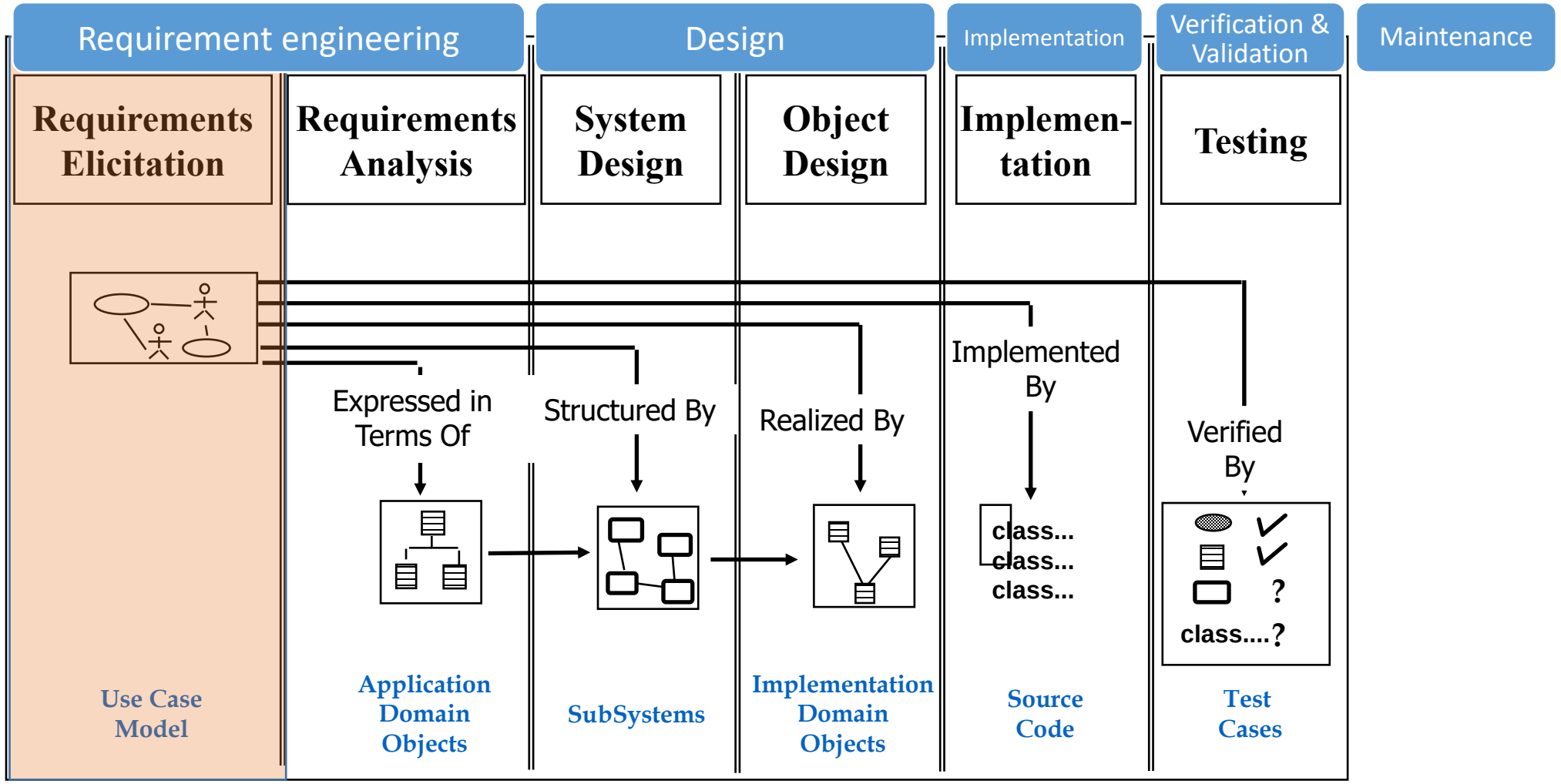
Requirement

Software Development for Robotics and Automation Engineering

Software Lifecycle Activities



Software Requirement

- Software requirements are description of features and functionalities of the target system.
- The requirements convey the expectations of users from the software product.
- The requirements can be obvious or hidden, known or unknown, expected or unexpected from client’s point of view.

Requirement Engineering (RE)

- The process to gather the software requirements from client, analyze and document them is known as requirement engineering.
- The goal of requirement engineering is to develop and maintain sophisticated and descriptive System Requirements Specification (SRS) document.
- Step of Requirement Engineering
  - Requirement Elicitation
  - Requirement Analysis
  - Requirement Documentation
  - Requirement Review

Software Requirement Specification

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, security, quality, limitations etc.
- The requirements received from client are written in natural language. (user requirements)
- It is the responsibility of team to analyst the user requirement document in technical language. (system requirements)
- So that the system requirements can be comprehended and useful by the software development team.

Software Requirement Validation

- After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements incorrectly.
- This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions :
  - If they can be practically implemented
  - If they are valid and as per functionality and domain of software
  - If there are any ambiguities
  - If they are complete
  - If they can be demonstrated

Software Requirements Characteristics

- Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined.
- A complete Software Requirement Specifications must be:

  - Clear
  - Correct
  - Consistent
  - Coherent
  - Comprehensible
  - Modifiable
  - Verifiable
  - Prioritized
  - Unambiguous
  - Traceable
  - Credible source

User Interface Requirements

- UI is an important part of any software or hardware or hybrid system. A software is widely accepted if it is easy to operate, quick in response, effectively handling operational errors and providing simple yet consistent user interface.
- User interface requirements are briefly mentioned as :
  - Content presentation
  - Easy navigation
  - Simple interface
  - Responsive
  - Consistent UI elements
  - Feedback mechanism
  - Default settings
  - Purposeful layout
  - Strategical use of color and texture
  - Provide help information
  - User centric approach
  - Group based view settings

Requirements Elicitation

- Challenging activity
- Requires collaboration of people with different backgrounds
  - User with application domain knowledge
  - Developer with solution domain knowledge (design knowledge, implementation knowledge)

- Bridging the gap between user and developer:
  - *Scenarios*: Example of the use of the system in terms of a series of interactions with between the user and the system
  - *Use cases*: Abstraction that describes a class of scenarios

Types of Requirements Elicitation

- Greenfield Engineering
  - Development starts from scratch, no prior system exists, the requirements are extracted from the end users and the client
  - Triggered by user needs
  - Example: Develop a game from scratch: Asteroids
- Re-engineering
  - Re-design and/or re-implementation of an existing system using newer technology
  - Triggered by technology enabler
  - Example: Reengineering an existing game
- Interface Engineering
  - Provide the services of an existing system in a new environment
  - Triggered by technology enabler or new market needs
  - Example: Interface to an existing game (Bumpers)

Requirements Elicitation Process

- **Requirements gathering -** The developers discuss with the client and end users and know their expectations from the software.
- **Organizing Requirements -** The developers prioritize and arrange the requirements in order of importance, urgency and convenience.
- **Negotiation & discussion -** If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, if they are, it is then negotiated and discussed with stakeholders. Requirements may then be prioritized and reasonably compromised. The requirements come from various stakeholders. To remove the ambiguity and conflicts, they are discussed for clarity and correctness. Unrealistic requirements are compromised reasonably.
- **Documentation -** All formal & informal, functional and non-functional requirements are documented and made available for next phase processing.

Requirements Elicitation Techniques

- Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development.
- There are various ways to discover requirements
  - Interviews
  - Surveys
  - Questionnaires
  - Task Analysis
  - Domain Analysis
  - Brainstorming
  - Prototyping
  - Observation

Scenarios / User Stories

- “A narrative description of what people do and experience as they try to make use of computer systems and applications” *[M. Carrol, Scenario-based Design, Wiley, 1995]*

- A concrete, focused, informal description of a single feature of the system used by a single actor.

- Scenarios can have many different uses during the software lifecycle

Scenario Example: Warehouse on Fire

- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, reports the emergency from her car.

- Alice enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene given that area appear to be relatively busy. She confirms her input and waits for an acknowledgment.

- John, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.

- Alice received the acknowledgment and the ETA.

Documentation schema for the scenario

Scenario name: warehouse0nFire

Participating actor instances:

bob, alice : FieldOfficer

john: Dispatcher

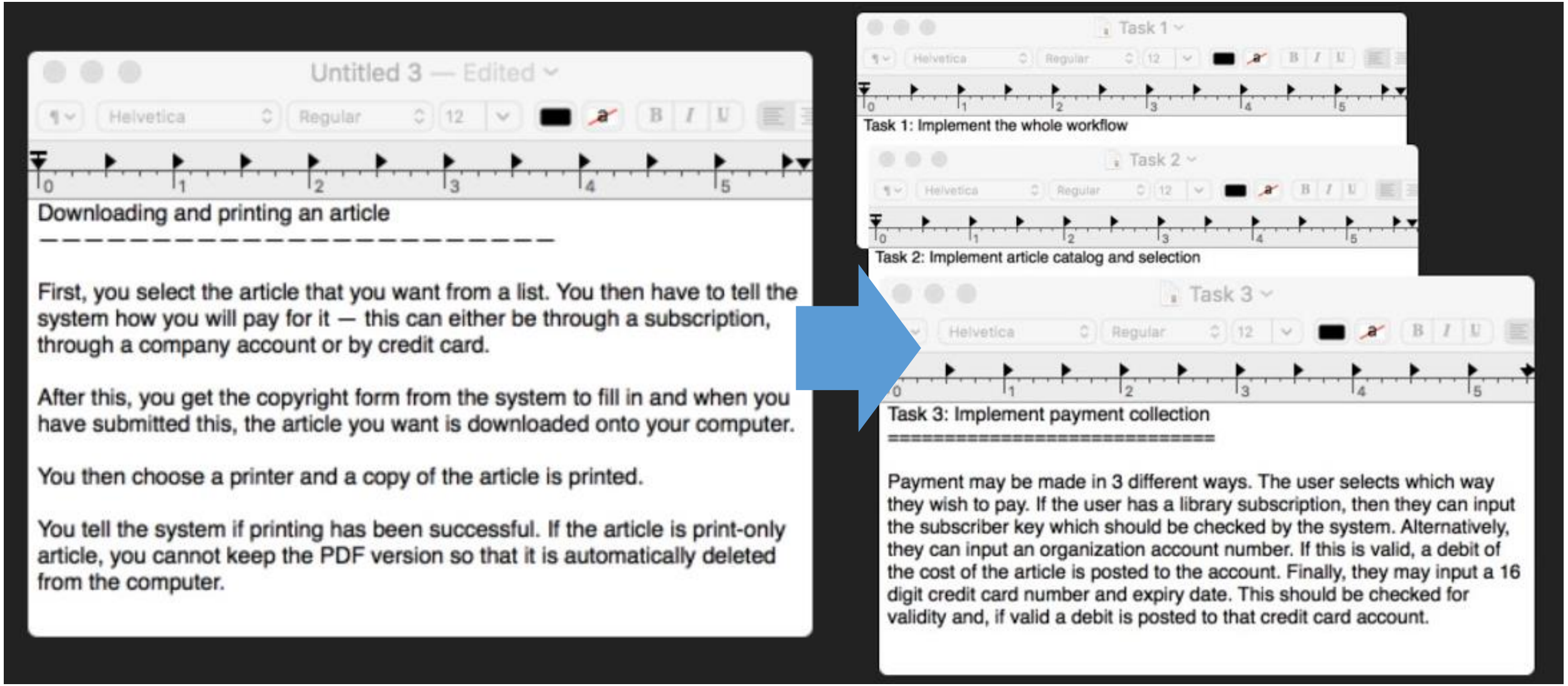
Flow of events :

1. Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, activates the “Report Emergency” function from her FRIEND laptop.
2. Alice enters the address of the building, a brief description of its location (i.e., northwest corner), and an emergency level. In addition to a fire unit, she requests severa1 paramedic units on the scene, given that the area appears to be relatively busy. She confirms her input and waits for an acknowledgment.
3. John, the Dispatcher , is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He creates allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.
4. Alice receives the acknowledgment and the ETA.

Types of Scenarios

- As-is scenario
  - Used in describing a current situation. Usually used during re-engineering. The user describes the system.
- Visionary scenario
  - Used to describe a future system. Usually described in greenfield engineering or reengineering.
  - Can often not be done by the user or developer alone
- Evaluation scenario
  - User tasks against which the system is to be evaluated
- Training scenario
  - Step by step instructions designed to guide a novice user through a system

User Story Card Example



Types of Requirements

Functional requirements describe **what the system should do**.

Nonfunctional requirements describe **how the system works**.

- **Functional requirements**: Describe the interactions between the system and its environment independent from implementation
  - The watch system must display the time based on its location
- **Nonfunctional requirements**: User visible aspects of the system not directly related to functional behavior.
  - The response time must be less than 1 second
  - The accuracy must be within a second
  - The watch must be available 24 hours a day except from 2:00am-2:01am and 3:00am-3:01am

Taxonomy of Nonfunctional Requirements

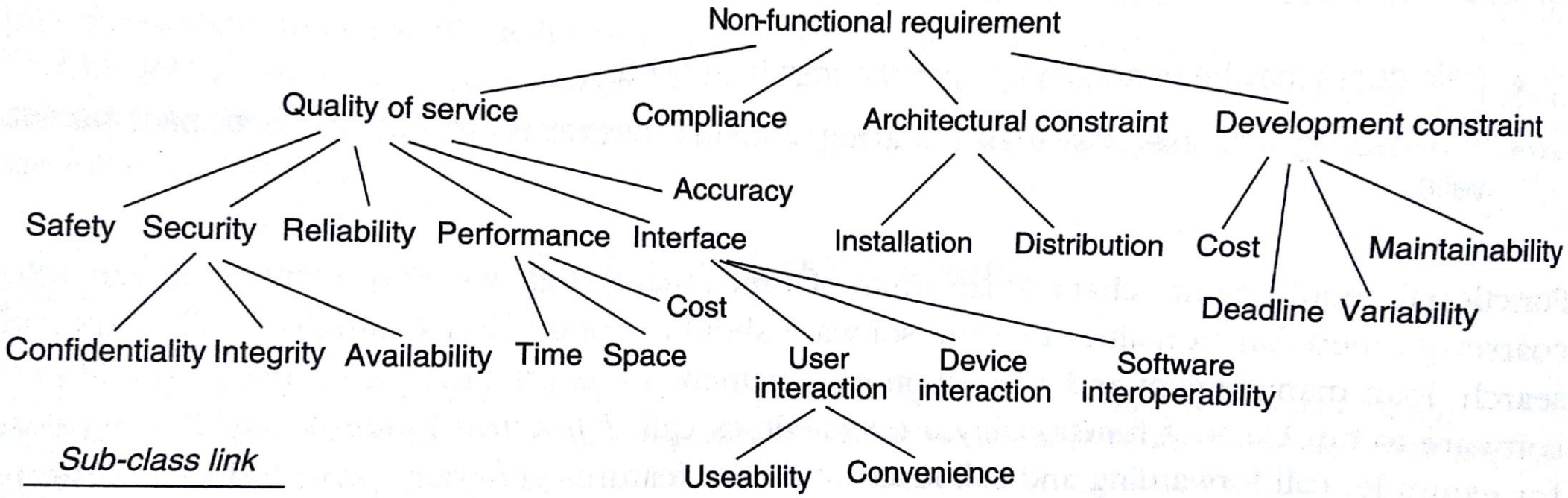
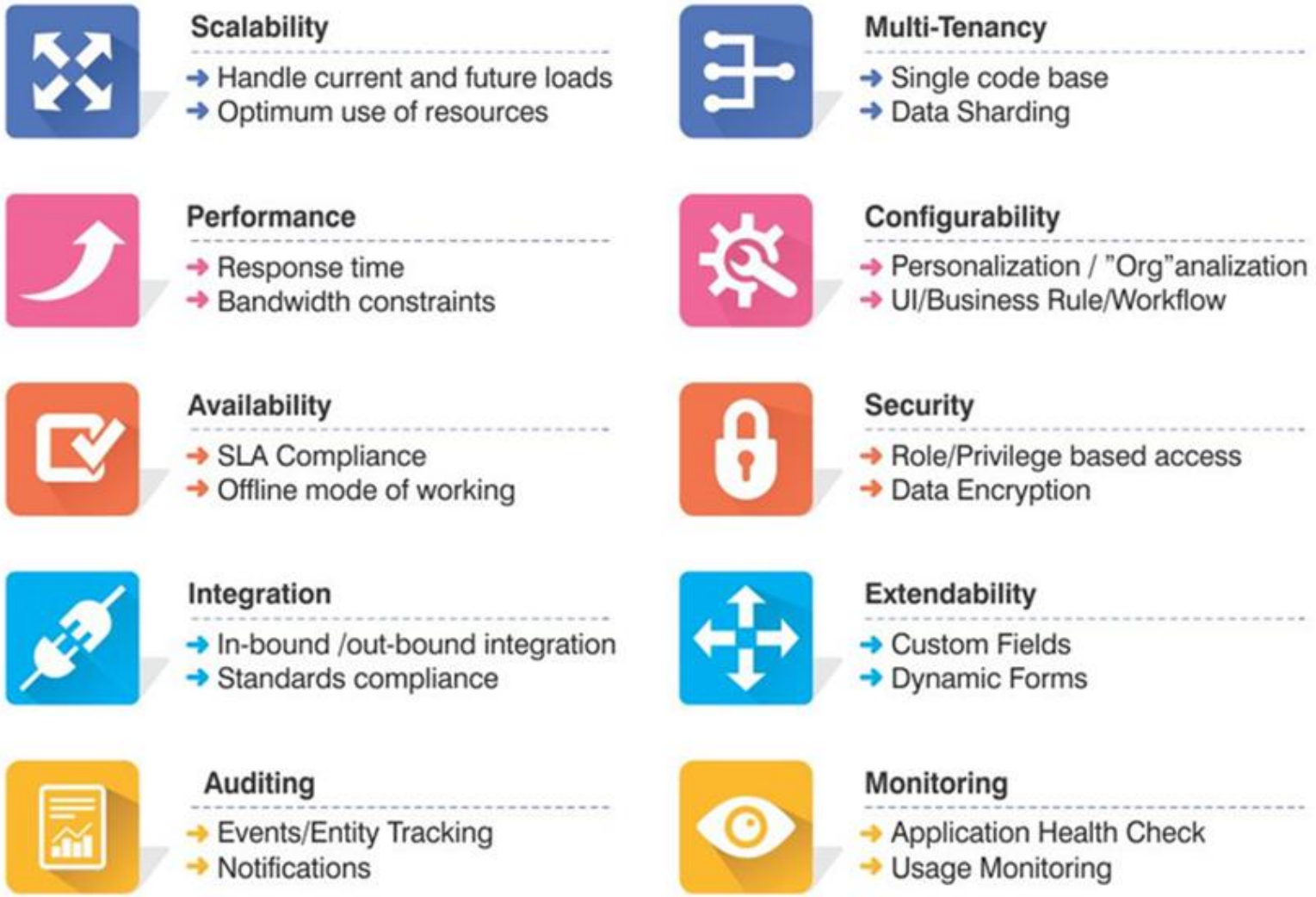


Figure 1.5 *A taxonomy of non-functional requirements*

Ref: Axel van Lamsweerde. Requirements Engineering: From System Goals to UML Models to Software. Specifications

Nonfunctional Requirements



https://blog.aspiresys.com/software-product-engineering/producteering/top-10-nfr-in-software-architecture-part-1/

Functional VS Nonfunctional Requirements

FUNCTIONAL	NONFUNCTIONAL
WHAT a system does	HOW *well* the system does it within design & resource constraints

Functional VS Nonfunctional Requirements

- Example

Functional testing	Nonfunctional testing
When inputs are valid, app login functions	After login, the dashboard loads within 3 seconds
When email notifications are on, and user receives a new message, an email notification is sent	The email notification is sent within 5 minutes
When a JPG file under 1MB is uploaded, the uploader accepts the file	When eight files or less (each under 1MB) are uploaded at the same time, the queues all
When the settings menu item is clicked, the settings page loads	The settings page has a matching appearance to the rest of the GUI

What is a User Interview?

- User interviews are where a researcher asks questions of, and records responses from, users. They can be used to examine the user experience, the usability of the product
- The ideal interview takes place with **two** UX researchers and one user. The first UX researcher focuses on asking questions and guiding the interviewee through the interview. The second takes notes. If a second researcher is unavailable for this – then videoing or audio recording an interview can be a good way to record the information elicited.
- Typical topics covered within user interviews include:
  - Background (such as ethnographic data)
  - The use of technology in general
  - The use of the product
  - The user's main objectives and motivations
  - The user's pain points

https://www.interaction-design.org/literature/article/how-to-conduct-user-interviews

Tip for User Interviews

- Create a script to ask questions
- Make sure you begin by explaining the purpose of the interview – what are you trying to achieve?
- Also explain how the person's data and any data you collect will be used from the interview.
- Try to keep leading questions to a minimum. A good question is "Do you use instant messaging?" rather than "How often do you use Snapchat?" The former lets you explore what the user actually does. The latter presupposes that user is working with Snapchat and that's the extent of their instant messaging activity.
- Keep it reasonably short. If you read the script aloud and it takes more than 10 minutes to read… it's probably too long. Interviews should, ideally, be less than 1 hour long and the majority of the time spent should be the interviewee talking and the researcher listening.

https://www.interaction-design.org/literature/article/how-to-conduct-user-interviews

How to Conduct a User Interview

- **Make your interviewee comfortable** – dress in a manner similar to them (you in a suit them in a tracksuit is going to make it feel like a job interview rather than a user test), make sure they understand you are testing a product or an idea and not the user themselves, offer them a drink (non-alcoholic), conduct a little small talk (but only a little) before you start, etc.
- **Try to keep the interview on time and heading in the right direction** – the reason scripts are useful is because you can reference them for this
- **Try to focus on the interviewee and not on note making** – it's just plain rude to bury your head in your notes. Maintain eye contact, keep a conversation flowing and record the interview rather than getting lost in note making.
- **Thank the interviewee at the end of the process** – not only is this polite but you can offer a chance for the interviewee to ask any question of their own at this point too.

https://www.interaction-design.org/literature/article/how-to-conduct-user-interviews