</talentlabs>

CHAPTER 4

Software Project Management

</talentlabs>

AGENDA

- Software Development Lifecycle (SDLC)
- Waterfall Methodologies
- Agile Methodologies
- Writing Business Requirements Document

talentlabs

Software Development Lifecycle (SDLC)

</talentlabs>

Comparing Software Projects with Non-software Projects

	Software Project	Other Project
Requirements	Not clear	Relative clear
Outcome Requirements	Strict, accurate	Relatively loose
Cost of re-do	High	Relatively low
Task Distribution	Hard to Split, a lot of dependencies	Relatively easy
Communication Skills	Hard, as most of the stakeholders have no technical background	Hard, but everyone can speak the same language
Delivery	Complicated, involved comprehensive testing and deployment design	Complicated, but most are about coordination between people

</talentlabs>

Quick Reflection

How do you manage your non-software projects?

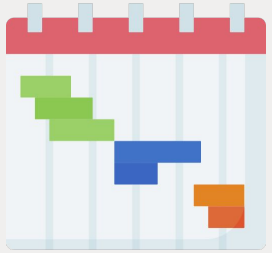
- *What are the tools that you use?*
- *What are the steps that you follow, starting from receiving the project?*
- *How do you split the work and distribute to your team?*
- *How do you know you have finished the project?*

</talentlabs>

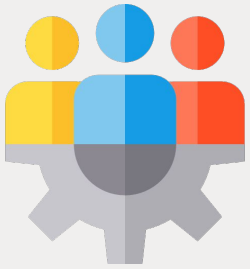
6 Key Challenges of Software Project Management

Communications with non-IT owners

Project scope and requirements definition



Timeline



Division of Labor

Testing & Quality Assurance

Launch and Deployment

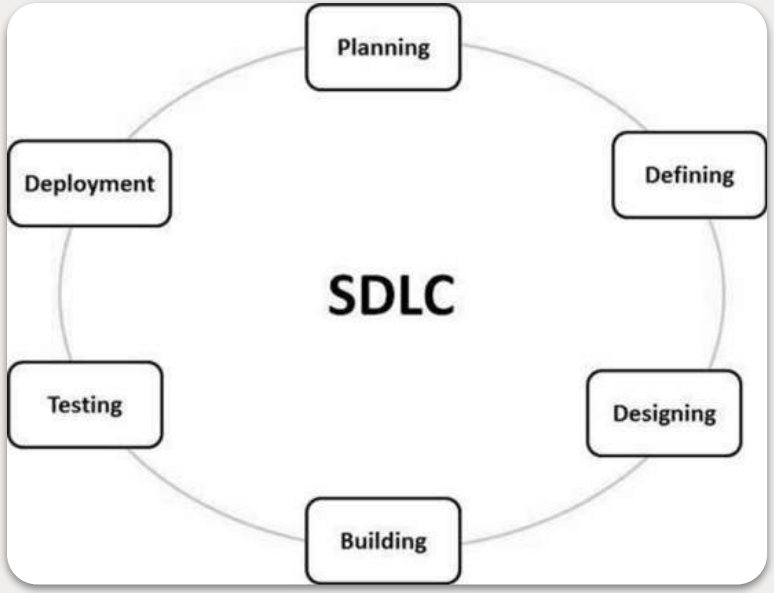
</talentlabs>

Software Development Lifecycle (SDLC)

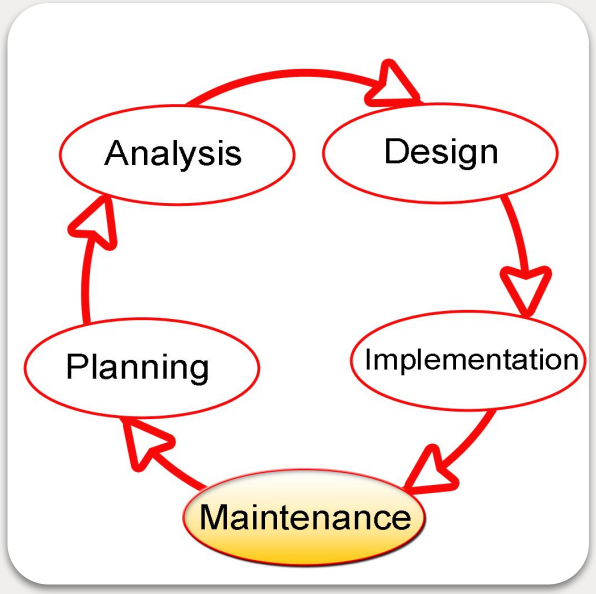
- A framework for you to plan and think about your software development project
- **No formal definition** (5-steps, 6-steps or 7-steps)
- Every company and team and modify it based on your business need

</talentlabs>

Common Versions of SDLC



Source: Tutorialspoint



Source: Wikipedia

Source: Kean University

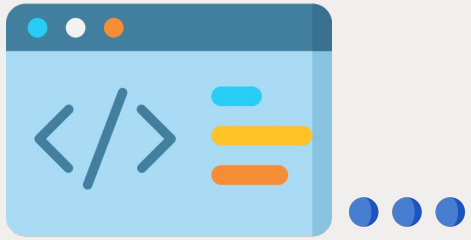
</talentlabs>

SDLC Features

01

Infinite Cycle

02



Coding is only a small part

03

Emphasis on business requirements and analysis

</talentlabs>

SDLC Feature 1 - Infinite Cycle

Software projects only ends when

- replacing with a different system
- system retirement or deprecation

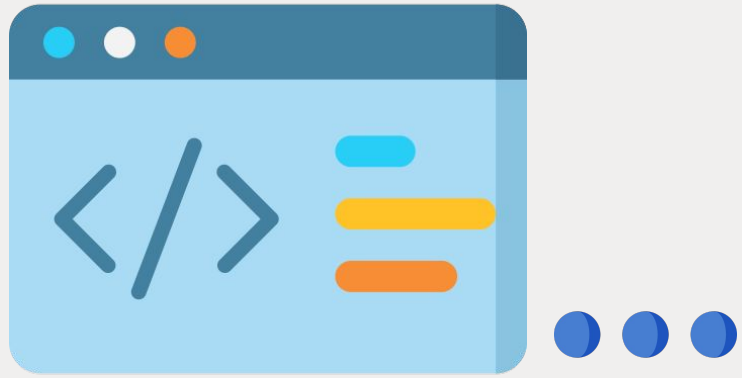
Otherwise, software teams will need to work on

- new features
- bug fixes
- regular upgrades

SDLC Feature 2 - Coding is only a small part

Implementation (aka coding) is **only one part of the cycle**, other common steps are

- Gather Requirements
- Design
- Testing and Maintenance
- Services after delivery, e.g. on-call support, data migration



</talentlabs>

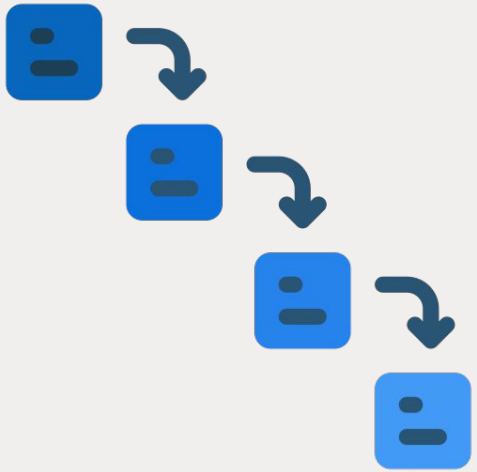
SDLC Feature 3 - Business Requirements and Analysis

Emphasis on **requirement capturing** and **analysis** stage

- Never start by writing code
- Be clear about the requirements and all the details before coding
- Software engineers **need to talk** to product and business owner

</talentlabs>

2 Common Types of SDLC Implementation



Waterfall Development

Agile Development

</talentlabs>

Waterfall Development

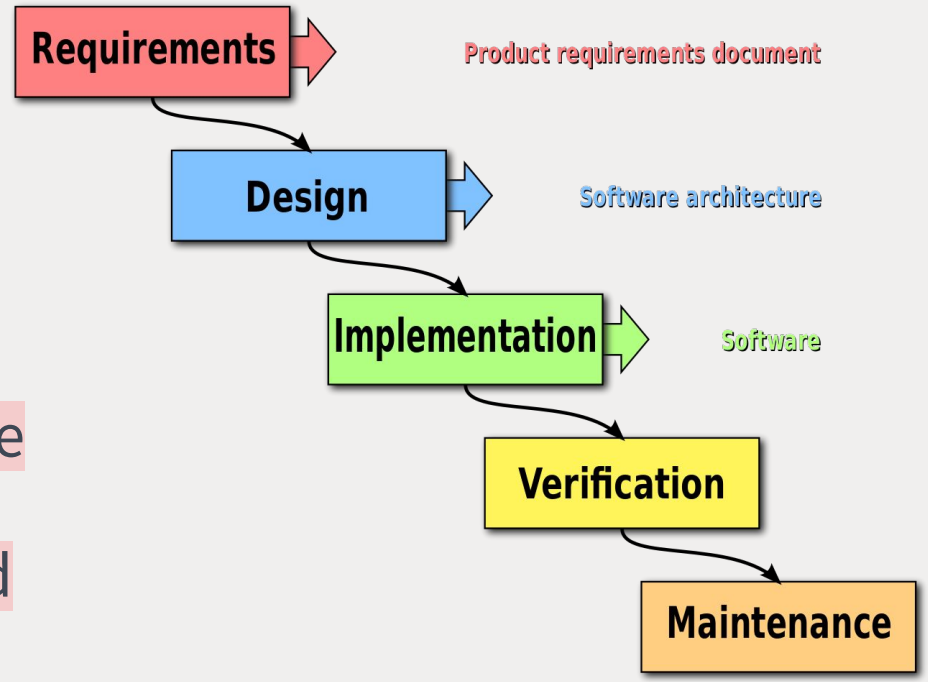
</talentlabs>

Waterfall Model

- One of the most commonly used SDLC model
- Designed back in 1976
- A lot of criticisms in recent years, but still being used by a lot of companies

Key Concepts

- Only proceed to the next step after completed the previous step
- Everything about the project is well-documented (including requirements, design, test cases etc)



</talentlabs>

Waterfall Step 1 - Requirements

Objective

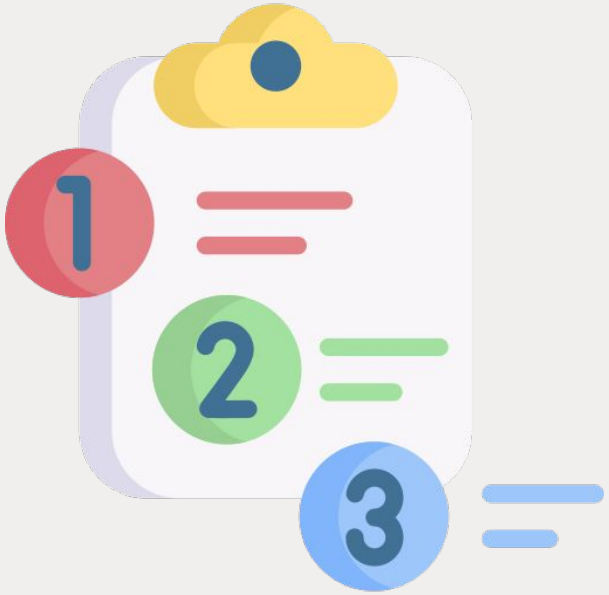
Capturing all the business requirements from the "users" or "business owner"

People Involved

- Business owner
- Project manager
- (Optional) Software engineers

Output

- Product Requirements Document
- Cover the details as much as possible



</talentlabs>

Waterfall Step 2 - Design

Objective

Come up with the system design and architecture, also the technologies to be used

People Involved

- Project Manager
- System architects and/or Senior Engineers

Output

- System architecture and design document

Waterfall Step 3 - Implementation

Objective

Build the system by coding, based on the technical design and architecture documents from step 2

People Involved

- System architects
- Software Engineers

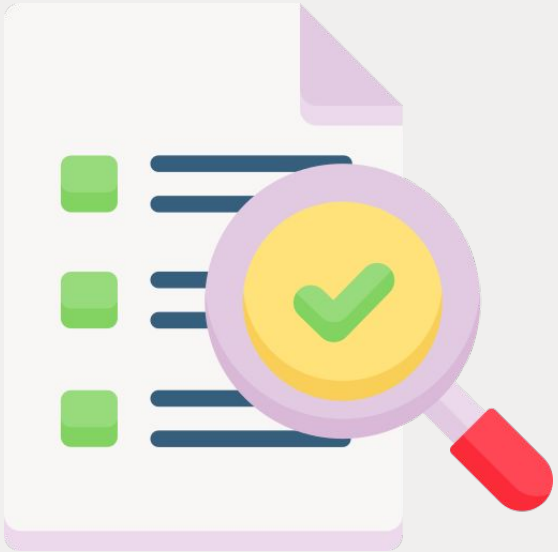
Output

- Software

Waterfall Step 4 - Verification

Objective

Test the software to see if it meets the requirements of the business and users



People Involved

- Testers (e.g. users, QA team, business team)
- Project Manager

Output

- List of bugs and issues to be fixed

Waterfall Step 5 - Maintenance

Objective

- Planning for rolling out the software (e.g. putting up the systems to servers), business rollout plan (e.g marketing, technical support, operations)
- Future support plan (e.g. bug fixes process, on-call support)

People Involved

- Business Team
- Project Manager
- System Architect

Output

- Software launch
- Future support plan

</talentlabs>

Problem with Waterfall Model

Fundamental assumption of Waterfall Model

- All the requirements can be defined clearly at the early stage of the project
- Software project can be managed as a step-by-step process

Reality

- Business or the users are not sure about the requirements
- Continuous input and clarification from business users is required

Agile Methodologies

</talentlabs>

Agile Revolution

- A movement against the “well-planned” software development methodologies (e.g. waterfall model)
- Assuming that no one can “plan” a software completely and perfectly

Plan for short future
(1 week to 1 month max.)

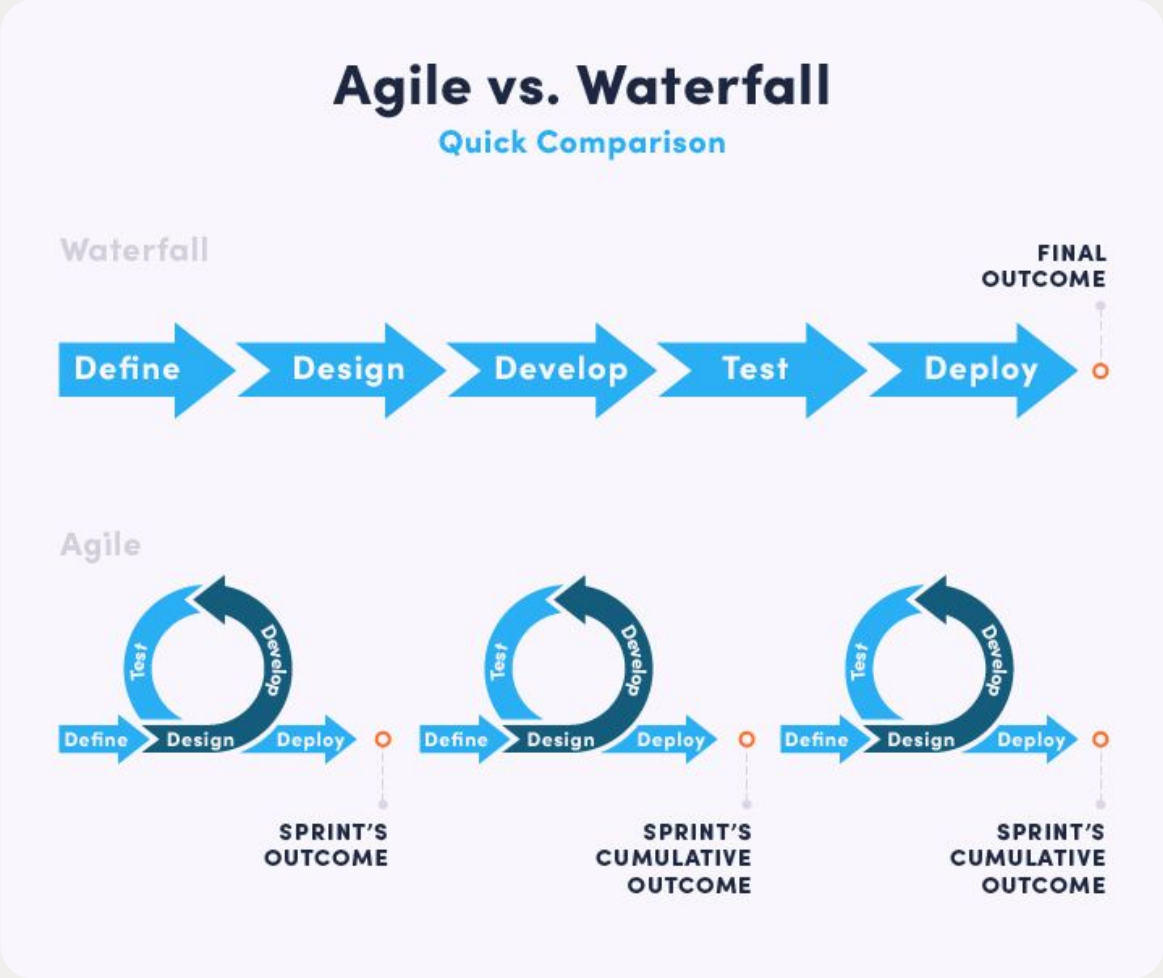
Release new versions frequently
(monthly is min. frequency)

Business & IT team
communicate frequently

</talentlabs>

Agile vs. Waterfall

	Agile Development	Waterfall Development
Release/Update Timeframe	Ranging from daily to monthly	Months
Release Size	Small (could be just small updates like fixing a bug)	Big (with major feature upgrade, a lot of changes)
Documentations and Admin Work	Minimal	A lot of processes and paperwork
Communication with Business	Anytime needed, and at least after every “sprint”/cycle	Before development, and after development (beginning and the end)
Testing	After every sprint	Once after the whole development

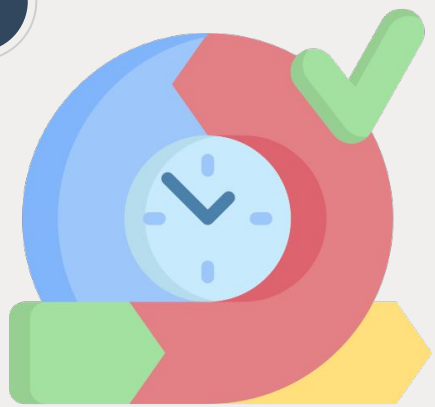


Source: selleo.com

</talentlabs>

Executing Agile Methodologies

01



Define the sprint duration
(1 week to 1 month)

02

Within each sprint, complete all the following tasks:

- Requirements capturing
- Design
- Implementation
- Testing
- Verification with business/users
- Release new version

03

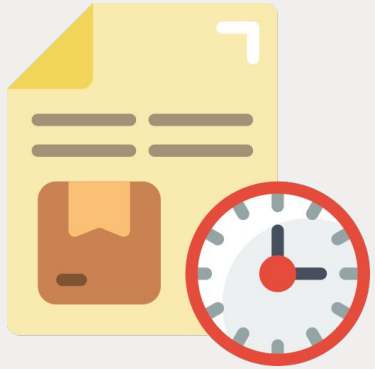
Keep asking business for feedback, to make sure deliverables are correct.

Don’t wait until verification day.

</talentlabs>

So many tasks in a week?

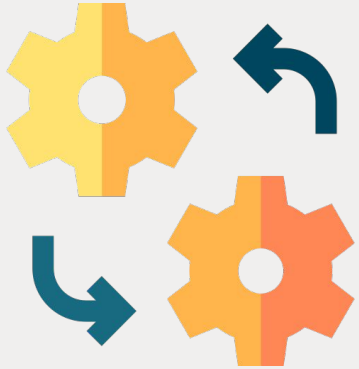
01



Minimise admin work and paperwork.

Less meetings, more working.

02



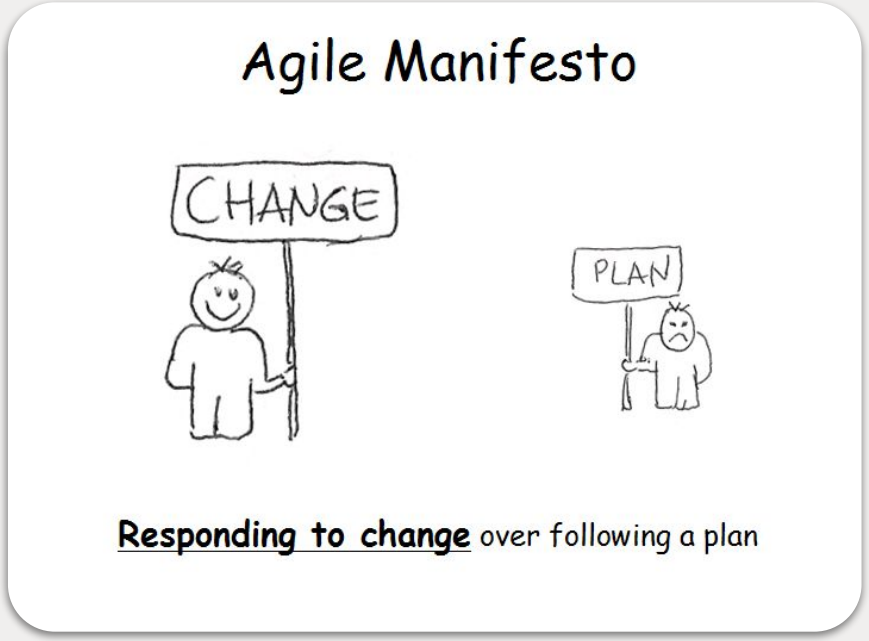
Rely on automation for software testing and deployment

</talentlabs>

Summarizing Agile

Given that things keep changing:

- Agile is a mindset shift from “well-planned” projects into “always-changing” projects
- React quickly to the **market changes** and **business requirements changes**



Source: https://wearecitizensadvice.org.uk/

</talentlabs>

Writing Business Requirements

</talentlabs>

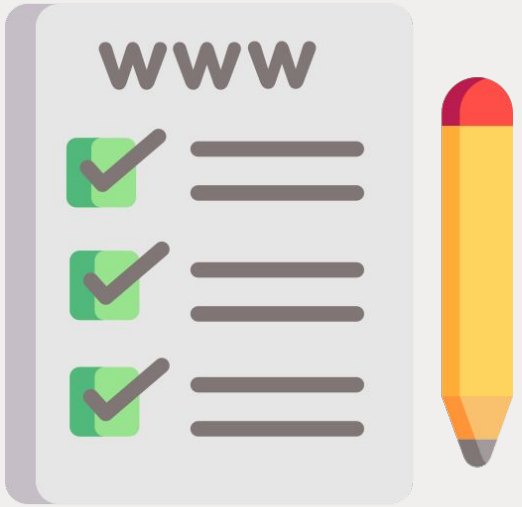
</talentlabs>

Function of Business Requirements Document (BRD)

Describe the high-level business needs

List out what functions need to be included in the software

Make sure all the teams are aligned on the project goal and expectations



</talentlabs>

Essential Parts of a BRD

1. A summary statement
2. Project objectives
3. Project scope
4. Financial statements
5. Functional requirements
6. Schedule & Timeline
7. Assumptions
8. Cost & Benefit

</talentlabs>

Writing Functional Requirements

Business Requirement

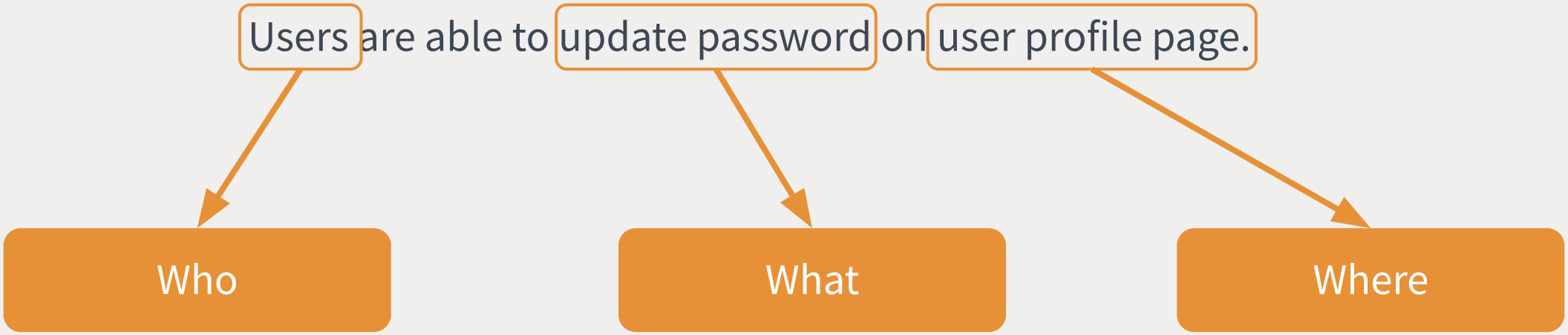
Membership Feature

Functional Requirements

- Users are able to register as member
- Users are able to login and logout
- Users are able to reset password
- Users are able to update password
- Users are able to update personal information

</talentlabs>

A Good Functional Requirement



</talentlabs>