Requirement Engineering

TELECOM Nancy 2024-2025

Sources

- Software Engineering 10 - Ian Sommerville - Pearson
- Applying UML and Patterns - Craig Larman
- Software Requirements (Third Edition) - Wiegers and Beatty
- Requirement Engineering – Dick, Hull and Jackson 2017 – Springer
- Software Engineering A Practitioner's Approach (Ninth Edition) – Pressman - Maxim

Topic covered

- Requirement Analysis
- Requirement Analysis Approaches
- Requirement Elicitation techniques
- Requirement Specification
- Requirement Verification and Validation
- Requirement Management
- Modeling techniques
- Requirement Tools
- Trends

What is a requirement ?

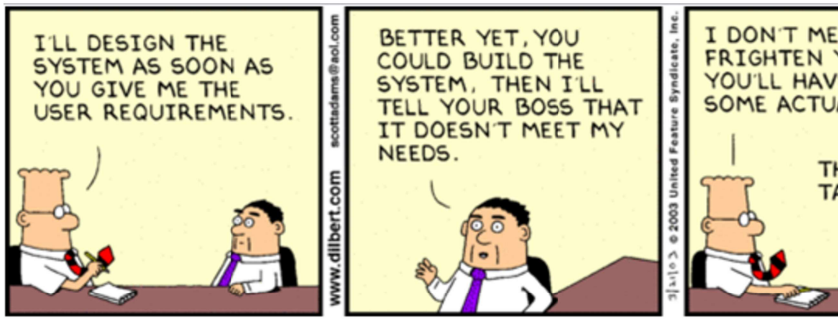
- The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:[1]
  - A condition or capability needed by a user to solve a problem or achieve a objective.
  - A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
  - A documented representation of a condition or capability as in 1 or 2.

    IEEE Computer Society (1990). "IEEE Standard Glossary of Software Engineering Terminology". *IEEE Standard*.

- What the system should do and the constraints of its operations

Requirements in practice

- A user describes his/her needs, requirements specification to in a specification document
- The software developer take the specification document and produce a piece of software thet meet these needs perfectly
- What could go wrong ?



Use case for the course :

We want to develop a shopping platform to allow local pro
to sell their products to consumers (aka thefoodassembly)

Producers are registered to an assembly. Every week they
propose their product with a quantity and a price

Consumers can list the products and decide what they nee

At some predefined day, producers deliver their product to
assembly and consumer can gather to collect their produc

An assembly manager is in charge to organize the distribut

The platform manager operate and maintain the system

Requirement

Description of the required functionnality

- A client can add a product to his shopping basket

Details of realisation

- The product is added to the basket
- For some product, the client can select a quantity. The quantity is limited by the available stock
- The total cost of the basket is updated
- The product stock is updated provisionally

Requirements may have different priorities

- High, medium, low
- Shall vs should

Requirement v2

Description of the required functionnality

- The system shall allows the client to select a product and add it to the basket

Details of realisation

- The product is added to the **basket**
- For some product, the client can select a quantity. The available stock limits the quantity
- The total cost of the basket is updated
- The product stock is updated provisionally

Requirements may have different priorities

- High, medium, low
- Shall vs should

Properties of a requirement

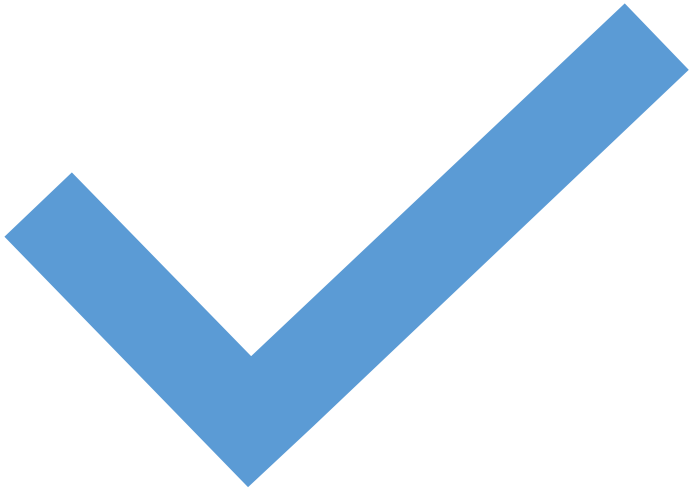
- Unambiguous
- Verifiable
- Clear
- Correct
- Understandable
- Feasible
- Independent
- Atomic
- Necessary
- Implementation-free
- Consistent
- Non Redundant
- Complete

https://www.informit.com/articles/article.aspx?p=1152528&seqNum=4

Unambiguous

- Only one way to interpret the requirement

- *A client shall be able to change the command at any time*
  - What means « change the command »
  - What means « at any time »

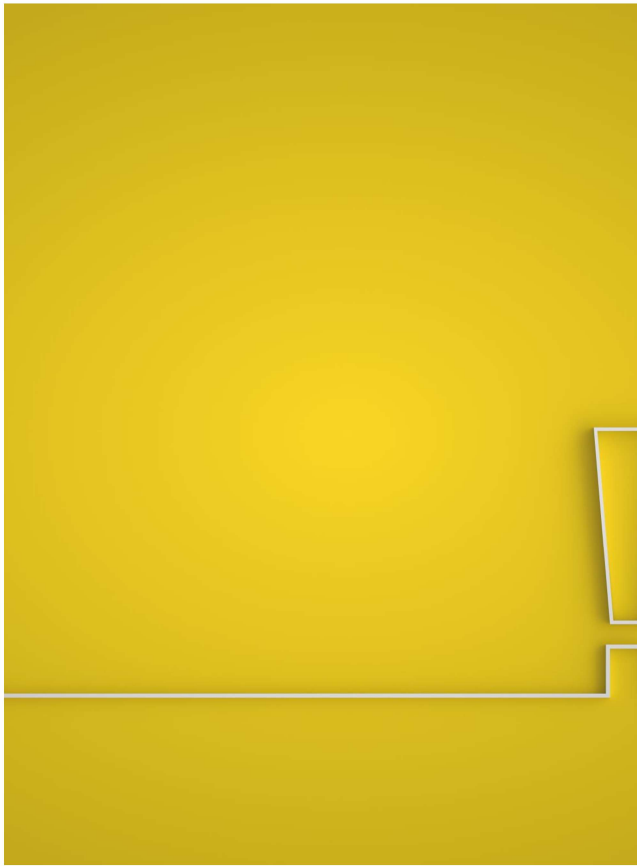


Verifiable

- It should be possible to test that a requirement is implemented correctly
- Be careful to word that are difficult to tes efficient, simple, user-friendly, flexible, quickly
- The system should be easy to use

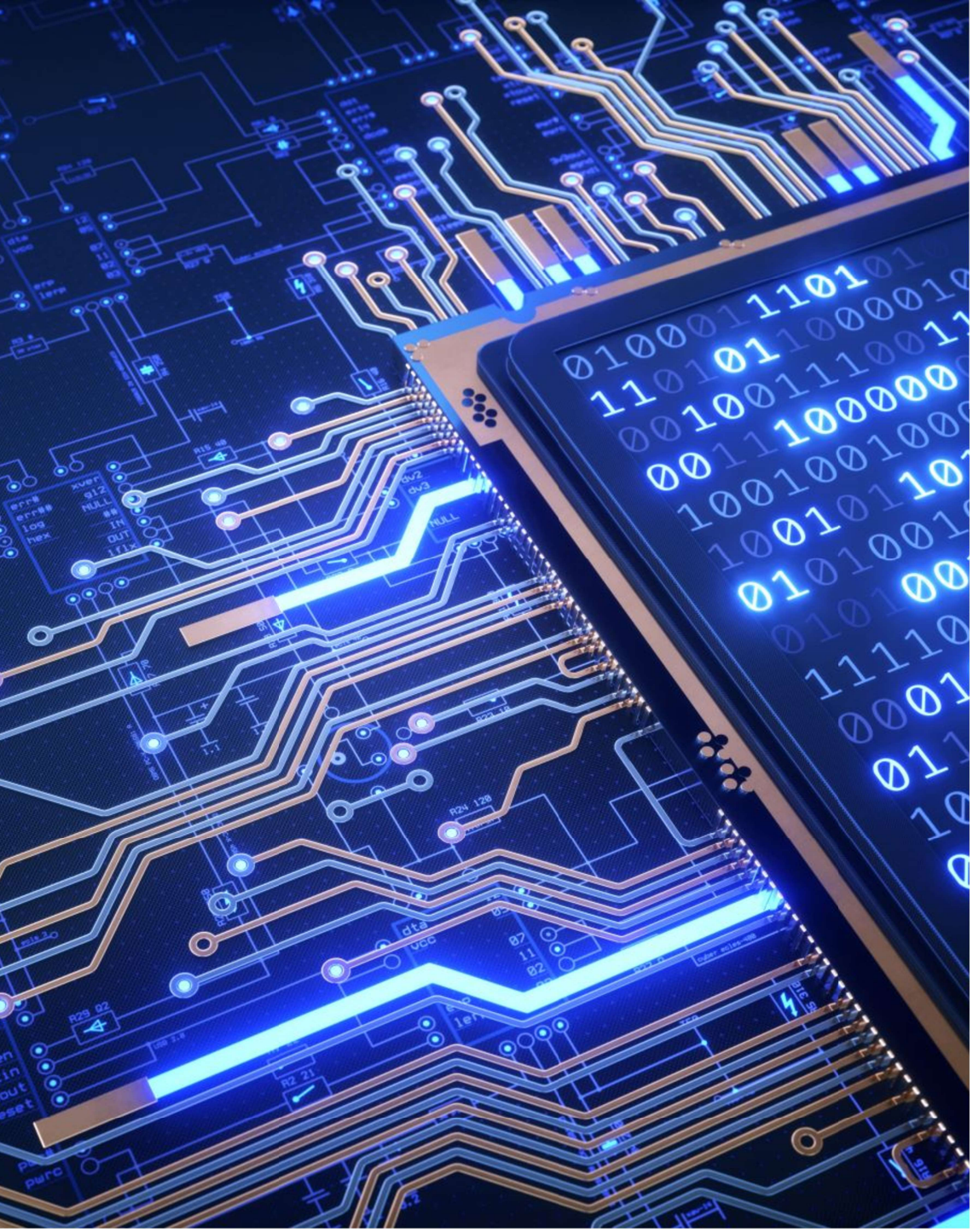
Understandable

- Requirements must be grammatically correct and written in a consistent style
  - *The system must allow users cancel their command*
  - *User must allow to chose among products*



Atomic

- The requirement should contain a single trace element: It will be difficult to trace a requiren that includes several requirements
  - *The producer shall be able to manage its products list*
  - *Manage = add, update, delete ?*

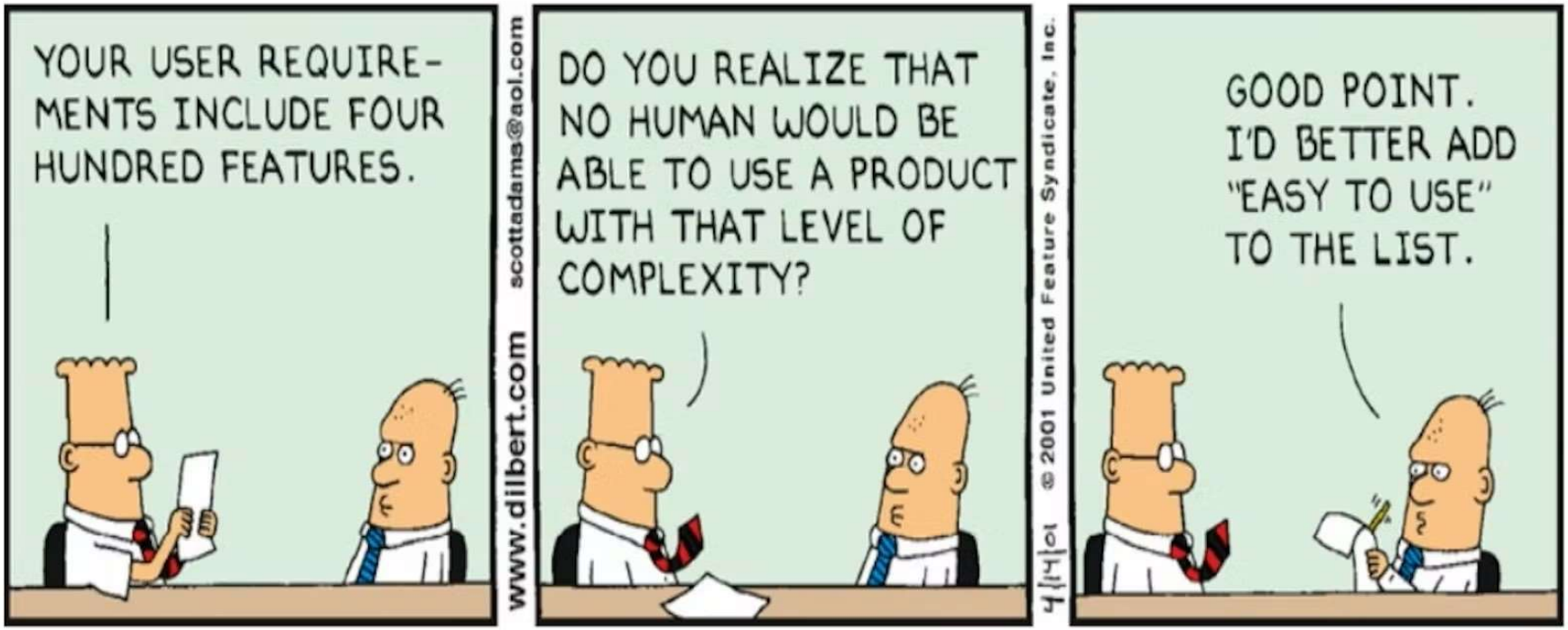


Implementation-Free

- Requirements should not contain unnece
  design and implementation information:
  - *The system must use CSV files to sto*
    assembly informations

Non-Redundant

- Each requirement should be expressed or
  once and should not overlap with anothe
  requirement:
    - The producer shall be able to chang
      the price of products
    - The producer shall be able to modif
      products and change their prices


YOUR USER REQUIREMENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.
www.dilbert.com scottadams3@aol.com
© 2001 United Feature Syndicate, Inc.
4/14/01

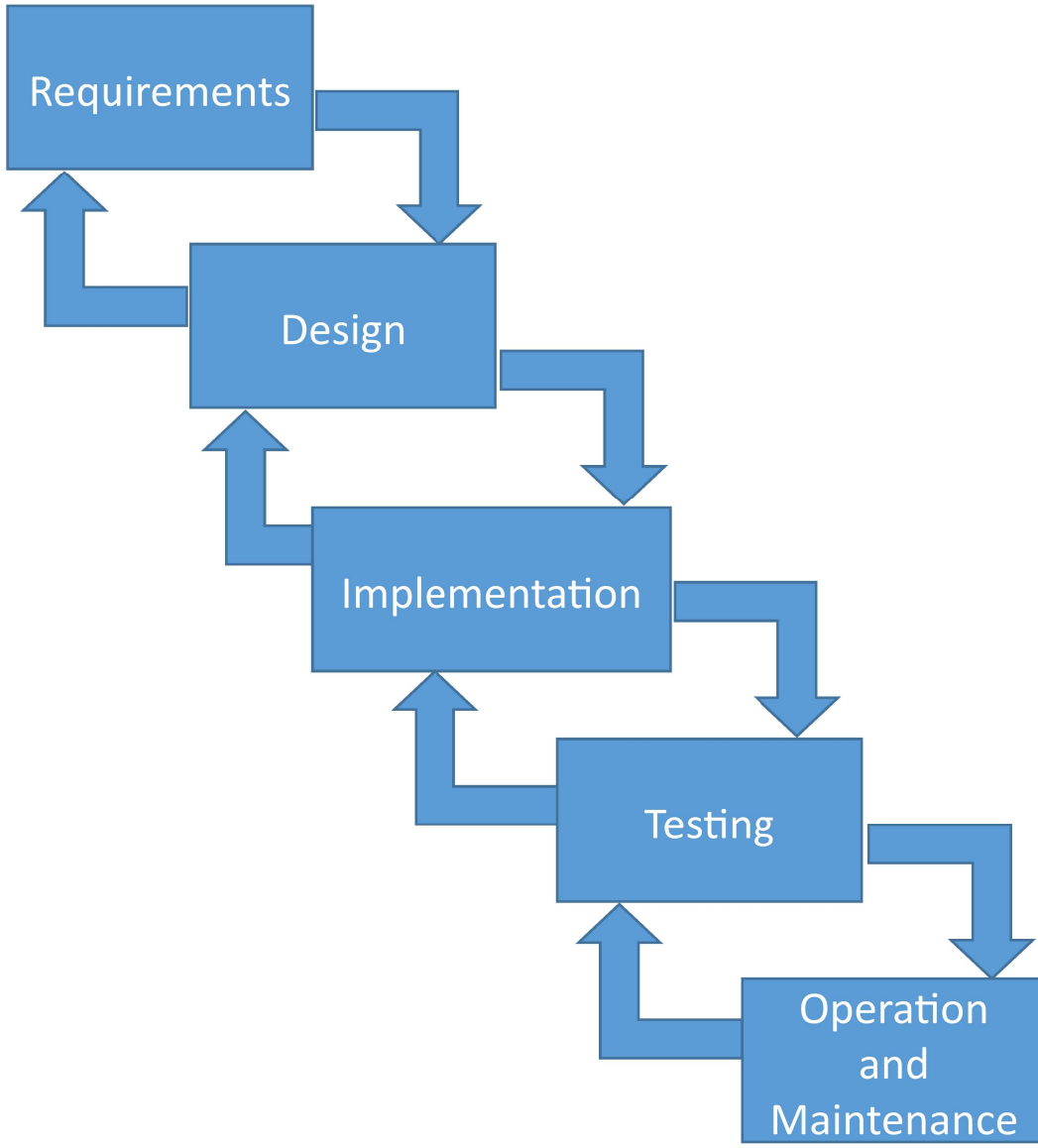
Requirement Analysis Approaches

Waterfall Model

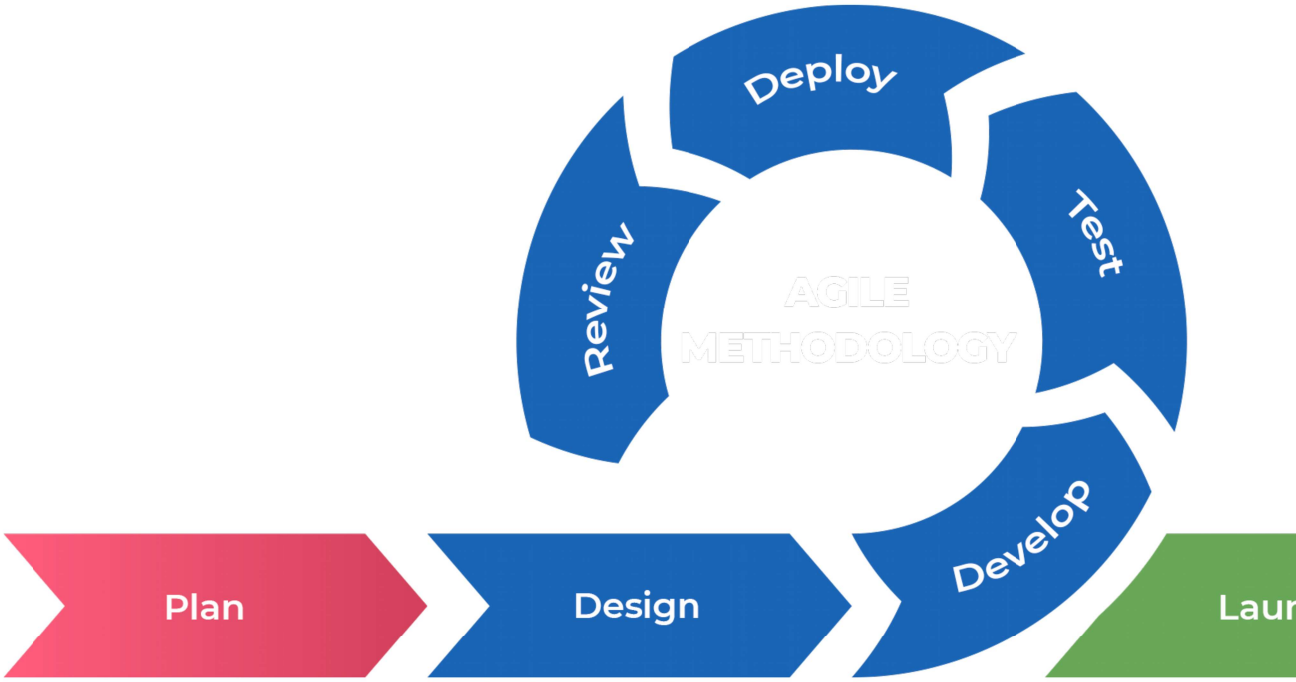
Agile Approaches and Iterative methods

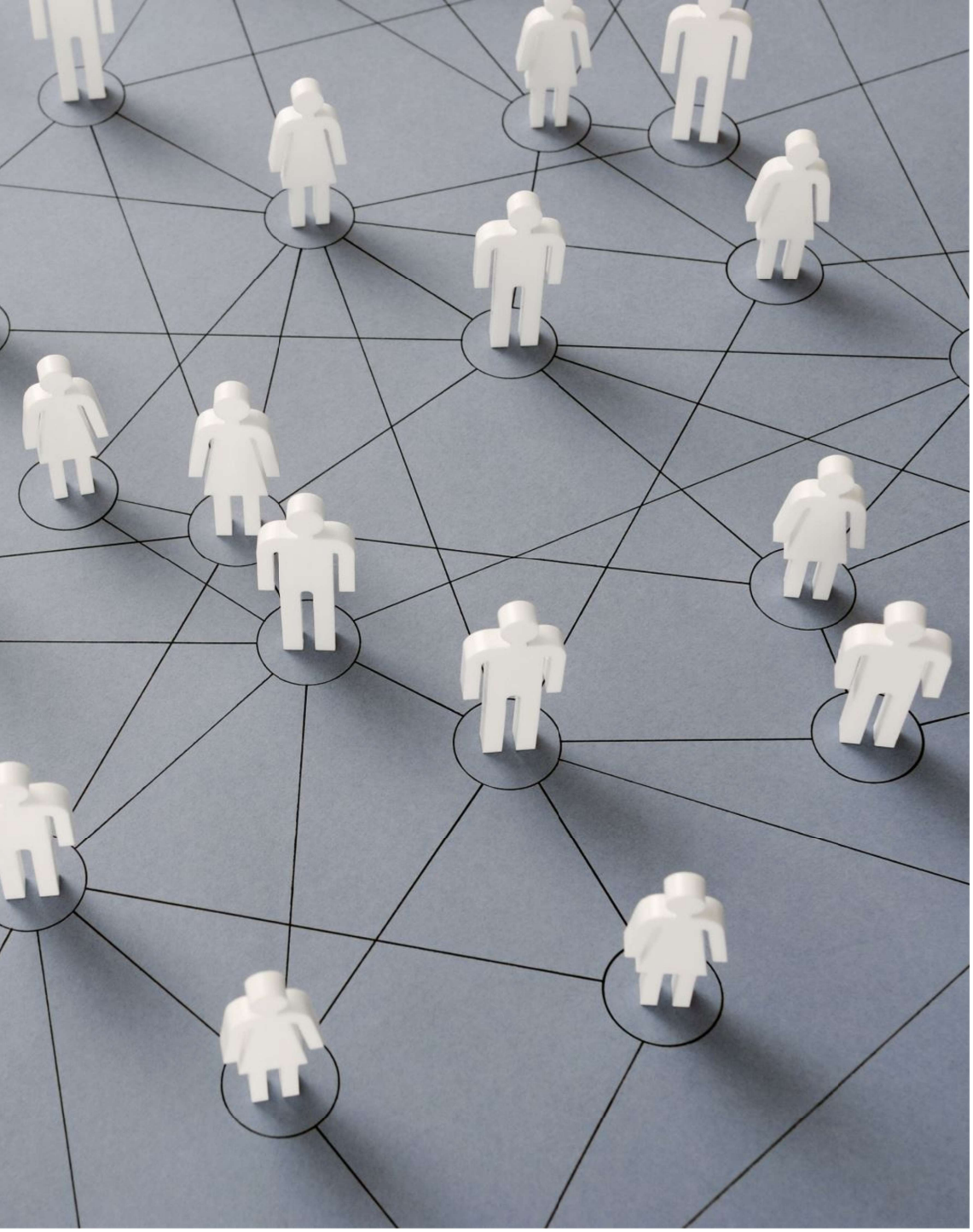
Hybrid Approaches and tailored methods

Waterfall Model



Agile Approaches and Iterative Methods

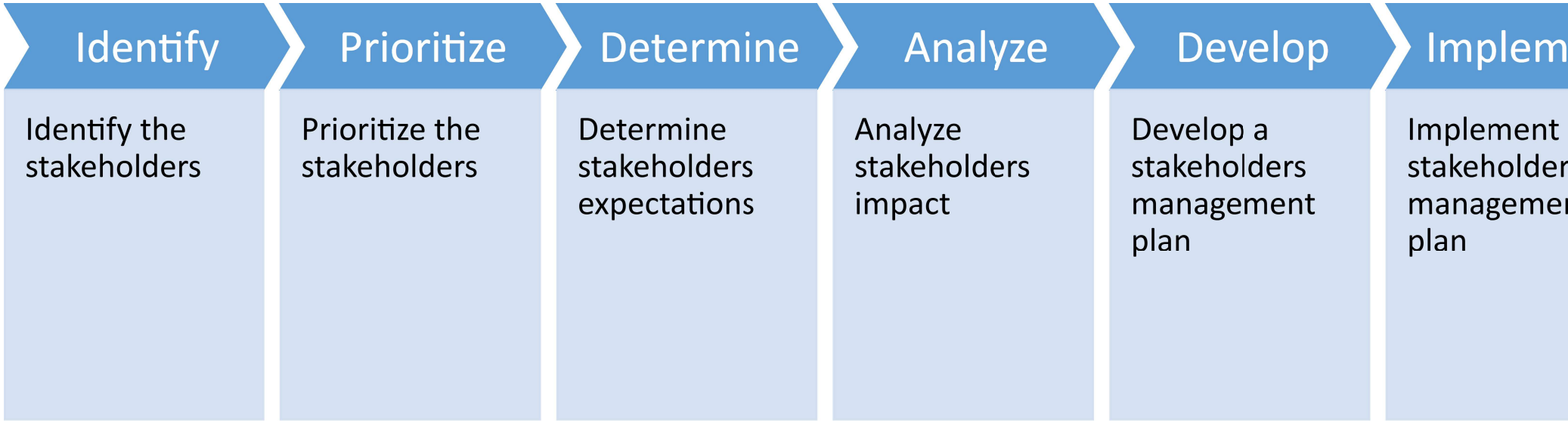




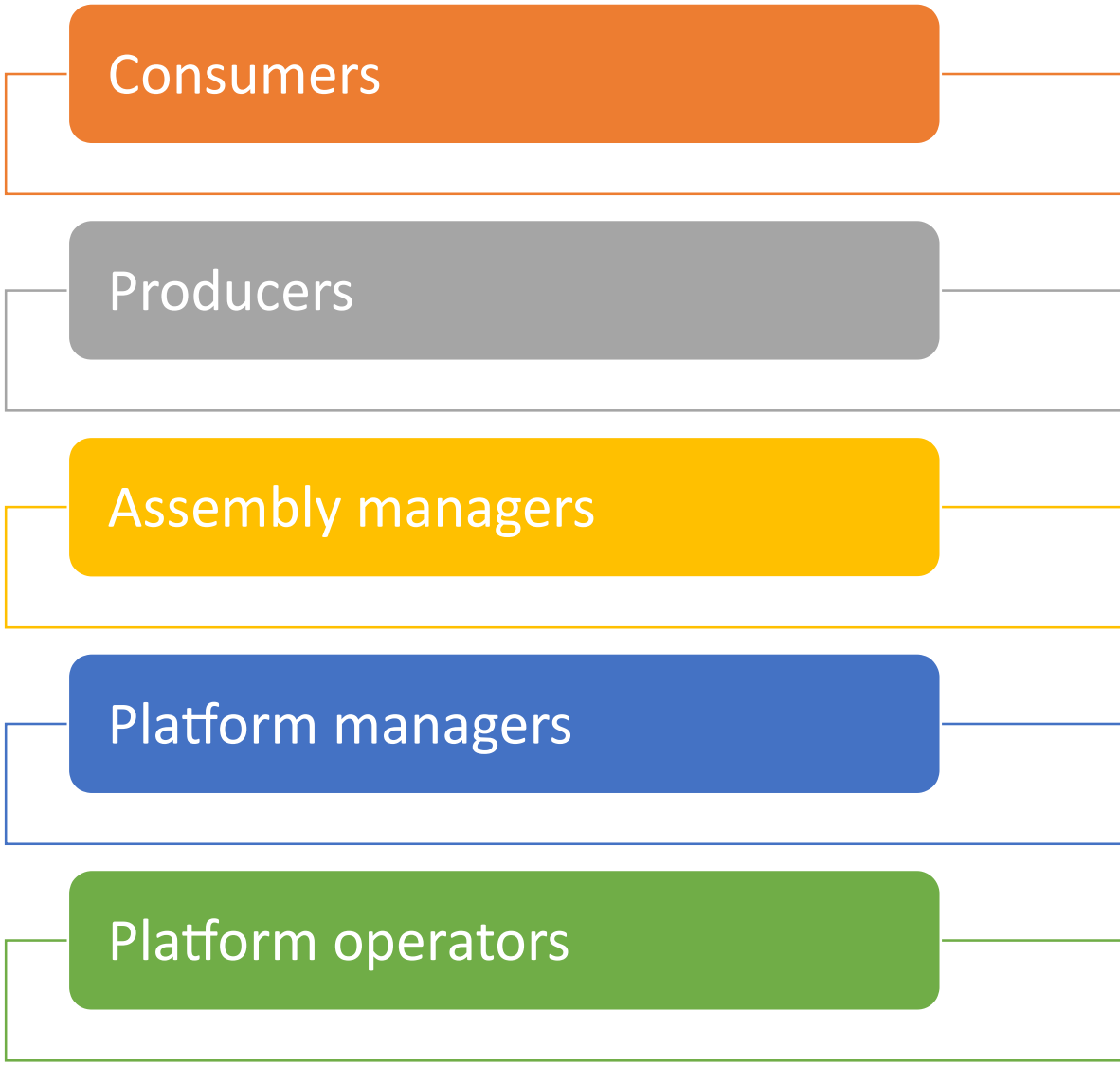
Stakeholders Analysis

- Any person or organization who is affecte the system in some way and so who has a legitimate interest, direct or indirect
- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Stakeholder analysis



Who are the stackholders for our example ?



Functional vs non-functional requirements

- Functional Requirements
- Non-Functional Requirements
- Domain requirements

Functional and non-functional requirements

Functional requirements

- Statements of services the system should provide, how system should react to particular inputs and how the system should behave in particular situations.
- May state what the system should not do.

Non-functional requirements

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Often apply to the system as a whole rather than individual features or services.

Domain requirements

- Constraints on the system from the domain of operatio

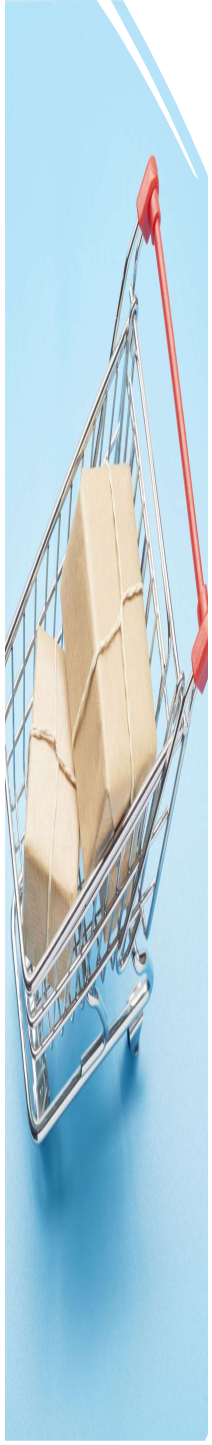
Functional requirements

Describe functionality or system services.

Depend on the type of software, expected use and the type of system where the software is u

Functional user requirements may be high-level statements of what the system should do.

Functional system requirements should describe the system ser in detail.



Example of functior requirements

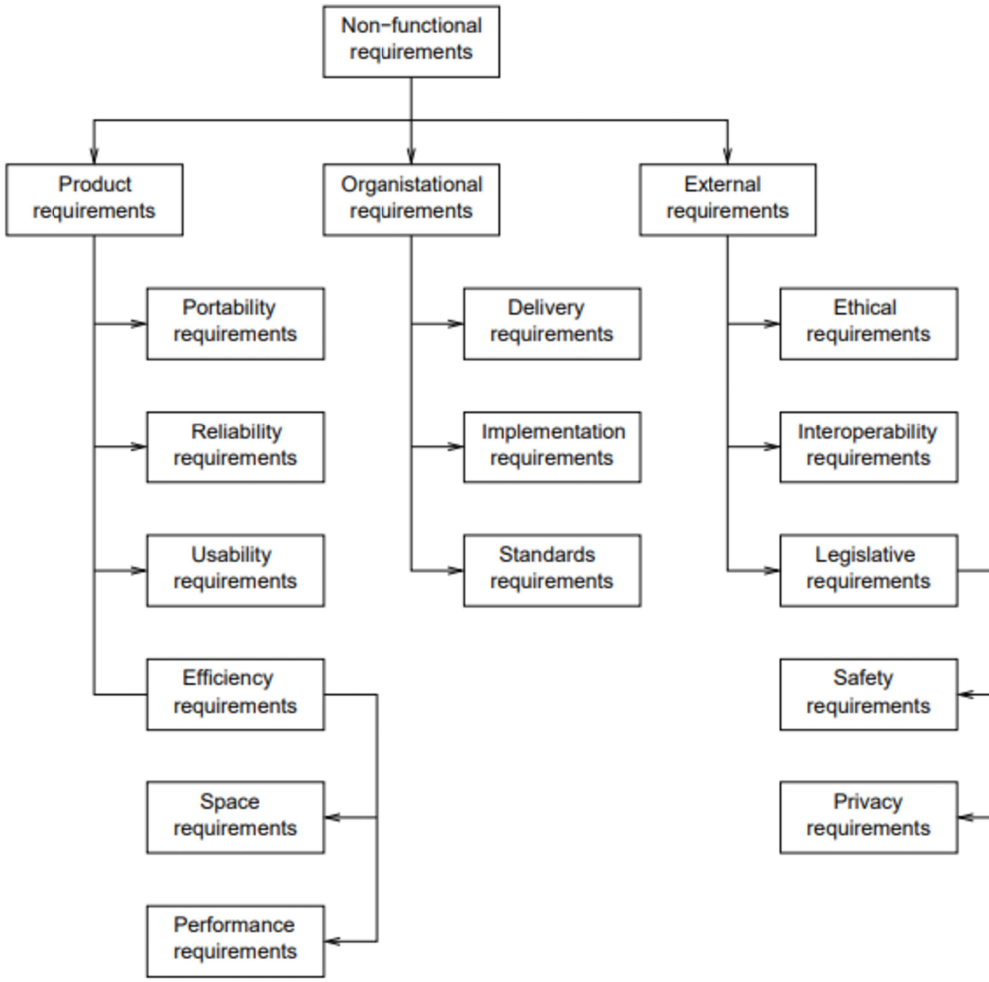
- The consumer shall be able to remove a product from its b
  - The product stock is updated
  - The cost of the basket is updated

- The producer shall be able to update the remaining stock product during a selling session
  - The stock is updated
  - If too many consumers have already ordered the pro the most recent orders are updated until the stock re a 0.
  - Consumers are notified

Non-functional requirements

- These define system properties and constraints e.g. reliability, response time a storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specifi mandating a particular IDE, programming language or development method.
- Non-functional requirements may be mor critical than functional requirements. If th are not met, the system may be useless.

Types of nonfunctional requirement



Non-functional requirements implementation

- Non-Functional Requirements Impact Sys Architecture
  - Non-functional requirements can affect the overall architecture of a system, rather than the individual components.
  - Performance requirements may require organizing the system to minimize communications between components.
  - A single non-functional requirement, such as security, may generate related functional requirements that define required system services.
  - Non-functional requirements may also gener restrictions on existing requirements.

Non-functional classifications

- Product requirements
  - Requirements which specify that the delivere product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its developme process e.g. interoperability requirements, legislative requirements, etc.

Goals and requirements

- Non-functional requirements may be challenging to state precisely, and imprec conditions may be difficult to verify.
- Goal
  - A general intention of the user, such as ease use.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users

Usability requirements

- Usability requirements are essential to ensure a system is easy to use and meets user needs.
- These requirements describe the user interface, interactions, and overall user experience.
- Usability requirements should be specific, measurable, and testable.
- They should address factors such as user satisfaction, learnability, and efficiency.
- Usability requirements may also cover accessibility for users with disabilities.
- Effective usability requirements can improve user adoption, productivity, and overall satisfact with a system.

The system shall meet WCAG 2.1 criteria level A

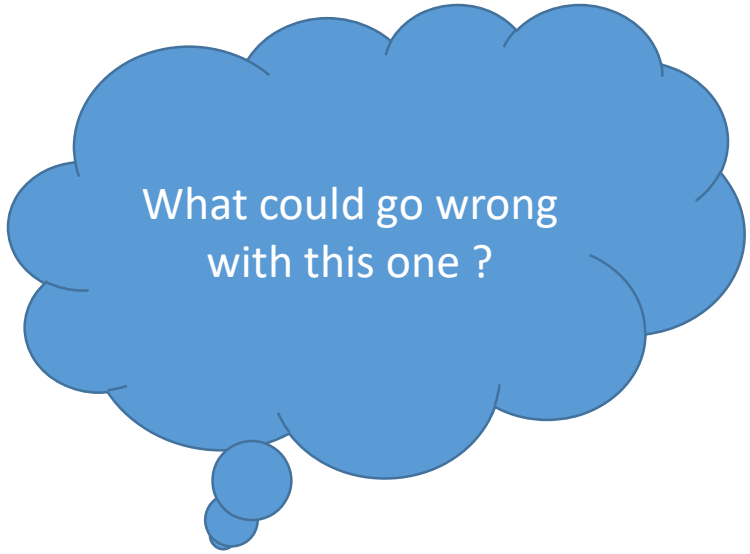
WCAG (Web Content Accessibility Guidelines) is the standard for accessibility criteria. Each guideline is testable.

Level A is the minimal level for accessibility. There are three levels A, AA and AAA

Performance requirement

- Performance requirements must be testable

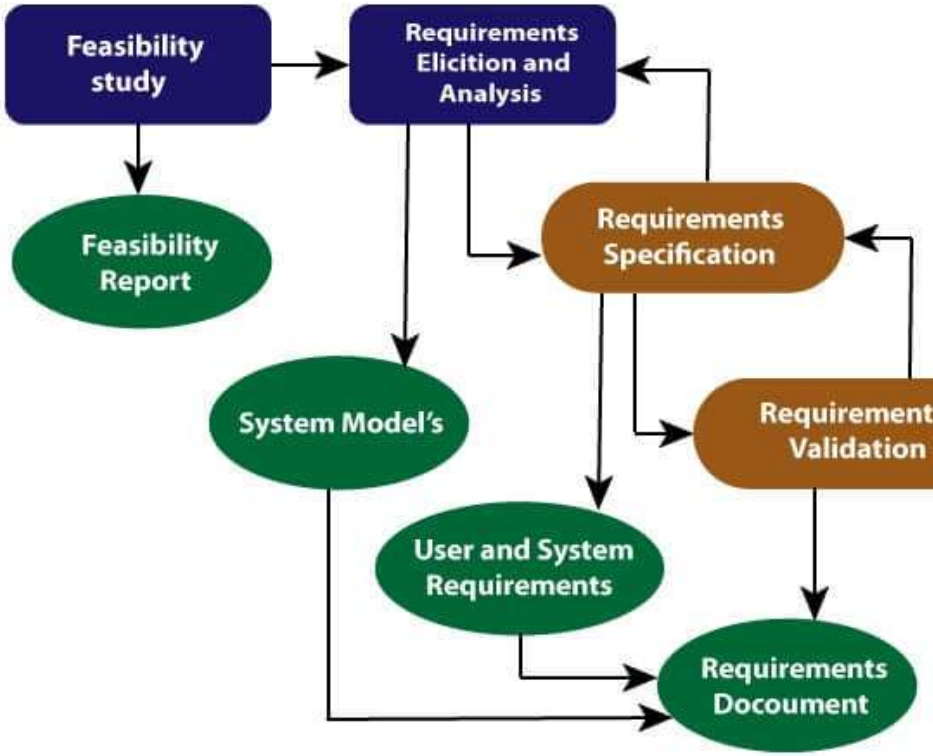
> Response Time: The website should load and display the requested page within 3 seconds or less, on average, for a typical user on a broadband connection.



Requirement Engineering process

- Process of defining, documenting and Maintaining requirements

- Generic activities
  - Requirements elicitation;
  - Requirements analysis;
  - Requirement Specification
  - Requirements validation;
  - Requirements management



Requirement Engineering Process

https://www.javatpoint.com/software-engineering-requirement-engineering

Requirements Elicitation: Finding What Users Need

Also known as requirements discovery, this process involves technical staff working with customers to ide system needs.

The focus is on understanding the application domain required services, and operational constraints.

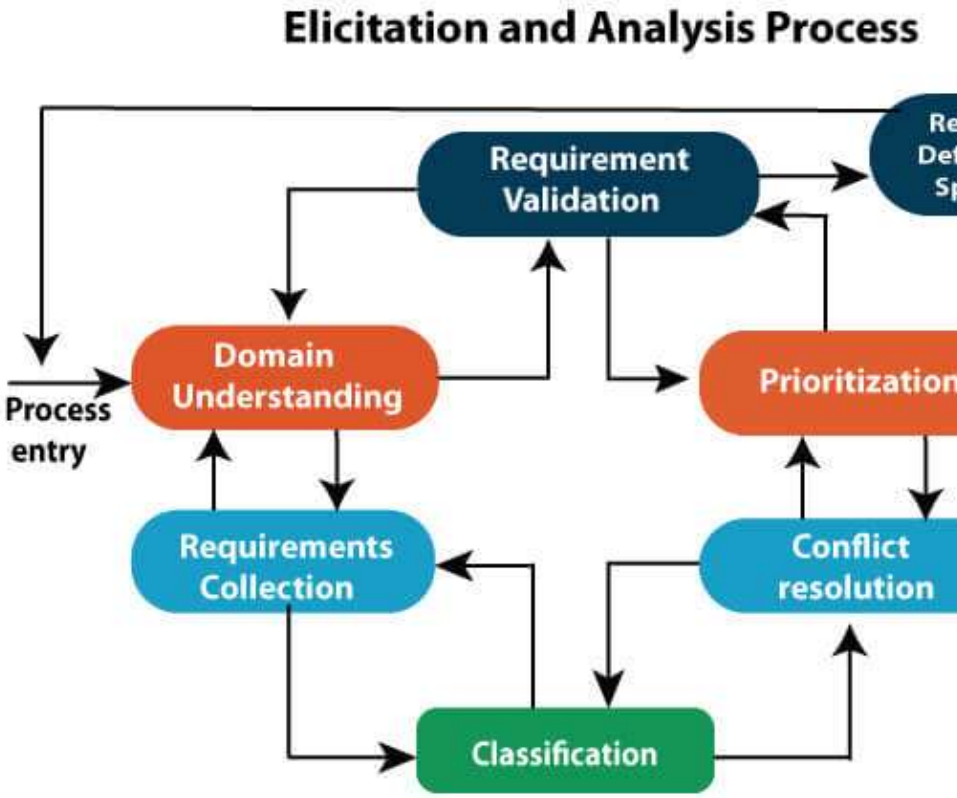
Stakeholders, including end-users, managers, engine domain experts, and trade unions, may be involved.

The goal is to ensure all stakeholder needs are consid and addressed in the system design.

Effective requirements elicitation can lead to a systen meets user needs, reduces development costs, and improves overall satisfaction.

Requirements elicitation

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.



Problems with Requirements Elicitati

- **Ambiguity and uncertainty**: Customers may have difficulty expressing their needs clearly or may not be aware of all requirements.
- **Conflicting requirements**: Stakeholders may have different needs or priorities that cannot be reconciled.
- **Scope creep**: As requirements are discovered, additional features or functions may be added to the project scope, leading to delays and increased costs.
- **Changing requirements**: Stakeholders may change their needs as they learn more about the system or the environment in which it operates.
- **Resistance to change**: Stakeholders may resist changes to existing systems or processes, making it difficult to identify and implement new requirements.
- **Cultural and language barriers**: Stakeholders from different backgrounds may have different ways of expressing themselves or may interpret requirements differently.
- **Time and resource constraints**: Eliciting requirements can be time-consuming and require significant resources, making it difficult to balance competing priorities.

Process activities

- Requirements discovery
  - Interacting with stakeholders to discov
    their requirements. Domain requireme
    are also discovered at this stage.
- Requirements classification and organisatio
  - Groups related requirements and orga
    them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving
    requirements conflicts.
- Requirements specification
  - Requirements are documented and inp
    into the next round of the spiral.

Requirements discove

- The process of gathering information to identify us system requirements.
- Involves interacting with system stakeholders, from managers to external regulators.
- Systems typically have a range of stakeholders, incl users, customers, developers, and operations staff.
- The goal is to distil the essential requirements from existing systems and user needs.

Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes
- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to lis to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

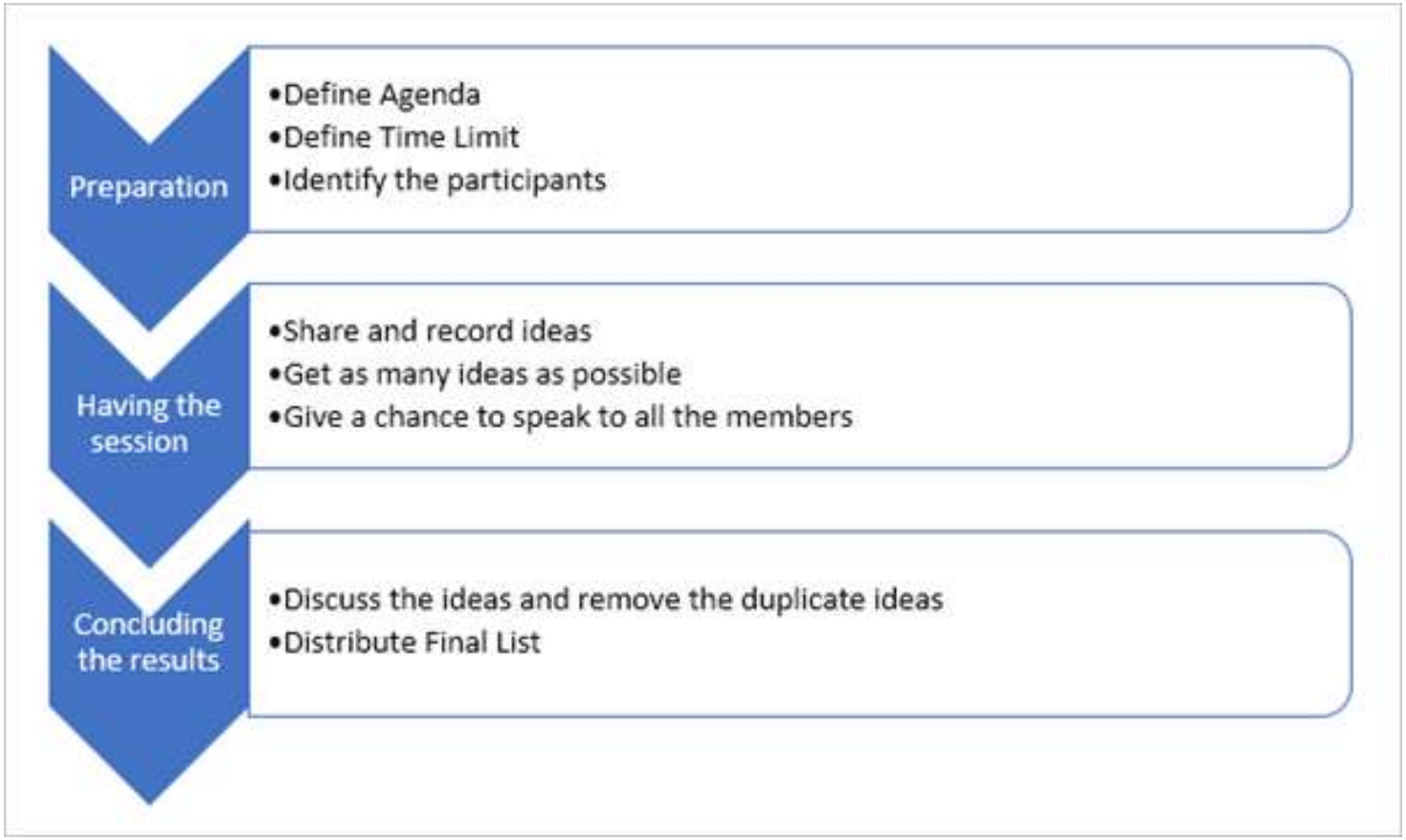
Lots of resources

https://www.axia-consulting.co.uk/requirements-gathering.htm

https://www.bridging-the-gap.com/what-questions-do-i-ask-during-requirements-elicitati

Brainstorming

- Generate ideas and find solutions



https://www.softwaretestinghelp.com/requirements-elicitation-techniques/

Document Analysis and Reviews

- Gather information by reviewing , examining materials that describe the business environment
  - Contracts
  - Process Cartography
  - Business Plans
  - Marketing Reports
  - Regulatory Compliance Documents
  - Competitor Analysis Reports

Ethnography

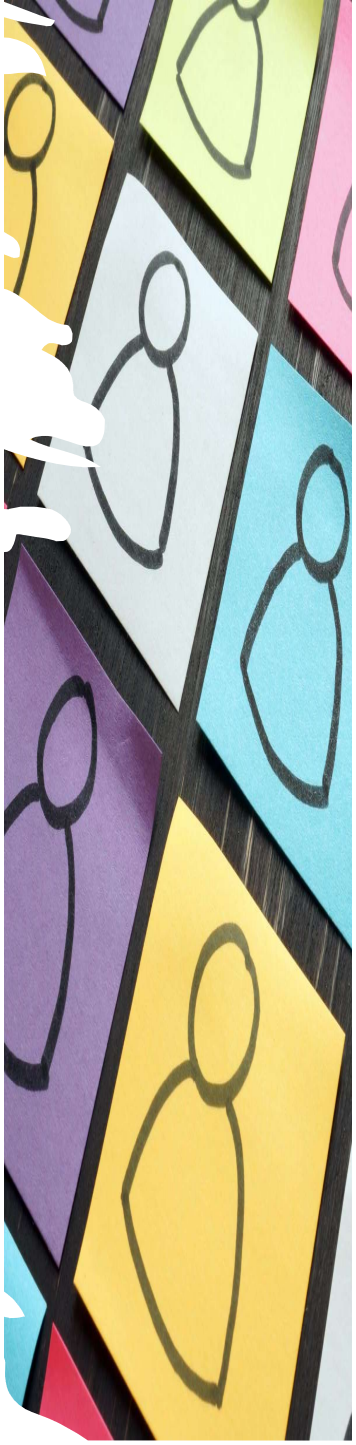
- A social scientist spends a considerable time obser
  and analysing how people actually work.
- People do not have to explain or articulate their wo
- Social and organisational factors of importance ma
  observed.
- Ethnographic studies have shown that work is usua
  richer and more complex than suggested by simple
  system models.

Scope of ethnography

- Requirements that are derived from the way that people actually work rather than the way I which process definitions suggest that they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.
    - Awareness of what other people are doing leads to changes in the ways in which we do things.
- Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Stories and scenarios

- Scenarios and user stories are real-life examples of how a system can be used.
- Stories and scenarios are a description of how a system may be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.



Examples

- Stories can lead to the development of features

1.Browse Products

1. As a customer, I want to browse all availabl products by category so that I can find item I'm interested in purchasing.

2.Search Functionality

1. As a customer, I want to use a search bar t find specific products so that I can quickly add them to my cart.

3.Place Orders

1. As a customer, I want to select the quantity of a product and add it to my cart so that I can purchase multiple items in one transaction.

4.Order Summary and Checkout

1. As a customer, I want to review my order summary and checkout securely so that I can confirm my purchase details are correc before payment.

5.Order History

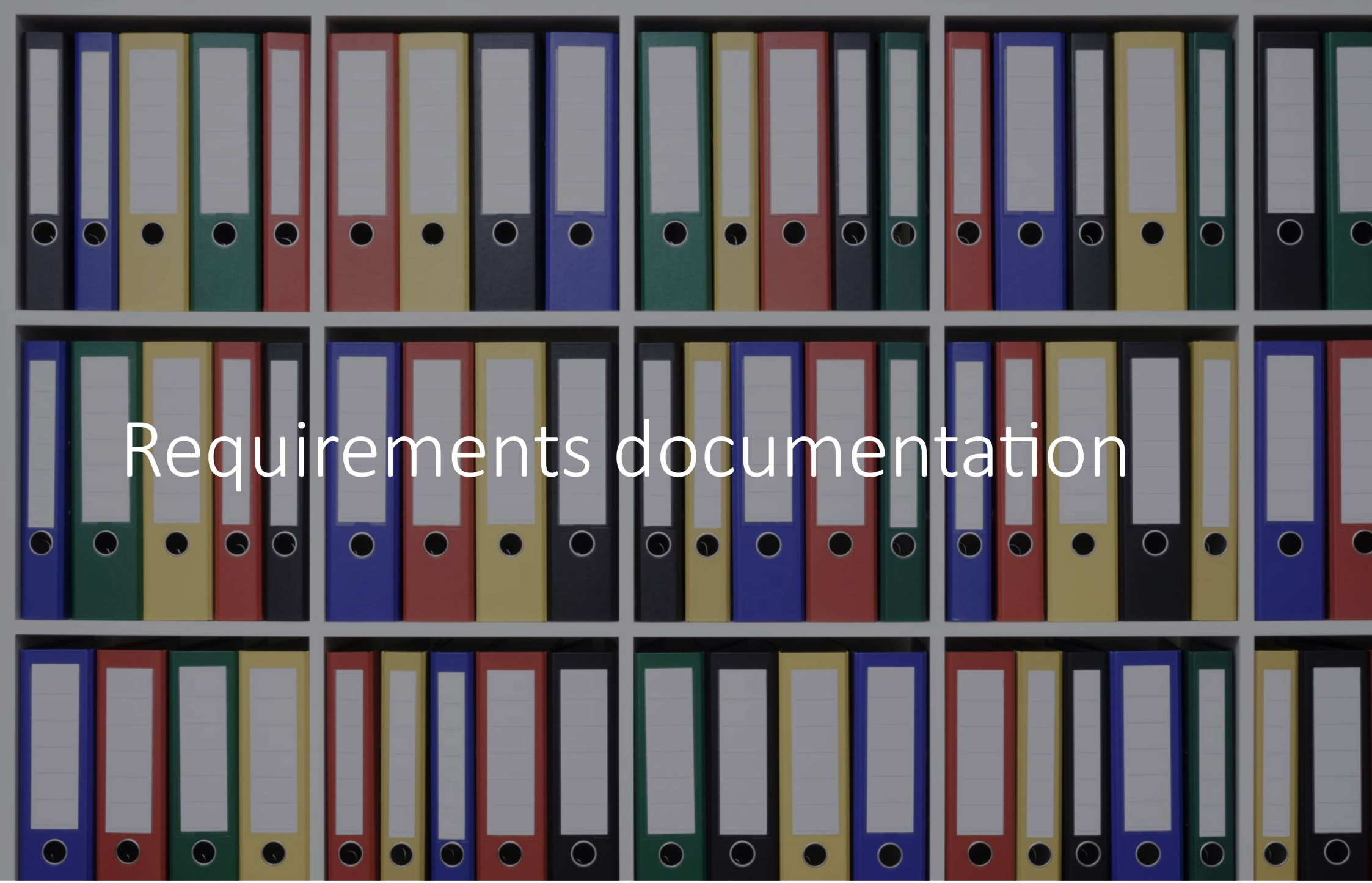
1. As a customer, I want to view my past orders so that I can reorder the same products easily.

6.Pickup Time Selection

1. As a customer, I want to select a pickup tim

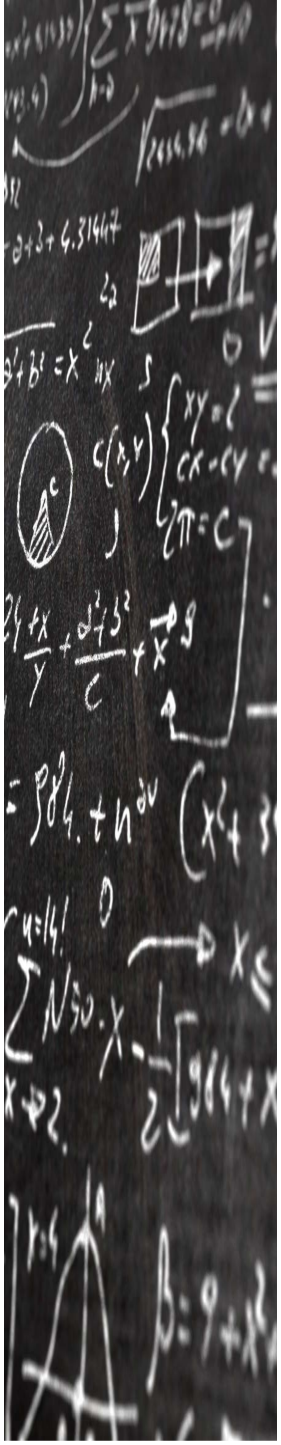
Features from stories

- **1. Browse Products by Category**
- **Feature: Product Categorization and Filtering**
  - Implement categories for all products (e.g., fruits, vegetables, dairy).
  - Allow users to filter products by categories.
  - Provide a clear and intuitive UI where categories are easily navigable.
- **2. Use Search Bar to Find Products**
- **Feature: Advanced Search Functionality**
  - Implement a search bar that supports keyword searches across produc names and descriptions.
  - Include auto-suggestion capabilities to help users find products quickly they type.
  - Allow users to filter search results by various parameters such as price category, and rating.


Requirements documentation

Requirements specification

- The process of writing down requirements in a requirements document.
- Requirements have to be understandable by end-u and customers who do not have a technical backgr and by developers that need to build the system.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as comp as possible.



Ways of writing a system requirements specificatio

- Natural Language: Common and flexible, but can be ambiguous
- Use Cases: Describes system use in specific scenarios, c but time-consuming
- User Stories: Brief descriptions from users' perspectives useful for Agile development
- Formal Methods: Precision and guarantees, but require specialised knowledge and tools
- Prototypes: Working models for feedback and refineme helpful for undefined or changing requirements

Guidelines for writing requirements

- Invent a standard format and use it for all requirem
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.
- Include an explanation (rationale) of why a requirer is necessary.

Form-based specifications

- Definition of the function or entity.
- Description of inputs and where they come from.
- Description of outputs and where they go to.
- Information about the information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current me in the safe zone between 3 and 7 units
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered
Destination	Main control loop
Action	CompDose is zero if the sugar level is stable or falling or if the level rate of increase is decreasing. If the level is increasing and the increasing, then CompDose is computed by dividing the difference sugar level and the previous level by 4 and rounding the result. If t to zero then CompDose is set to the minimum dose that can be deli
Requires	Two previous readings so that the rate of change of sugar level can
Pre-condition	The insulin reservoir contains at least the maximum allowed single
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side-effects	None

Figure from Soft. Eng 10

Formal Specification

- A method of writing a specification document using mathematical notation and formal logic
- Used in safety-critical systems where precision and correctness are critical
- Provides a way to reason about system behaviour and prove properties
- Can be checked mechanically for consistency and completeness

- Example:
- Z notation
- Developed by Jean-Raymond Al in the 1970s for formal specification of software system
- Uses mathematical notation to describe data types, operations and constraints
- Specifies system behaviour in terms of pre- and post-conditio invariants, and operations

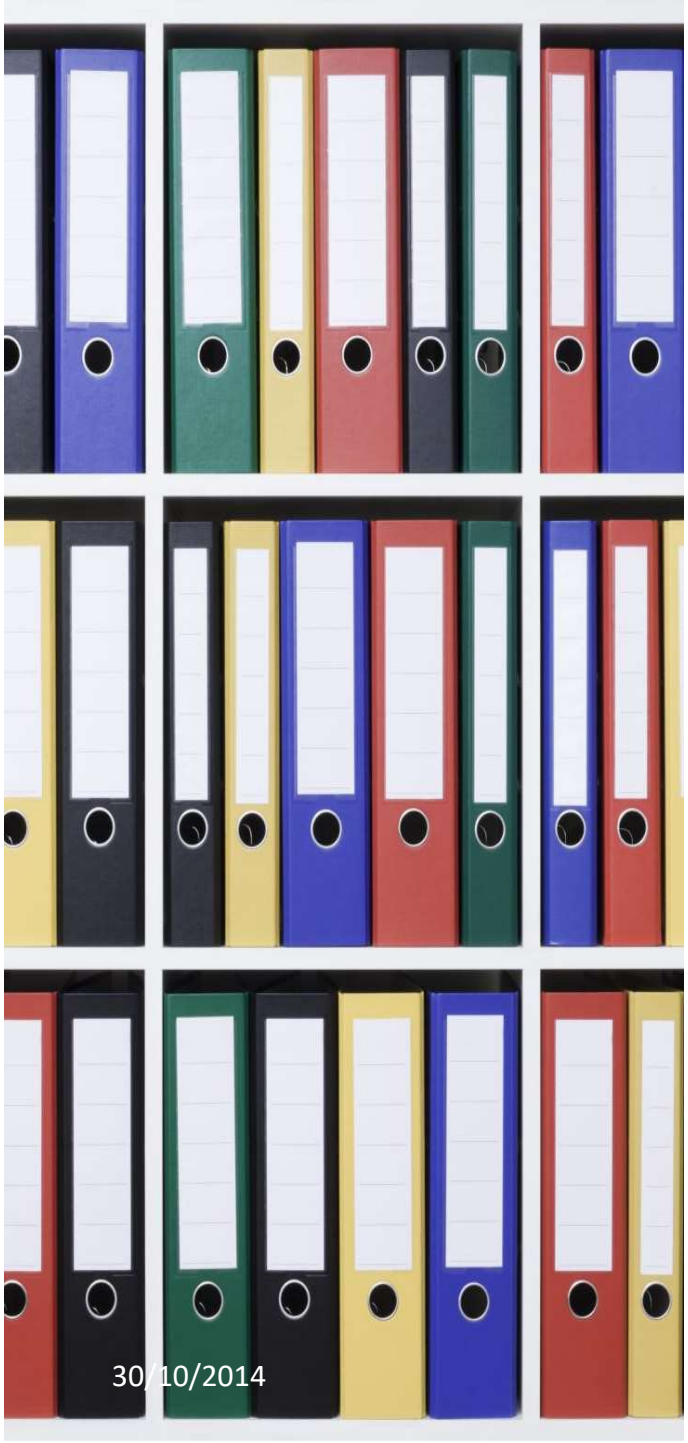
Examples of requirements in Z

- The system must check if the desired quantity of a product is available before allowing a customer to add to their cart.

```
∀ prod : Product; qty : ℕ | prod ∈ Products ∧ qty
≤ prod.quantityAvailable:
CanAddToCart(prod, qty) ⇔ True
```

- The checkout process must ensure that the total amount to be paid matches the sum of prices of all items in the cart times their quantities.

```
checkoutValid(cart: Cart) ⇔
∑ {p.price * c.quantity | p ∈ cart.products ∧ c ∈ cart.contents ∧
p.id = c.productId} = cart.totalAmount
```



The software requirements docume

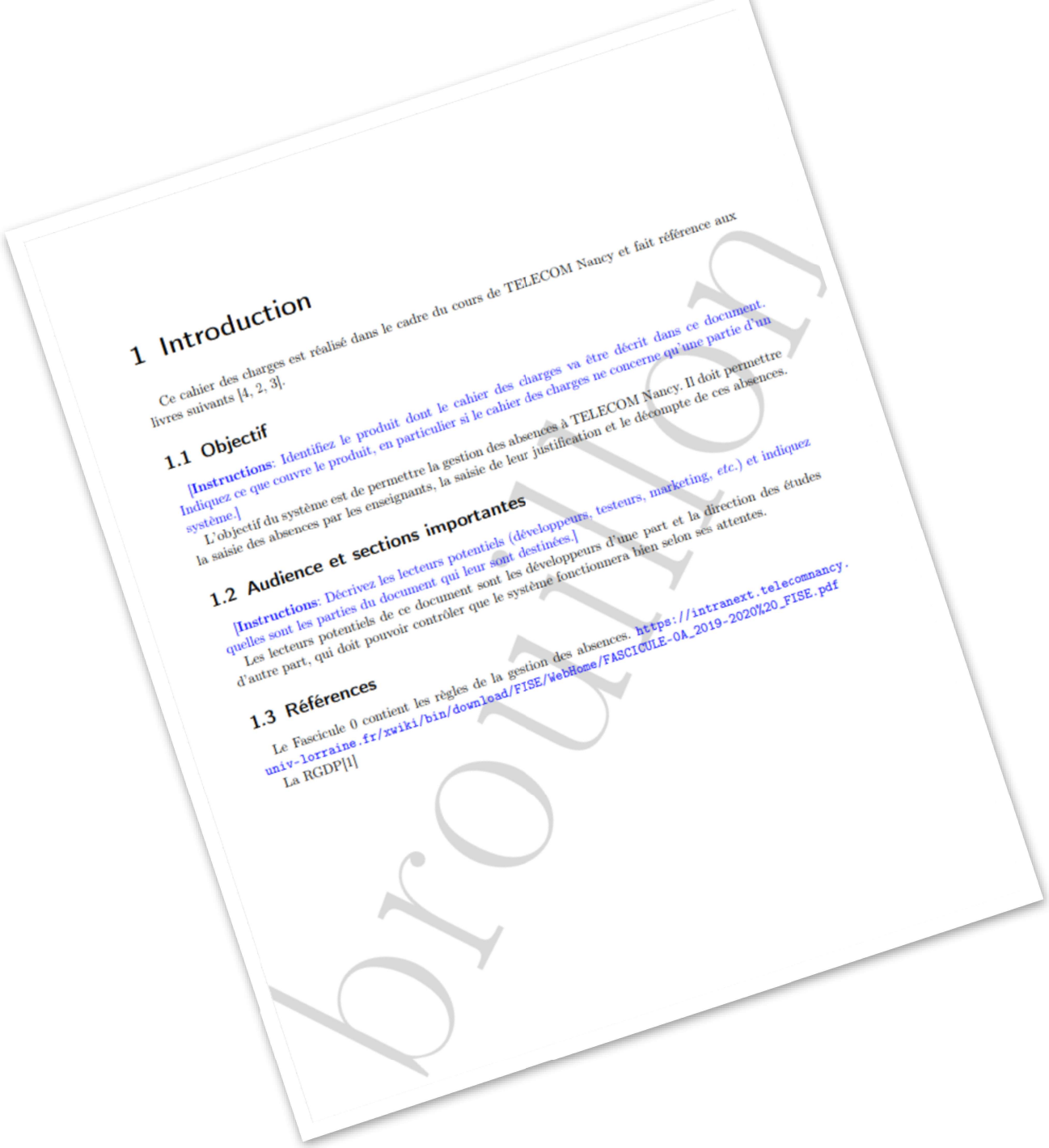
- The software requirements document is the official statement of what is required of the system develo
- Should include both a definition of user requiremer a specification of the system requirements.
- It is NOT a design document. As far as possible, it sh set of WHAT the system should do rather than HOW should do it.

Requirements document variability

- Information in requirements document depends on type system and the approach to development used.
- Systems developed incrementally will, typically, have less detail in the requirements document.
- Requirements documents standards have been designed IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The document model of the project

Introduction



1 Introduction

Ce cahier des charges est réalisé dans le cadre du cours de TELECOM Nancy et fait référence aux livres suivants [4, 2, 3].

1.1 Objectif

[**Instructions**: Identifiez le produit dont le cahier des charges va être décrit dans ce document. Indiquez ce que couvre le produit, en particulier si le cahier des charges ne concerne qu'une partie d'un système.]

L'objectif du système est de permettre la gestion des absences à TELECOM Nancy. Il doit permettre la saisie des absences par les enseignants, la saisie de leur justification et le décompte de ces absences.

1.2 Audience et sections importantes

[**Instructions**: Décrivez les lecteurs potentiels (développeurs, testeurs, marketing, *etc.*) et indiquez quelles sont les parties du document qui leur sont destinées.]

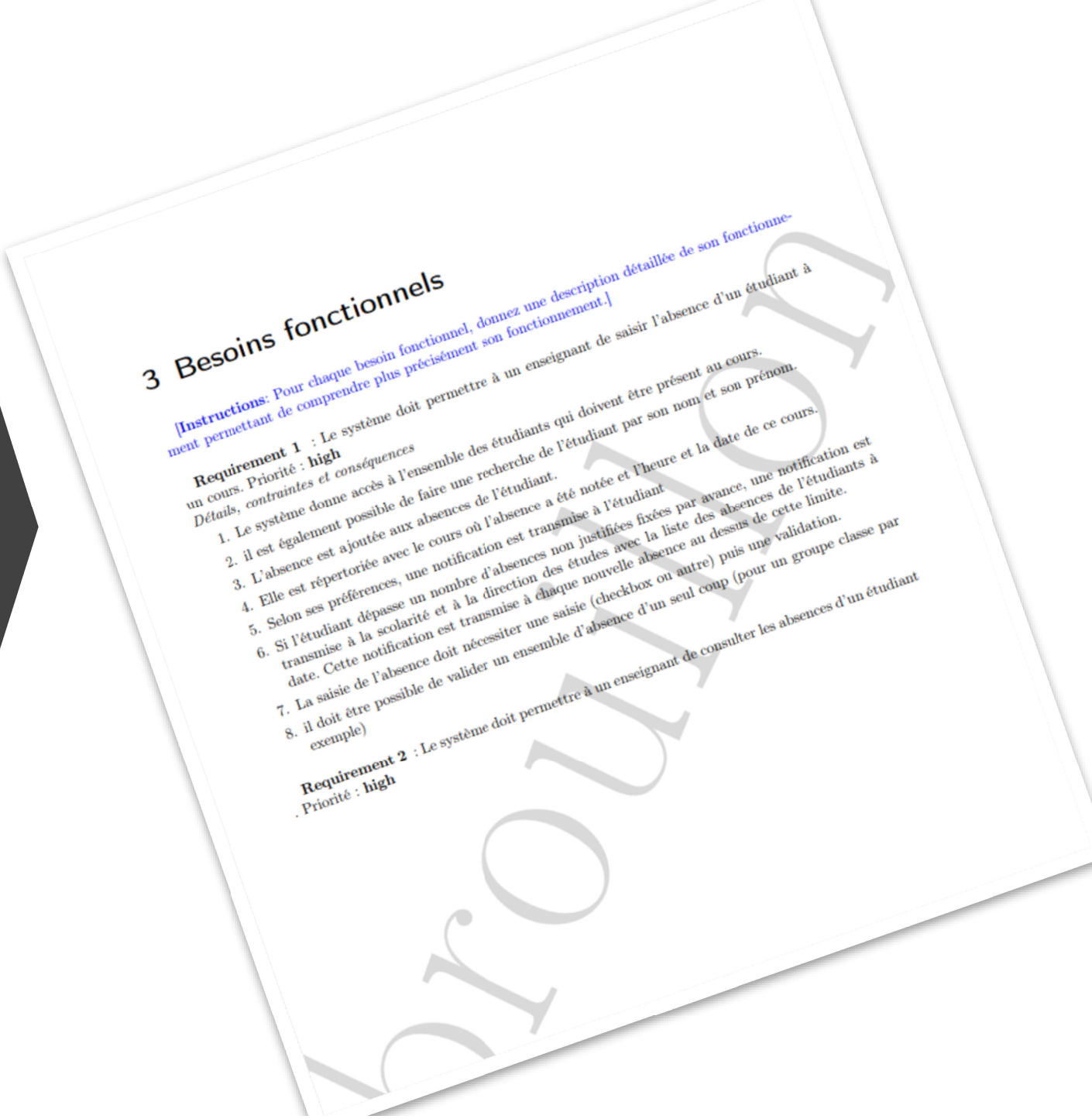
Les lecteurs potentiels de ce document sont les développeurs d'une part et la direction des études d'autre part, qui doit pouvoir contrôler que le système fonctionnera bien selon ses attentes.

1.3 Références

Le Fascicule 0 contient les règles de la gestion des absences. `https://intranext.telecomnancy.univ-lorraine.fr/xwiki/bin/download/FISE/WebHome/FASCICULE-0A_2019-2020%20_FISE.pdf`

La RGDP[1]

Fonctional Requirements



3 Besoins fonctionnels

[**Instructions**: Pour chaque besoin fonctionnel, donnez une description détaillée de son fonctionnement permettant de comprendre plus précisément son fonctionnement.]

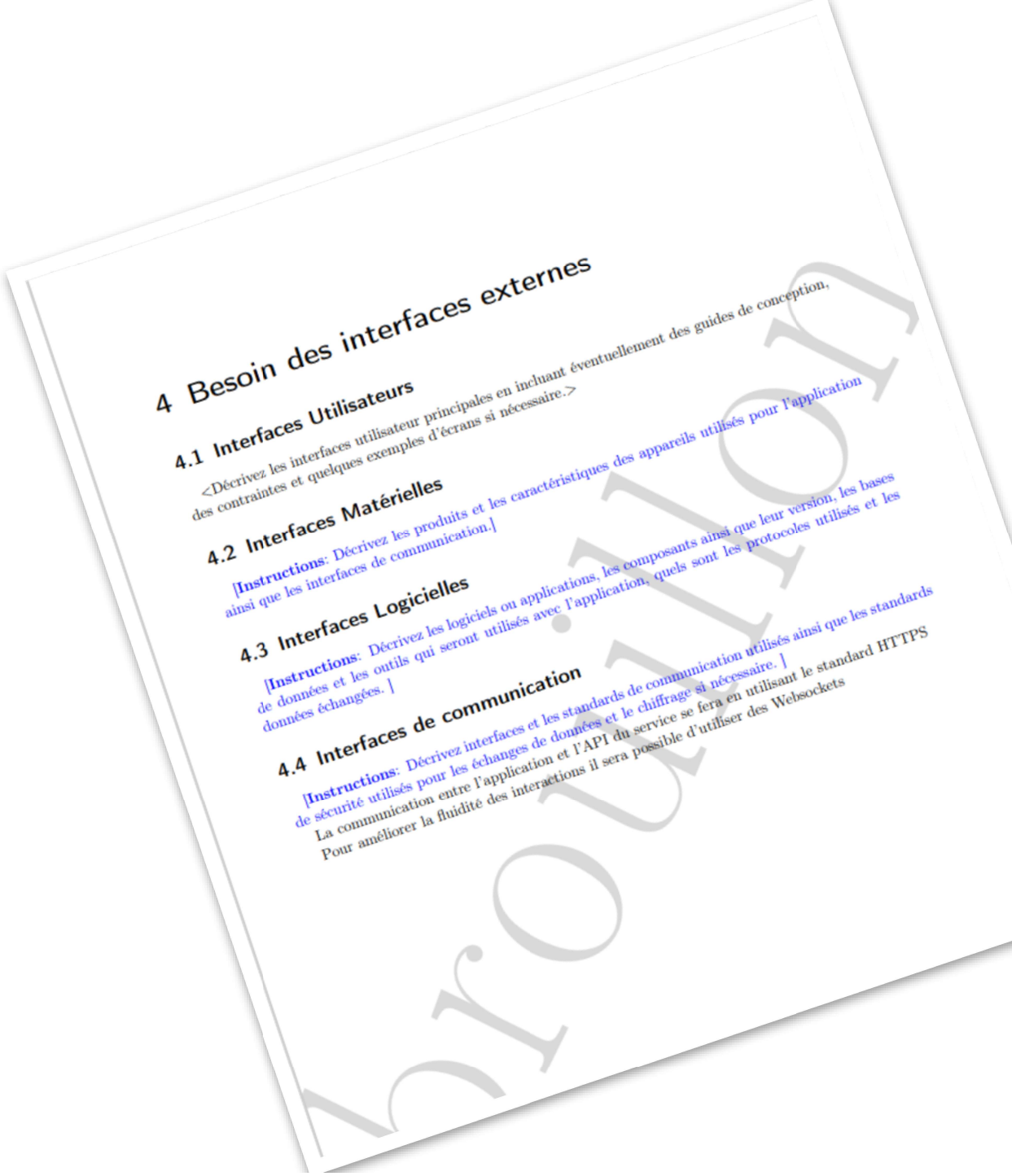
Requirement 1 : Le système doit permettre à un enseignant de saisir l'absence d'un étudiant à un cours. Priorité : **high**

Détails, contraintes et conséquences

1. Le système donne accès à l'ensemble des étudiants qui doivent être présent au cours.
2. il est également possible de faire une recherche de l'étudiant par son nom et son prénom.
3. L'absence est ajoutée aux absences de l'étudiant.
4. Elle est répertoriée avec le cours où l'absence a été notée et l'heure et la date de ce cours.
5. Selon ses préférences, une notification est transmise à l'étudiant
6. Si l'étudiant dépasse un nombre d'absences non justifiées fixées par avance, une notification est transmise à la scolarité et à la direction des études avec la liste des absences de l'étudiants à date. Cette notification est transmise à chaque nouvelle absence au dessus de cette limite.
7. La saisie de l'absence doit nécessiter une saisie (checkbox ou autre) puis une validation.
8. il doit être possible de valider un ensemble d'absence d'un seul coup (pour un groupe classe par exemple)

Requirement 2 : Le système doit permettre à un enseignant de consulter les absences d'un étudiant . Priorité : **high**

Besoins des interfaces externes



4 Besoin des interfaces externes

4.1 Interfaces Utilisateurs

<Décrivez les interfaces utilisateur principales en incluant éventuellement des guides de conception, des contraintes et quelques exemples d'écrans si nécessaire.>

4.2 Interfaces Matérielles

[**Instructions**: Décrivez les produits et les caractéristiques des appareils utilisés pour l'application ainsi que les interfaces de communication.]

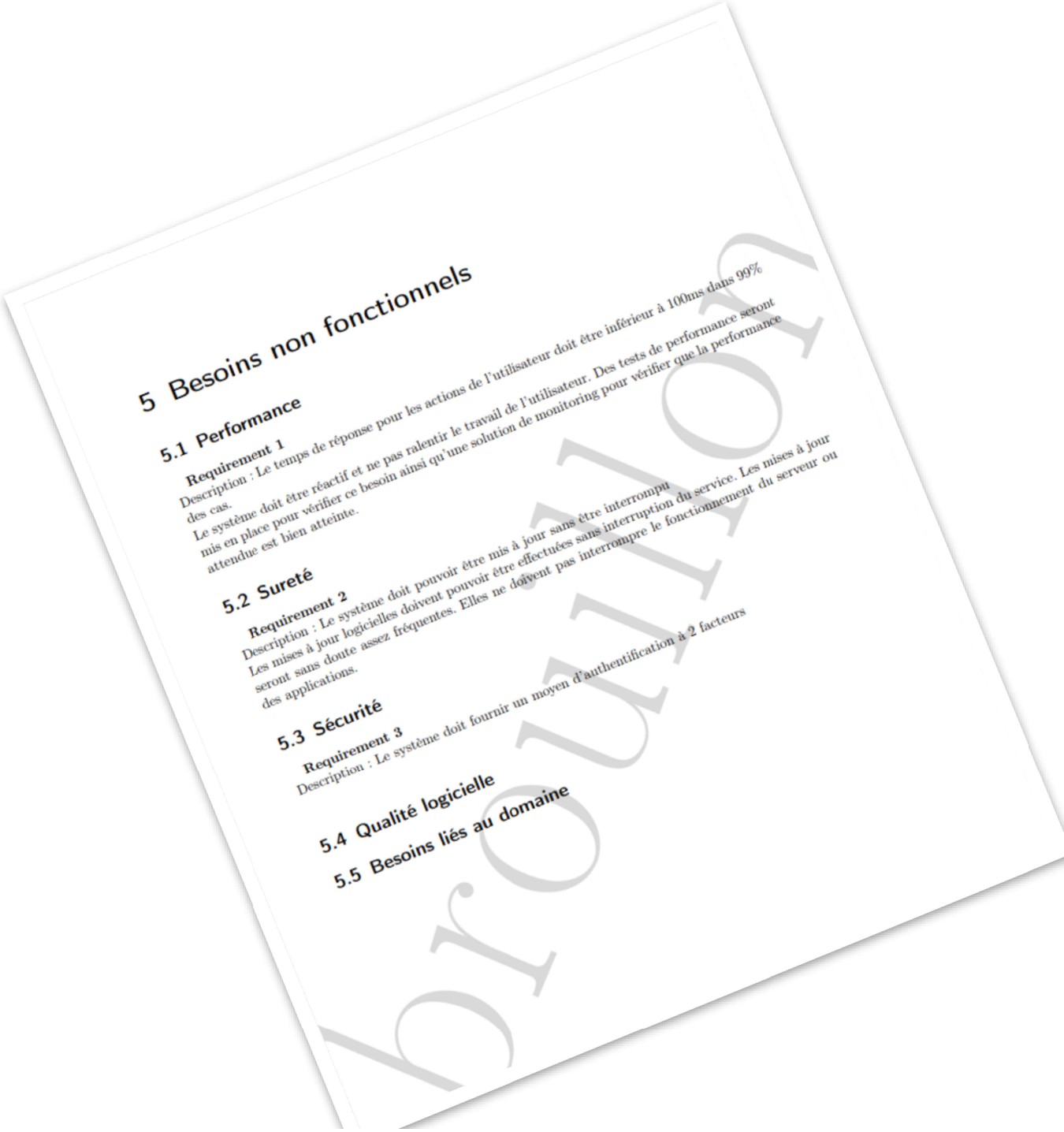
4.3 Interfaces Logicielles

[**Instructions**: Décrivez les logiciels ou applications, les composants ainsi que leur version, les bases de données et les outils qui seront utilisés avec l'application, quels sont les protocoles utilisés et les données échangées.]

4.4 Interfaces de communication

[**Instructions**: Décrivez interfaces et les standards de communication utilisés ainsi que les standards de sécurité utilisés pour les échanges de données et le chiffrage si nécessaire.]
La communication entre l'application et l'API du service se fera en utilisant le standard HTTPS
Pour améliorer la fluidité des interactions il sera possible d'utiliser des Websockets

Besoins non fonctionnels



5 Besoins non fonctionnels

5.1 Performance

Requirement 1
Description : Le temps de réponse pour les actions de l'utilisateur doit être inférieur à 100ms dans 99% des cas.
Le système doit être réactif et ne pas ralentir le travail de l'utilisateur. Des tests de performance seront mis en place pour vérifier ce besoin ainsi qu'une solution de monitoring pour vérifier que la performance attendue est bien atteinte.

5.2 Sureté

Requirement 2
Description : Le système doit pouvoir être mis à jour sans être interrompu
Les mises à jour logicielles doivent pouvoir être effectuées sans interruption du service. Les mises à jour seront sans doute assez fréquentes. Elles ne doivent pas interrompre le fonctionnement du serveur ou des applications.

5.3 Sécurité

Requirement 3
Description : Le système doit fournir un moyen d'authentification à 2 facteurs

5.4 Qualité logicielle

5.5 Besoins liés au domaine

Appendices

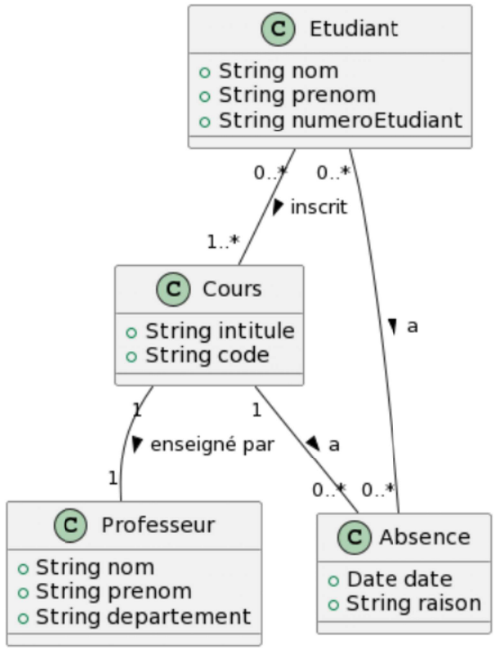


FIGURE 6.1 – Exemple de diagramme de classes

6 Appendices

6.1 Appendice A : Glossaire

6.2 Appendice B : Modèles d'analyse

Modèle de données et Dictionnaire de données

[**Instructions**: Diagramme entité association ou diagramme de classes correspondant aux infor
tions nécessaires à l'application. Vous ajouterez un dictionnaire de données permettant de compre
le rôle des différents attributs]

Dictionnaire de données

- Étudiant
  - nom (String) : Le nom de famille de l'étudiant.
  - prenom (String) : Le prénom de l'étudiant.
  - numeroEtudiant (String) : Un identifiant unique pour chaque étudiant.
- Cours
  - intitule (String) : Le nom du cours.
  - code (String) : Un code unique identifiant le cours.
- Absence
  - date (Date) : La date à laquelle l'absence a été enregistrée.
  - raison (String) : La raison de l'absence de l'étudiant, si elle est fournie.
- Professeur
  - nom (String) : Le nom de famille du professeur.
  - prenom (String) : Le prénom du professeur.
  - departement (String) : Le département auquel le professeur est rattaché.
- Relations
  - Étudiant-Cours : Cette relation montre quel étudiant est inscrit à quels cours.
  - Cours-Absence : Cette relation associe chaque absence à un cours spécifique.
  - Etudiant-Absence : Cette relation relie les absences à l'étudiant spécifique concerné.
  - Cours-Professeur : Chaque cours est associé à un professeur qui l'enseigne.

Requirements validation

Requirements validation

A process of verifying that requirements a
complete, correct, and consistent

Helps to ensure that the system will meet
the needs of its stakeholders

Can involve various methods of testing,
analysis, and review

Requirements error costs are high, so
validation is critical

- Fixing a requirements error after delivery may cost u
to 100 times the cost of fixing an implementation er

Examples of requirement validation methods:

- Functional testing: Testing the system to ensure that it meets functional requirements
- Non-functional testing: Testing the system to ensure that it me non-functional requirements such as performance, reliability, a security
- Formal verification: Using mathematical proof to verify that the system meets its requirements
- Peer reviews: Having other team members or stakeholders revi the requirements to identify issues or ambiguities
- Prototyping: Building a partial or full-scale version of the syster test its functionality and validate requirements

Requirements reviews

A process of systematically examining requirements to ensure they are complete, correct, and consistent

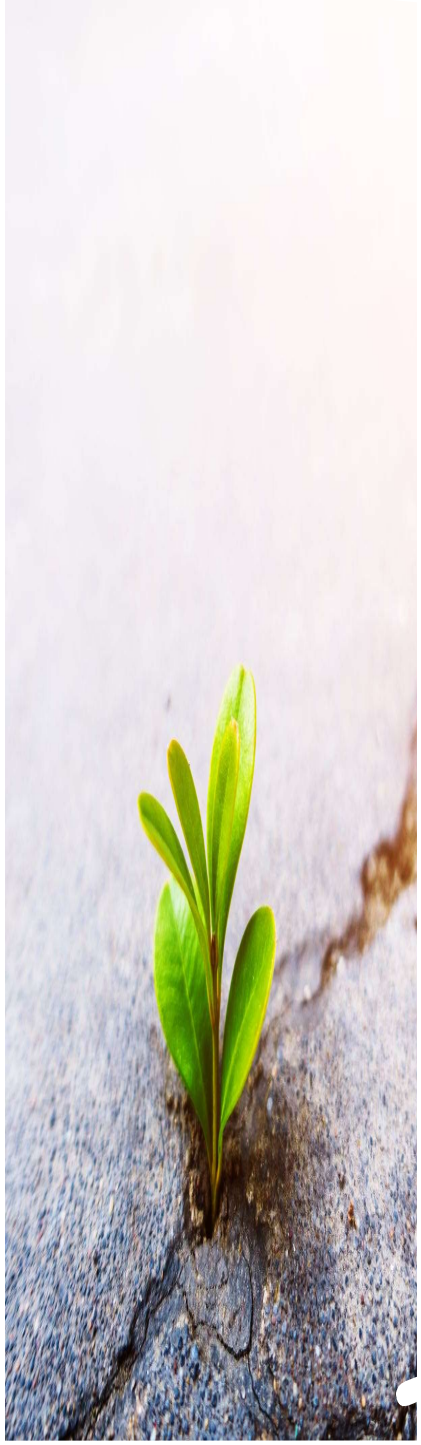
Can be performed by various stakeholders, including developers, testers, and end-users

Helps to identify issues and improve the quality of requirements

- Peer review: Reviewing requireme by other team members or stakeholders
- Walkthrough: A step-by-step revie the requirements with a group of stakeholders
- Inspection: A more formal review process with a designated leader a documented findings
- Quality review: Reviewing requirements against established quality criteria, such as completen consistency, and verifiability

Review checks

Verifiability	• Is the requirement realistically testable?
Comprehensibility	• Is the requirement properly understood?
Traceability	• Is the origin of the requirement clearly stated?
Adaptability	• Can the requirement be changed without a large impact on other requirements?



Requirements chang management

- The process of managing changes to requirements throughout the development lifecycle
- Ensures that changes are made in a controlled and systematic way to avoi unintended consequences
- Helps to maintain the quality of the system and ensure that it meets stakeholders' needs

Key activities in change managemer

- Change request: A formal request to change a requirement, including a description of the pro change and the reason for it
- Impact analysis: Assessing the potential impact change on the system, including costs, risks, an benefits
- Approval: Reviewing the change request and approving or rejecting it based on the impact a
- Implementation: Making the necessary change the system and ensuring that they are adequate tested and validated
- Verification: Ensuring that the change has been adequately implemented and does not have an unintended consequences

Key points

Requirements for a software system set out what it should do and define its operation and implementation constraints.

Functional requirements are statements of the services the system must prov or descriptions of how some computations must be carried out.

Non-functional requirements often constrain the developed system and proc used.

They often relate to the system’s emergent properties and apply to the entire system.

Key points

The requirements engineering process is iterative and includes requirements elicitation, specification and validation.

Requirements elicitation is an iterative process representing a spiral of activit requirements discovery, classification and organisation, requirements negotia and documentation.

You can use various techniques for requirements elicitation, including intervie and ethnography. User stories and scenarios may be used to facilitate discussions.

Key points

Requirements specification formally documents the user and system requirements and creates a software requirements document.

The software requirements document is an agreed statement of the syste requirements and should be organised so that system customers and developers can use it.

Key points

Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.

Business, organizational and technical changes inevitably lead to changes the requirements for a software system. Requirements management is th process of managing and controlling these changes.