Software Engineering

A Faculty of Engineering Course: CSEN 406

Requirements Engineering

3

Dr. Iman Awaad

iman.awaad@giu-uni.de

GERMAN INTERNATIONAL UNIVERSITY
الجامعة الألمانية الدولية

Acknowledgments

The slides are **heavily** based on the **slides** by **Prof. Dr. John Zaki**.

They are also **heavily** based on the slides and textbook by **Ian Somerville**.

Their contribution is gratefully acknowledged.

Any additional sources are referenced.

Requirements Engineering

- Functional & non-functional requirements
- The software requirements document
- Requirements specification
- Requirements engineering processes
- Requirements elicitation and analysis
- Requirements validation
- Requirements management

How do we ensure a product meets stakeholder needs?

What are the consequences of bad requirements engineering?

How are these requirements obtained?

How do we categorise and specify them?

What is requirements engineering?

What is requirements engineering?

Requirements engineering

…is the **systematic process** of **defining**, **documenting**, **analyzing**, and **managing** the **needs** and **expectations** of **stakeholders**.

What were the requirements?

Dacia Bigster

Ford Mustang

https://upload.wikimedia.org/wikipedia/commons/8/8d/Dacia_Bigster_DSC_8686.jpg

https://www.topgear.com/car-reviews/ford/mustang

What were the requirements?

Polaroid Project

- 2.5 minute wait
- No sound
- Couldn’t play it on a TV

https://www.pexels.com/photo/camera-design-electronics-equipment-584404/

Consequences…

F-35

- Fly in formation
- Persistent challenges in developing and testing its software!
- Target error, seeing double… Software errors!!

  “Software problems caused Lockheed Martin to store 72 F-35 jets in its facilities for over a year, though the backlog was cleared by July 2025”

FAIL

https://commons.wikimedia.org/wiki/File:F-35_Side_View.jpg

Consequences…

CareFusion Alaris Pump Recall 02.2020

- Pump overdose when not needed
- Pump delay infusion
- Fatal software errors

FAIL

https://www.mpo-mag.com/wp-content/uploads/sites/7/2024/08/12_main-10.jpg

Consequences…

An integrated digital patient record system…

National Health Service, civilian IT, UK

- Electronic health records, digital scanning, and integrated IT systems across hospitals and community care.
- Scope creep
- “Fiasco”, "the biggest IT failure ever seen”
- …At a cost of 12 billion GBP

https://museumoffailure.com/exhibition/nhs-national-programme-for-it

Consequences…

An integr

ital
ystems
nity care.

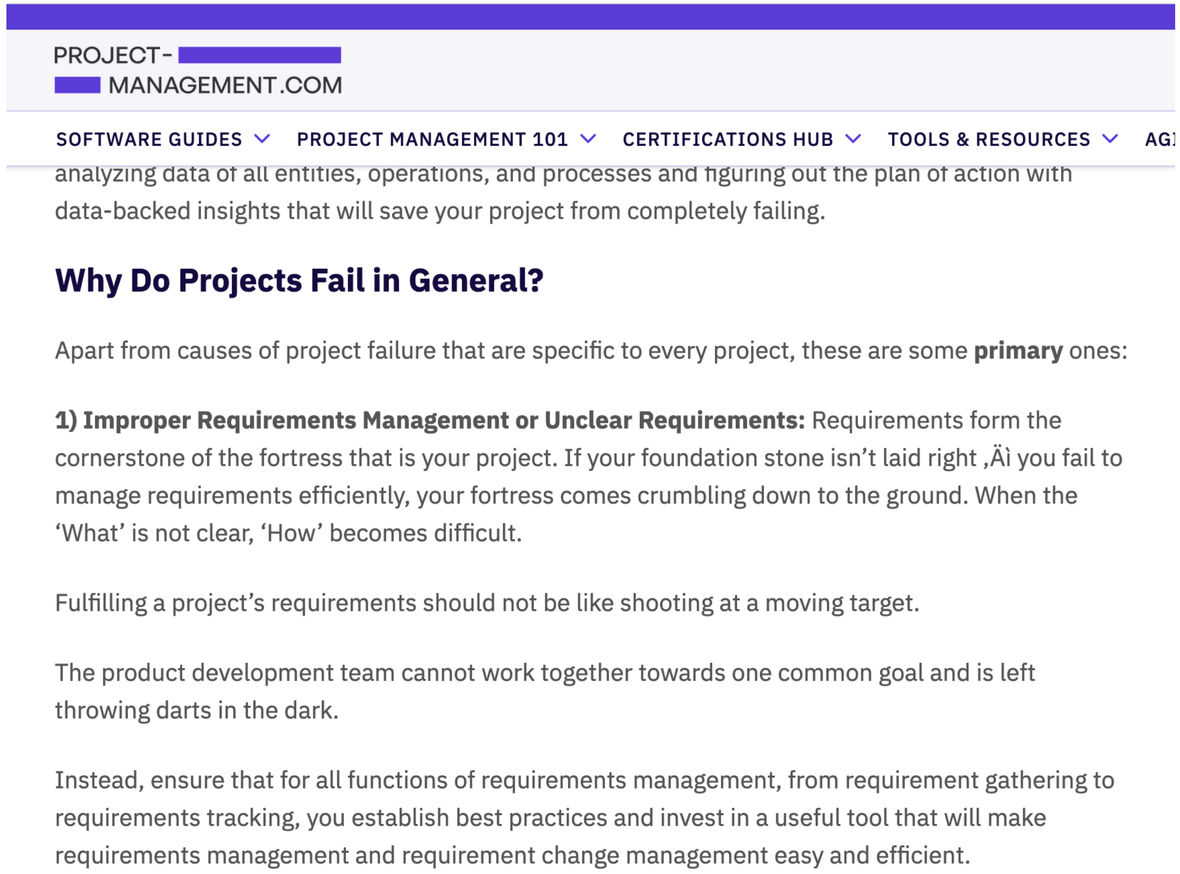
re ever

National H

- Resistance created by **top-down decisions** being made on behalf of local organizations
- Scale of the project was grossly **underestimated**
- The programme's **innovative procurement procedures** caused crippling **contractual problems** with suppliers
- **End users** were **not adequately involved** in the development process, so the system had an '*unprecedented scale and boundless complexity*' that made it **difficult for healthcare professionals to use**
- …

https://museumoffailure.com/exhibition/nhs-national-programme-for-it

Why do projects fail?



PROJECT- [redacted]
[redacted] MANAGEMENT.COM

SOFTWARE GUIDES | PROJECT MANAGEMENT 101 | CERTIFICATIONS HUB | TOOLS & RESOURCES | AGI

analyzing data of all entities, operations, and processes and figuring out the plan of action with data-backed insights that will save your project from completely failing.

Why Do Projects Fail in General?

Apart from causes of project failure that are specific to every project, these are some **primary** ones:

1) Improper Requirements Management or Unclear Requirements: Requirements form the cornerstone of the fortress that is your project. If your foundation stone isn't laid right ,Äì you fail to manage requirements efficiently, your fortress comes crumbling down to the ground. When the 'What' is not clear, 'How' becomes difficult.

Fulfilling a project's requirements should not be like shooting at a moving target.

The product development team cannot work together towards one common goal and is left throwing darts in the dark.

Instead, ensure that for all functions of requirements management, from requirement gathering to requirements tracking, you establish best practices and invest in a useful tool that will make requirements management and requirement change management easy and efficient.

1. Improper requirements management or unclear requirements
2. Gaps in communication: Poor communication escalates into a catastrophe very quickly because it is a silent killer (pun intended).
3. Improper planning
4. Using complex tools without prior practice
5. Absence of good project leadership
6. Inefficient project tracking

https://project-management.com/a-project-managers-guide-to-failure-and-getting-back-up-after-you-fall/

Take a minute…

What is requirements engineering?

Take a minute…

Why is requirements engineering important?

Requirements v constraints

Requirements

What the system should do

Constraints

What the system should **NOT** do

What are requirements?

- **Anything that drives design choices…**
- Both the **user's view** of external system behaviour & the **developer's view** of internal characteristics
- A **property** (or attribute) of a product **must have** to provide value to a stakeholder, including both **behaviour of the system *under specific conditions*** and those **properties** that make it **suitable & enjoyable for use**
- **Specifications** of what should be implemented
- **Descriptions** of how the system should behave
- **Constraint** on the development process of the system

How do we get these?

Process of **elicitation**, **analysis specification**, **verification & validation**, and **management** of the stakeholder requirements

How things work…

Company defines its needs in a sufficiently abstract way that a solution is not pre-defined…

The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs…

Once a contract has been awarded, the **contractor must write a system definition for the client** in more detail so that the **client understands** and can **validate** what the software will do.

Both of these documents
may be called the **requirements document** for the system.

How things work…

Document 1

Company defines its needs in a sufficiently abstract way that a solution is not pre-defined.

The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs.

Document 2

Once a contract has been awarded, the **contractor must write a system definition for the client** in more detail so that the client understands and can validate what the software will do.

Both of these documents
may be called the **requirements document** for the system.

Types of Requirements

Document 1

Functional Requirements

User requirements

Can be **high-level statements** of **what** the system or actors can do

Explained in **natural language & diagrams**

Represents **system**, **it's functions + constraints**

"**detailed requirements**": more **detailed**

Document 2

Non-functional Requirements

System requirements

Types of Requirements

Document 1

aka system requirements doc / functional specification doc

Functional Requirements

User requirements

Can be **high-level statements** of **what** the system or actors can do

Explained in **natural language & diagrams**

Represents **system**, **it's functions + constraints**

"**detailed requirements**": more **detailed**

Non-functional Requirements

System requirements

Detail the system's **functions**, **services** & **operational constraints**

Must be precise, defining **exactly** what is to be implemented.

(See also **Domain** requirements: …come from the application domain of the system and reflect characteristics and constraints of that domain. They may be functional or non-functional requirements)

Types of Requirements

Document 1

aka system requirements doc / functional specification doc

Functional Requirements (FR)

User requirements

Can be **high-level statements** of **what** the system or actors can do

Explained in **natural language & diagrams**

Represents **system**, **it's functions + constraints**

"**detailed requirements**": more **detailed**

Non-functional Requirements (**NFR**)

System requirements

Detail the system's **functions**, **services** & **operational constraints**

Must be precise, defining **exactly** what is to be implemented.

Include *e.g.* security, reliability, usability, performance…

Types of Requirements

Document 1

Functional Requirements (FR)

User requirements

Can be **high-level statements** of **what** the system or actors can do

Explained in **natural language & diagrams**

Represents **system**, **it's functions + constraints**

"**detailed requirements**": more **detailed**

aka system requirements doc / functional specification doc

Non-functional Requirements (**NFR**)

System requirements

Detail the system's **functions**, **services** & **operational constraints**

Must be precise, defining **exactly** what is to be implemented.

Include *e.g.* security, reliability, usability, performance…

A non-functional requirement may lead to a functional one!

User v system requirements: Libsys example

aka Functional Req.

User requirement definition

1. LIBSYS shall keep track of all data required by copyright licensing agencies in the UK and elsewhere.

→ Client managers
System end-users
Client engineers
Contractor managers
System architects

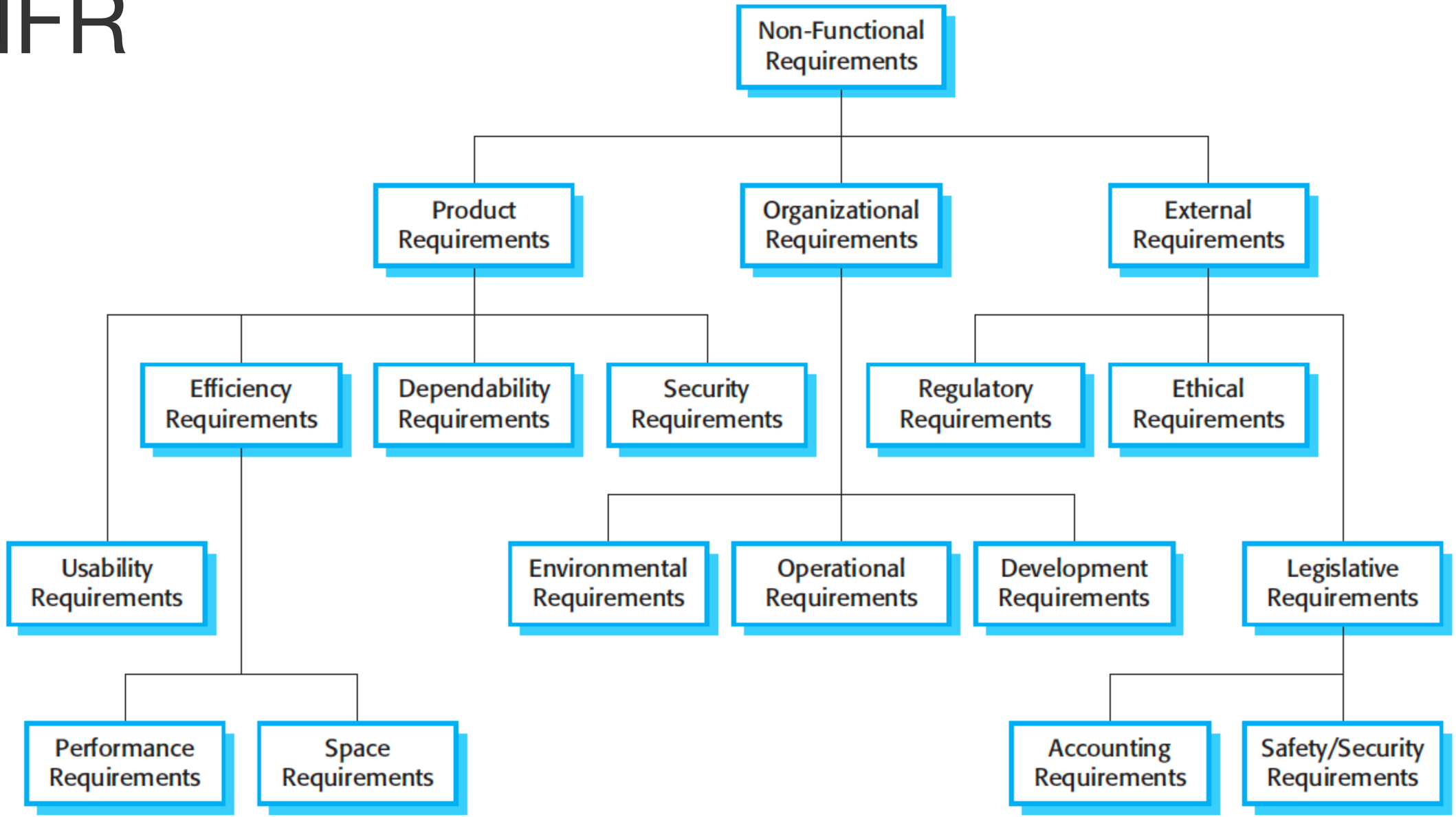
aka Non-Functional Req.

System requirements specification

1.1 On making a request for a document from LIBSYS, the requestor shall be presented with a form that records details of the user and the request made.
1.2 LIBSYS request forms shall be stored on the system for five years from the date of the request.
1.3 All LIBSYS request forms must be indexed by user, by the name of the material requested and by the supplier of the request.
1.4 LIBSYS shall maintain a log of all requests that have been made to the system.
1.5 For material where authors' lending rights apply, loan details shall be sent monthly to copyright licensing agencies that have registered with LIBSYS.

→ System end-users
Client engineers
System architects
Software developers

NFR


Non-Functional Requirements
Product Requirements
Organizational Requirements
External Requirements
Efficiency Requirements
Dependability Requirements
Security Requirements
Regulatory Requirements
Ethical Requirements
Usability Requirements
Environmental Requirements
Operational Requirements
Development Requirements
Legislative Requirements
Performance Requirements
Space Requirements
Accounting Requirements
Safety/Security Requirements

Non-functional Requirements

Product Req.

Specify or constrain software behaviour

Organisational Req.

broad system req.

derived from policies and procedures in the customer's & developer's organizations

External Req.

all requirements that are derived from factors external to the system and its development process

Non-functional Requirements

Product Req.

Specify or constrain software behaviour

e.g.

- performance requirements:
  - how fast the system must execute
  - how much memory it requires
- reliability requirements:
  - acceptable failure rate
  - security requirements
  - usability requirements

Non-functional Requirements

Consider a Patient Management System

Product Req.	Organisational Req.	External Req.
Specify or constrain software behaviour	broad system req. derived from policies and procedures in the customer's and developer's organizations	all requirements that are derived from factors external to the system and its development process
The system shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30).	Users of the system shall authenticate themselves using their health authority identity card.	The system shall implement patient privacy provisions as set by law.
Downtime within normal working hours shall not exceed five seconds in any one day.		

Metrics for specifying non-functional requirements

Speed

Size

Ease of use

Reliability project management

Robustness

Portability

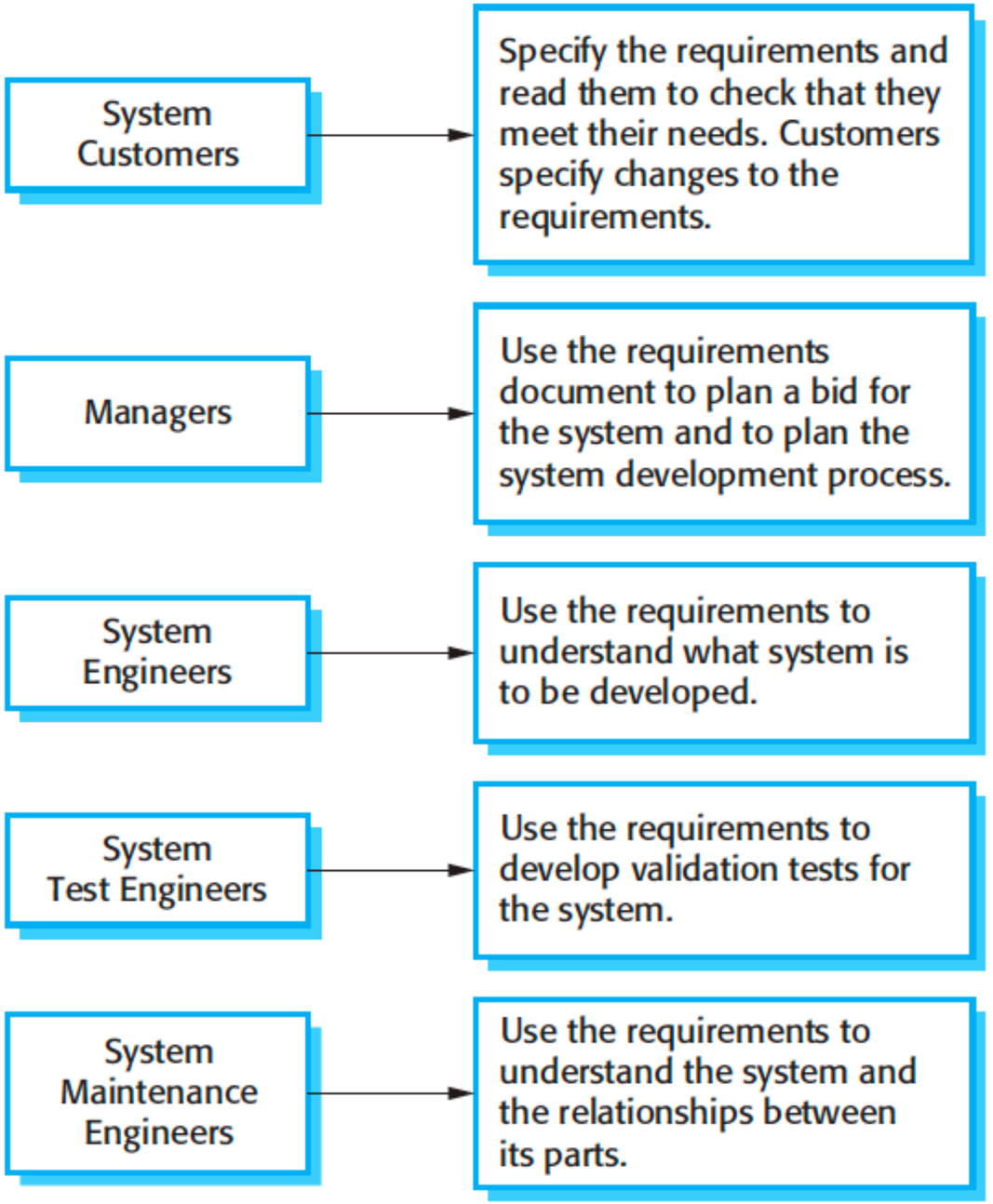
Take a minute…

Why are Non-functional requirements more important than functional requirements?

Users of the software requirements document

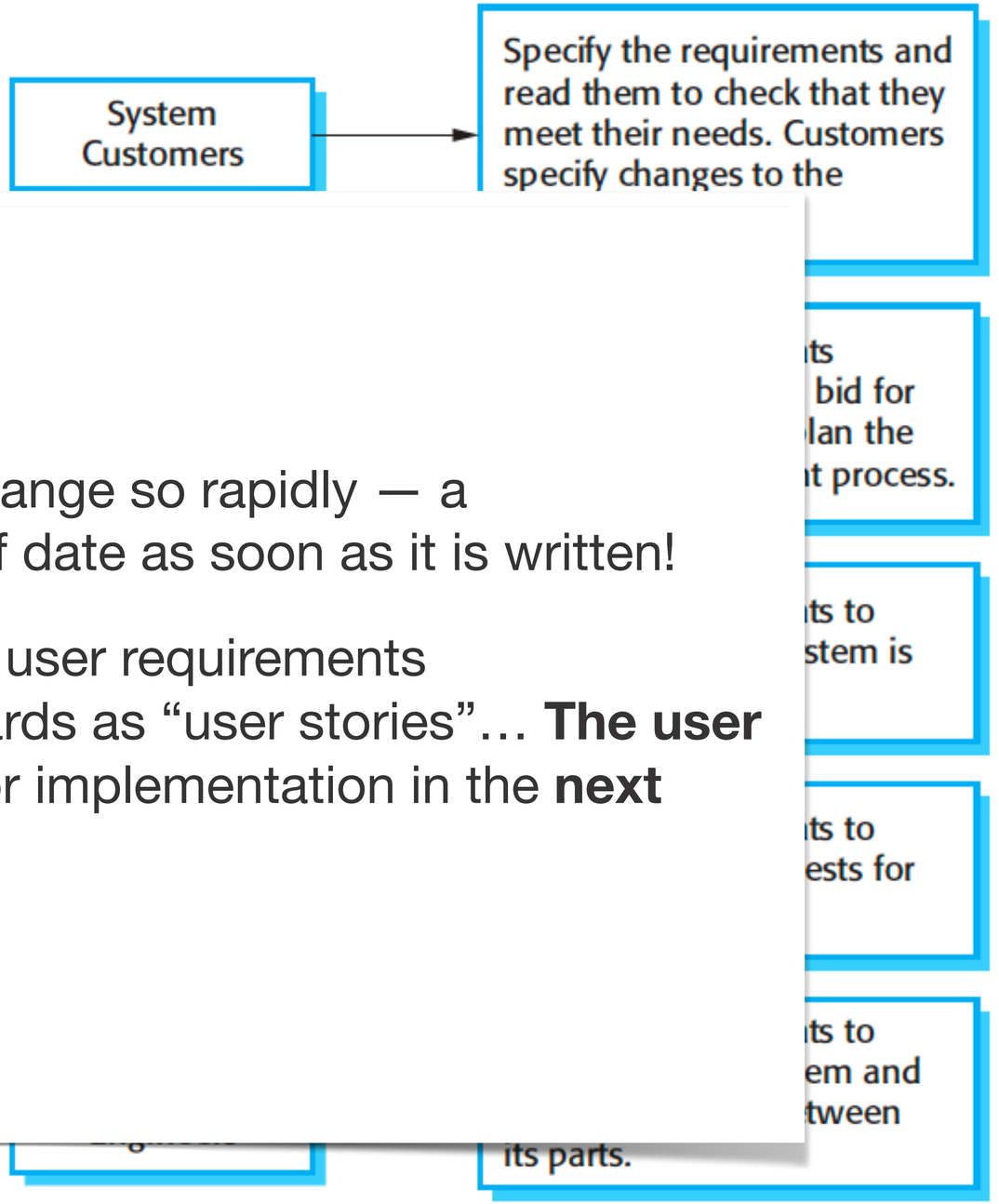
aka software requirements specification (SRS)

User requirements + detailed specification of system requirements



Users of the re

Use
specific



- **Agile** methods: requirements change so rapidly — a requirements document is out of date as soon as it is written!
- **Extreme programming:** collect user requirements **incrementally**, write them on cards as “user stories”… **The user then prioritizes** requirements for implementation in the **next increment** of the system.

Software requirements specification

see Fig. 4.7

What is included depends on:

- type of software
- approach to development

For your project: focus on defining **user requirements** and high-level, **non-functional system requirements**…

Specify only the **external behavior** of the system, no system architecture or design details: write them in natural language, with simple tables, forms, and intuitive diagrams…

Expanded versions of user requirements: they add detail and explain how the user requirements should be provided by the system. Should be a complete and detailed specification of the whole system.

Non-functional Requirements

Consider a Patient Management System

Remember this!?

Product Req.

Specify or constrain software behaviour

The system shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30).

Downtime within normal working hours shall not exceed five seconds in any one day.

Organisational Req.

broad system req. derived from policies and procedures in the customer's and developer's organizations

Users of the system shall authenticate themselves using their health authority identity card.

External Req.

all requirements that are derived from factors external to the system and its development process

The system shall implement patient privacy provisions as set by law.

System requirements specification

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract.

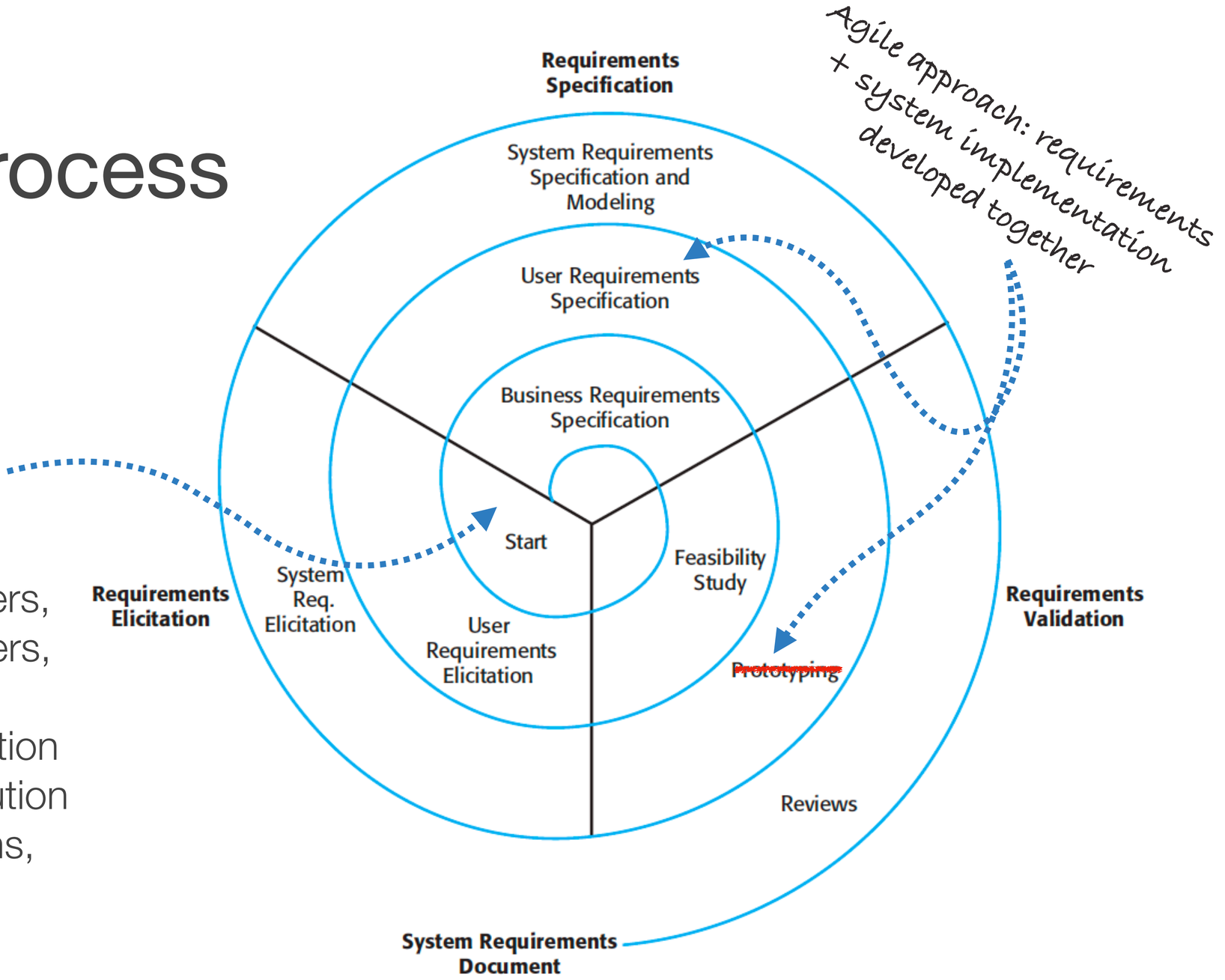
Natural language + UML diagrams + Tables

see Sec. 4.3

Requirements engineering process

Phase 1: **Inception**

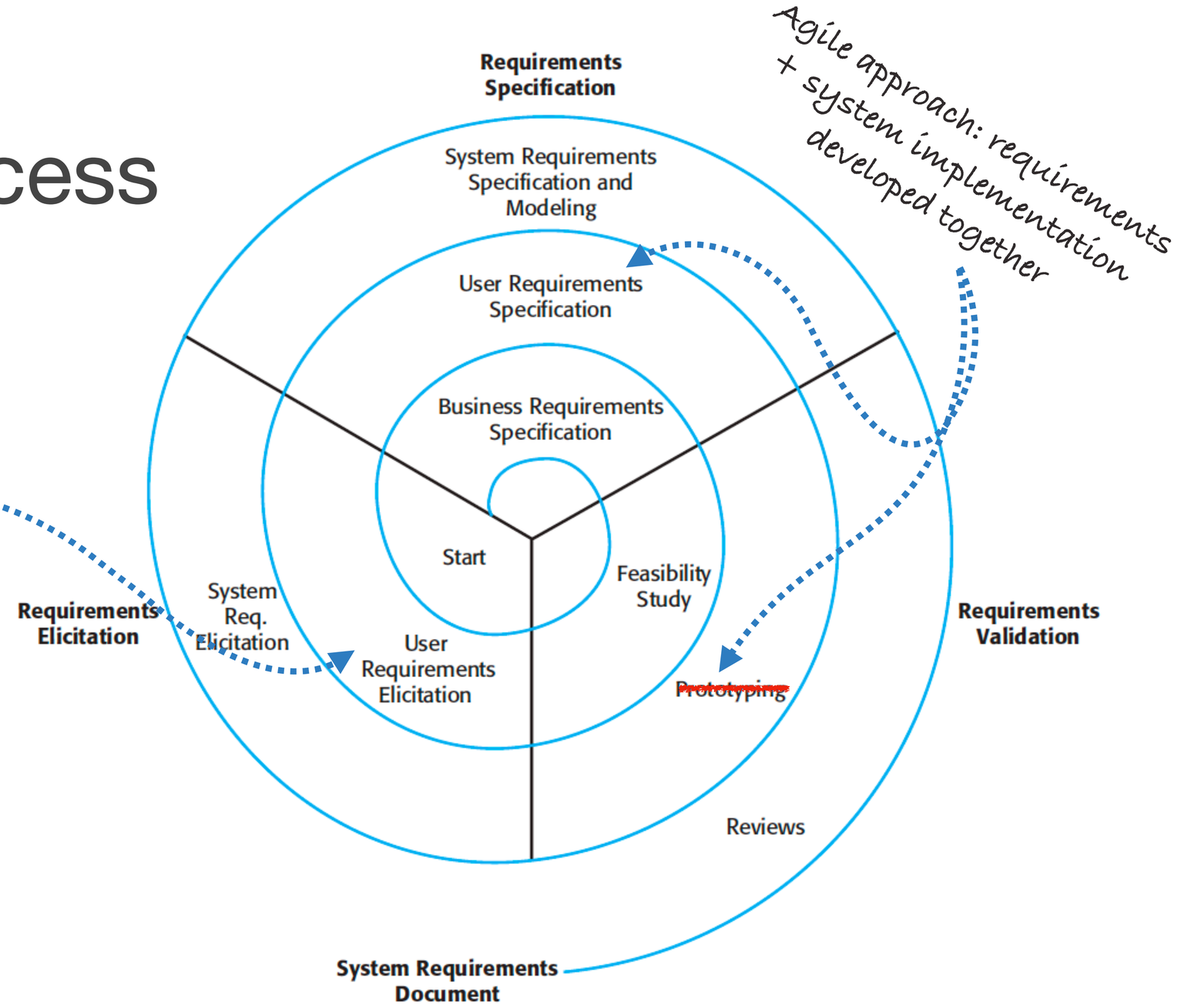
- Understand the market
- Develop the business case & feasibility study
- Talk to the different stakeholders, managers, marketing, end users, se, developers,…
- Understand nature of the solution and develop a preliminary solution through brainstorming sessions, joint application development meetings,…



Requirements engineering process

Phase 2: **Elicitation & analysis**

- Observe users in their jobs.
- Issues: volatility, ill-defined or too detailed req., little understanding between developers & customer.
- Collect from stakeholders: documentation, existing systems, domain experts through brainstorming, interviews, observation, use cases, scenarios,…

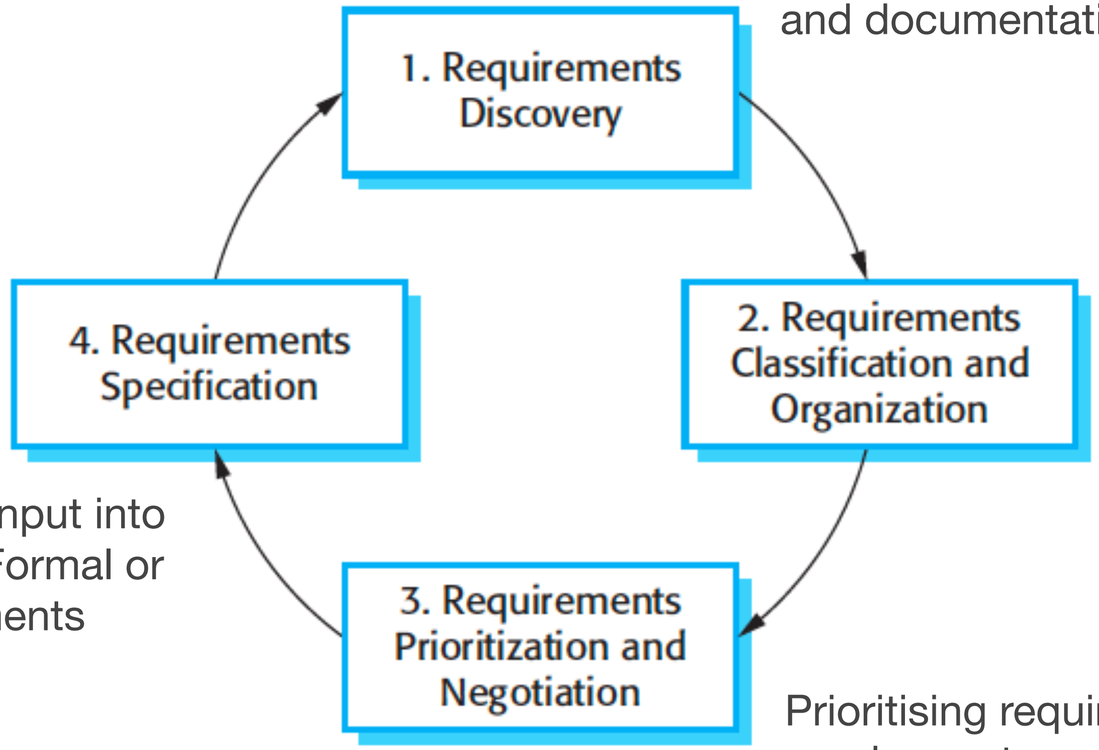


Requirements elicitation & analysis

…is a **process** during which software engineers **work with customers** and system **end-users** to find out about the **application domain**, what **services** the system should provide, the **required performance** of the system, **hardware constraints**, …

Elicitation & analysis process

Interacting with stakeholders to collect their requirements (including domain requirements) and documentation.



Group related requirements and organises them into coherent clusters.

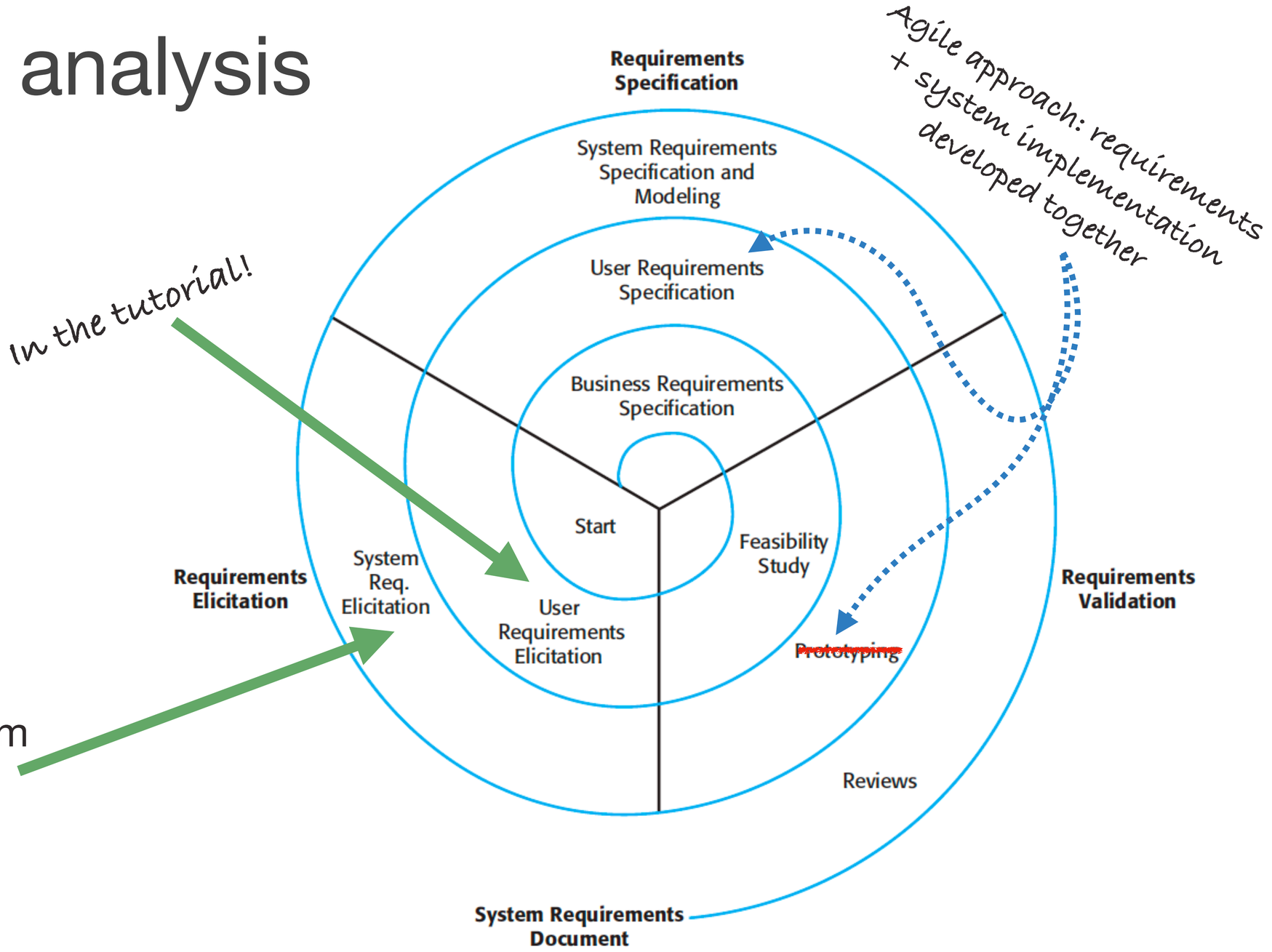
Prioritising requirements, finding & resolving requirements conflicts through negotiation.

Requirements documented, input into the next round of the spiral. Formal or informal requirements documents produced.

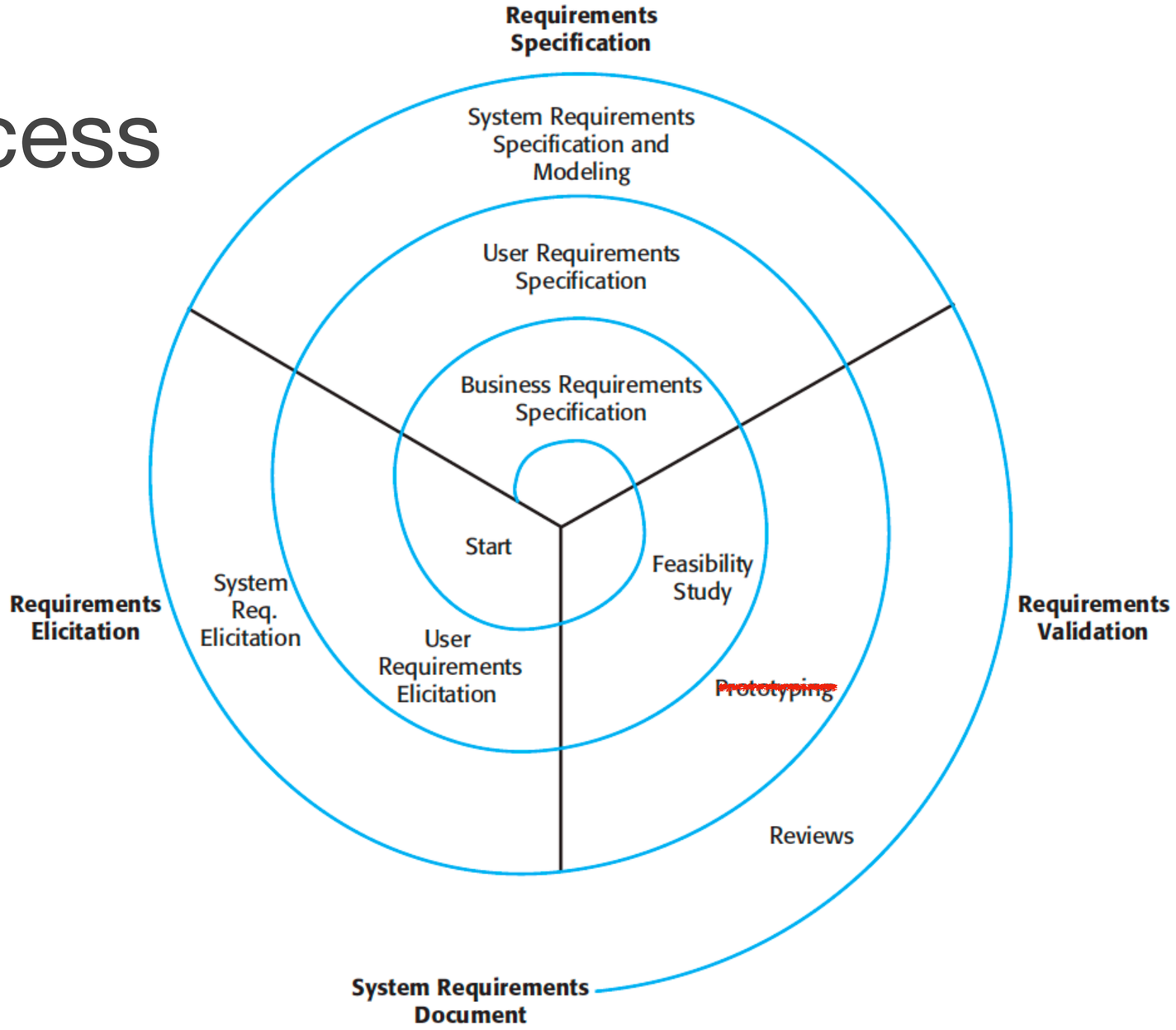
Elicitation & analysis process

1 High-level business, NFR and user requirements

2 Detailed system requirements



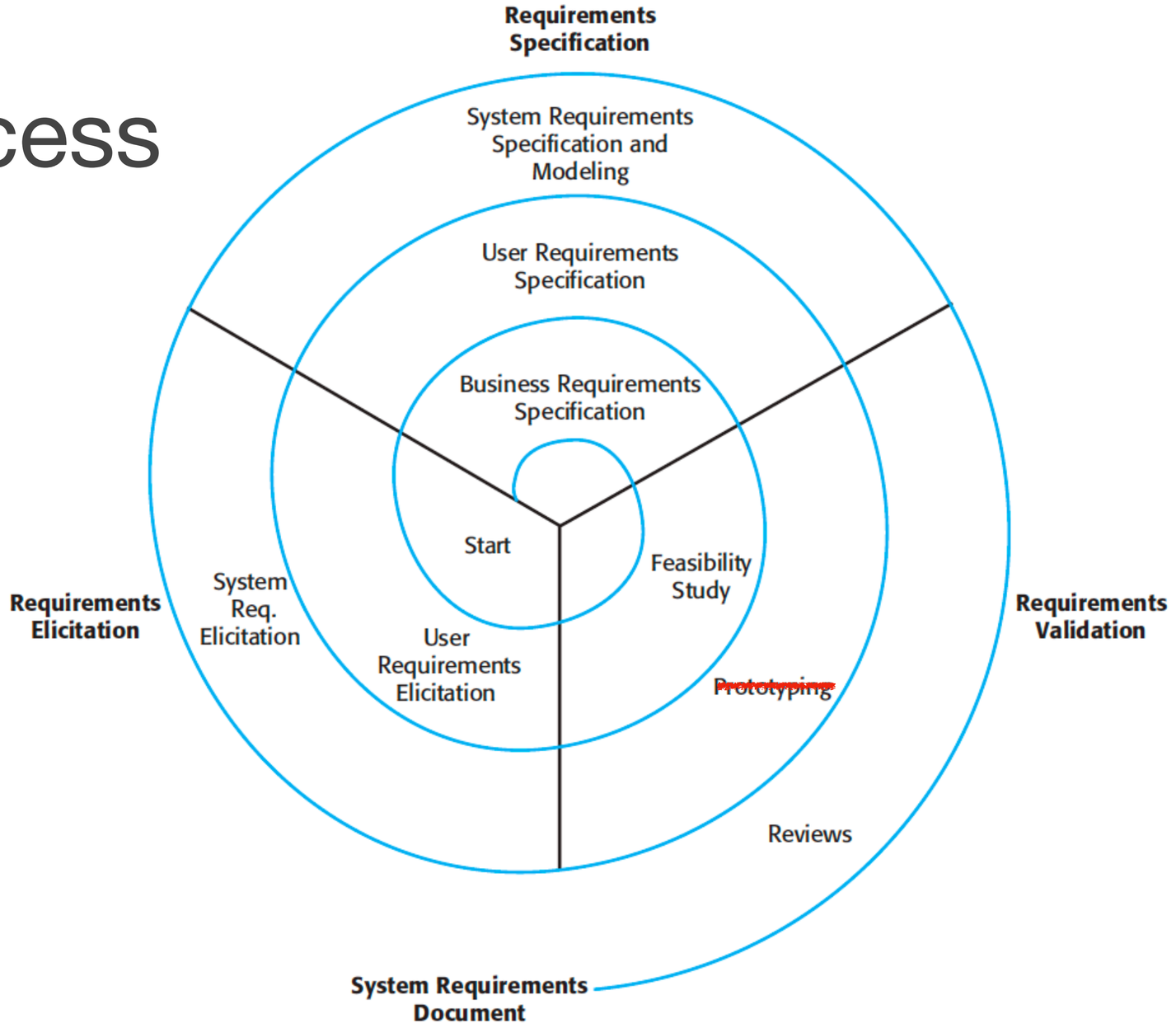
Requirements engineering process



Phase 3: **Elaboration**

- Refine what was done in the previous phases, expanding and looking deeper, …
- Carry out modelling activities
- analysis models,…

Requirements engineering process



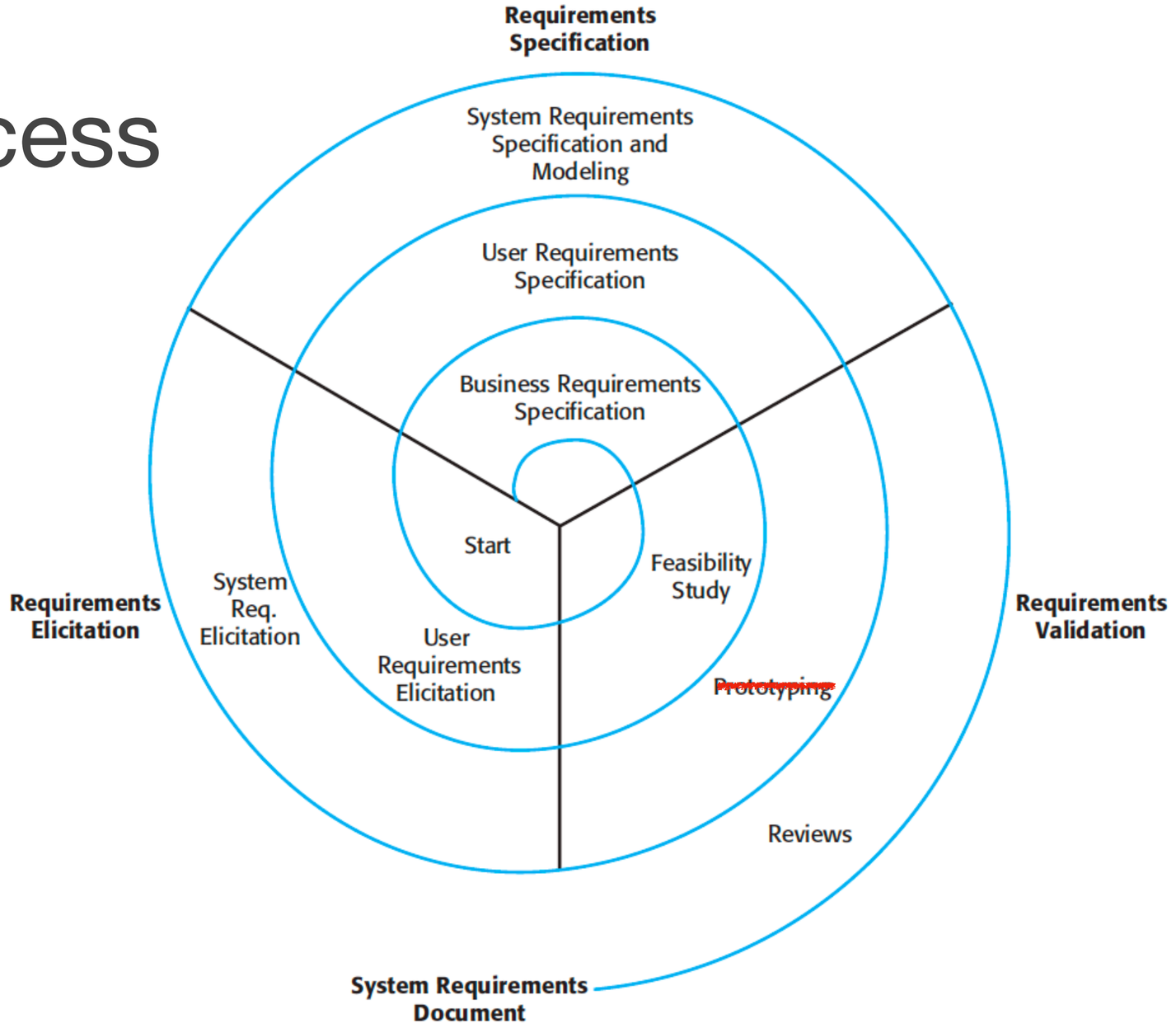
Phase 4: **Negotiation**

- Negotiate needs & wants: what to eliminate perhaps, prioritize the requirements, risk factored in
- Discussions around
- Availability of resources, delivery time, cost, scope of requirements & remove conflicts.

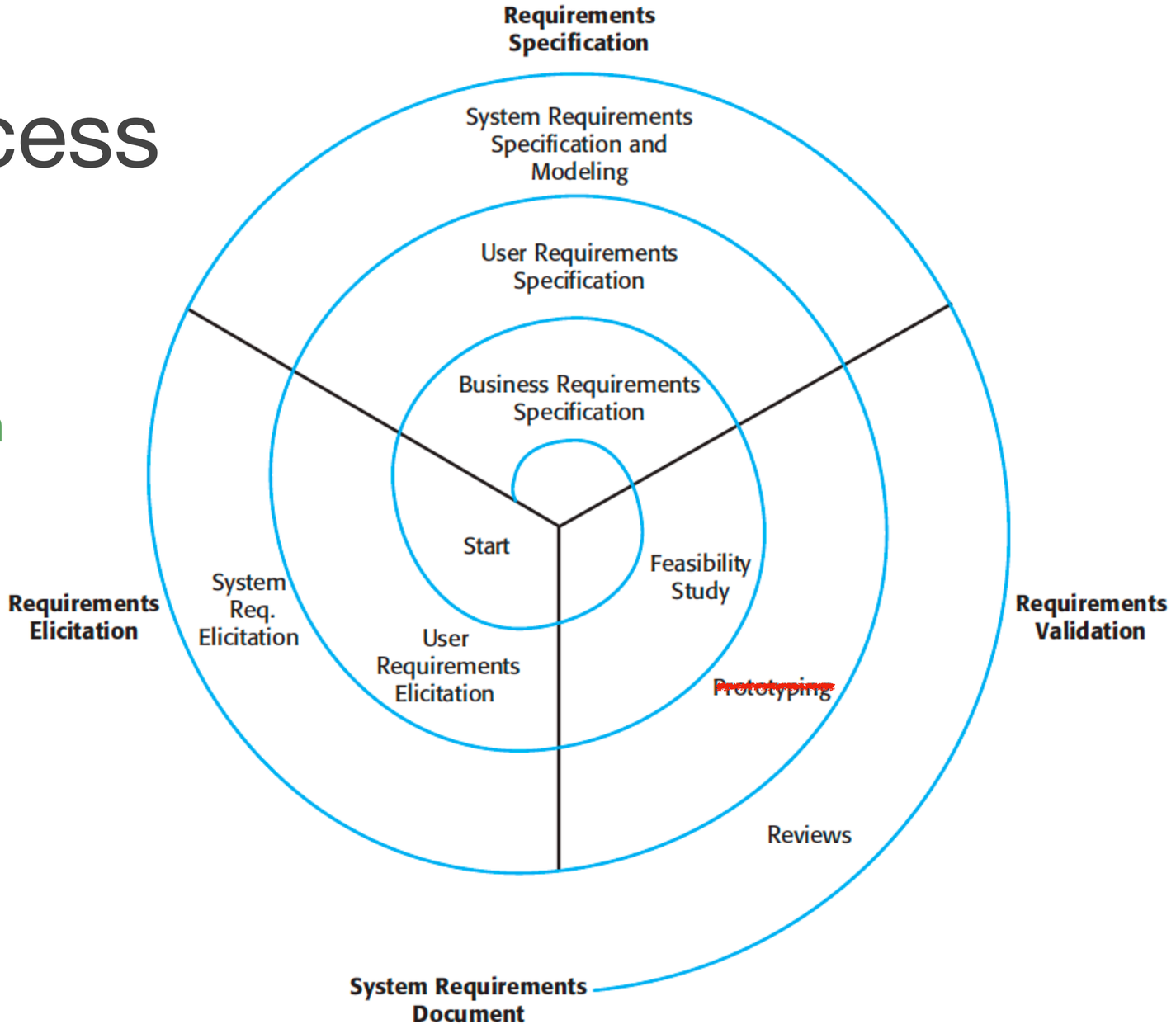
Requirements engineering process

Phase 5: **Specification**

- Written specifications (SRS), set of models, use cases, prototype, UML, UI/UX
- Document functions, features, or constraints
- Talk to different stakeholders, managers, marketing, end users
- Submit the document to the customer
- Written in a language that he/she understands (+ UML diagrams)



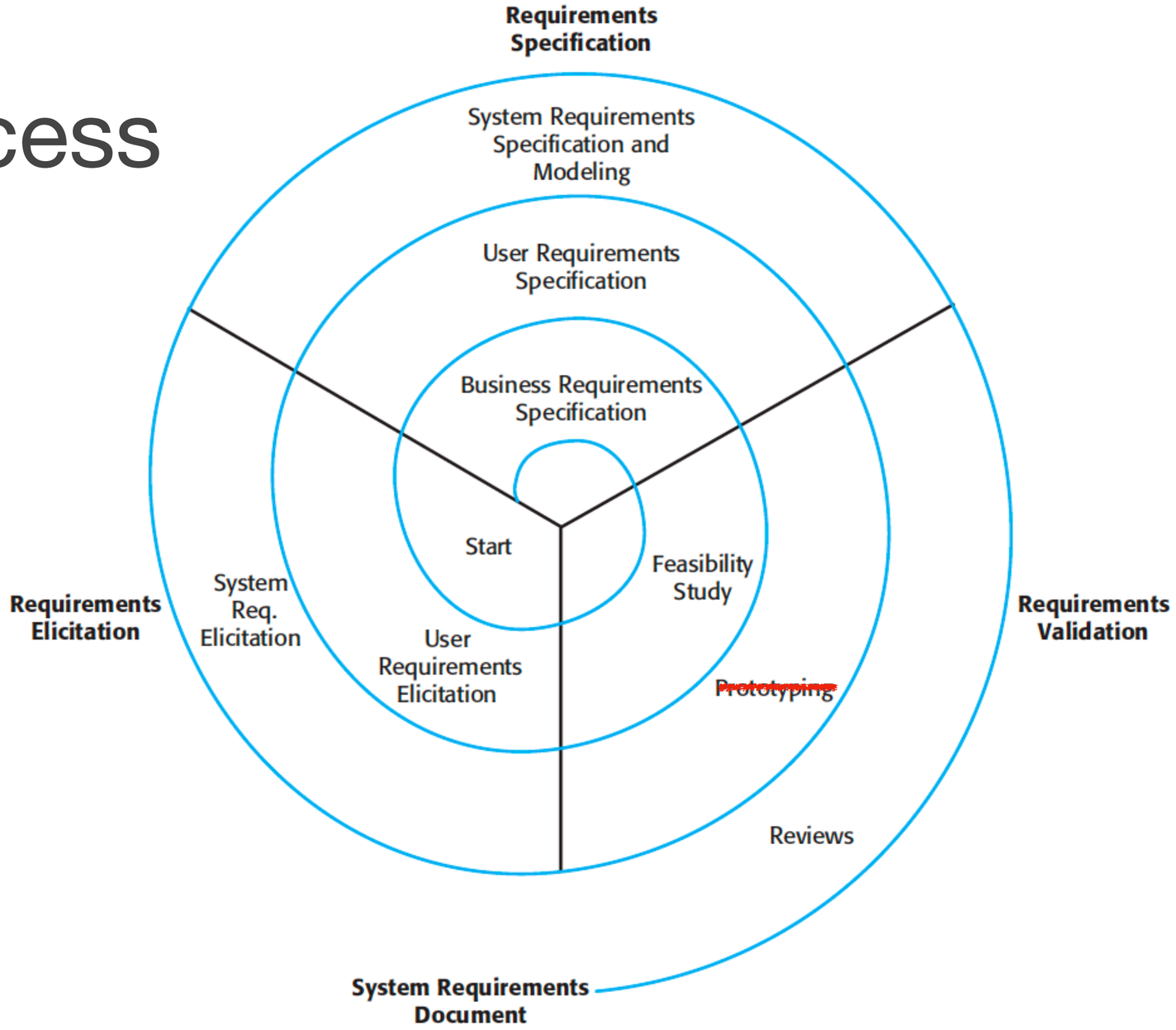
Requirements engineering process



Phase 6: **Verification & Validation**

- Technical review, missing information, checking errors, built according to standards: Simple sanity check, test-case generation, inspection, requirement review, coded prototypes, design prototypes
- **Verification**:
  built the **product correctly**….
- **Validation**:
  built the **correct product**…

Requirements engineering process



Phase 7: **Management**

- Identify, control, and track the req. for successful and smooth implementation.
- Requirements change over time…

Requirements elicitation & analysis

Remember this?

…is a **process** during which software engineers **work with customers** and system **end-users** to find out about the **application domain**, what **services** the system should provide, the **required performance** of the system, **hardware constraints**, …

Planning the Elicitation Process

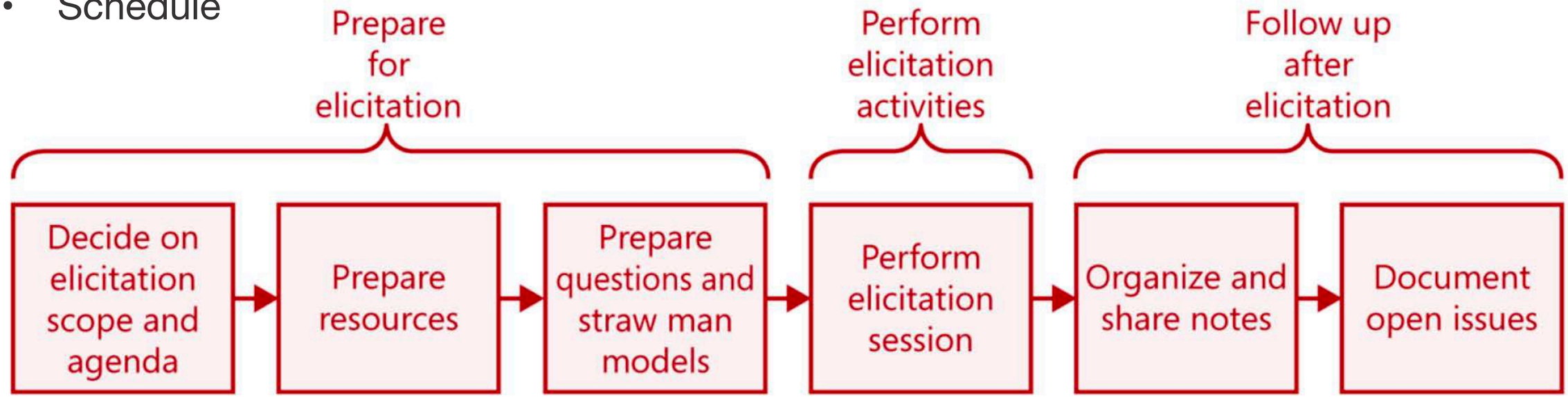
- Elicitation objective
- Documents needed
- Planned techniques
- Elicitation risks
- Schedule

Needs: Number of users, functions, features,…
Constraints: Time, cost,…

Planning the Elicitation Process

- Elicitation objective
- Documents needed
- Planned techniques
- Elicitation risks
- Schedule

Needs: Number of users, functions, features,…
Constraints: Time, cost,…



Planning the Elicitation Process

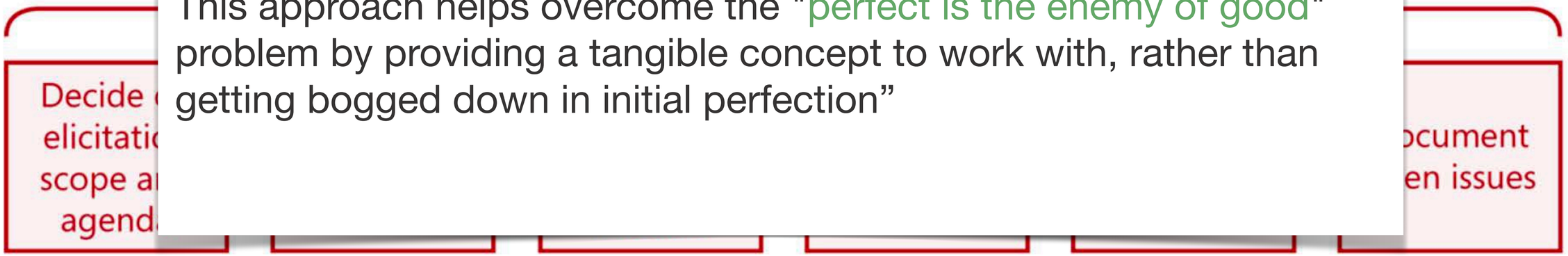
- Elicita
- Docur
- Planne
- Elicita
- Sched

eatures,…
e, cost,…

“A straw man model is is a preliminary, simplified draft of a product, plan, or proposal designed to stimulate discussion, feedback, and iterative improvement within a team or with stakeholders.

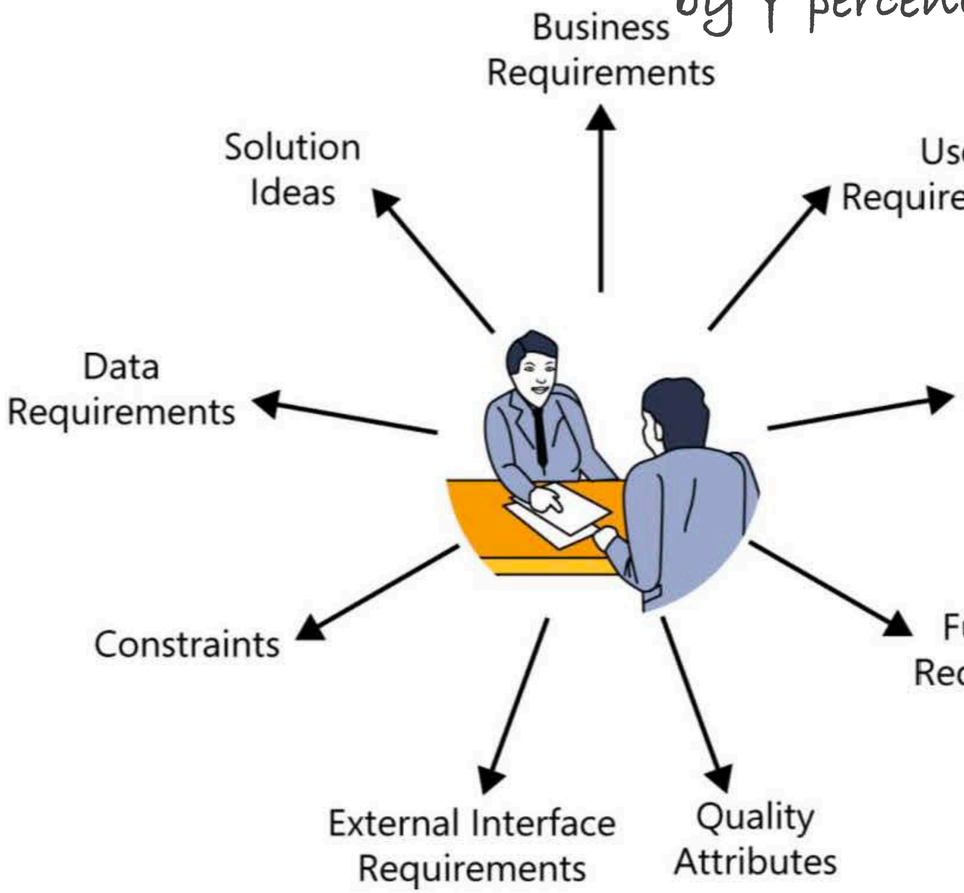
It's an **intentionally imperfect starting point**, built on incomplete information, that serves as a hypothesis to be quickly tested, criticised, and refined into a more robust final solution.

This approach helps overcome the "perfect is the enemy of good" problem by providing a tangible concept to work with, rather than getting bogged down in initial perfection”



Classifying customer input

Increase market share in region X
by Y percent within Z months



I need to print a mailing label for a package

A new client must pay 30% of the estimated consulting fee and travel expenses in advance

The user must be able to sort the project list in forward and reverse alphabetical order

How do you know when you’re done?

Elicitation techniques?

User interface analysis

Focus groups

System interface analysis

Interviews

Observation

Document analysis

Workshops

Surveys & questionnaires

When is which technique appropriate?

	Interviews	Workshops	Focus groups	Observations	Questionnaires	System interface analysis	User interface analysis	Document analysis
Mass-market software	x		x		x			
Internal corporate software	x	x	x	x		x		x
Replacing existing system	x	x		x		x	x	x
Enhancing existing system	x	x				x	x	x
New application	x	x				x		
Packaged software implementation	x	x		x		x		x
Embedded systems	x	x				x		x
Geographically distributed stakeholders	x	x			x			

Interviews

- Prepare questions
- Prepare straw man models
- Stay in scope
- Suggest ideas
- Actively listen
- Establish rapport

For example…

What are the problems?

What is the system?

How will the system solve the problems?

What are the goals of the system?

How will the system be used daily?

What are the constraints on the system?

How is the work is done now (frequency, transaction volume,…)?

Any performance considerations?

Interviews

- Prepare questions
- Prepare straw man models
- Stay in scope
- Suggest ideas
- Actively listen
- Establish rapport

Common interview mistakes?

For example…

What are the problems?

What is the system?

How will the system solve the problems?

What are the goals of the system?

How will the system be used daily?

What are the constraints on the system?

How is the work is done now (frequency, transaction volume,…)?

Any performance considerations?

Workshops

- Stay in scope
- Time box discussions
- Use parking lots
- Fill all the team roles
- Plan an agenda
- Keep everyone engaged
- Small team, right stakeholders
- Establish & enforce ground rules

…facilitated sessions with multiple stakeholders & formal roles

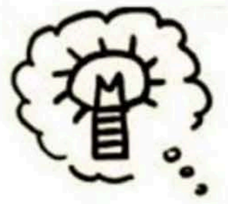
…include several types of stakeholders (*e.g.* users, developers, testers)

…resource intensive: must be planned well! Rarely useful to start on a clean slate… **use other methods before workshops**

Workshops: Team roles

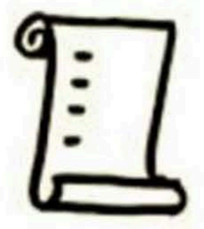
- Stay in scope
- Time box discussions
- Use parking lots
- Fill all the team roles
- Plan an agenda
- Keep everyone engaged
- Small team, right stakeholders
- Establish & enforce ground rules

FACILITATOR



AGITATOR

TIMEKEEPER



NOTE TAKER

OBSERVER

FOCUS MAKER

Focus Groups

…are a **representative group of users** who meet in a **facilitated elicitation activity** to **generate input and ideas** on a product's **functional and quality requirements**

…are useful for **exploring users' attitudes**, **impressions**, **preferences**, and **needs**

…are particularly valuable if you **don't have access to end users** within your company

https://flatworldknowledge.lardbucket.org/books/sociological-inquiry-principles-qualitative-and-quantitative-methods/s15-01-focus-groups.html

Observation

… is based on the saying:
harder to describe, easier to do…

Tasks are habitual

Beware: observations are time consuming!

Great for important or high-risk tasks

Can be **silent** or **interactive**

Questionnaires

…used to **survey large groups of users** to understand their needs

Pros:

- Inexpensive
- Easily administered across geographical boundaries
- Analyzed results can be input to other elicitation techniques

Preparing well-written questions is the biggest challenge with questionnaires!

System Interface Analysis

How will your product fit into the rest of the system?

This activity reveals **functional requirements regarding the exchange of data and services between systems**

Identify functionality in the other system that might lead to requirements for your system:

- What data **to pass to** the other system
- What data **is received** from it
- What are rules governing that data?

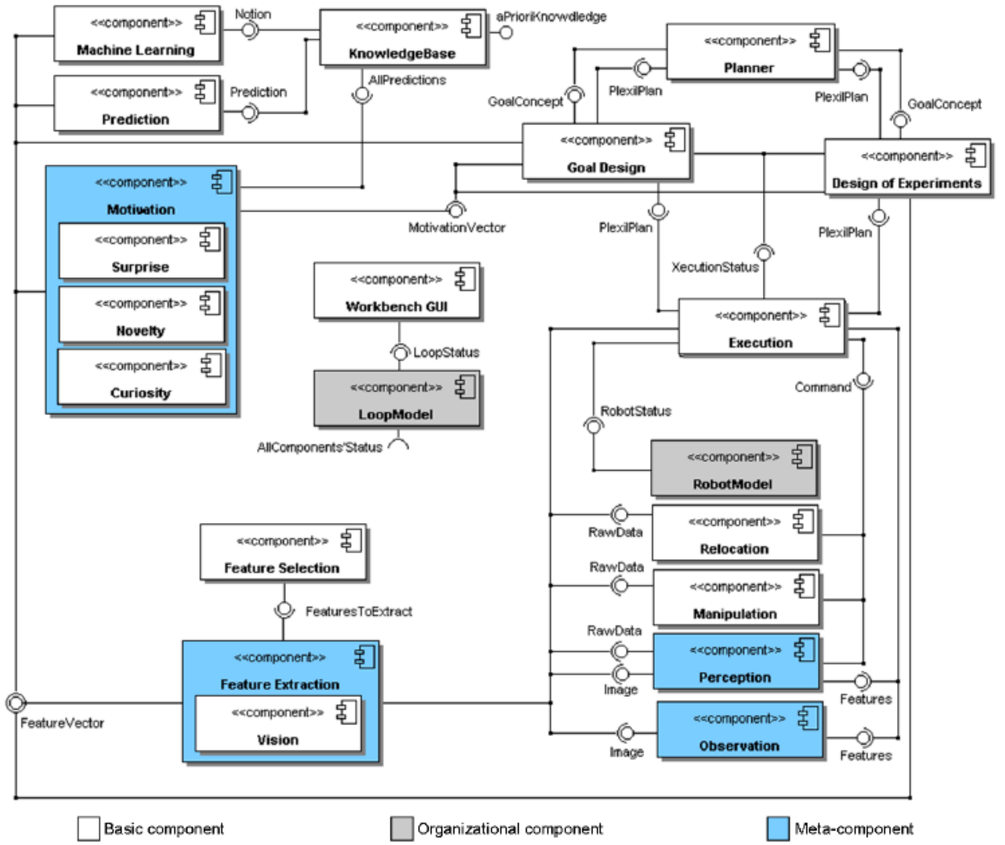


Figure 4.10: A component diagram showing the data flow between XPERSIF components.

System Interface Analysis

- Study (direct interaction) existing systems: extract user & functional requirements
- If necessary, use screen shots
- No current system? Look at similar products
- Draft use cases to review with users

Do **not** assume:

Certain functionality is needed in the new system

Flow must be the same in your system

Document Analysis

…is the process of **examining existing documentation** for **potential software requirements**.

Comparative reviews give you a competitive advantage!

Can reveal info people don't tell you!

Drawbacks: **out-of-date** or **poorly-written documentation** (*e.g.* not very thorough)

Documents:

- can describe corporate
- industry standards
- regulations with which the product must comply
- SRS
- User Manuals
- Development Documents
- …

Software quality begins with the quality of the requirements.

Pearl Zhu