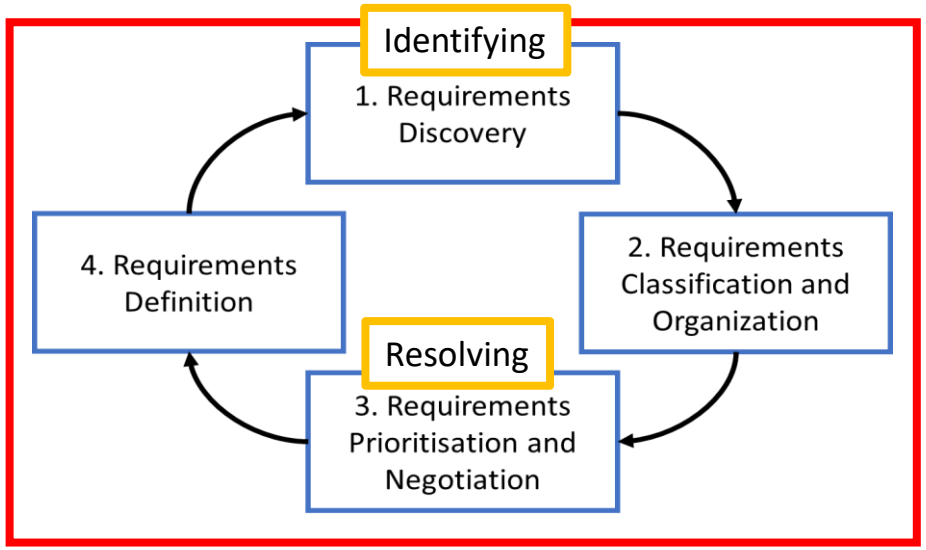
The initially **unorganized set of requirements is grouped** based on their relevance and organized into coherent categories.

Focuses on **prioritizing requirements and resolving conflicts** among multiple stakeholders – to address disparities and reach consensus on compromise requirements.

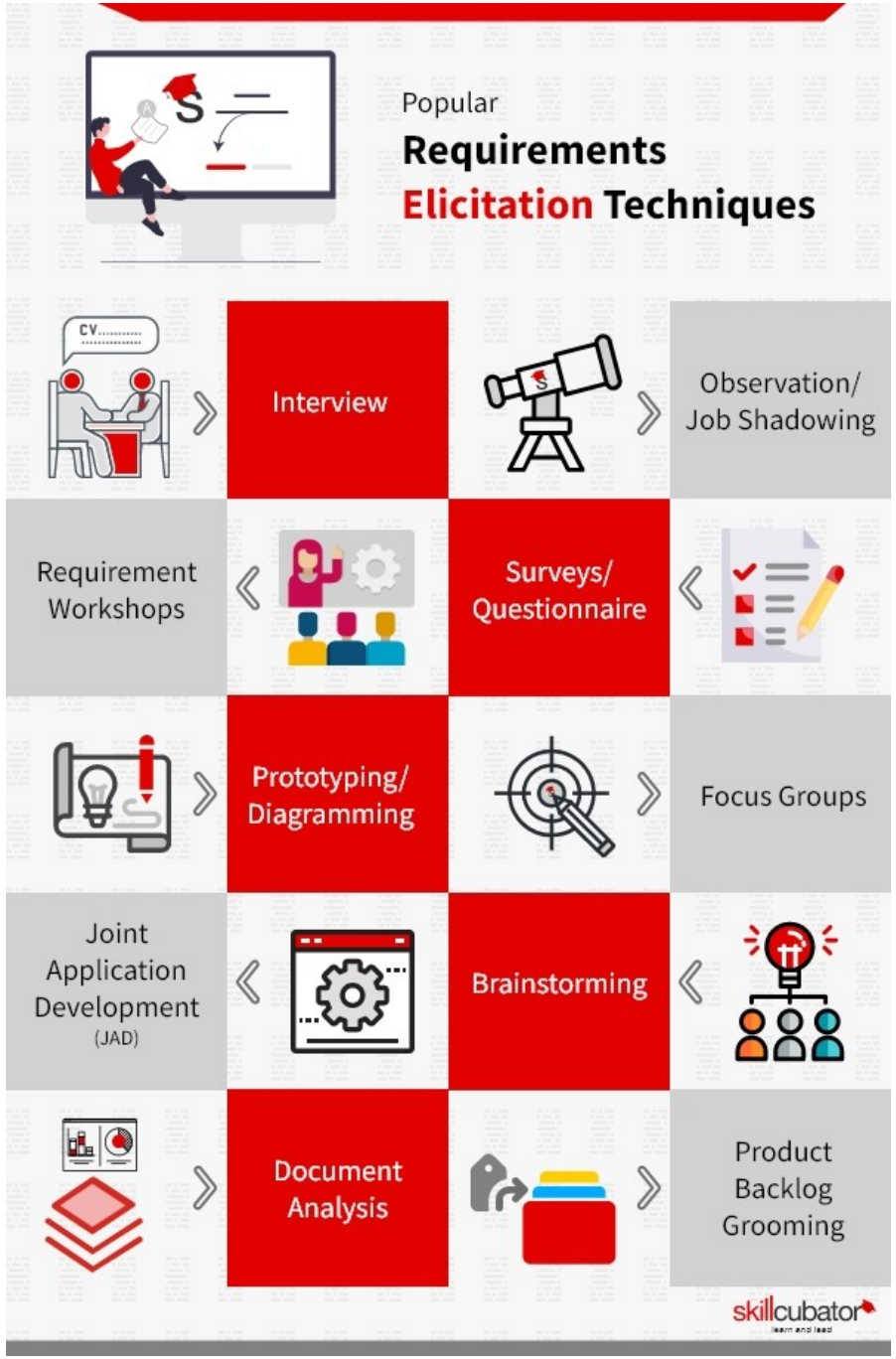
Requirements Elicitation Techniques

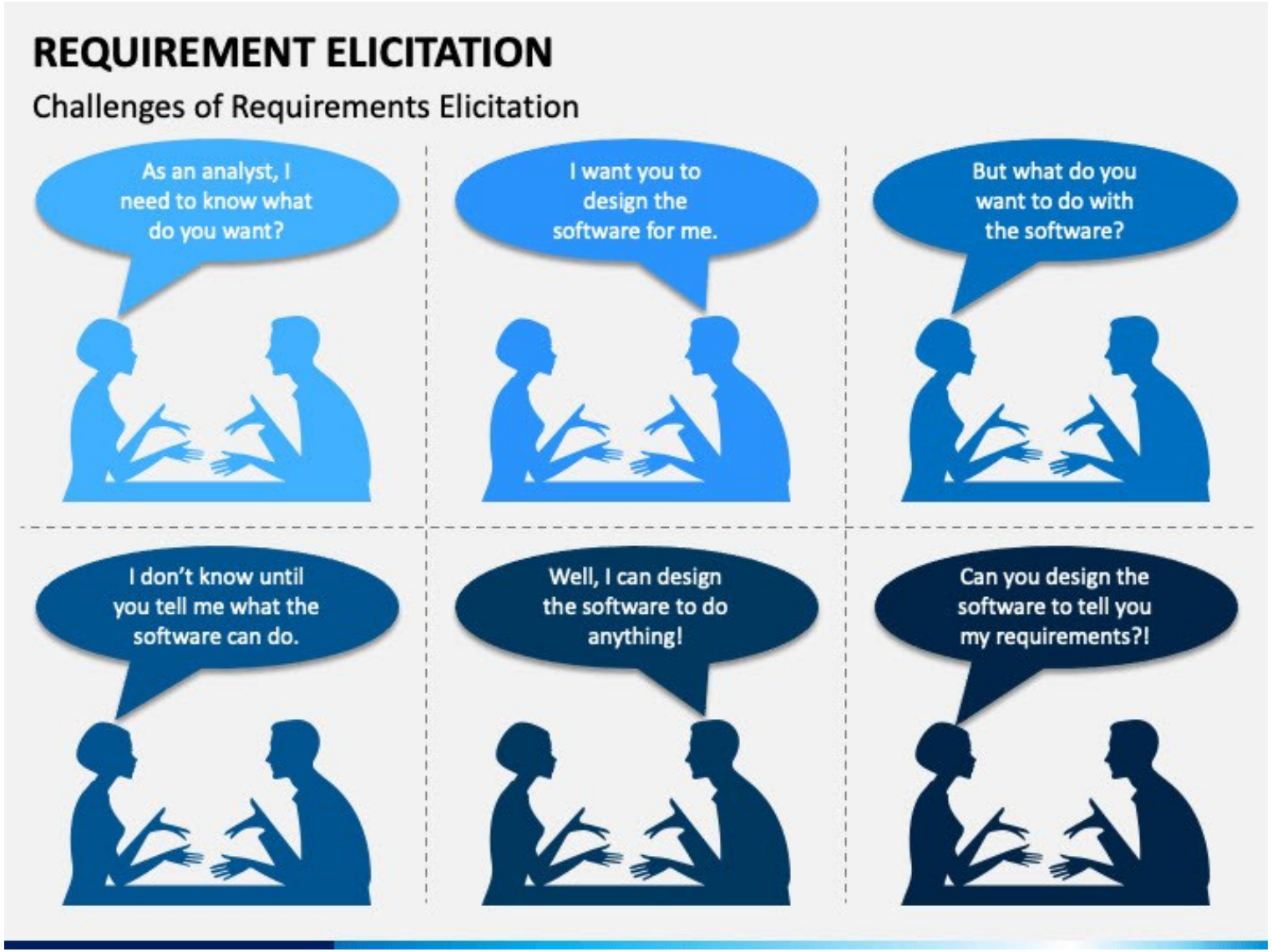
Requirements Elicitation Techniques

- To **engage with stakeholders** from various backgrounds to gather insights about the envisaged system.
- This process may be complemented by leveraging insights from existing systems and their usage, as well as information extracted from diverse documents.
- Important to invest time in comprehending users' workflows, the output they generate, their interactions with other systems, and the potential adjustments required to integrate a new system seamlessly.



How to Engage?





Requirements Elicitation Techniques

- It’s all about the question you ask.
  - You should have questions from the lab.
  - Because there is a lack of clarify in the initial requirements brief.
- It’s rarely enough, even, to accept the initial answer to a question.
  - Because that should highlight more things to ask about.
- Use the **initial questions** you had from the brief.
  - To set **investigation goals**, or main topics to discuss.
  - **Choose a technique** to best learn about these things.

Surveys/Questionnaire

- **Good for contacting lots of people**, getting a majority view.
- Bad for understanding something in detail.
- Most common **mistake**: Sending a badly made questionnaire.
  - There are many types of question you can ask.
  - These can be ‘designed well’ to extract a good answer.
- 2nd common mistake: reinvent the wheel.
  - There are well tested, proven surveys you can use.
  - Perceived Ease of Use Scale, Usability Questionnaires, NASA-TLX
- Other common mistakes
  - Two parts questions, leading questions, surveys are too long.

Interview/Focus Group

- **Allow you the freedom to ask follow up questions**.
- Strong interviews have 2 characteristics.
  - Interviewer is open minded, unbiased, **ready to listen**.
  - Interviewer gets interviewee to be **relaxed and chatty** and involved.
- Strongly recommended: Get people **to do more than just talk**.
  - Draw diagrams/explanations.
  - Show them things to discuss.
- Common mistake:
  - Start an interview without a plan: Should always go with a schedule of topics (even if you leave it behind).
  - This should be a todo list – not a script – be flexible: Specifically designed to get the most from whoever you are talking to.

Interview/Focus Group

Pros

- More poeple at once.
- Faster coverage of users.
- **Discuss differences & opposing opinions**.

Cons

- Possible conflict.
- Can get a ‘dominant speaker’.
- Takes longer to discuss per question.

8-12 people for breadth, 5-7 for depth, must do more than 1, 2.

Ideally, 2 organisers: 1 leader + 1 notetaker.

Observation

Observation

- Allows you to **see things people didn't say**.
- People rarely know the whole picture (just their slice).
- People are not good at realising everythng that's important for you to know.
- Watch and observe what people do:
  - Where people are.
  - Where things are.

Advanced Technique – Technology Tours

- Helps you **identify what things** the “new software” will work with.
- Aim:
  - Discover all related technology in a task.
- Approach:
  - Find out about all the technology in a room.

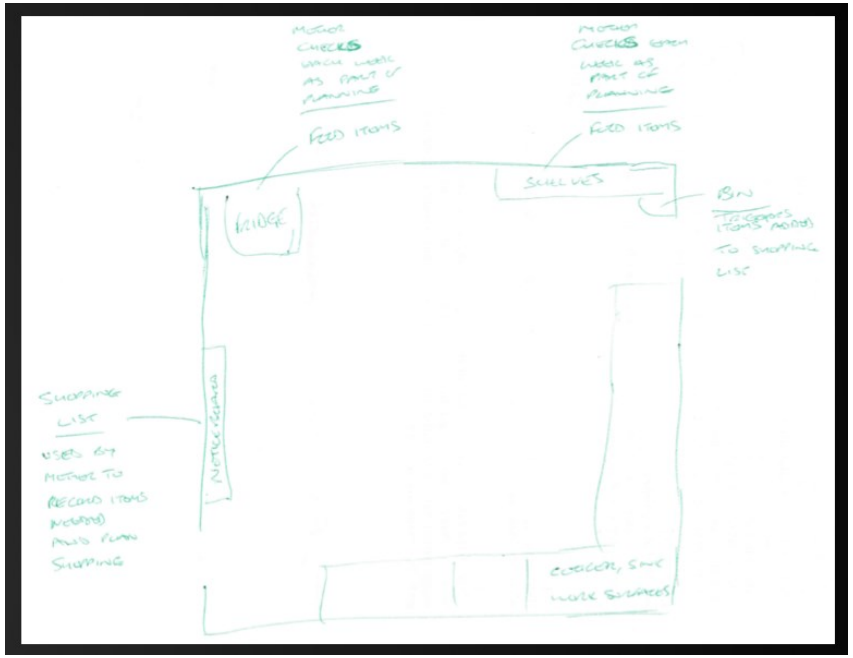
UPLC

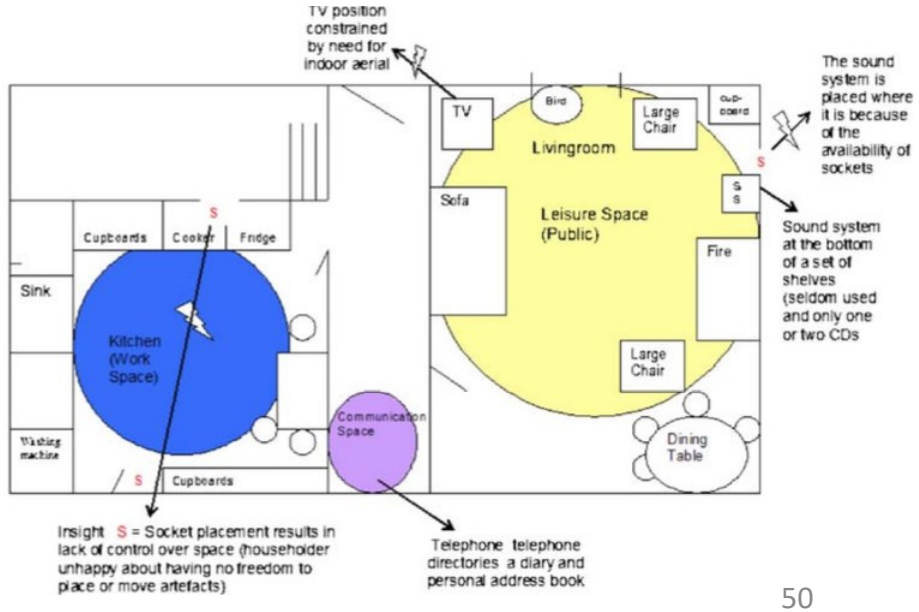
Advanced Technique – Technology Tours

- **Key Questions**:
  - What technology is present in this room?
  - Where is it placed?
  - Who uses the technology?
  - What activities does it support?
- Can be helpful activity in interviews, or Contextual Inquiry.

Advanced Technique – Technology Tours

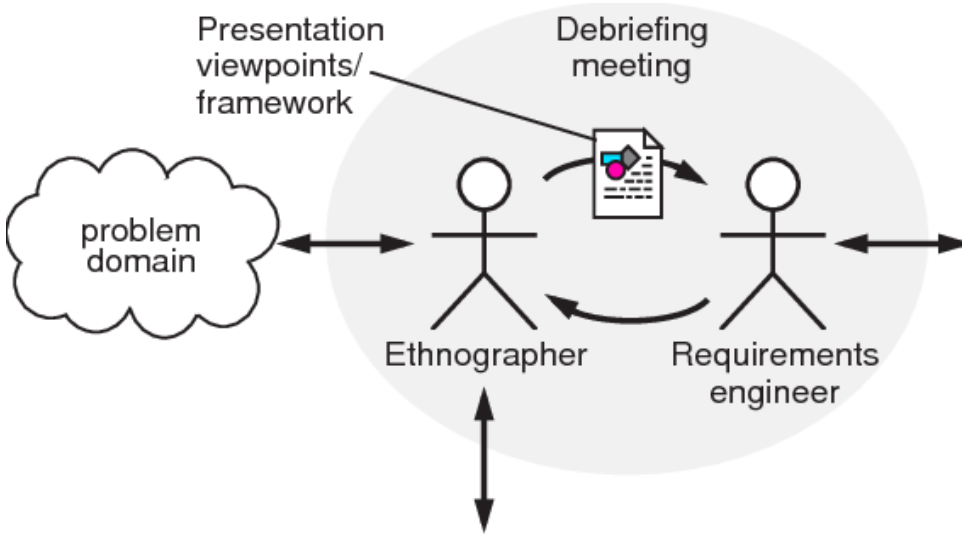
- **Create diagrams** of this layouts.
- Don't limit yourself to "computers".
  - Is a whiteboard important in the process?
  - Why is it?
  - Maybe it compensates for technology they want?
- You could extend this to a virtual idea.
  - E.g., servers that the workers use to do tasks.





Advanced Technique – Ethnography

- There's no better way to learn than **doing**.
- It got an anthropological background to the idea.
- **Explore the client's work culture**.
  - Get involved, sit and work in their office for a day.
- Traditionally – the longer the better.
  - But any taking part is better than not.
- Focused Ethnography
  - Targetting specific tasks, or roles in the company.
  - Don't wait (if you can't) for a task to happen naturally.



Requirements Elicitation Techniques – Overall Strategy

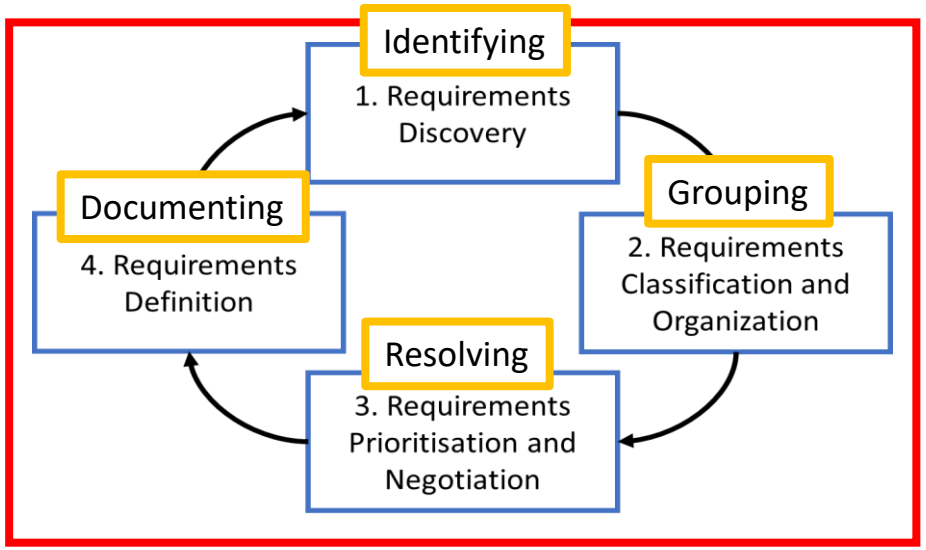
- Most mistakes are **because people don't have a strategy**.
  - They send out a survey before they know what the survey should be about.

- One method can **help understand the results of another**.
  - You can use a survey to better understand something from an interview.
  - You can interview people problems that a survey brings up.
  - You might do an interview after you observed someone for a while.
  - You might decide who to interview based on observing an office.

- Don't forget – decide what you want to learn more about.
  - Identify the stakeholder to investigate and pick a method to suit.

Summary

- Why requirements are so important.
  - That **over-budget costs are mostly associated with bad requirements**.
- What "Requirements Engineering" is (RE) – the first step of SE phases.
  - User requirements
  - System requirements (or system specifications)
- **Solution requirements** – difference between functional and non-functional requirements.
- **Stakeholder requirements** – primary, secondary and tertiary stakeholders.
- RE consists of three main activities:
  - **Requirements Elicitation and Analysis** (starts today!)
    - Requirements Elicitation Techniques focusing on the "**Identifying**" and "**Resolving**".
  - Requirements Definition
  - Requirements Validation




THANK
YOU