Software Engineering

Plan

- SRS
- Requirement Engineering
- Requirement Gathering
- User Story & Use Cases

1 SRS

“A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development”

SRS

A Sequence Diagram fully describes what the software will do and how it will be expected to perform.

Benefits:

- It provides feedback to the customer.
- It decomposes the problem into component parts.
- It serves as an input to the design specification.
- It serves as the parent document

Includes:

- Interfaces
- Functional Capabilities
- Performance Levels
- Data Structures/Elements
- Safety
- Reliability
- Security/Privacy
- Quality
- Constraints and Limitations

SRS Structure

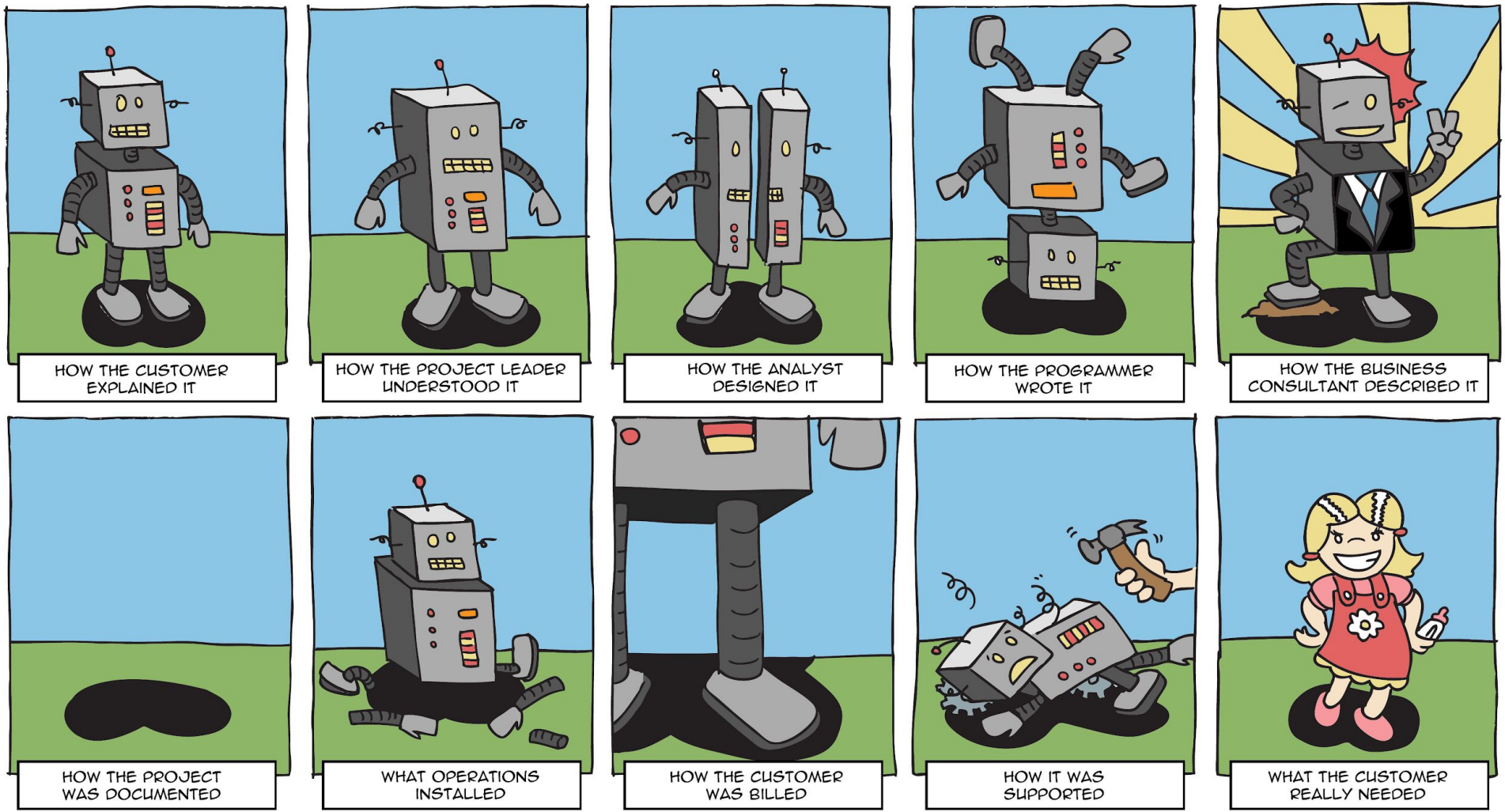
1. Introduction
   - 1.1. Purpose
   - 1.2. Document conventions
   - 1.3. Intended audience
   - 1.4. Additional information
   - 1.5. Contact information/SRS team members
   - 1.6. References
2. Overall Description
   - 2.1. Product perspective
   - 2.2. Product functions
   - 2.3. User classes and characteristics
   - 2.4. Operating environment
   - 2.5. User environment
   - 2.6. Design/implementation constraints
   - 2.7. Assumptions and dependencies
3. External Interface Requirements
   - 3.1. User interfaces
   - 3.2. Hardware interfaces
   - 3.3 Software interfaces
   - 3.4 Communication protocols and interfaces
4. System Features
   - 4.1. System feature A
     - 4.1.1. Description and priority
     - 4.1.2. Action/result
     - 4.1.3. Functional requirements
   - 4.2. System feature B
5. Other Nonfunctional Requirements
   - 5.1. Performance requirements
   - 5.2. Safety requirements
   - 5.3. Security requirements
   - 5.4. Software quality attributes
   - 5.5. Project documentation
   - 5.6. User documentation
6. 6. Other Requirements

Appendix A: Terminology/Glossary/Definitions list

Appendix B: To be determined

Requirement Engineering

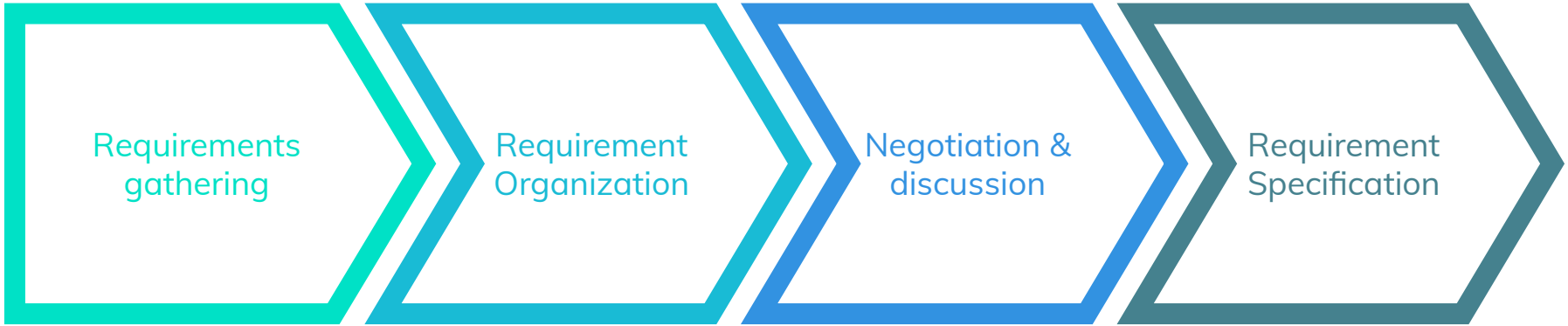
“Requirements engineering (RE) refers to the process of defining, documenting and maintaining requirements”


HOW THE CUSTOMER EXPLAINED IT
HOW THE PROJECT LEADER UNDERSTOOD IT
HOW THE ANALYST DESIGNED IT
HOW THE PROGRAMMER WROTE IT
HOW THE BUSINESS CONSULTANT DESCRIBED IT
HOW THE PROJECT WAS DOCUMENTED
WHAT OPERATIONS INSTALLED
HOW THE CUSTOMER WAS BILLED
HOW IT WAS SUPPORTED
WHAT THE CUSTOMER REALLY NEEDED

RE Process

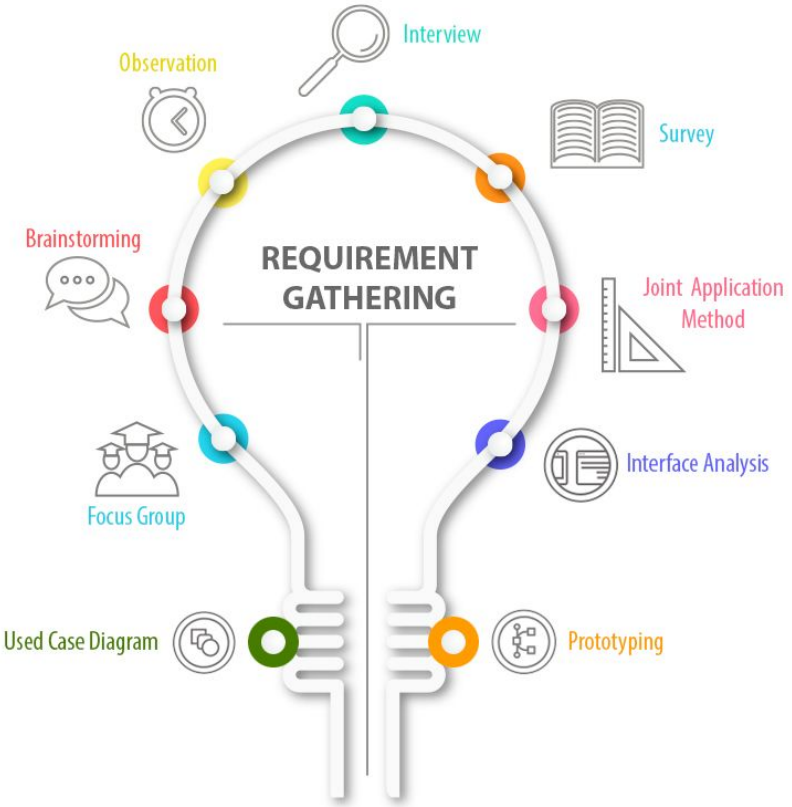
Requirement Gathering

Requirement Gathering (Elicitation)



RG Techniques

- Interviews
- Surveys
- Questionnaires
- Task analysis
- Domain analysis
- Brainstorming
- Prototyping
- Observation



RG Characteristics

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

Software Requirements

Functional, defines what a system is supposed to do.

Example:

- User should be able to mail any report to management.
- Users can be divided into groups and groups can be given separate rights.

Non-Functional, defines how a system is supposed to be.

- Security
- Logging
- Storage
- Configuration
- Performance
- Cost
- Interoperability
- Flexibility
- Disaster recovery
- Accessibility

Software Requirements

Must Have

Software cannot be said operational without them.

Should Have

Enhancing the functionality of software.

Could Have

Software can still properly function with these requirements.

Wish List

These requirements do not map to any objectives of software.

User Story & Use Cases

“Goal is to structure your work: from the largest objectives down to the minute details.”

Use Cases

Use cases are used to capture user (actor) point of view while describing the functional requirements of the system. They describe the step by step process a user goes through to complete that goal using a software system. Use cases are generally divided into:

- Name
- Summary
- Rational
- Users
- Pre-conditions
- Basic Course of Events
- Alternative Paths
- Postconditions

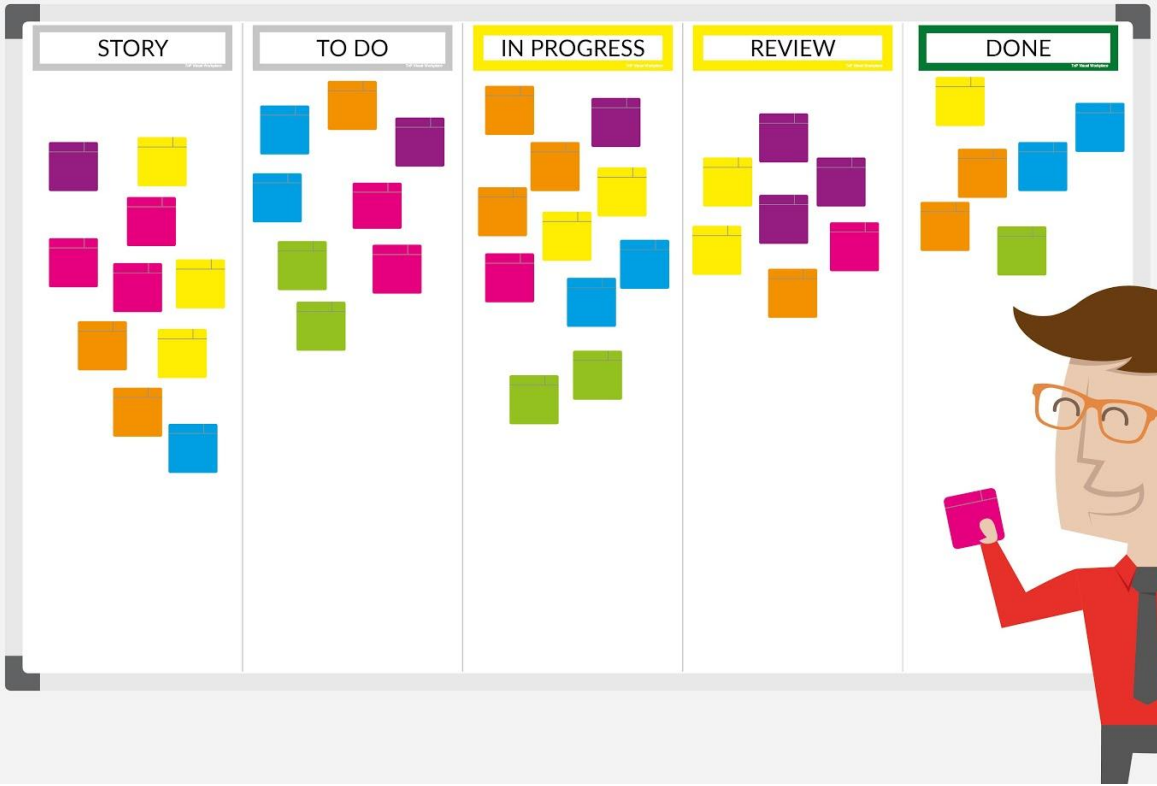
Use Cases

Name	Search and Replace
Summary	All occurrences of a search term are replaced with replacement text.
Rationale	While editing a document, many users find that there is text somewhere in the file being edited that needs to be replaced, but searching for it manually by looking through the entire document is time-consuming and ineffective. The search-and-replace function allows the user to find it automatically and replace it with specified text. Sometimes this term is repeated in many places and needs to be replaced. At other times, only the first occurrence should be replaced. The user may also wish to simply find the location of that text without replacing it.
Users	All users
Preconditions	A document is loaded and being edited.

Use Cases Cont.

Basic Course of Events	1. The user indicates that the software is to perform a search-and-replace in the document. 2. The software responds by requesting the search term and the replacement text. 3. The user inputs the search term and replacement text and indicates that all occurrences are to be replaced. 4. The software replaces all occurrences of the search term with the replacement text.
Alternative Paths	1. In Step 3, the user indicates that only the first occurrence is to be replaced. In this case, the software finds the first occurrence of the search term in the document being edited and replaces it with the replacement text. The postcondition state is identical, except only the first occurrence is replaced, and the replacement text is highlighted. 2. In Step 3, the user indicates that the software is only to search and not replace, and does not specify replacement text. In this case, the software highlights the first occurrence of the search term and the use case ends. 3. The user may decide to abort the search-and-replace operation at any time during Steps 1, 2, or 3. In this case, the software returns to the precondition state.
Postconditions	All occurrences of the search term have been replaced with the replacement text.

The scrum board



User Stories

User stories are simple, clear, brief descriptions of functionality that will be valuable to either a user or purchaser of a product. User stories should be:

- **Independent** - Dependencies between stories lead to prioritization and planning problems.
- **Negotiable** - they are not written story cards are short!
- **Valuable** - each story must bring some business value.
- **Estimable** - each story must have an estimated time & cost.
- **Small** - to estimate and track progress within a sprint.
- **Testable** -must be a criteria of “done”

User Stories

User Story ID	As a <type of user>	I want to <perform some task>	So that i can <achieve some goal>
1	All	Replace all occurrences of a word in a document	Correct my spelling mistake of a word
2	Project manager	View a status report from each team member	Ensure the project stays on track
3	Employee	Be reminded of upcoming deadlines	Complete my tasks on time
4	Director	See the big picture view of department work	Stay

User Stories

- **Stories** - are short requirements or requests written from the perspective of an end user.
- **Epics** - are large bodies of work that can be broken down into a number of smaller stories.
- **Initiatives** - are collections of epics that drive toward a common goal.
- **Themes** - are large focus areas that span the organization.

Thanks!

Any questions?

References

- http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf
- https://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/
- http://asingh.com.np/blog/ieee-srs-recommendations/
- https://techwhirl.com/writing-software-requirements-specifications/
- https://www.versionone.com/agile-101/
- https://www.atlassian.com/agile/project-management/epics-stories-themes