INTRODUCTION TO SOFTWARE ENGINEERING

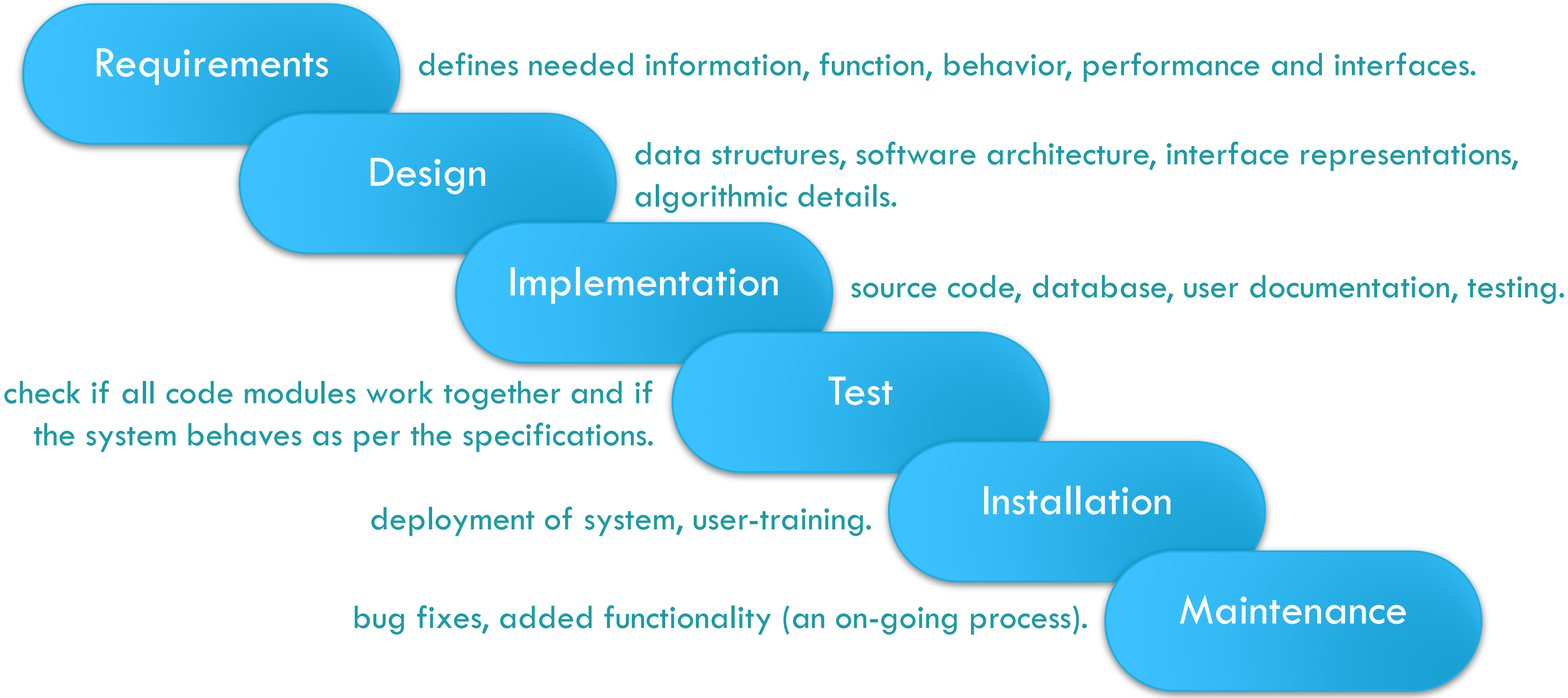
Lesson 02 – Introduction to project management

OUTLINE

1. Waterfall model
2. Agile methodologies (Scrum, Kanban)
3. Comparison and selection criteria

1. WATERFALL MODEL

The flow is waterfall-like from Requirements, Design, implementation, testing, installation, and maintenance of software application.



Requirements: defines needed information, function, behavior, performance and interfaces.

Design: data structures, software architecture, interface representations, algorithmic details.

Implementation: source code, database, user documentation, testing.

Test: check if all code modules work together and if the system behaves as per the specifications.

Installation: deployment of system, user-training.

Maintenance: bug fixes, added functionality (an on-going process).

1.1. REQUIREMENTS

Milestone schedule

No.	Item	Time			
		12	13	14	15
2.1.MB	Begin Lesson 02		◆		
2.1.1.M1	Complete Waterfall model		◇		
2.1.2.M2	Complete Agile methodologies			◇	
2.1.MF	Finish Lesson 02			◇	
2.2.MB	Begin Lab 02				◇
2.2.1.M1	Complete Gitlab manage issues				◇

┆ Start date

╎ Current date

◆ Completed

◇ Planned

1.1. REQUIREMENTS

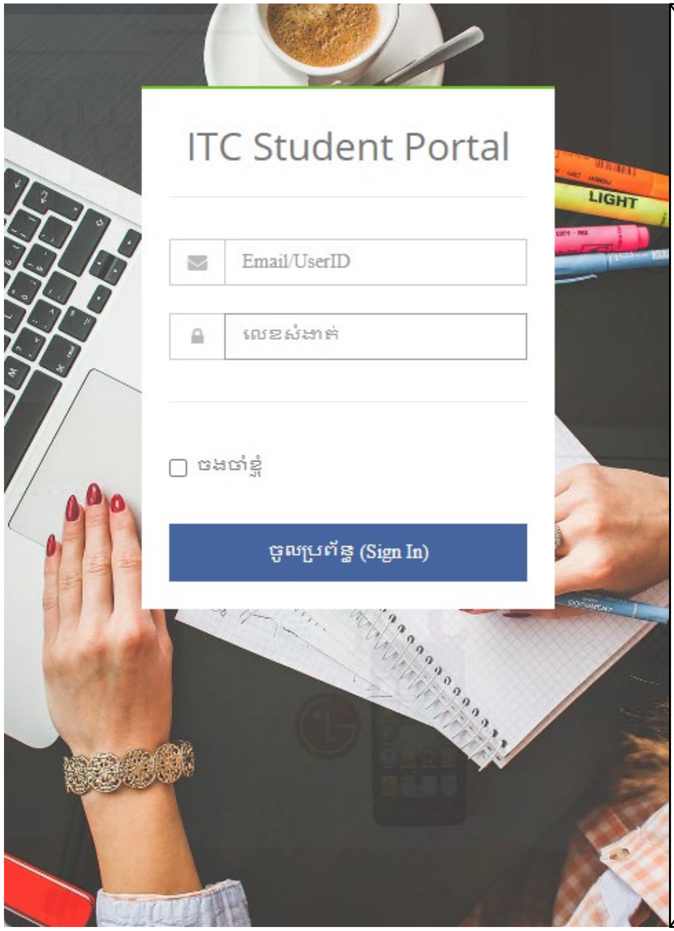
Steps in requirement analysis are:

- **Identify stakeholders**: Identify and prioritize stakeholders, such as clients, end-users, and team members. You can use techniques like interviews, surveys, workshops, or stakeholder mapping.
- **Understand the project goal**: Understand the objectives of the project.
- **Gather requirements**: Collect all the necessary information from stakeholders to create a comprehensive list of what is expected.
- **Elicit requirements**: Actively gather detailed information from stakeholders about what the software needs to do. You can use methods like interviews, questionnaires, or user observation.
- **Analyze requirements**: Determine the quality of the requirements by identifying if they are unclear, incomplete, ambiguous, or contradictory.
- **Prioritize requirements**: Prioritize the requirements.
- **Validate requirements**: Validate the requirements.
- **Obtain stakeholder sign-off**: Obtain sign-off from stakeholders.
- **Document and review requirements**: Document and review the requirements.

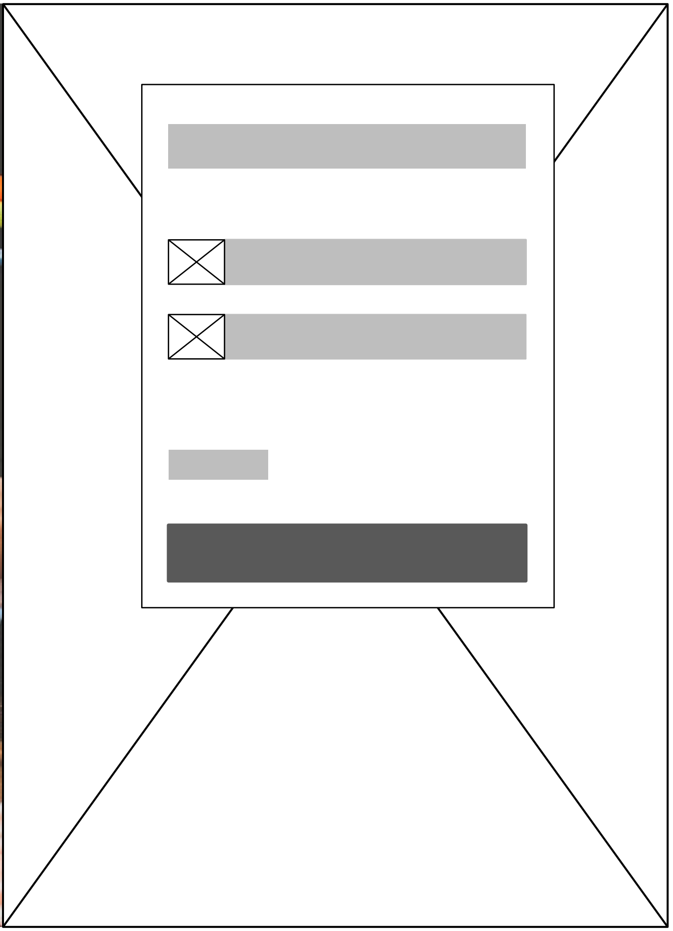
➔ So many steps? We will detail in lesson **requirements elicitation.**

1.2. DESIGN

Layout
Branding
Colors
Typography
Iconography



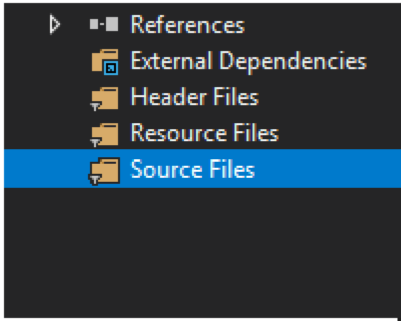
UI



UX

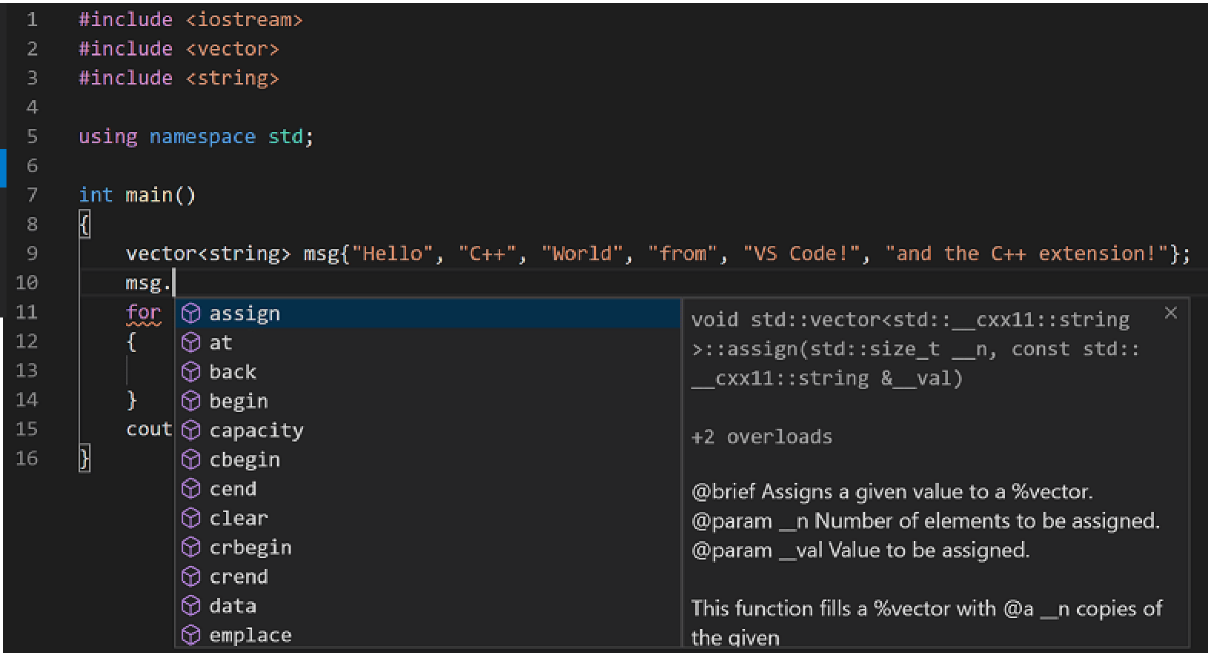
UX Strategy
User personas
Structure
Wireframes
Usability tests

1.3. IMPLEMENTATION



- References
- External Dependencies
- Header Files
- Resource Files
- Source Files

Folder structure



```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
{
    vector<string> msg{"Hello", "C++", "World", "from", "VS Code!", "and the C++ extension!"};
    msg.
    for
    {

    }
    cout
}
```

```
assign
at
back
begin
capacity
cbegin
cend
clear
crbegin
crend
data
emplace
```

```
void std::vector<std::__cxx11::string
>::assign(std::size_t __n, const std::
__cxx11::string &__val)

+2 overloads
```

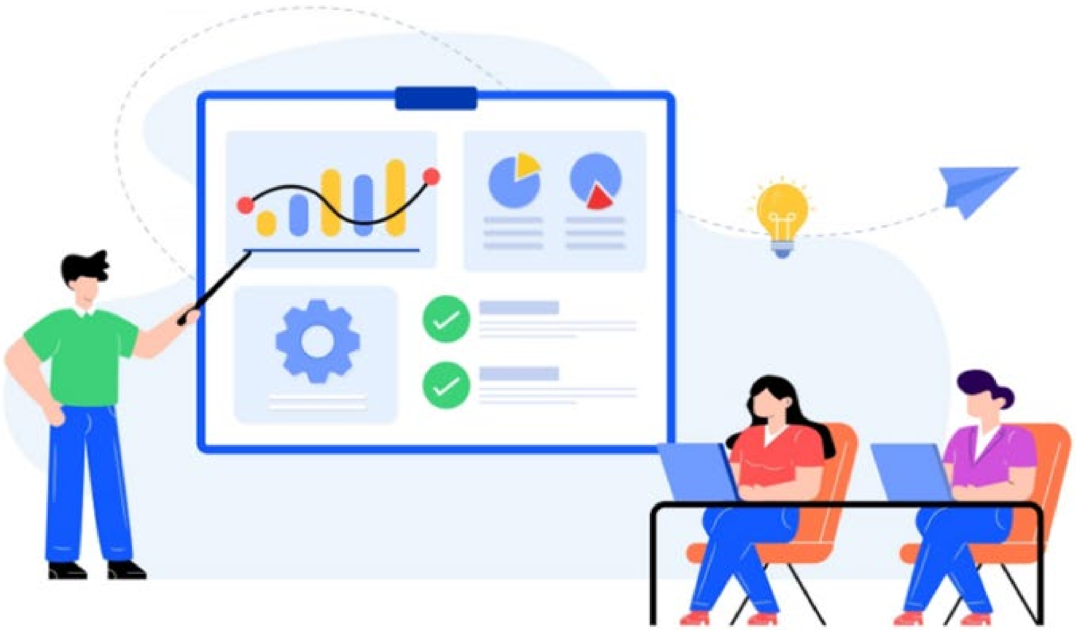
@brief Assigns a given value to a %vector.
@param __n Number of elements to be assigned.
@param __val Value to be assigned.

This function fills a %vector with @a __n copies of the given

Codes

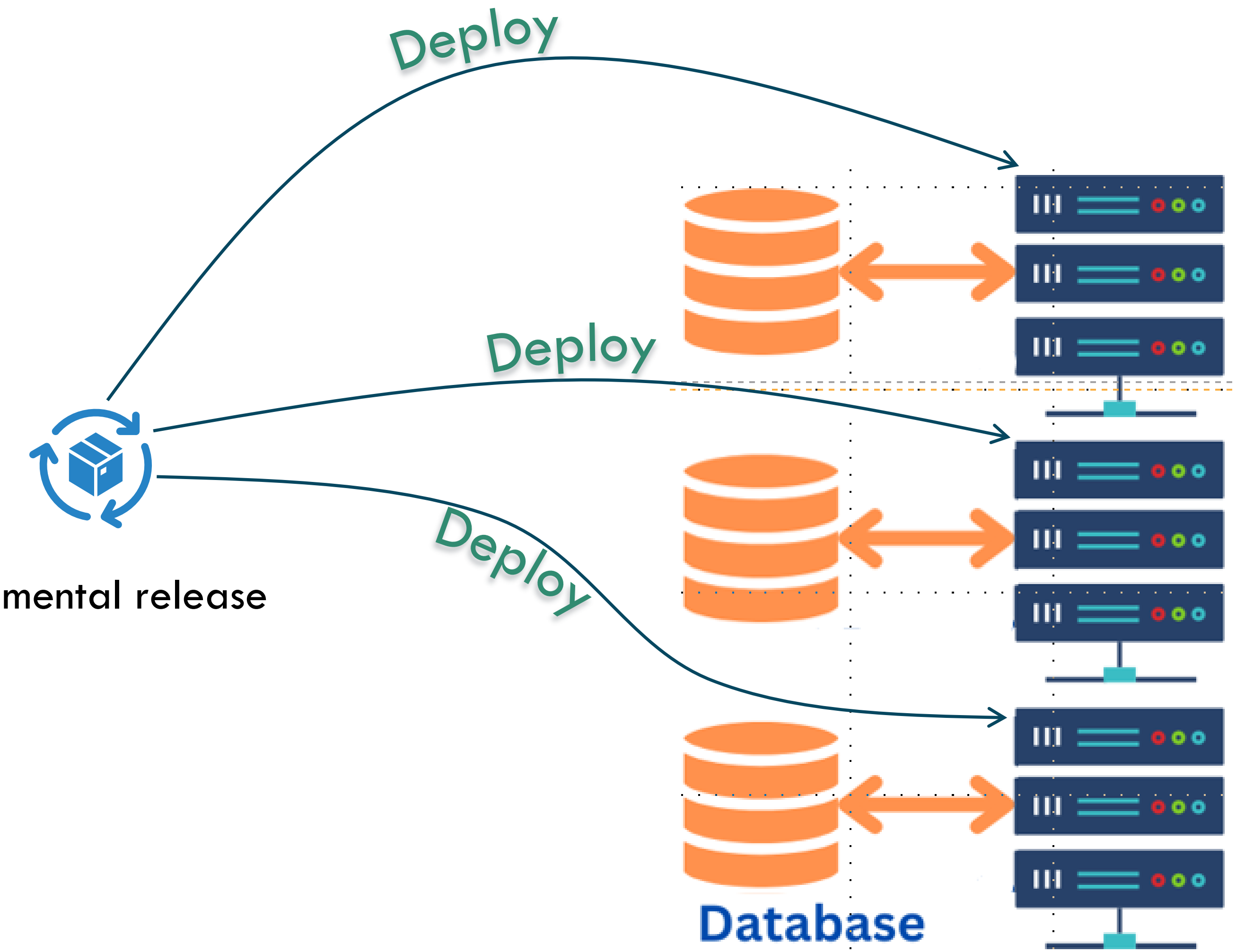
1.4. TEST

Demonstrate the work to customer and get feedbacks for improvement the next increment.

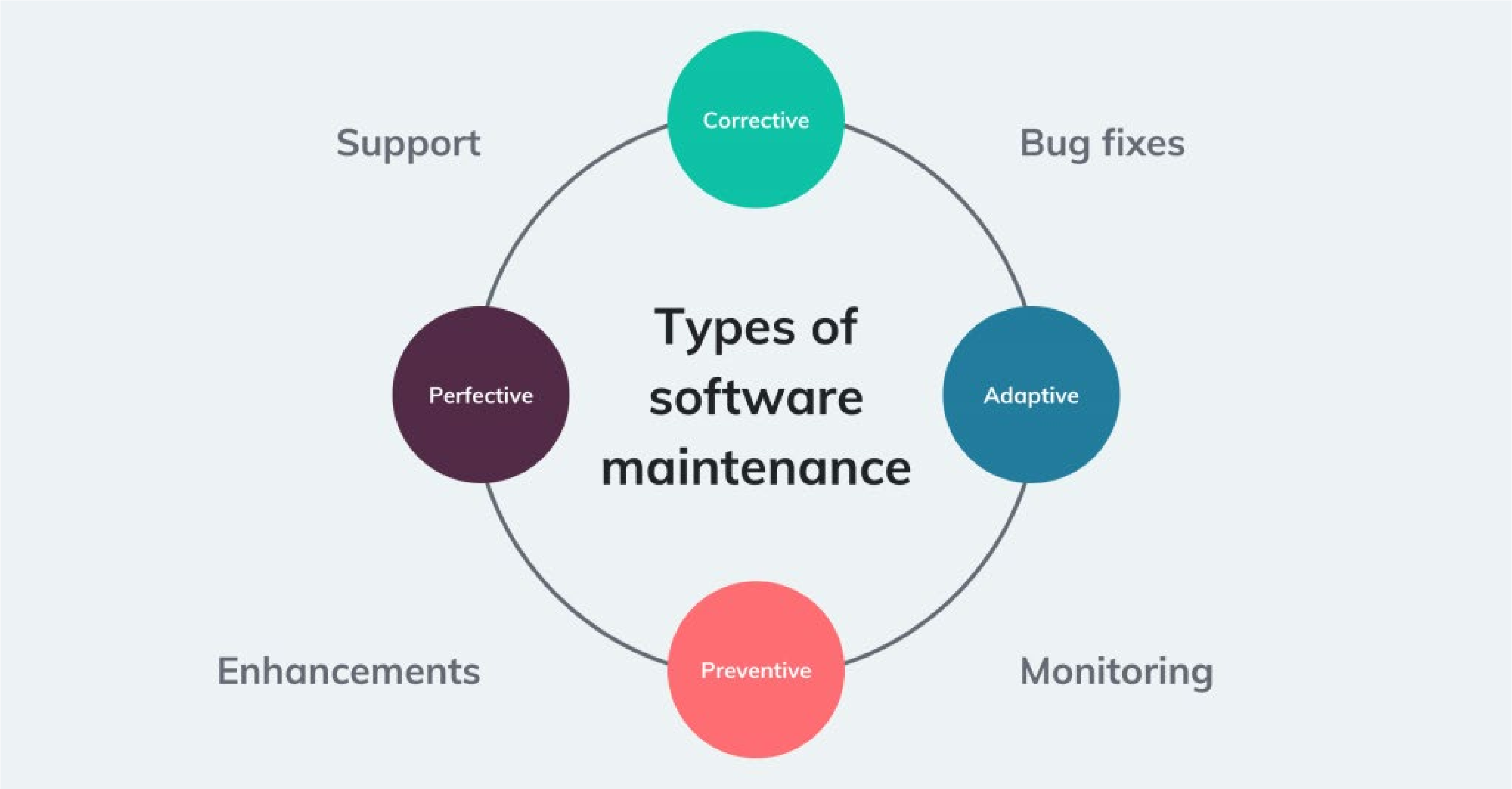


Take notes of what are done, what needs to improve, and what are added features.

1.5. INSTALLATION



1.6. MAINTENANCE



SCHEDULE

Milestone schedule

No.	Item	Time			
		12	13	14	15
2.1.MB	Begin Lesson 02		◆		
2.1.1.M1	Complete Waterfall model		◆		
2.1.2.M2	Complete Agile methodologies			◇	
2.1.MF	Finish Lesson 02			◇	
2.2.MB	Begin Lab 02				◇
2.2.1.M1	Complete Gitlab manage issues				◇

⋮ Start date

¦ Current date

◆ Completed

◇ Planned

2. AGILE METHODOLOGIES (SCRUM, KANBAN)

Agile methodology is a project management framework that breaks projects into phases, called sprints, and uses an iterative approach. After each sprint, teams reflect on what could be improved and adjust their strategy for the next sprint.

1991

Rapid Application Development (RAD)

1994

Unified Process (UP)

1995

Scrum

1996

Extreme Programming (XP)

1997

Feature-Driven Development (FDD)

2.1. TYPES OF AGILE FRAMEWORKS

1. **Kanban**
2. **Scrum**
3. **Lean**
4. **DSDM or Dynamic Systems Development Method ·**
5. **XP or Extreme Programming**
6. **FDD or Feature Driven Development**
7. **Crystal**
8. **Scaled Agile Framework (SAFe)**

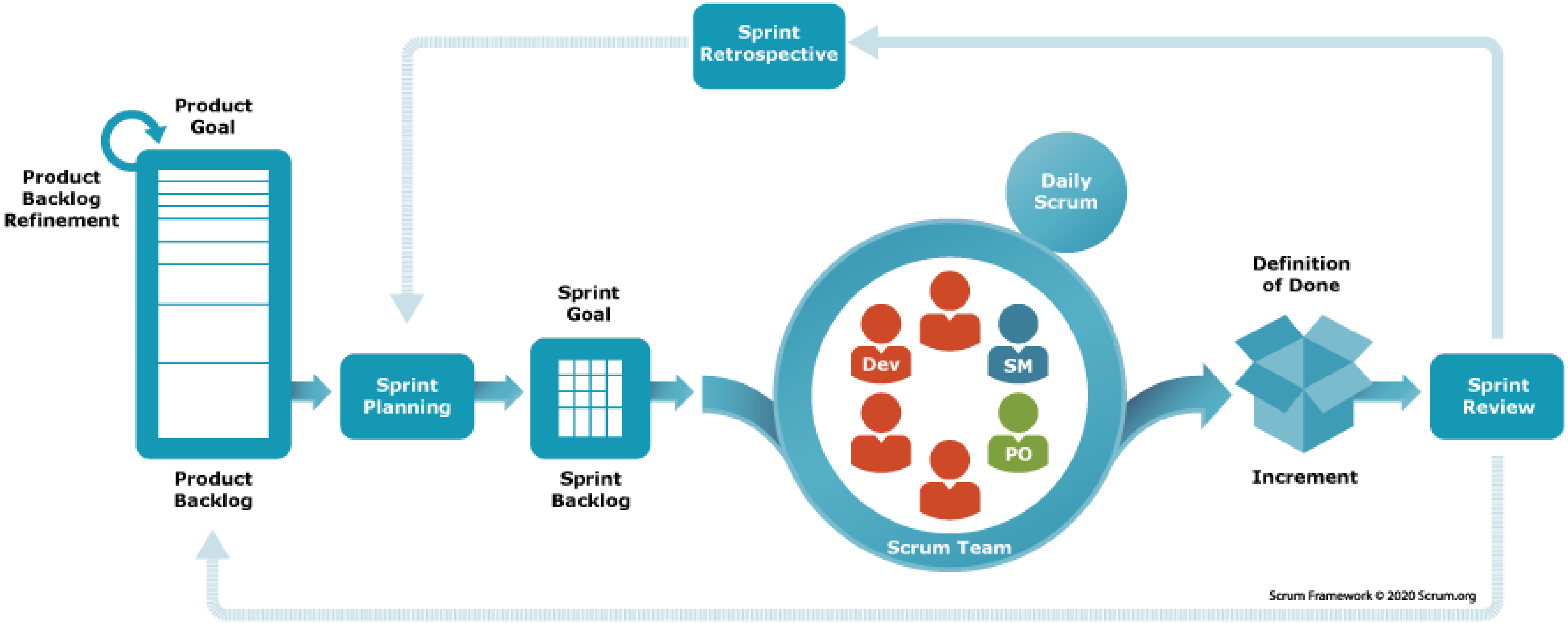
2.2. KANBAN

Kanban is about envisioning the existing workflow in terms of steps. These steps can be created on the whiteboard. The goal of the Kanban execution is to ensure work items move to the next steps quickly to realize business value faster.

Kanban is needed when:

- Dynamic/ frequent changing requirements which need to be delivered faster.
- In case of changing priorities, the team can pull the prioritized work as soon as the Work In Process limit drops.
- Frequent releases are there (Periodically).
- When incoming work is continuous.
- Where task priority needs to be decided dynamically based on task nature and type.

2.3. SCRUM

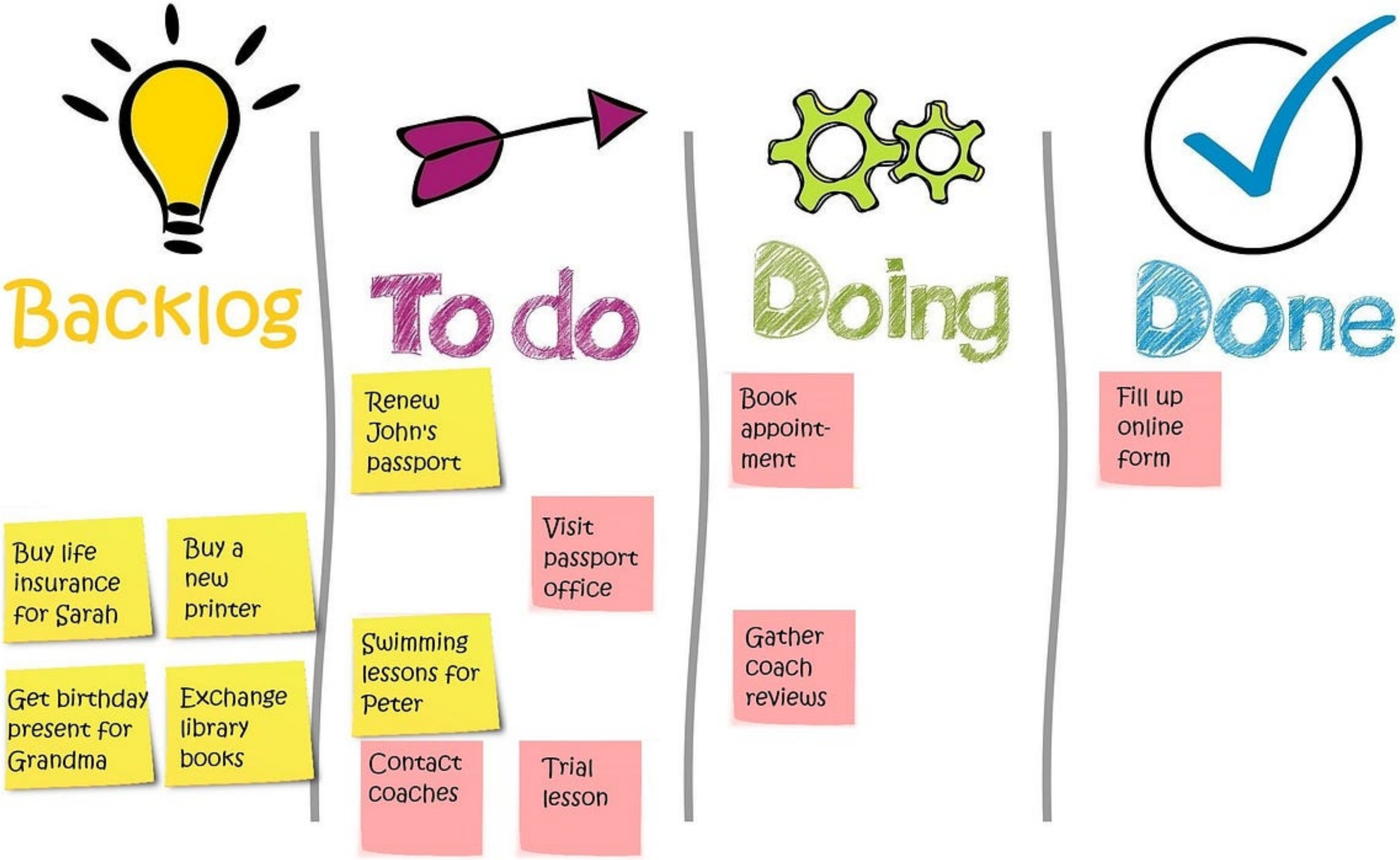


Scrum Framework © 2020 Scrum.org

2.3. SCRUM (TERMINOLOGIES)

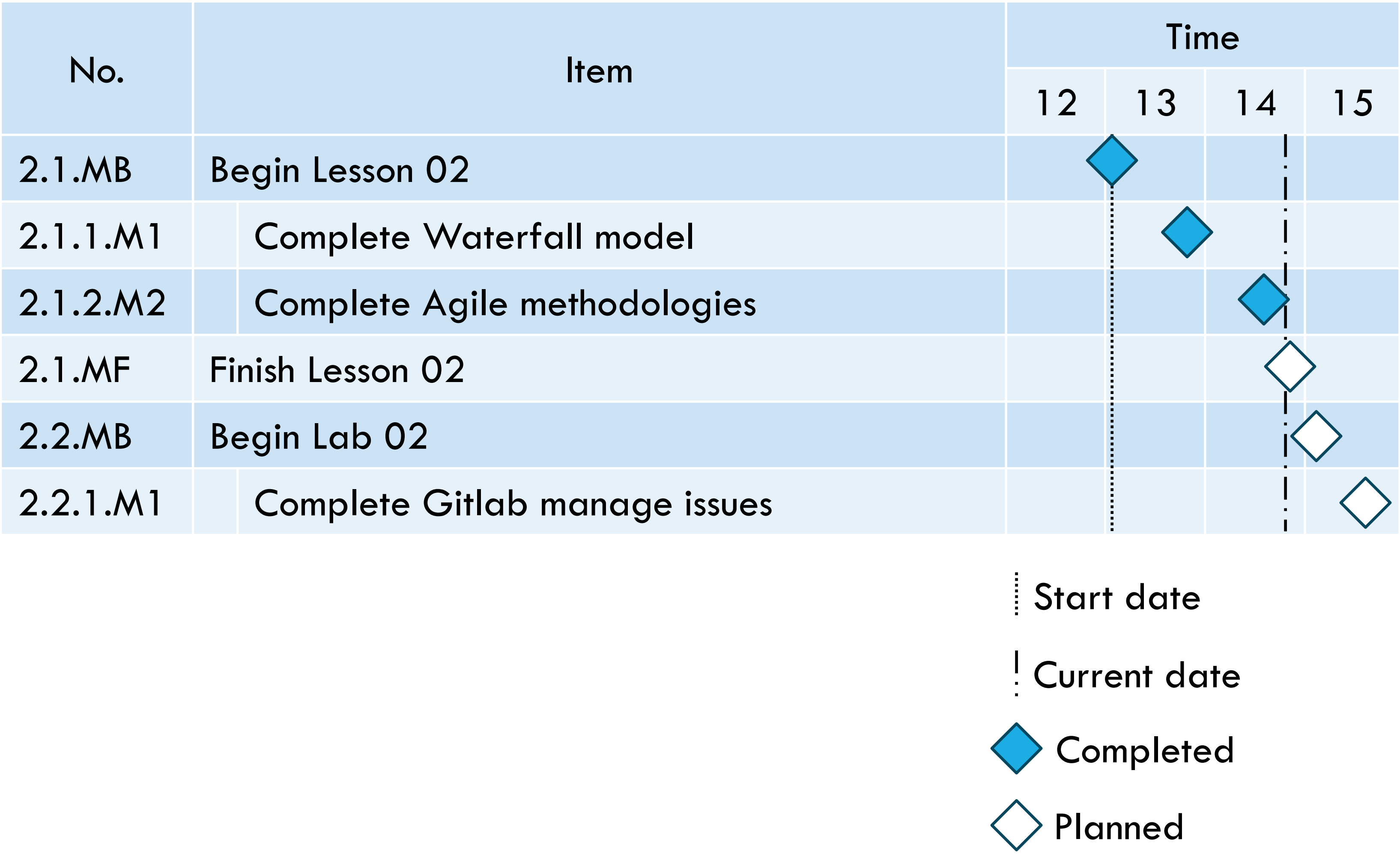
- **Product backlog**: a backlog of work to be completed (high level, valuable). Product backlog is made up of prioritized User Stories.
- **User Story**: statement of user’s need and/or business value.
- **Sprint**: a 1 to 4 week cycle of development of scrum team to deliver fully tested, production ready software.
- **Sprint Planning**: an event in Scrum that scrum team takes the most prioritized stories from Product backlog and work out.
- **Sprint Backlog**: list of user stories in a sprint with status and assignments (usually in a form of “Information Radiator” or “Big board”).
- **Release**: an increment of the product that is intended to be deployed into live use.
- **Daily Scrum**: a short event (5-15 minutes) during which the team share information on their progress and motivate focus on product increment to be delivered at the end of the Sprint.
- **Sprint Review**: an event in Scrum that is a demonstration of working software developed during the Sprint fulfilling Stories that are acknowledged as “Done” by PO.

2.3. SCRUM (BOARD)



SCHEDULE

Milestone schedule



No.	Item	Time			
		12	13	14	15
2.1.MB	Begin Lesson 02		◆		
2.1.1.M1	Complete Waterfall model		◆		
2.1.2.M2	Complete Agile methodologies			◆	
2.1.MF	Finish Lesson 02			◇	
2.2.MB	Begin Lab 02				◇
2.2.1.M1	Complete Gitlab manage issues				◇

┆ Start date

╎ Current date

◆ Completed

◇ Planned

3. COMPARISON AND SELECTION CRITERIA

No.	Kanban	Scrum
1.	Planning, release, and process improvement can have separate cadences.	Iteration is timeboxed.
2.	For planning and process improvement, the lead time is used as the default metric.	For planning and process improvement, Velocity is used as the default metric.
3.	Cross-functional teams are optional.	Cross-functional teams prescribed.
4.	Project Tracking: CFD can be used to understand workflow progress.	Project Tracking: Burndown chart is prescribed.
5.	WIP limited directly (per workflow state).	WIP limited indirectly (per sprint).
6.	Can add new items whenever the WIP limit falls.	Cannot add items to ongoing iteration.