Applied Software Engineering SIT725

T1 2025

Dr Niroshinie Fernando

Lecture 2
Software Requirements Specifications (SRS)

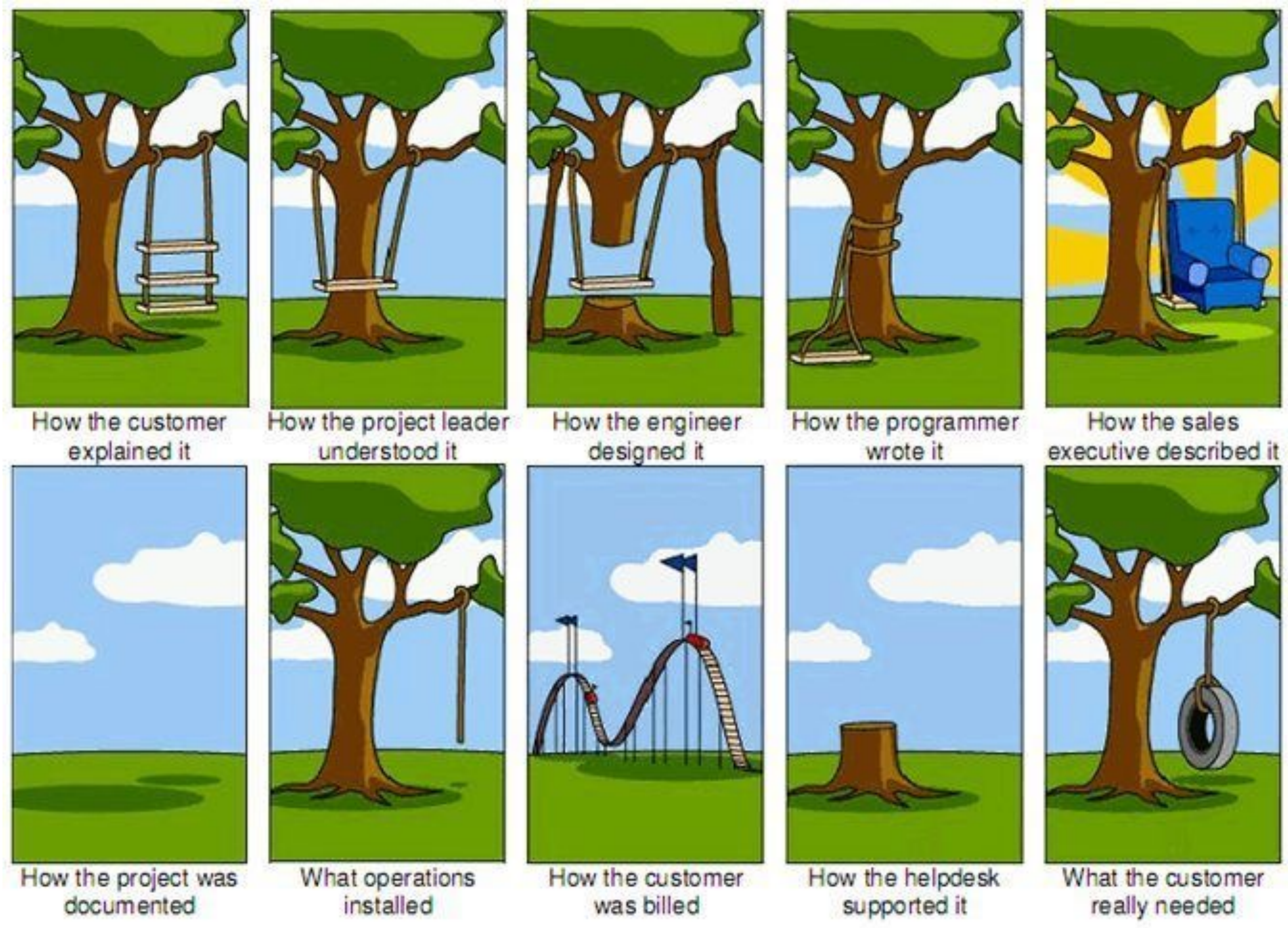
What is an SRS?

a detailed document that defines what a software system is expected to do and how it should perform.

a contract between stakeholders and developers, ensuring that all parties are aligned on the software's goals, functionality, and constraints before development begins.

Why do we need an SRS?

Before we build the software right, we need to build the right software


How the customer explained it
How the project leader understood it
How the engineer designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

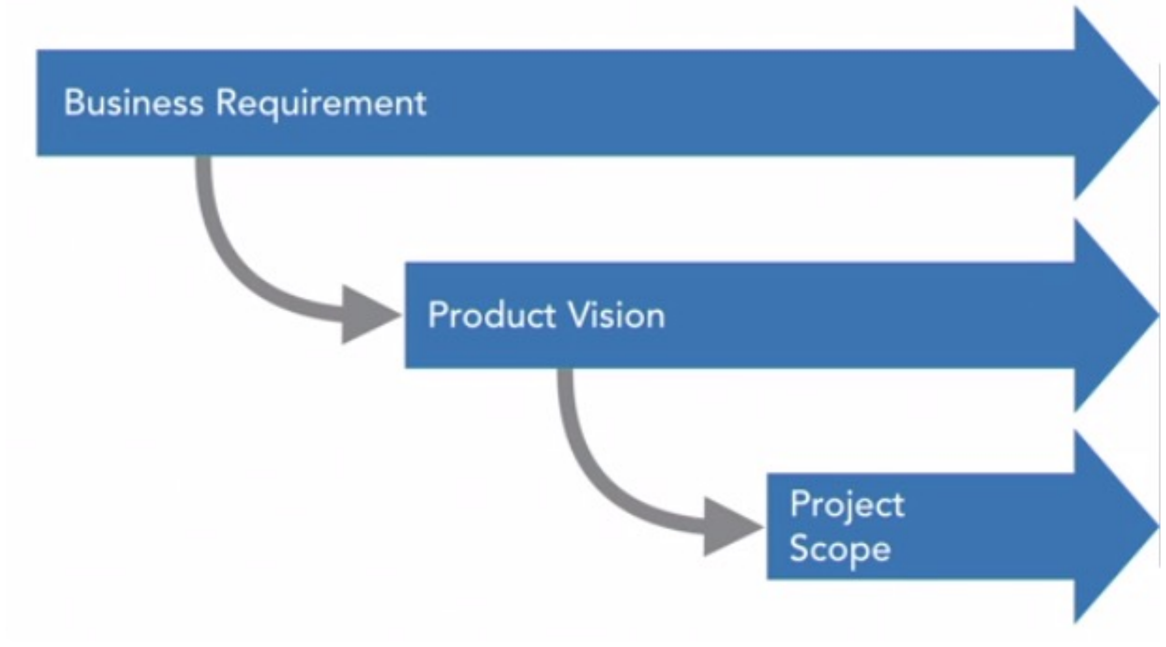
Why SRS?

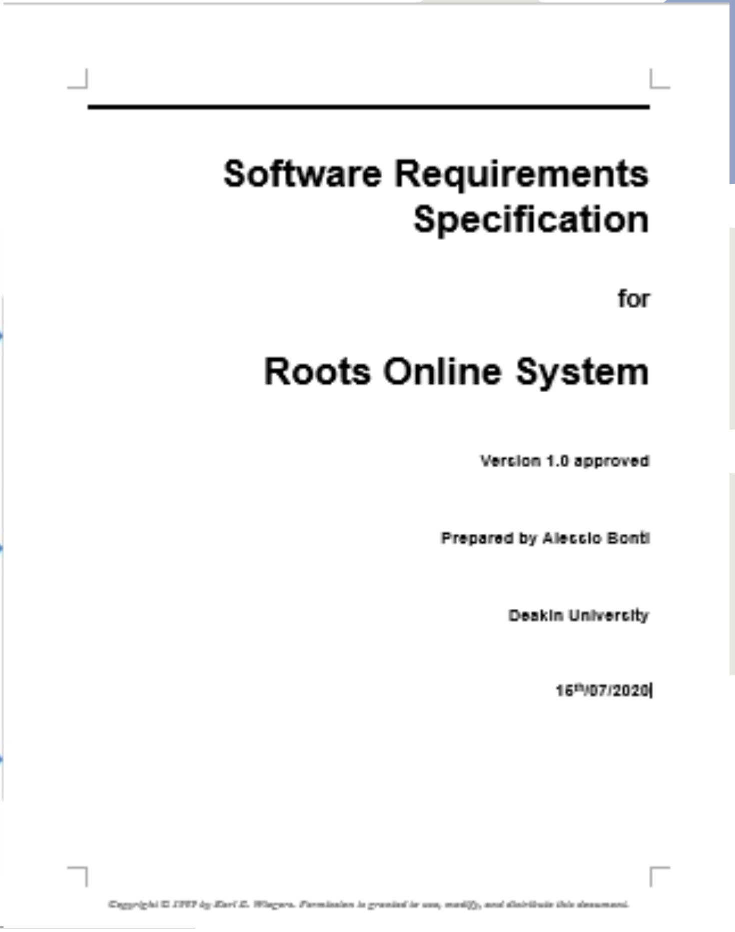
"It is not enough to do your best: you must KNOW what to do, and THEN do your best."

W.Edwards Deming

SRS- Project definition

Business requirements drive the product vision which then define the project scope.





Software Requirements Specification

for

Roots Online System

Version 1.0 approved

Prepared by Alessio Bonti

Deakin University

16th/07/2020

SRS- Stakeholders

- Meet your stakeholders.
- Who are the stakeholders?
- Are there any domain experts?
- What are their needs and wants?

SRS- Business requirements

- At a very high level, what problem are you trying to solve, and what would solve it ?

Task	Explanation
Problem	Graduate Job Market being more competitive and expecting beyond university learnt skills
Affects	Some students who complete their degree are not professionally ready or have competitive advantage
Which impacts	Their employability once they graduate
A successful solution would be	A software solution similar to LinkedIn which -Tracks the student growth throughout his degree - Provides professional related information - Allows external people to view the students profile

SRS- Product vision

- How is your product special ? And how is it different from what is already in the market?

Task	Explanation
For	Students
Who	Want to increase their professional skills and visibility
The product	Is a Professional Social Network similar to linkedin
That	Allows students to create a professional profile connected to their coursework, so that they can develop skils and be more marketable
Unlike	LinkdedIN
Our product	Is completly developed for students, hence everyone is at the same level

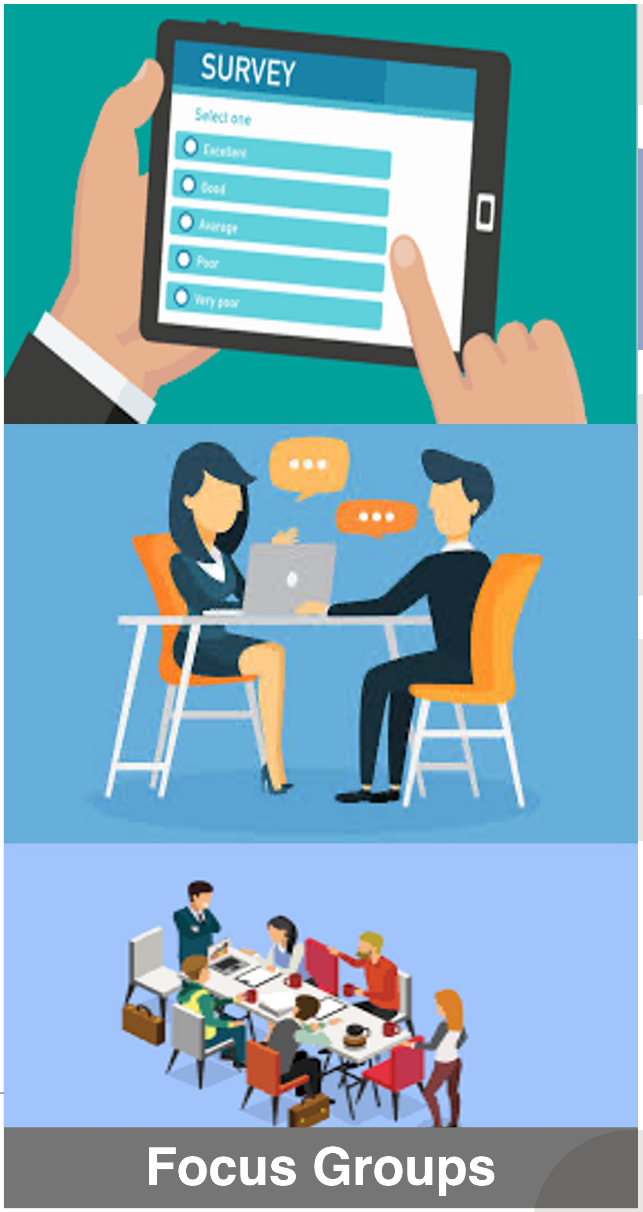
SRS-Collecting requirements

- Gathering information is essential to the creation of a product.
- Each technique can provide different outcomes depending on the culture, the aim and the nature of the product.

Surveys

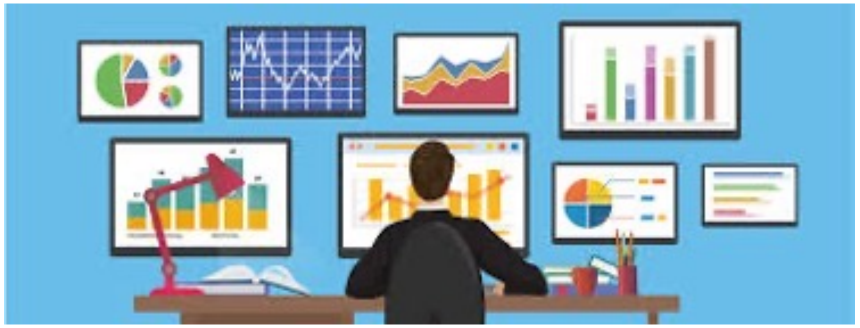
Interviews

Observation



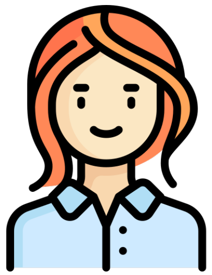
SRS- Requirements analysis

To understand what we need to build, we need to understand the functionalities. We will focus on **USER STORIES**.



USER STORIES allow us to:

- Identify conditions
- Understand **acceptance criteria**
- Define domain classes

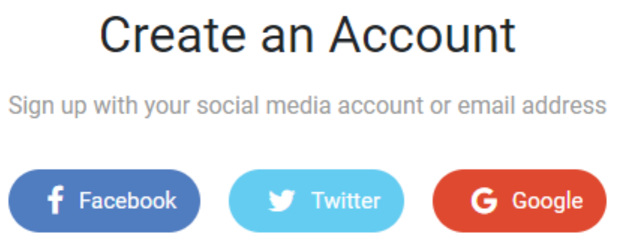


User Stories

- A user story is a clear way to understand what the software is required to do.
- Large applications are generally made up of many user stories, each story contributes to one key aspect.
- Analysing a story reveals many requirements which may or may not have discovered in other stories before.

User story: Sign up

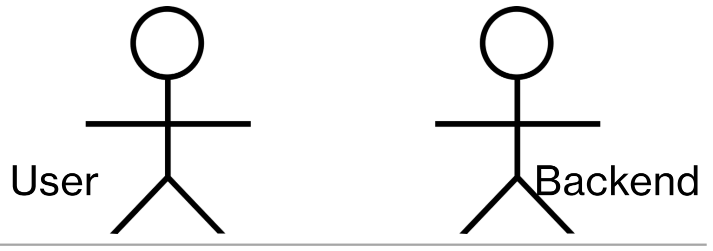
As a user, I want to be able to create an account using my social account (Facebook, twitter, or Google), so what I can easily join as a new user


Create an Account

Sign up with your social media account or email address

Use Case

- Once we know what is required to satisfy the user, we can now try to design the use case.
- The simplest way is to follow the natural flow of things, from top to bottom, following what may call the "User journey"
- Do not skip steps, keep them separated, this way you can validate one by one.



Primary actor : User

Secondary Actor : Signup backend.

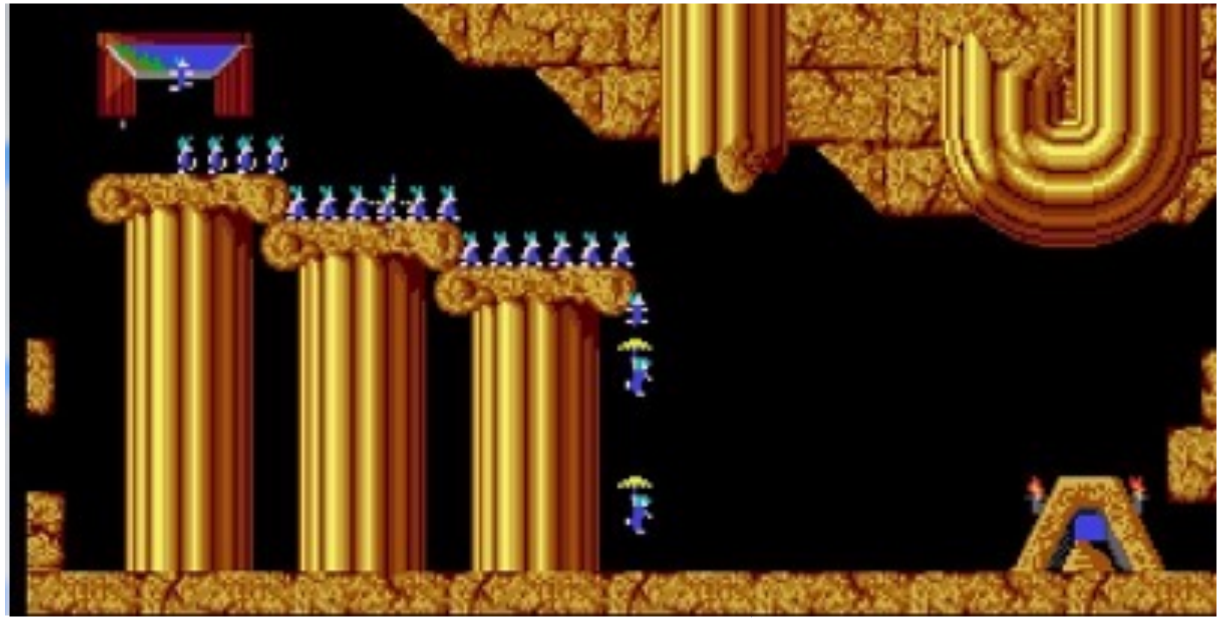
Description : every user is required to signup in order to log into the system, the user is required tc create an account, which can be done either through a sign up form, or through a social account.

Basic Flow :

1. User lands on home page.
2. User clicks on sign up button
3. User is redirected to sign up form.
4. User submit all the details for the signup.
5. The account is created on the Roots system.
6. An email is sent to the user to validate.
7. The user clicks on the link in the email and is now able to log into Roots

Alternative Flow :

4. User clicks on FB or Google SignUp
5. Is redirected to Fb for validation
6. User can now log into Roots



Edge Case

Go through the story, can you think of anything that you may have missed out? It could be very small, but it can happen.

An edge case could be something as simple as having two users who **share the same name, last name**

Don’t forget about the edge cases

Domain diagrams

- From the Use Case we could infer the presence of the USER, in this case we can start creating our USER Model or User Class. What data is required for the creation of this user?
- How is the user digitally represented?
- Developer Tip : Don’t get too attached to this model, it will likely change in the future, once you find other specifications.

CLASS MODEL: USER

Name : string

Last Name : String

DOB : date

Email : string

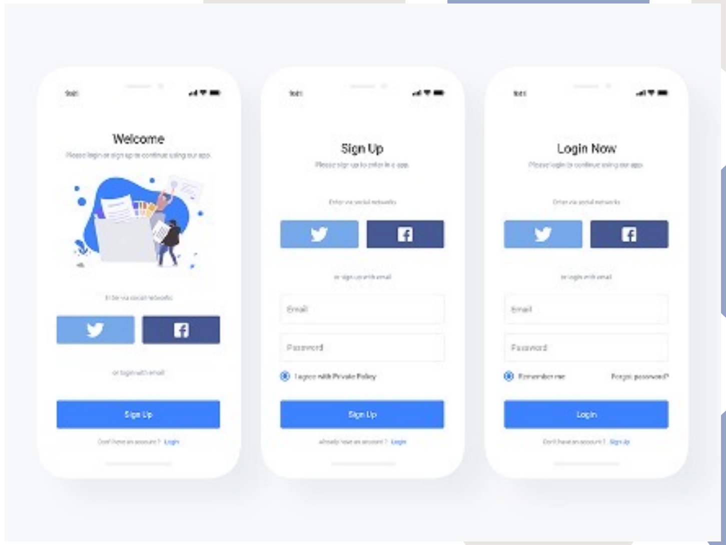
Verified : Boolean

As you can see, most of these are easy to get, but look at Verified, this is not straight forward because it is the result of the application behavior.

Validation

- The simplest one is the paper mockup.
- Programs like Adobe XD (Free) can help you achieve this easily.
- Coding is the last thing you are supposed to do.

- In this class, you are meant to validate your design first, and then implement it. Programming bugs can be because of knowledge, design bugs are a sign of untidiness.



SRS

- An SRS may be written by
- **Customer**
  - Is a call for proposals
  - Must be general enough to yield a good selection of bids and specific enough to exclude unreasonable ones.
- **Bidders**
  - Is a proposal to implement a system
  - Must be specific enough to demonstrate feasibility and competence
- **Developers**
  - Reflects the developer’s understanding of the customer needs
  - Forms the basis of evaluation of contractual performance.

SRS - Structure

1. **Introduction**
2. **Overall Description**
3. **Specific Requirements**
4. **Supporting Information**

Check out the sample SRS for a hypothetical streaming service 'StreamFlix' given in the unit site

References:
https://web.cs.dal.ca/~arc/teaching/CS3130/Templates/SRS%20and%20Project%20Plan%20Templates/SRS-template1.doc

https://github.com/jam01/SRS-Template

No single universally standard SRS template is. But in general, templates based on IEEE Std 830 and ISO/IEC/IEEE 29148:2018 are widely accepted

SRS - Structure

1. Introduction

- 1.1 Document Purpose: <*Describe the purpose of the SRS (identify the product whose software requirements are specified in this document) and the intended audience of this document.*>
- 1.2 Product Scope: <*Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals.*>
- 1.3 Document Overview: <*Describe what the rest of the document contains and how it is organized.*>
- 1.4 Definitions, Acronyms and Abbreviations: <*Define all the terms necessary to properly interpret the SRS*>

SRS - Structure

2. Overall Description

- 2.1 Product Perspective: <*How it fits into existing systems.*>
- 2.2 Product Functions: <*High-level system features.*>
- 2.3 User Characteristics: <*Who are the users.*>
- 2.4 Constraints: < *operating environment, limitations).*>
- 2.5 Assumptions and Dependencies: <*Dependencies on external systems.*>

different user types, their expected technical proficiency

SRS - Structure

Specify all the software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements, and to enable testers to test that the software system satisfies those requirements.

3. Specific Requirements

- 3.1 External Interfaces: *< User Interfaces, hardware & software interfaces>*
- 3.2 Functional Requirements: *<Detailed system functionalities>*
- 3.3 Non-Functional Requirements: *<Performance, security, usability, etc.>*

SRS - Structure

4. Supporting Information

- System architecture diagrams: *<high-level system architecture diagram, deployment architecture diagram, sequence diagrams>*
- Use cases, Use case diagrams
- API documentation (for third-party integrations).
- Legal compliance
- References

*** May not have all the above and may have other information not given above**

SRS- Cheat sheet

Requirements Specs have several purposes:

- Communication
- Contractual
- Basis for Verification
- Basis for Change Control

Requirements Specs have several audiences:

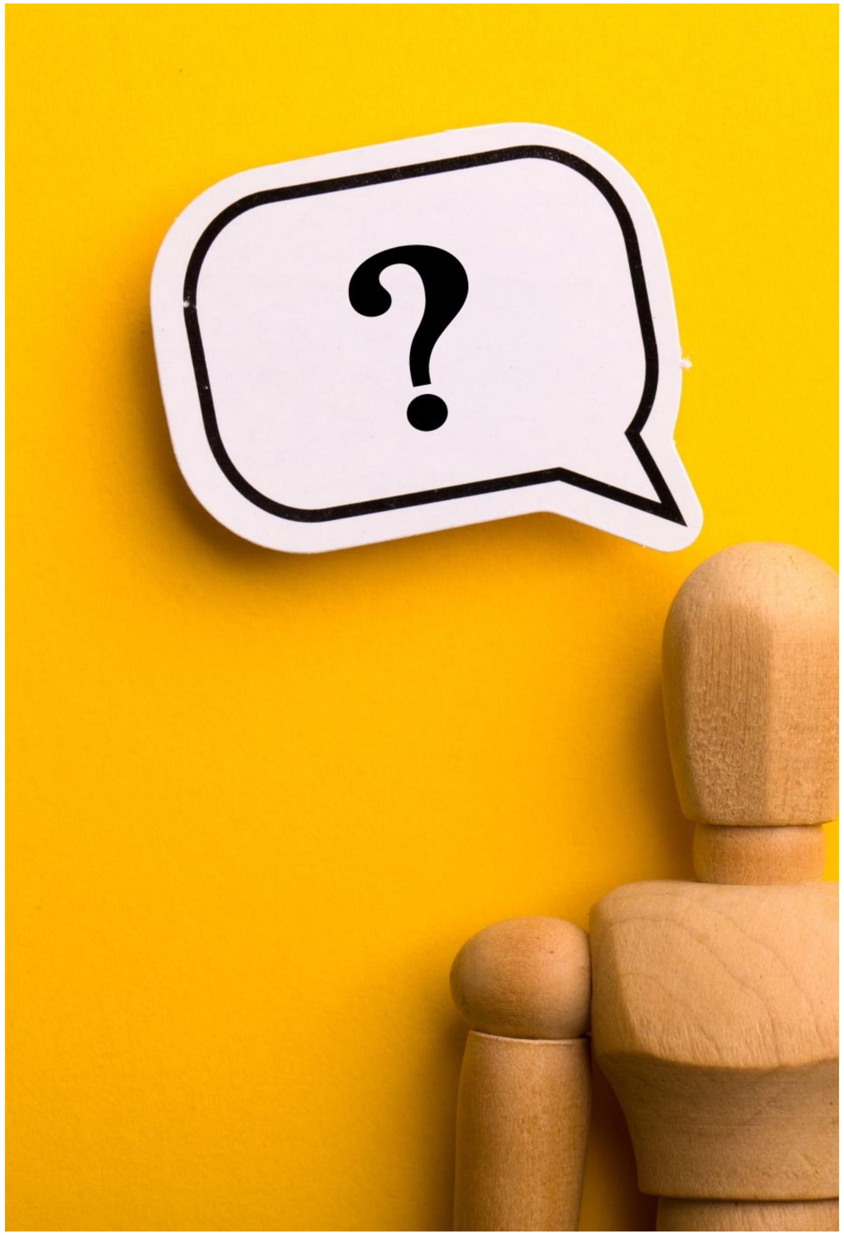
- Technical and non-technical

Good Specs are hard to write

- Complete, consistent, valid, unambiguous, verifiable, modifiable, traceable

Project needs vary

- The amount of effort put into getting the spec right should depend on the possible consequences of requirements errors



Thank you!

Questions?