Robotics Corner

Robotics Corner

Software Engineering

01 Definitions in the Software Industry and SDLC

02 Requirements Analysis

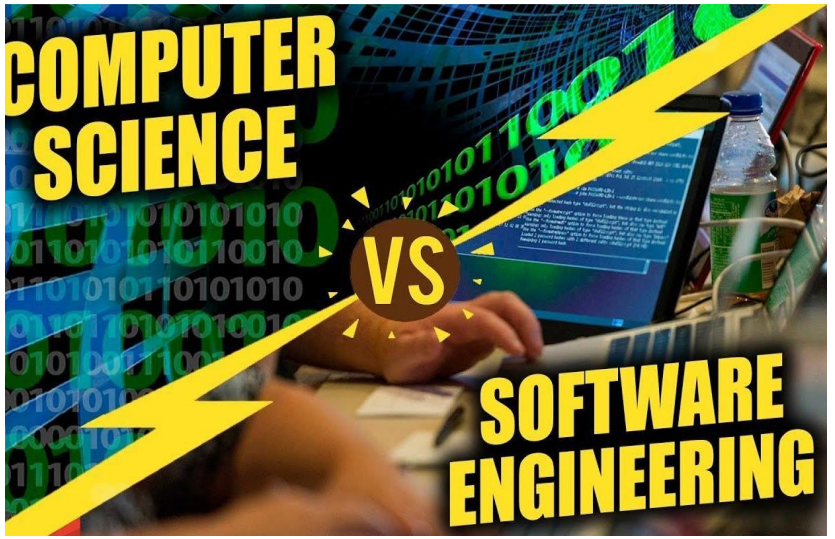
03 Design phase

04 Testing and validation

05 SDLC Process Models

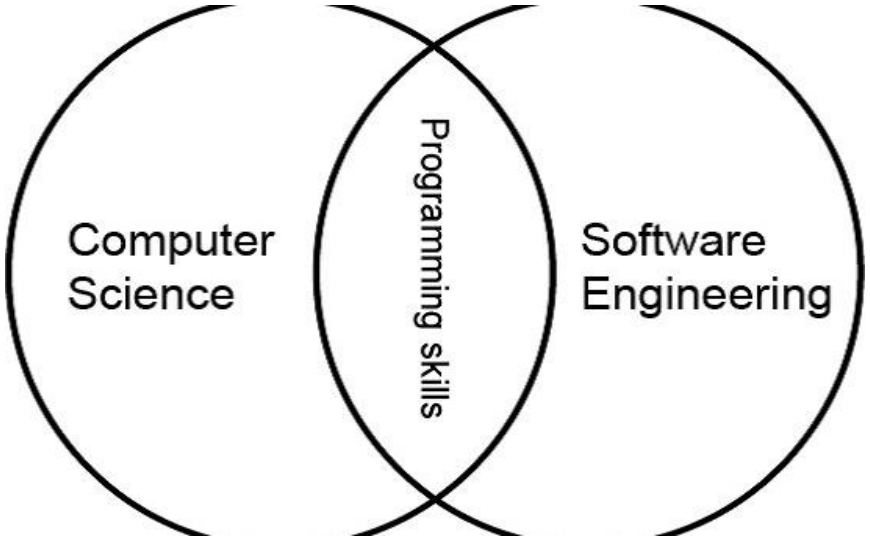
ROBOTICS CORNER

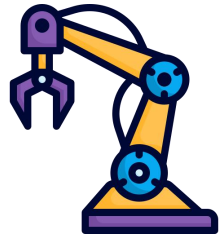
Software Engineering vs Computer Science



- **Computer Science** is concerned with theory
- and
- Fundamentals, while **Software Engineering** is concerned with the practicalities of development and delivering a useful Software

- **Software**: Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed *for a general market.*

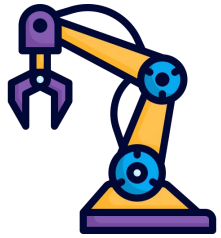




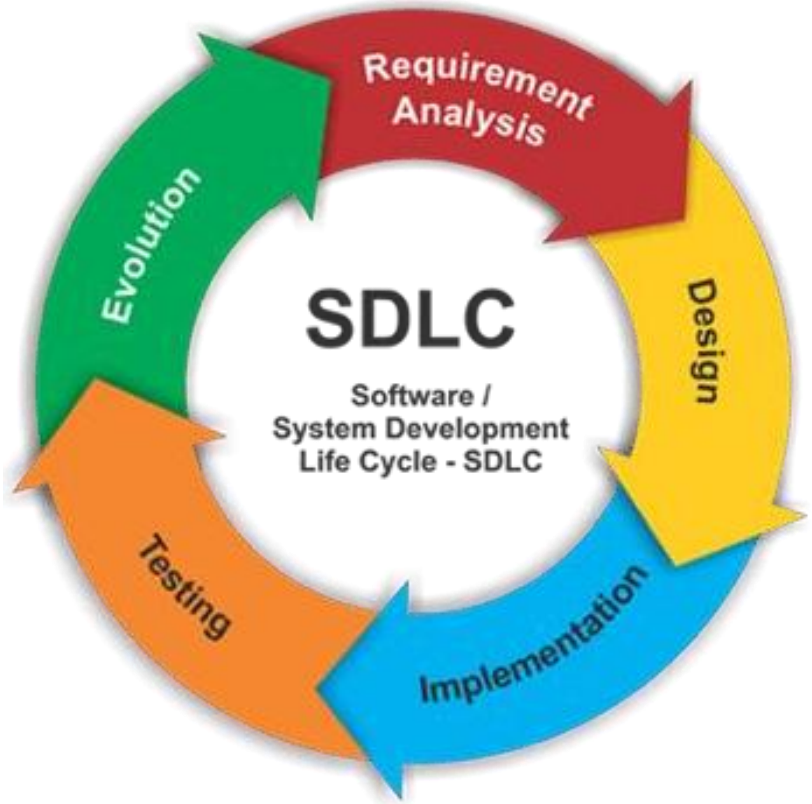
Attributes of a good Software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

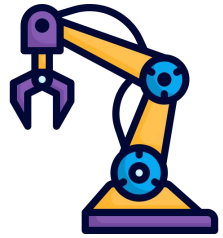
ROBOTICS CORNER



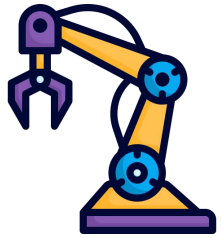
- Requirements
- Design
- Implementation
- Testing
- Evolution



ROBOTICS CORNER

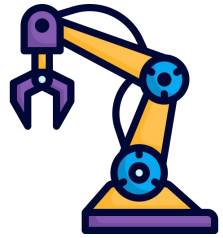


- **Requirement Types**
  - User requirements
    - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
  - System requirements
    - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.



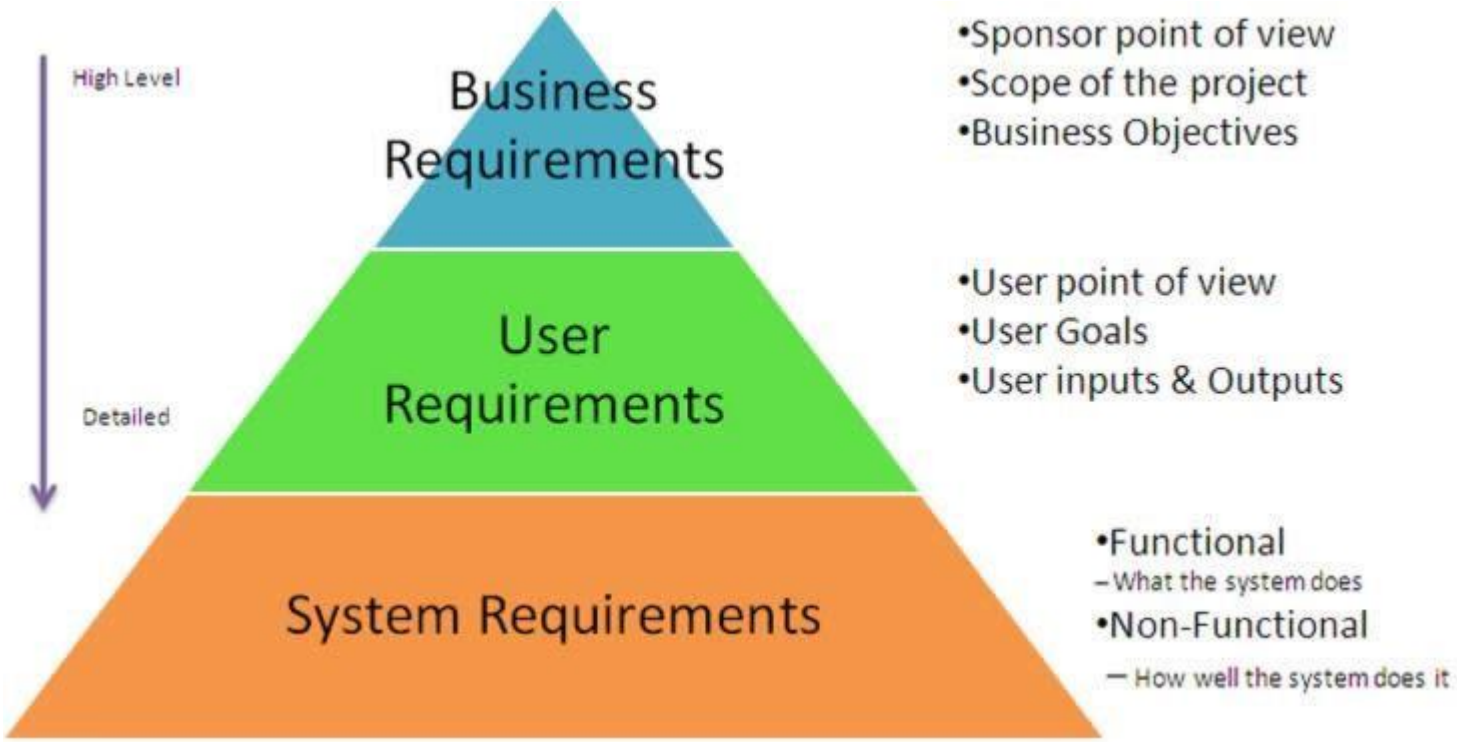
Requirements

- **Business requirements.** These include high-level statements of goals, objectives, and needs.
- **Stakeholder requirements.** The needs of discrete stakeholder groups are also specified to define what they expect from a particular solution.
- **Solution requirements.** Solution requirements describe the characteristics that a product must have to meet the needs of the stakeholders and the business itself.
  - **Functional** requirements describe how a product must behave, what its features and functions.
  - **Nonfunctional** requirements describe the general characteristics of a system. They are also known as *quality attributes*.



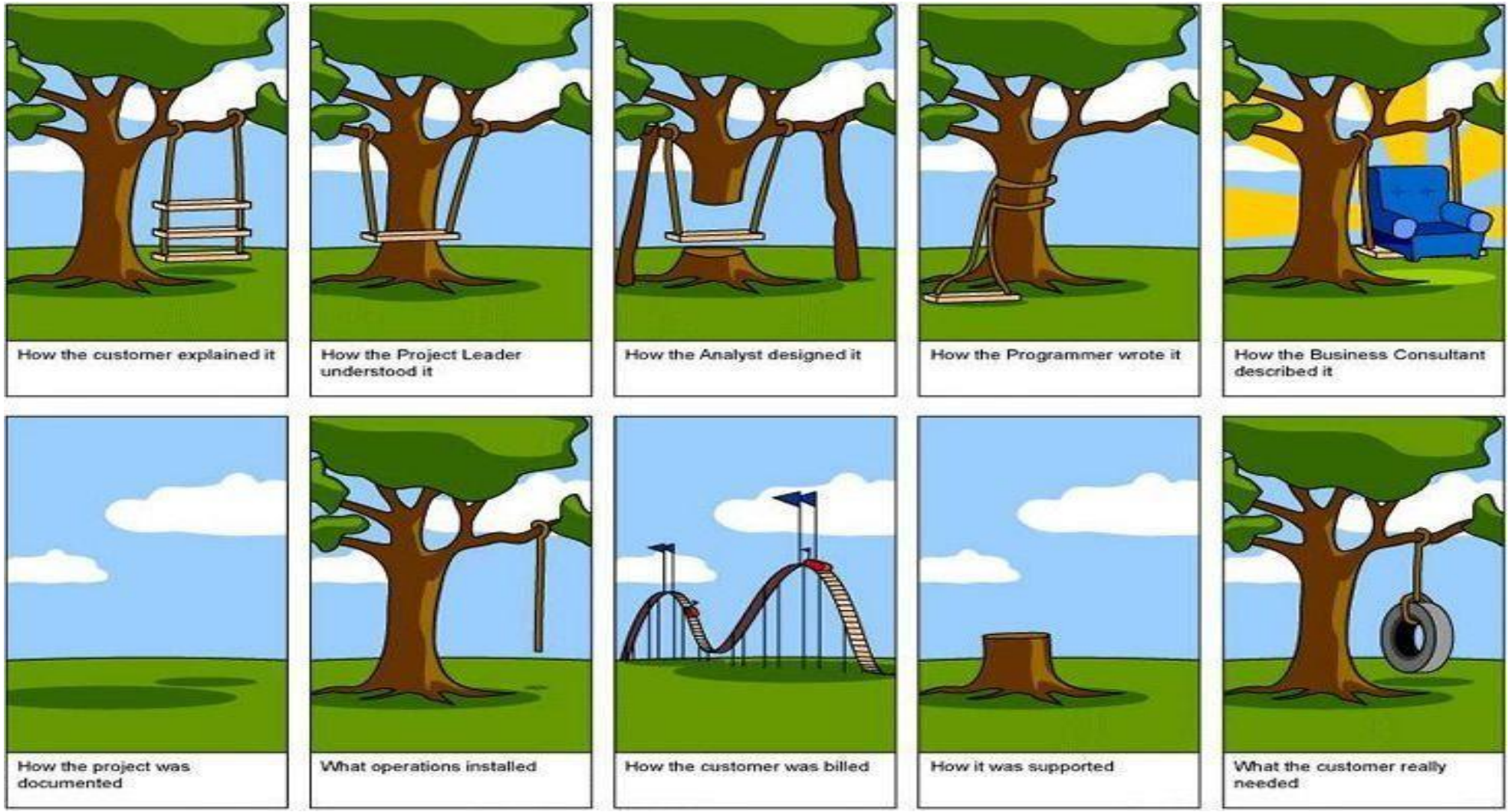
Software Requirement Specifications

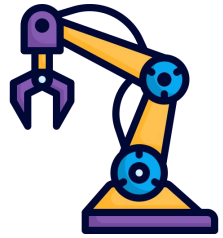
- TODO



ROBOTICS CORNER

Requirements

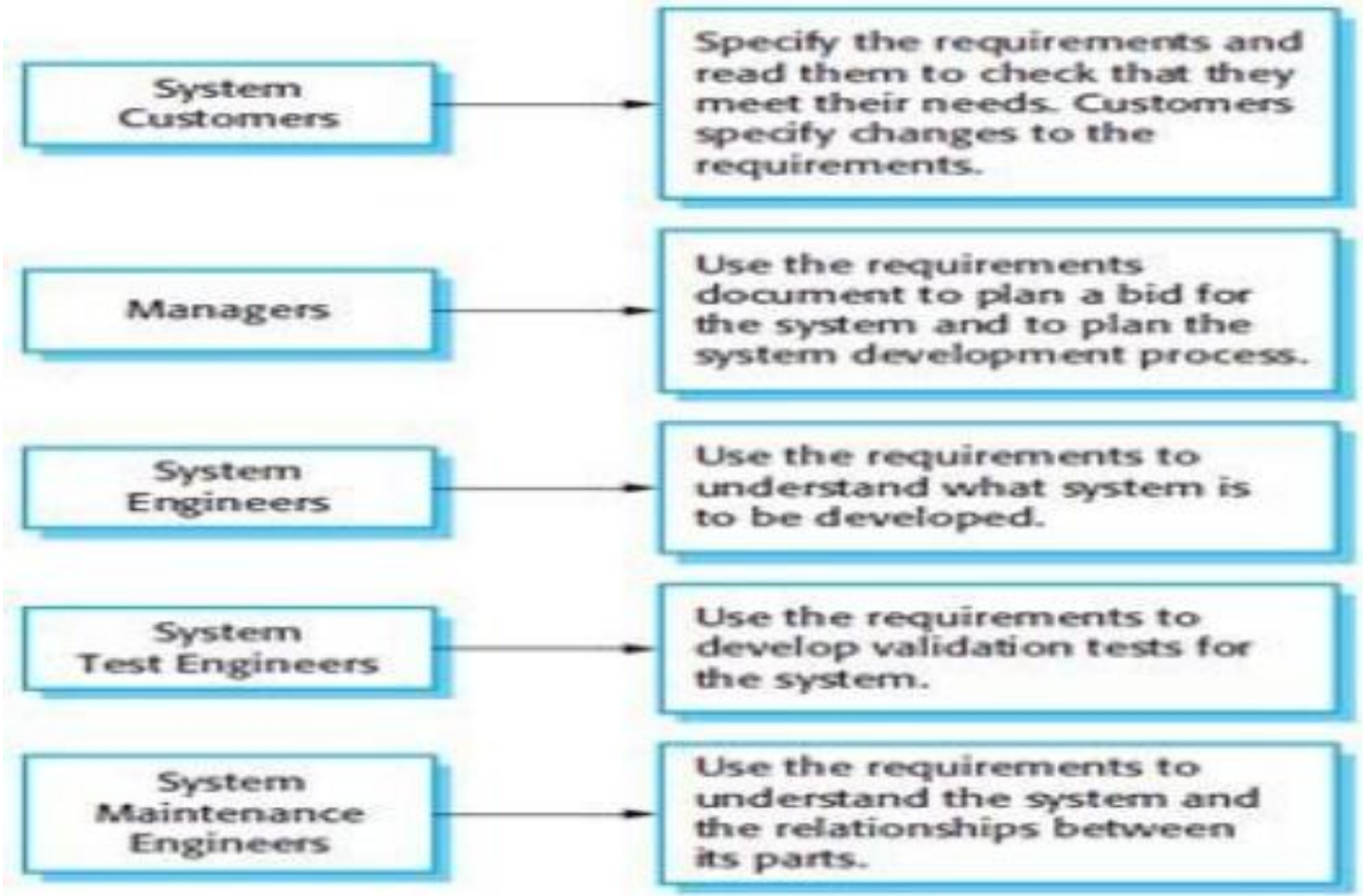




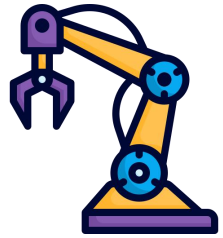
Requirement Documents

- The usage of requirement documents by the system Actors from system Customers, Managers, System Engineers, and quality control engineers

- Each actor uses the requirements to do his job

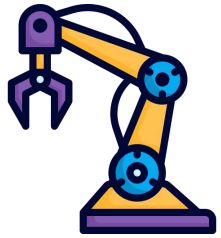


ROBOTICS CORNER



Design

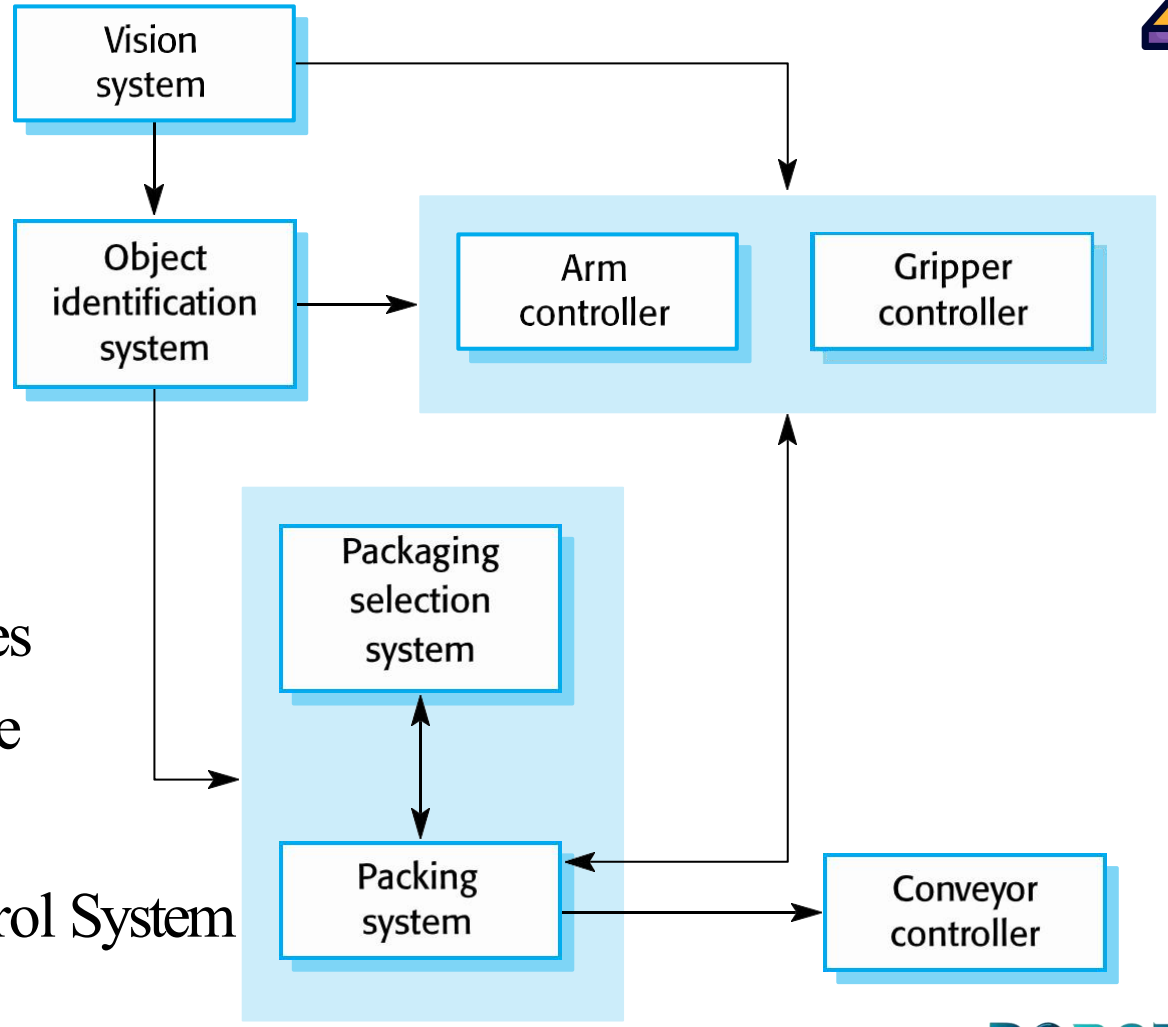
- Designing The system is the first stage by High-Level Design HLD
- Breaking down the system into components each has its own component Detailed Design CDD
- Design types are:
- Dynamic Design which describes how the system behaves
- Static Design describes the Architecture and structure of the System which relates to files and implementation



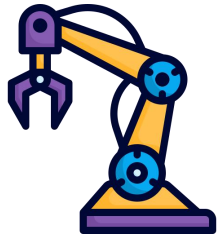
Architecture Design

- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system.
- Architectural design is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them.

This is Architecture Model of packing Robot Control System

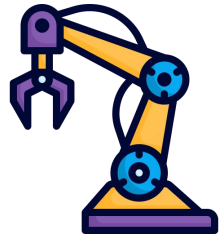


UML



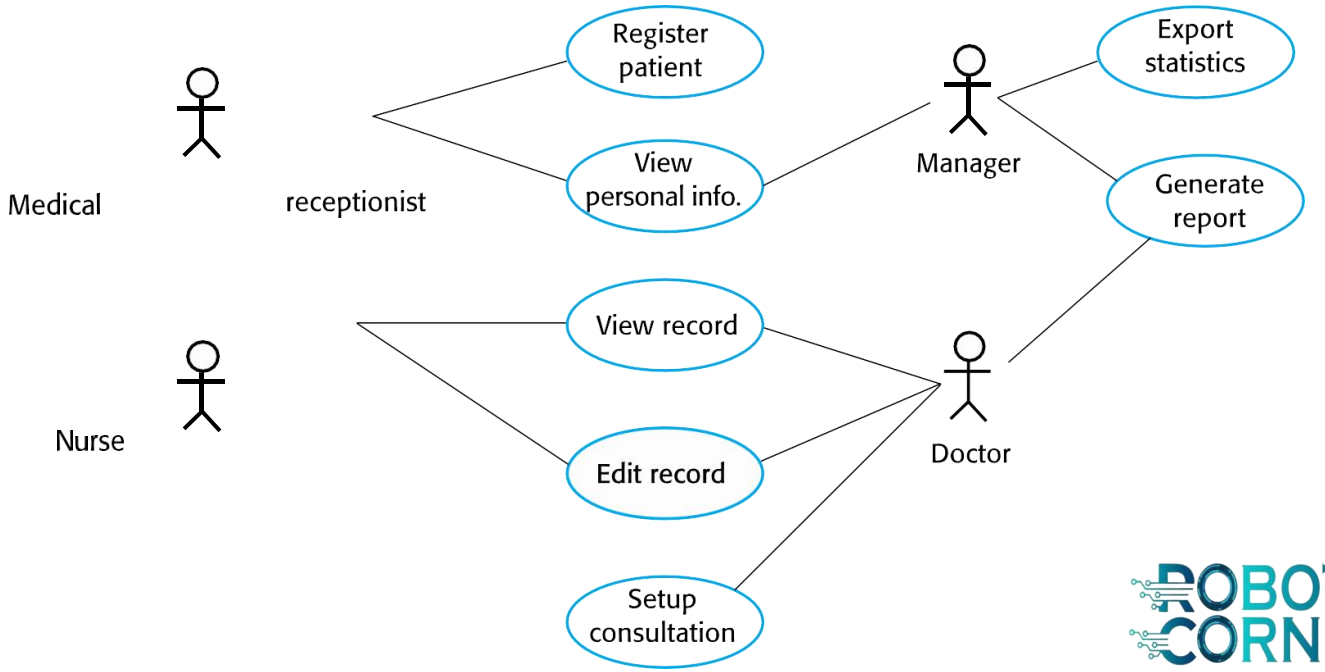
- **UML** stands for Unified Modeling Language
- UML is a diagramming language designed for Object Oriented programming
- UML can be used to describe:
- the organization of a program
- how a program executes
- how a program is used
- how a program is deployed over a network
- …and more
- **FAQ**: Why UML is used in Embedded Systems?
- UML is not just class diagrams, it includes state diagrams and flowcharts, which are useful also outside of an object oriented setting
- Also, many C programs are written in a way that borrows a few concepts from OOP.

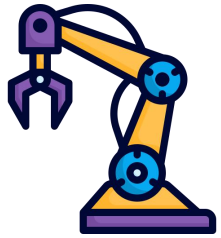
ROBOTICS CORNER



Use Case

- **Actors.** These are the users outside the system that interact with the system.
- **System**.The system is described by functional requirements that define an intended behavior of the product.
- **Goals**.The purposes of the interaction between the users and the system are outlined as goals.

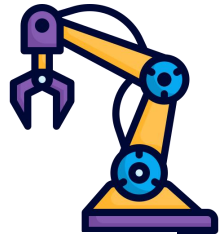




Sequence Diagram

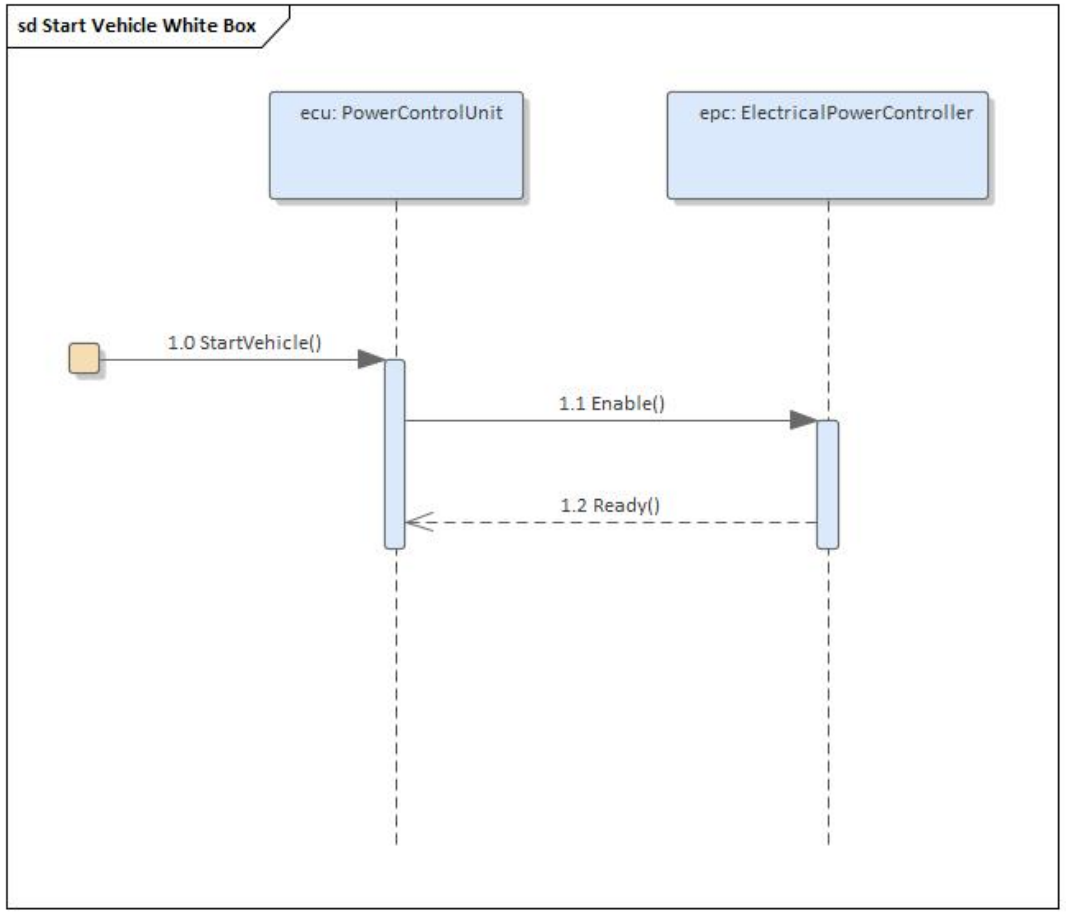
- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

ROBOTICS CORNER

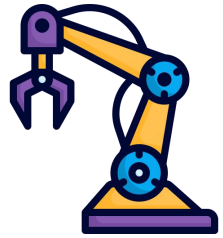


Sequence Diagram

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

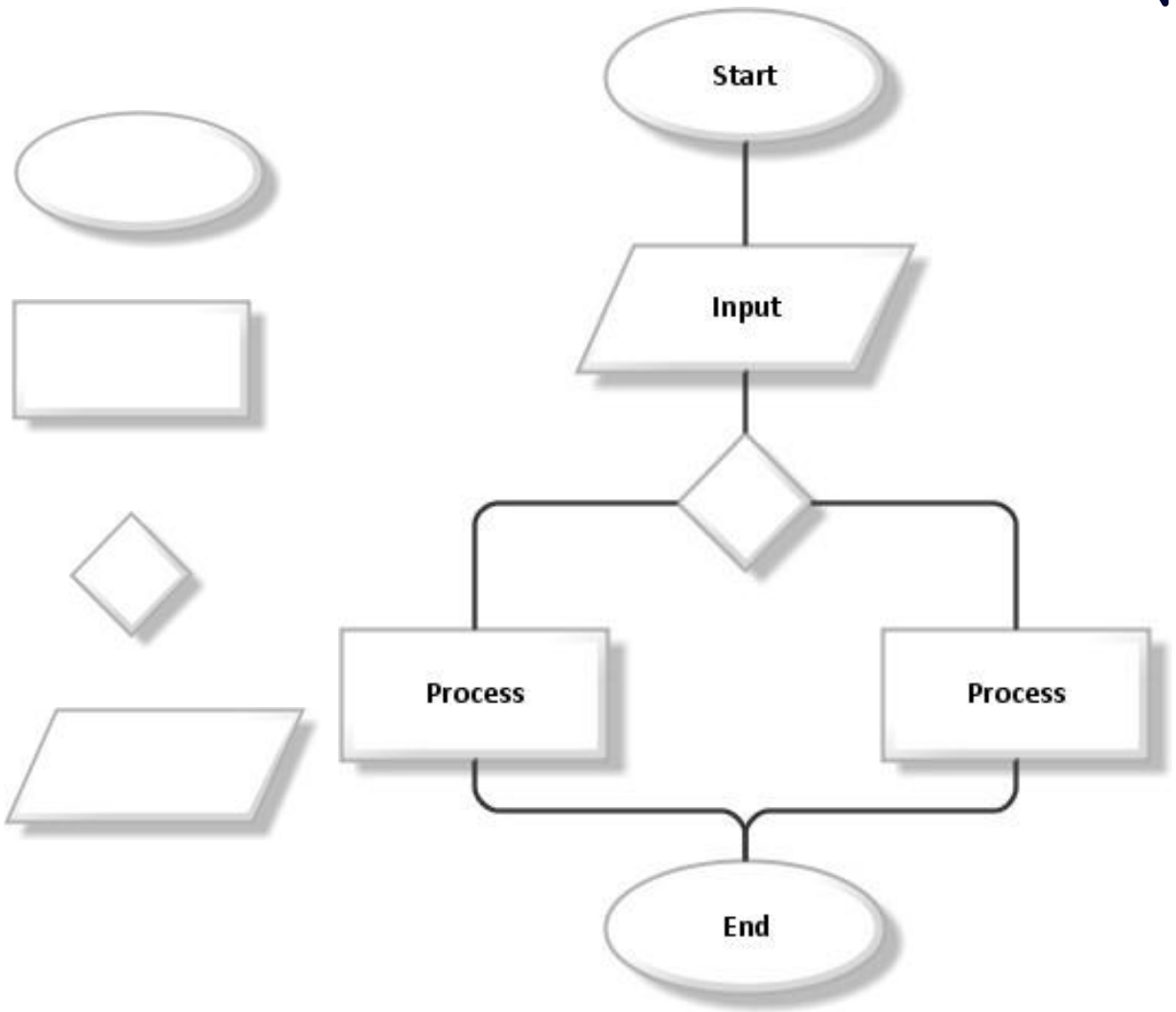


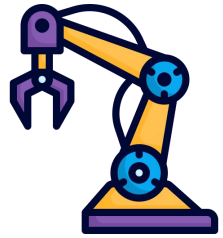
ROBOTICS CORNER



Flow Chart

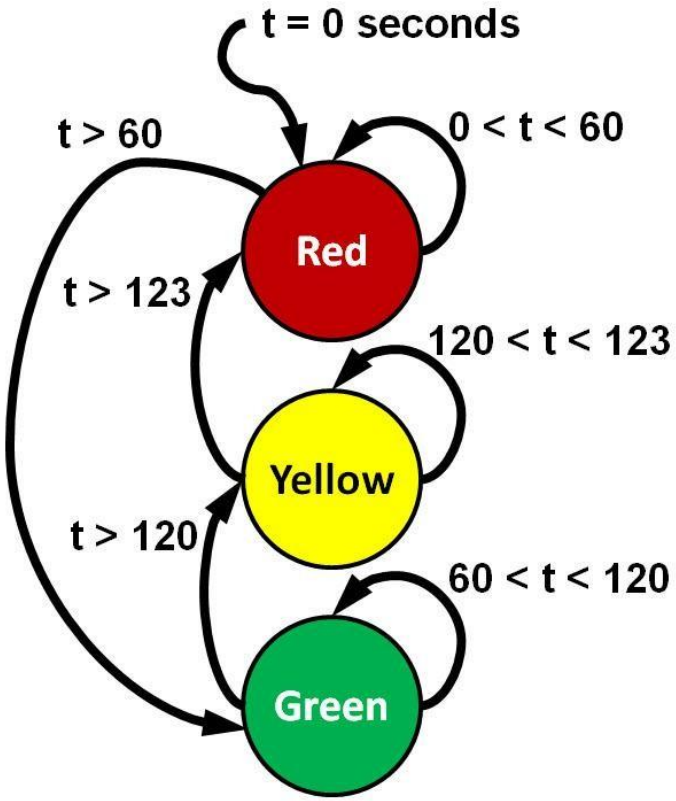
- Used for start and end
- Processing steps
- Decision 1 input multiple Output
- Data illustrates input and output





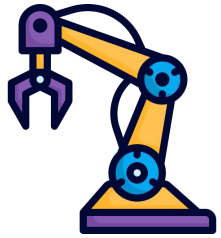
Finite State Machine

- Every system has to have finite number of states
- The system transients from state to another state according to the input of the state which is the output of the previous state



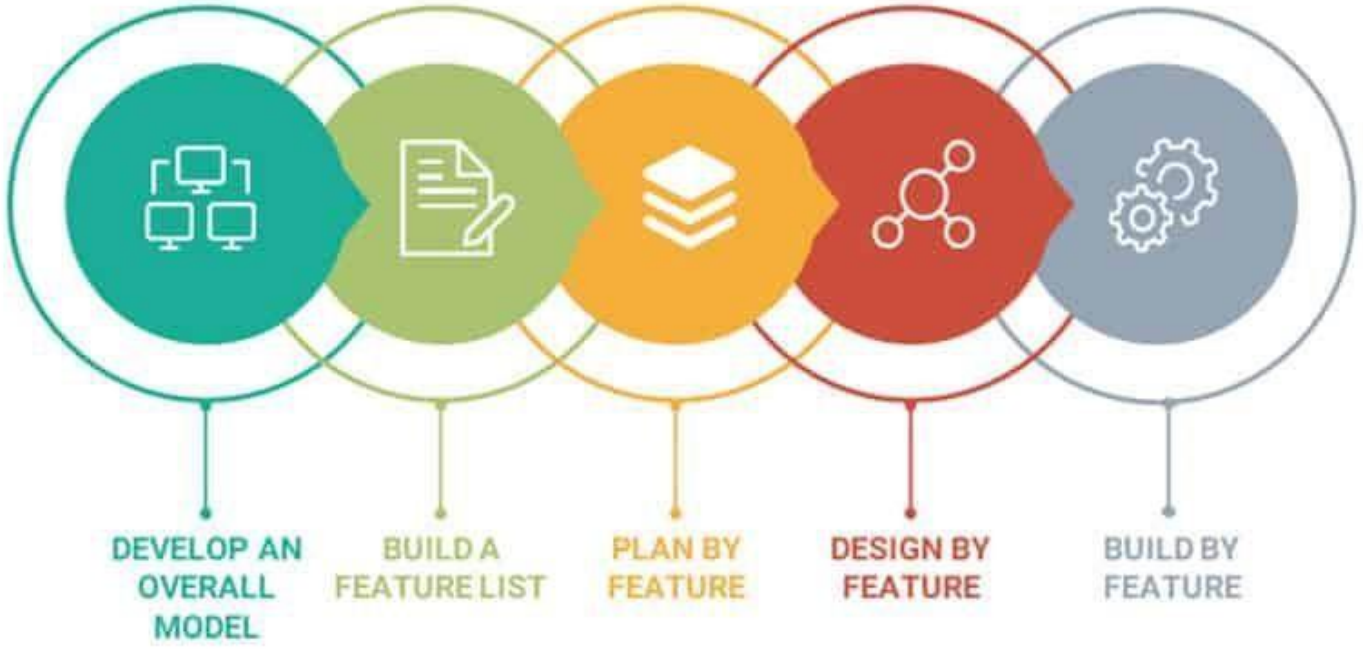
Traffic Light

- Example Traffic Light System :
- Inputs are the timer value
- **RED** state when time=0 stay for 60 s
- **GREEN** state when the timer = 60 &&time <120
- **YELLOW** state stay for 3s then go to **RED**
- **RED** reset the time and start over

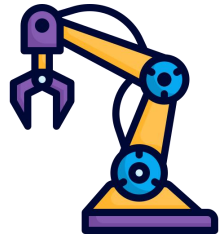


Implementation/Development Techniques

- Plan Driven Development
- Feature Driven Development

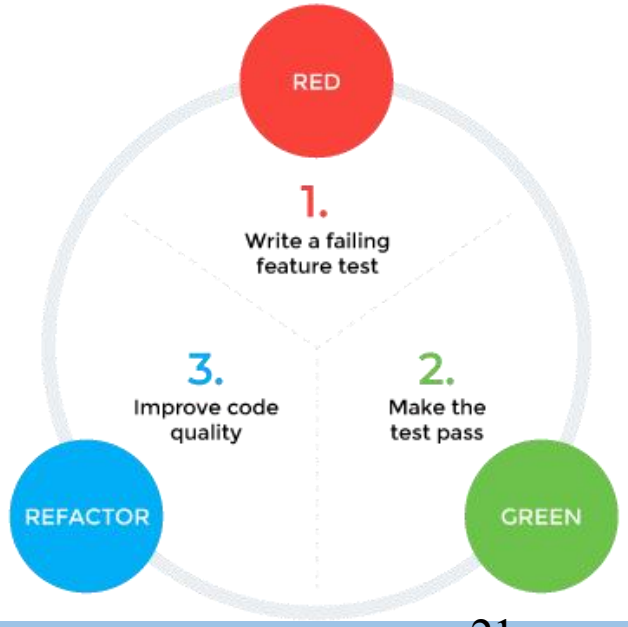


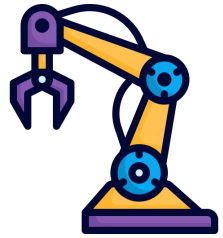
ROBOTICS CORNER



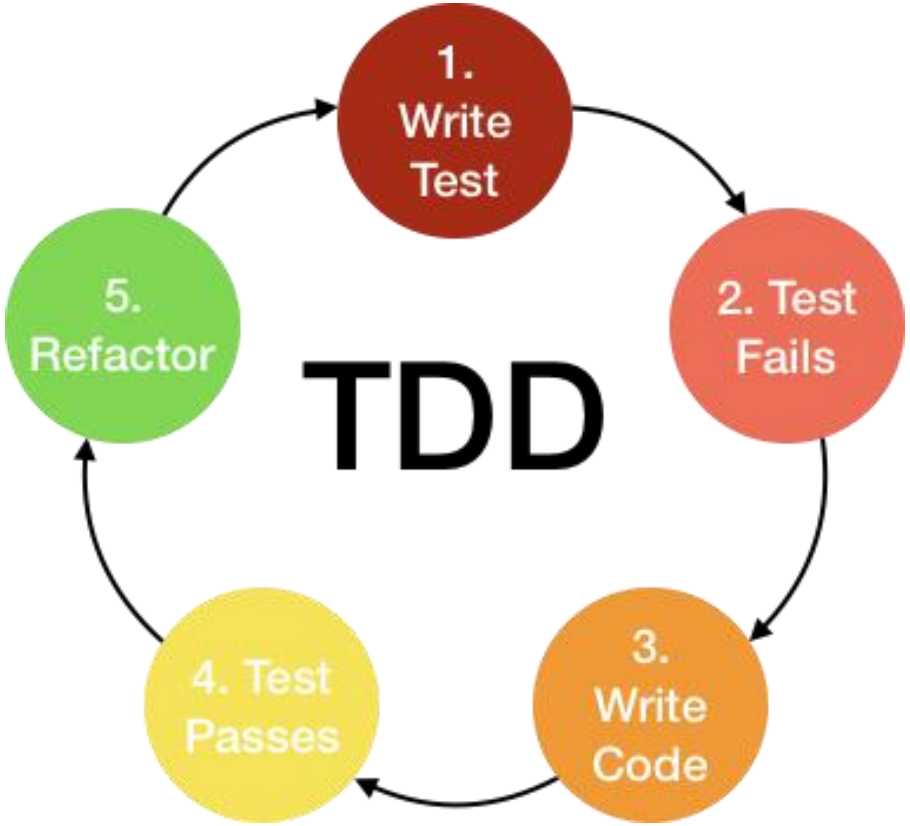
Implementation Techniques

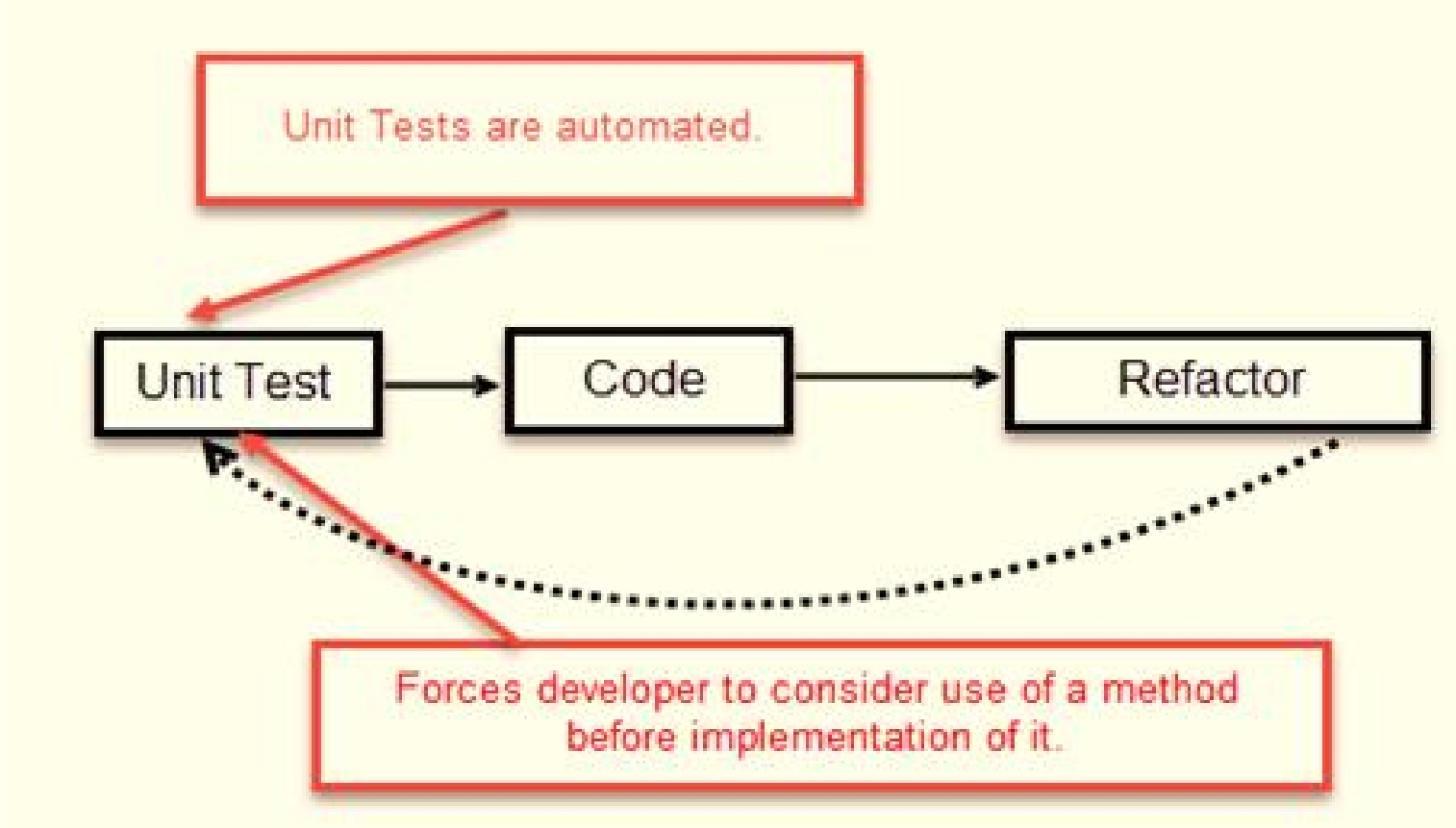
1. **Test-Driven Development(TDD)** approach first, the test is developed which specifies and validates what the code will do. In simple terms, test cases are created before code is written. The purpose of TDD is to make the code clearer, simple and bug- free.
2. **Behavior-Driven Development(BDD)** is also a test-first approach, but differs by testing the actual behavior of the system from the end users perspective.

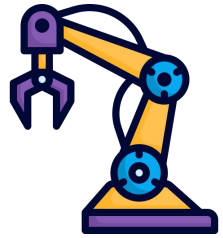




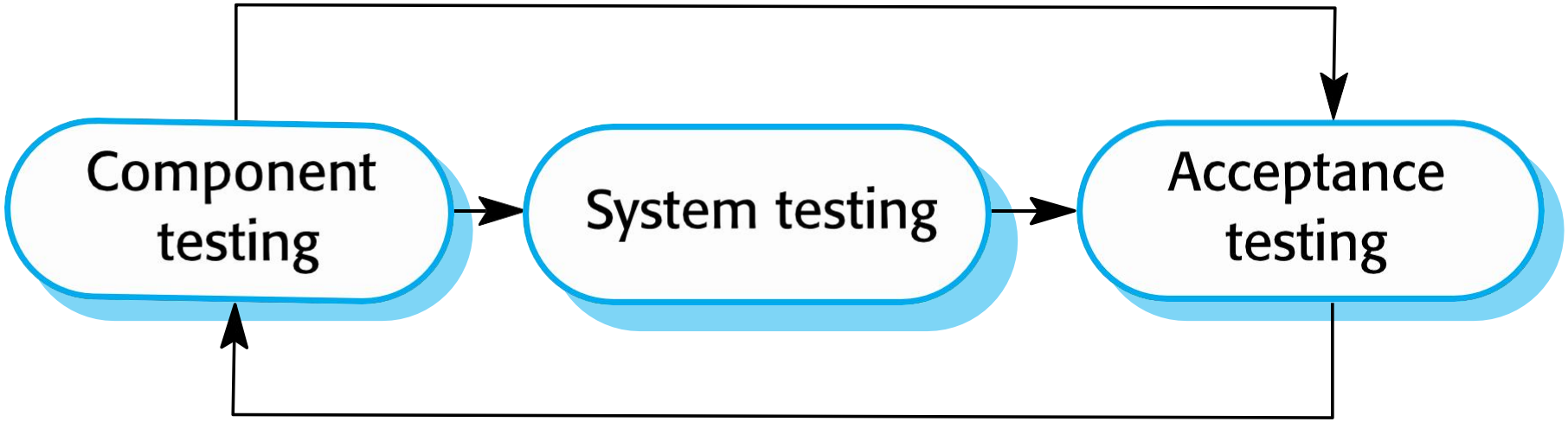
TDD Techniques

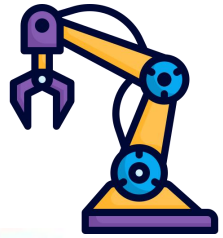




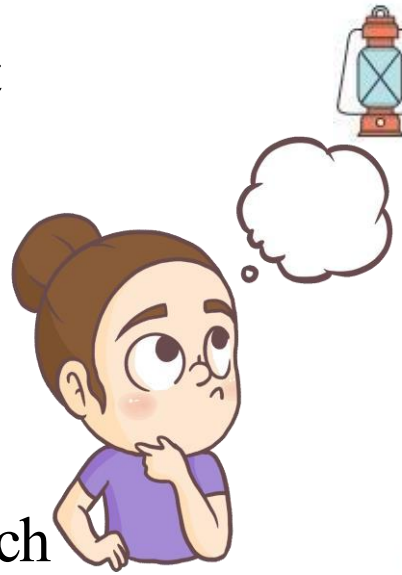


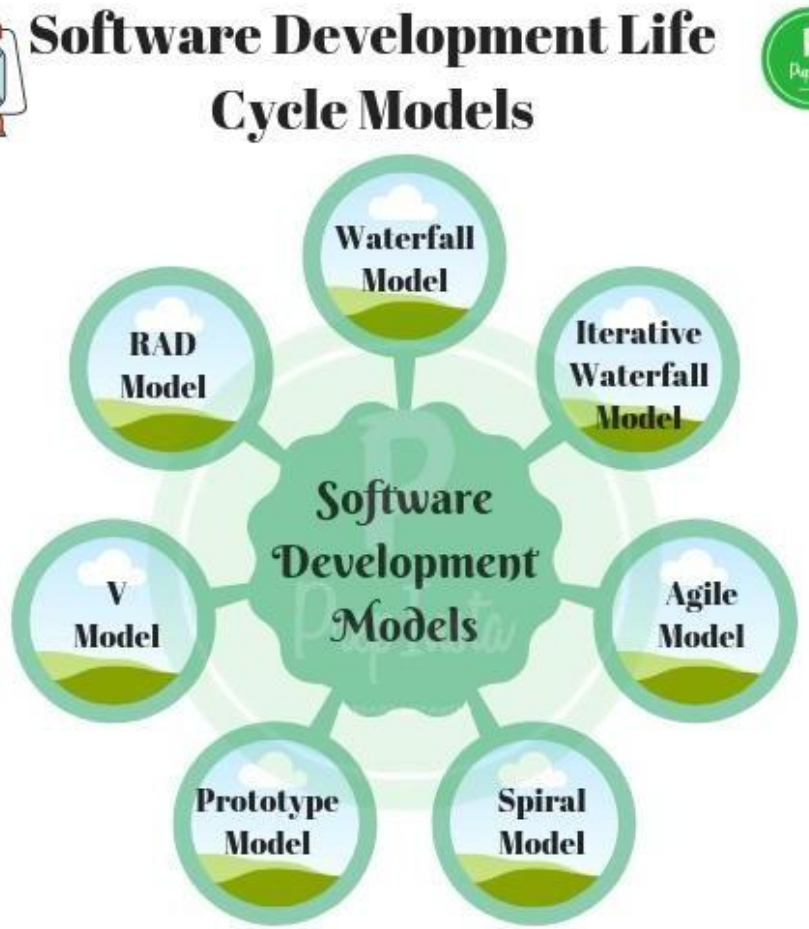
1. Each Component will need to be have its own unit Testing and when its integrated with other components will be a part of Integration Testing then System Testing and the customer will do the acceptance testing

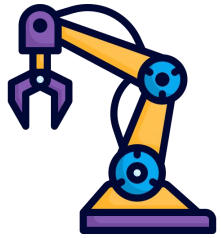




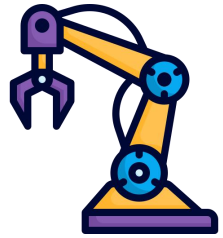
- Why The need of Software development Models?
- Different types of models
- The Advantages and the drawbacks of each model



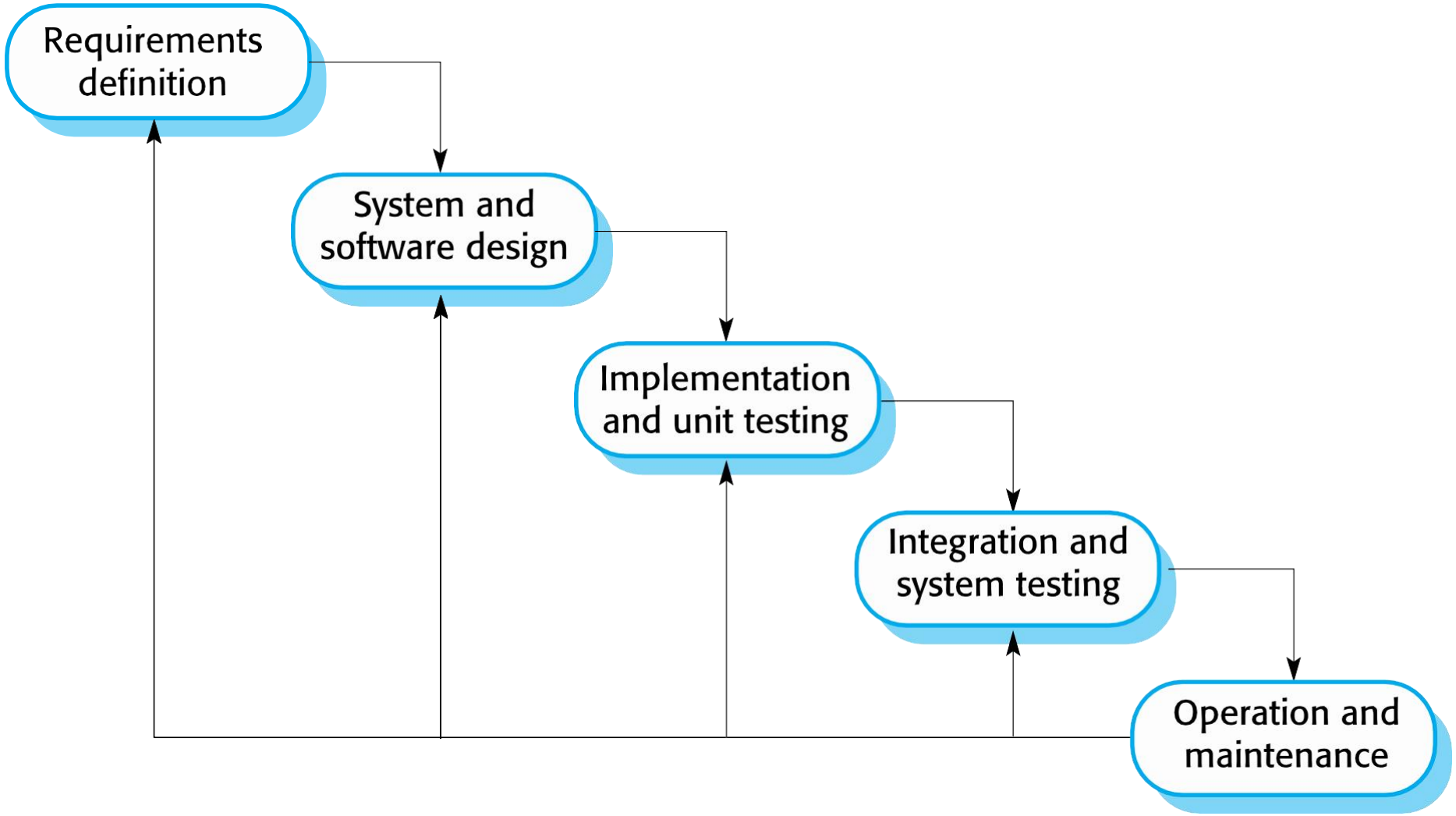




- **SDLC phases are not enough by their own to produce a software product and combine it with the business needs, cost estimation, and customer satisfaction.**
- **So Software development phases have to fit in a model that serves the overall aim which might be adapting to the change of customer requirements or developing high-quality Software**
- **Some of the models are customer centric where the customer plays an important role in the feedback process and the development team adapts according to the change of requirements i.e *Agile* used for new products and innovations**
- **Other SDLC Models are used when quality control is the key essence of the product such as *V-Model* which is used in Safety critical applications such as Automotive.**

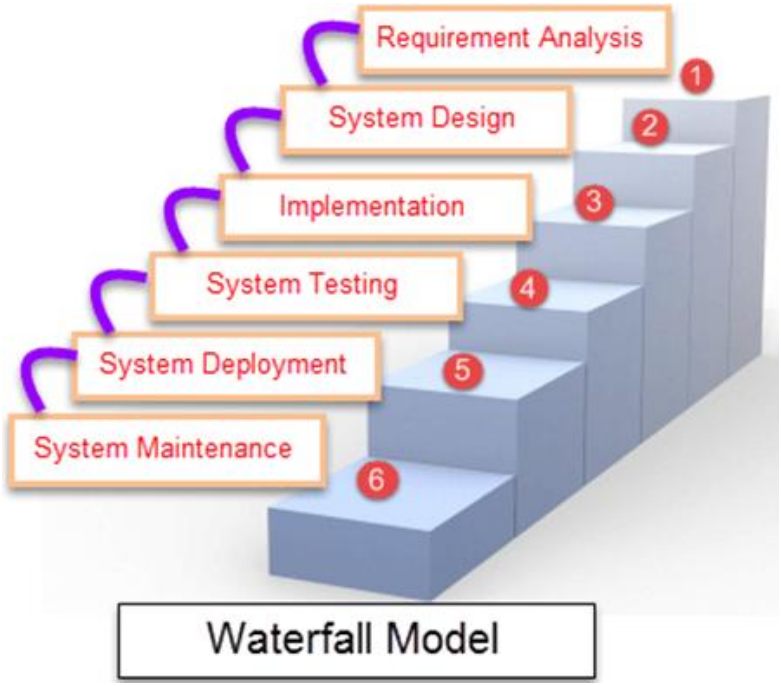


Waterfall Model



Advantages and Disadvantages

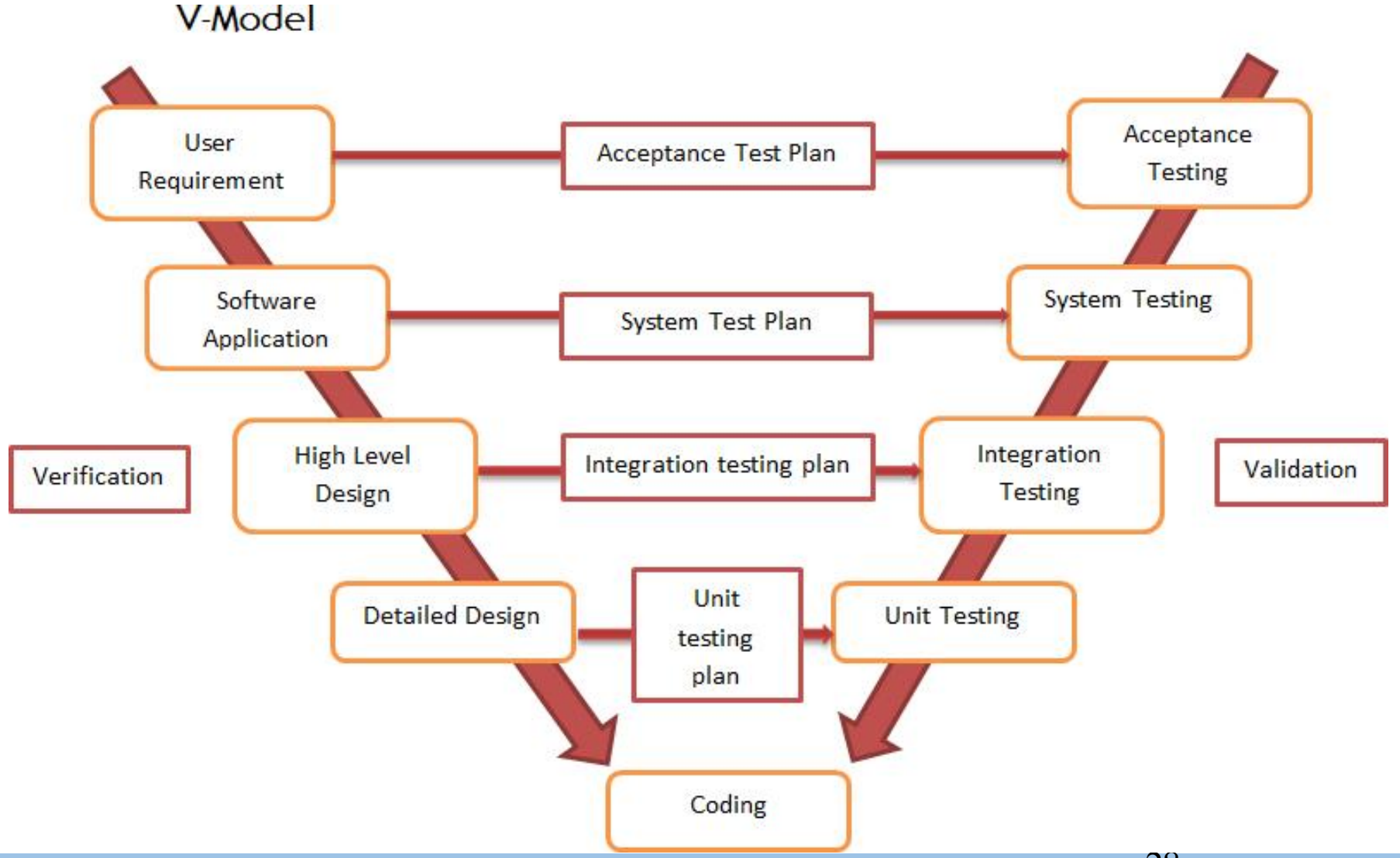
When to use Waterfall?
Projects which not focus on changing the requirements, for example, projects initiated from a request for proposals,the customer has a very clear documented requirements



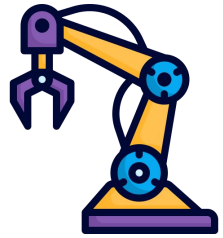
Waterfall Model

Advantages	Disadvantages
•Easy to explain to the users. •Structures approach. •Stages and activities are well defined. •Helps to plan and schedule the project. •Verification at each stage ensures early detection of errors/misunderstanding. •Each phase has specific deliverables.	•Assumes that the requirements of a system can be frozen. •Very difficult to go back to any stage after it finished. •A little flexibility and adjusting scope is difficult and expensive. •Costly and required more time, in addition to the detailed plan.

V-Model

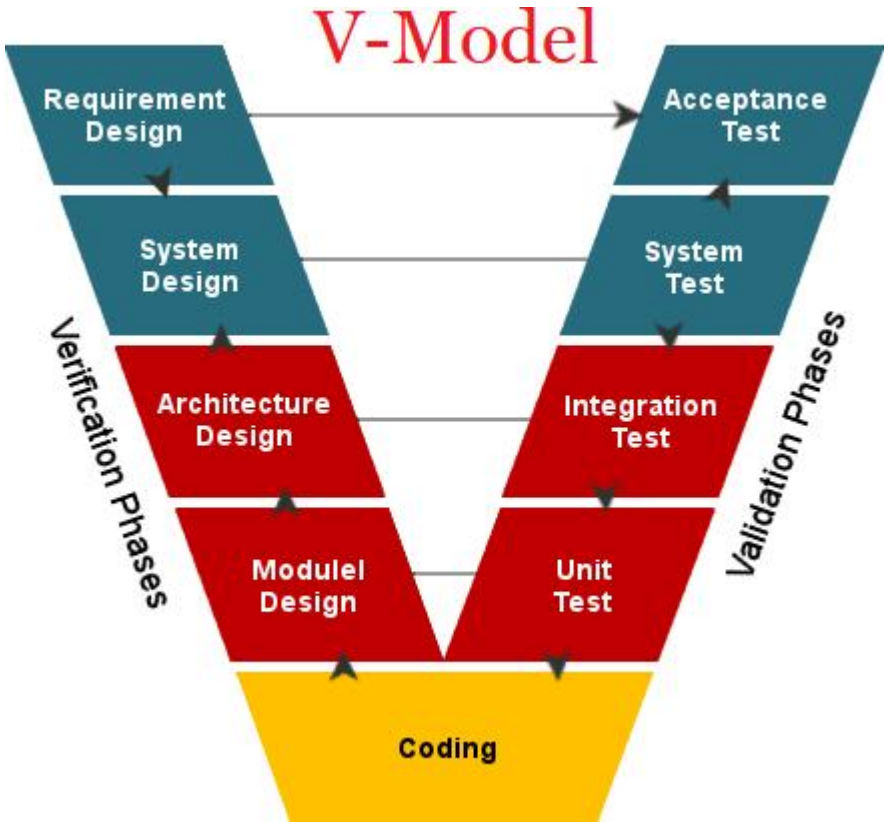

V-Model
User Requirement
Acceptance Test Plan
Acceptance Testing
Software Application
System Test Plan
System Testing
Verification
High Level Design
Integration testing plan
Integration Testing
Validation
Detailed Design
Unit testing plan
Unit Testing
Coding

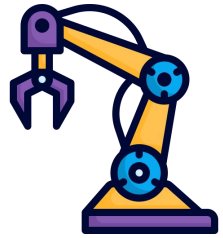
V-Model



- **The usage**
- Software requirements clearly defined and known
- Software development technologies and tools are well-known

Advantages	Disadvantages
•Simple and easy to use •Each phase has specific deliverables. •Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle. •Works well for where requirements are easily understood. •Verification and validation of the product in the early stages of product development.	•Very inflexible, like the waterfall model. •Adjusting scope is difficult and expensive. •The software is developed during the implementation phase, so no early prototypes of the software are produced. •The model doesn't provide a clear path for problems found during testing phases. •Costly and required more time, in addition to a detailed plan

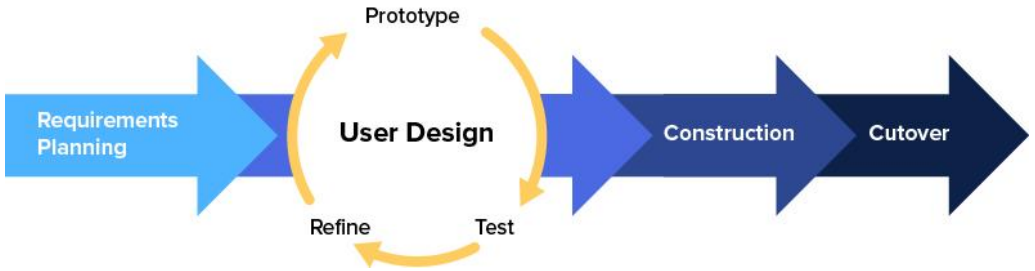


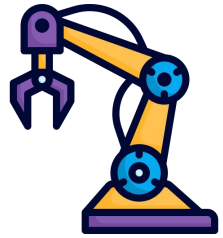


- **Rapid development and delivery is now often the most important requirement for software systems**
  - Businesses operate in a fast **changing requirement** and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs.

- **Plan-driven development** is essential for some types of system but does not meet these business needs.
- **Agile development** methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems

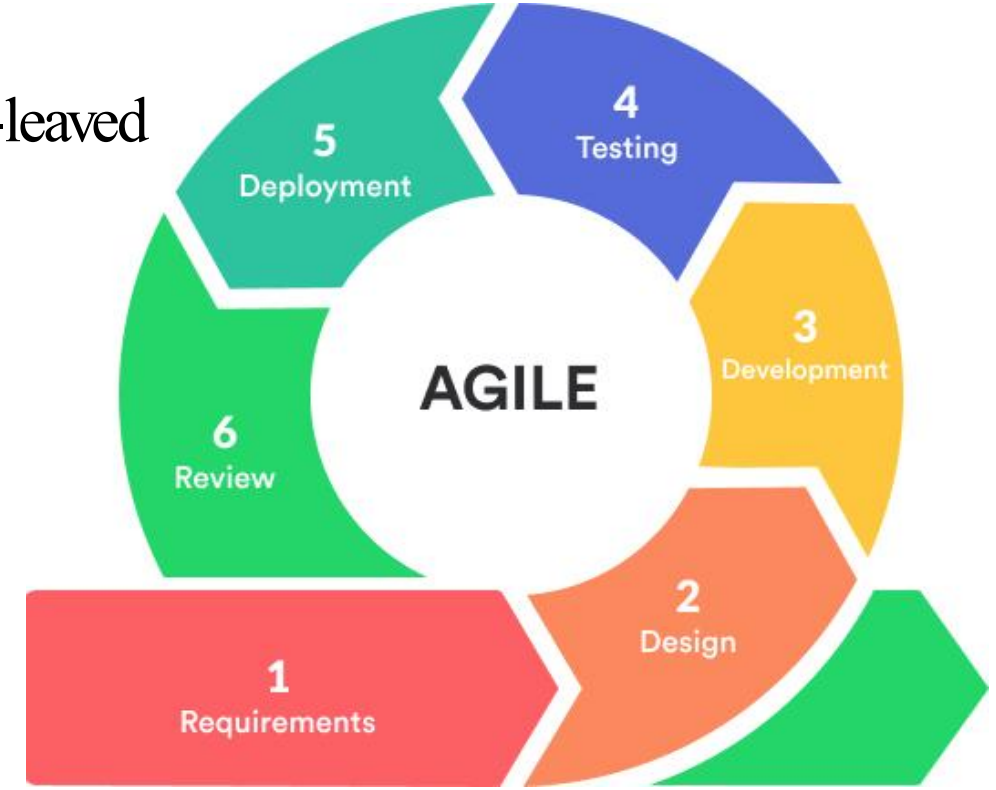
Rapid Application Development (RAD)





Agile development

- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation – focus on working code

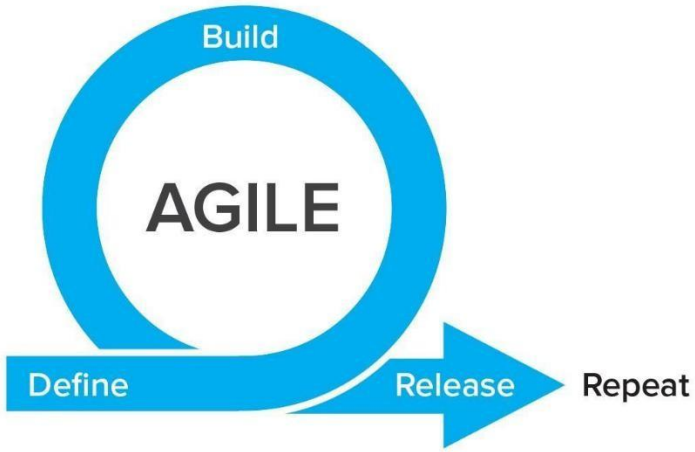


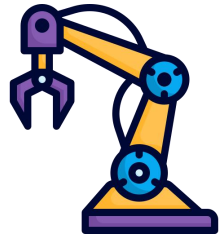
Quoted from Chapter 3 Agile Software Development

When to use Agile?

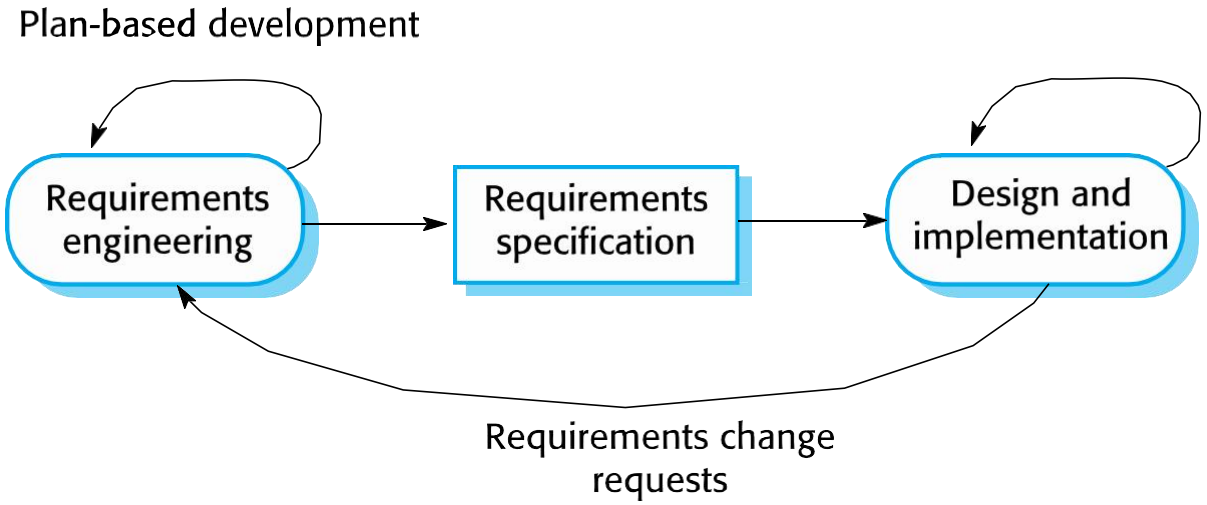
- It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, we can use it when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough.
- This will enable more valuable and workable piece for software early which also increase the customer satisfaction.

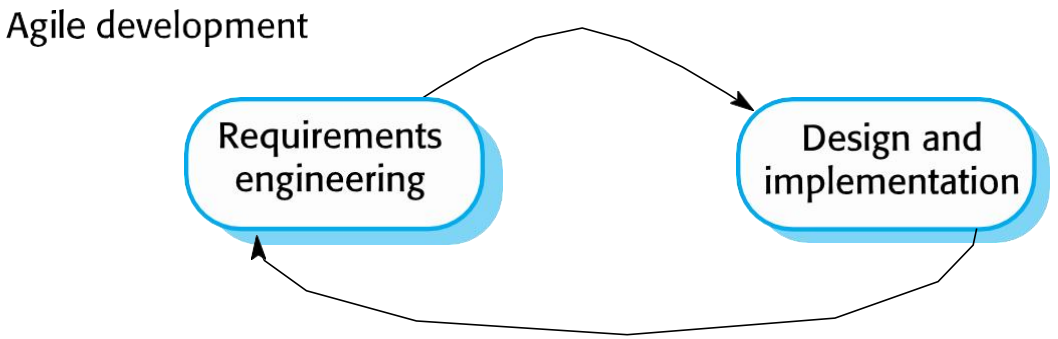
Advantages	Disadvantages
•Decrease the time required to avail some system features. •Face to face communication and continuous inputs from customer representative leaves no space for guesswork. •The end result is the high-quality software in the least possible time duration and satisfied customer.	•Scalability. •The ability and collaboration of the customer to express user needs. •Documentation is done at later stages. •Reduce the usability of components. •Needs special skills for the team.



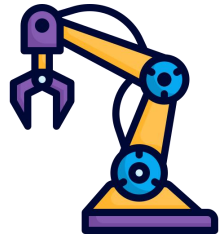


Plan-driven vs Agile development

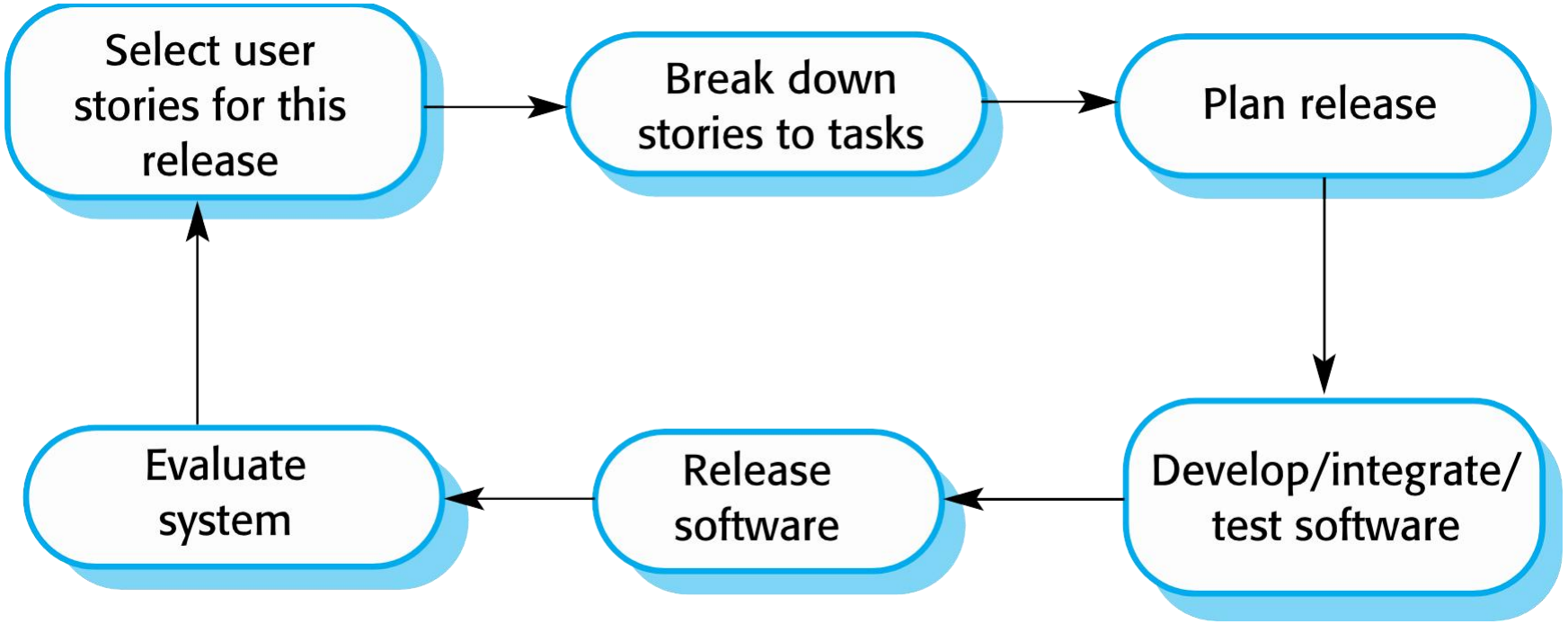




Quoted from Software Engineering : Chapter 3 Agile Software Development

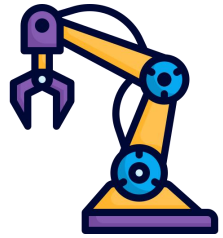


How Agile works



Quoted from Software Engineering : Chapter 3 Agile Software Development

Examples of task cards for prescribing medication



User Story Characteristics

Invest:
I) Independent
N) Negotiable
V) Valuable
E) Estimable
S) Small
T) Testable

Smart:
S) Specific
M) Measurable
A) Achievable
R) Relevant
T - Time-boxed

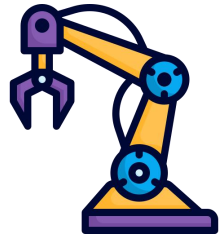
Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

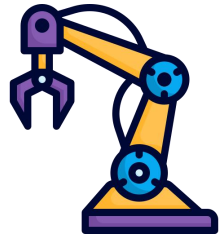
ROBOTICS CORNER



SCRUM

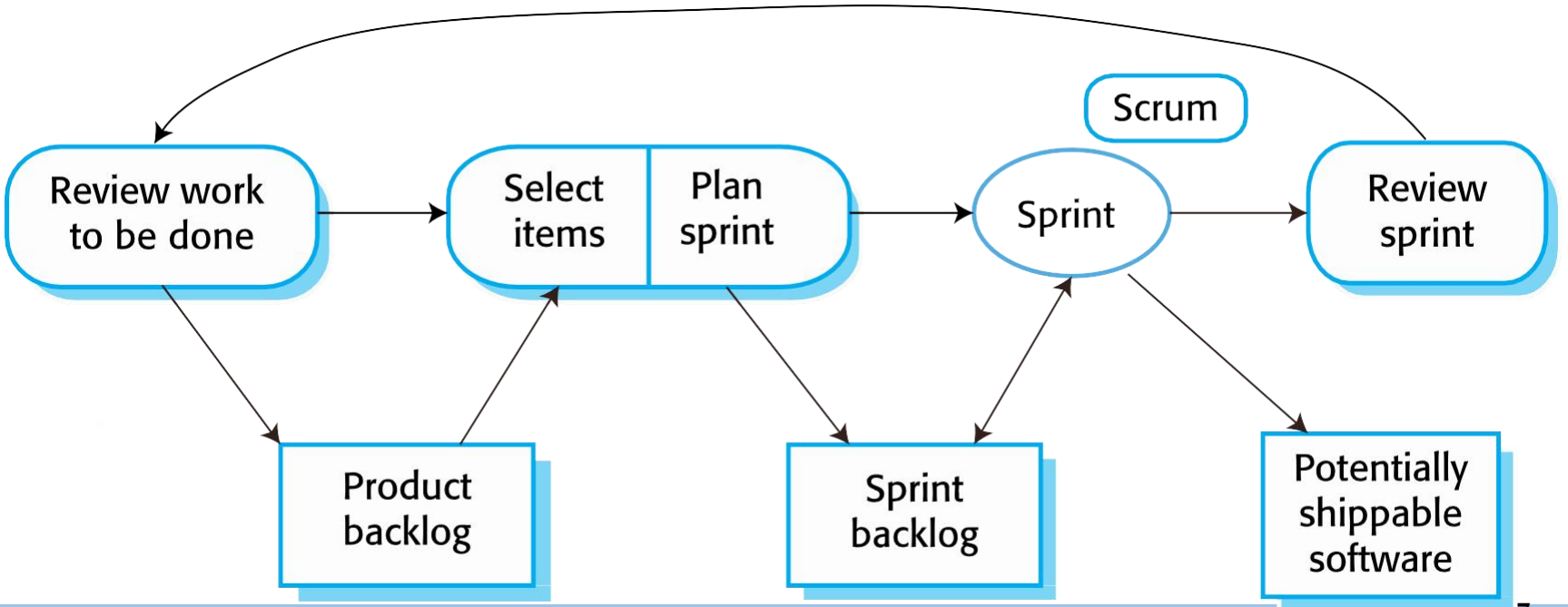
- **Scrum** is an agile method that focuses on managing iterative development rather than specific agile practices.
- There are three phases in Scrum.
  - The **Initial phase** is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  - This is followed by a series of **sprint cycles**, where each cycle develops an increment of the system.
  - The **project closure** phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

ROBOTICS CORNER



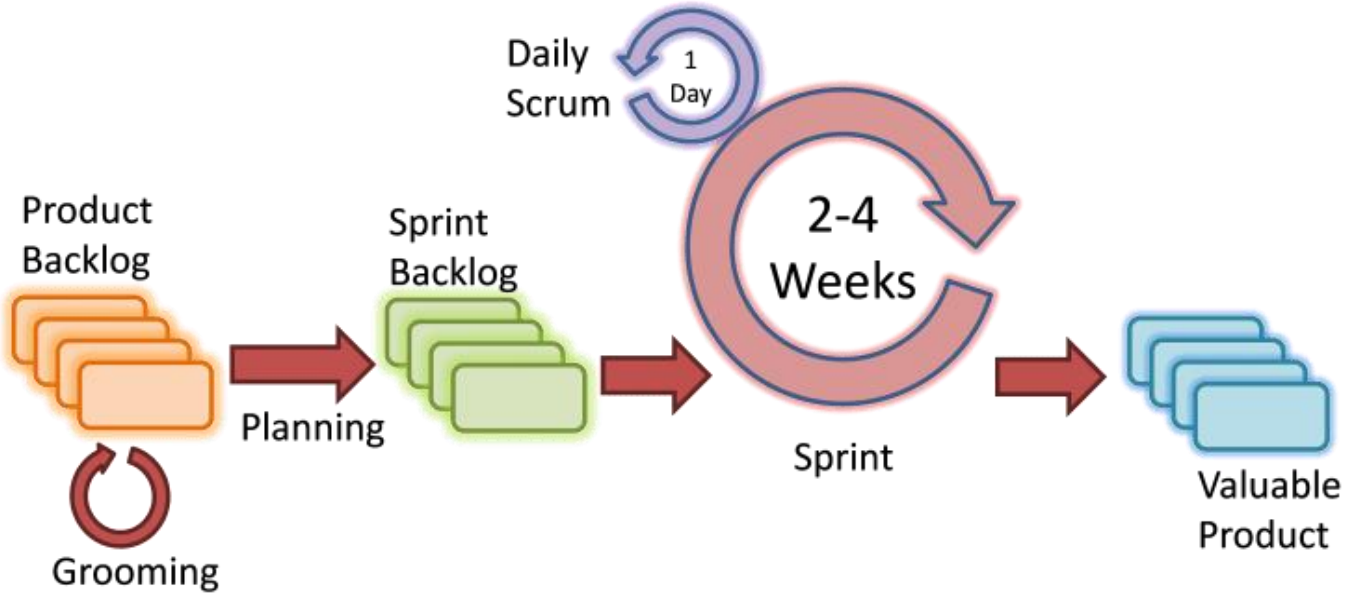
Scrum Sprint cycle

- **Sprints** are fixed length, normally 2–4 **weeks**.
- The starting point for planning is the product **backlog**, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality from the product backlog to be developed during the sprint.

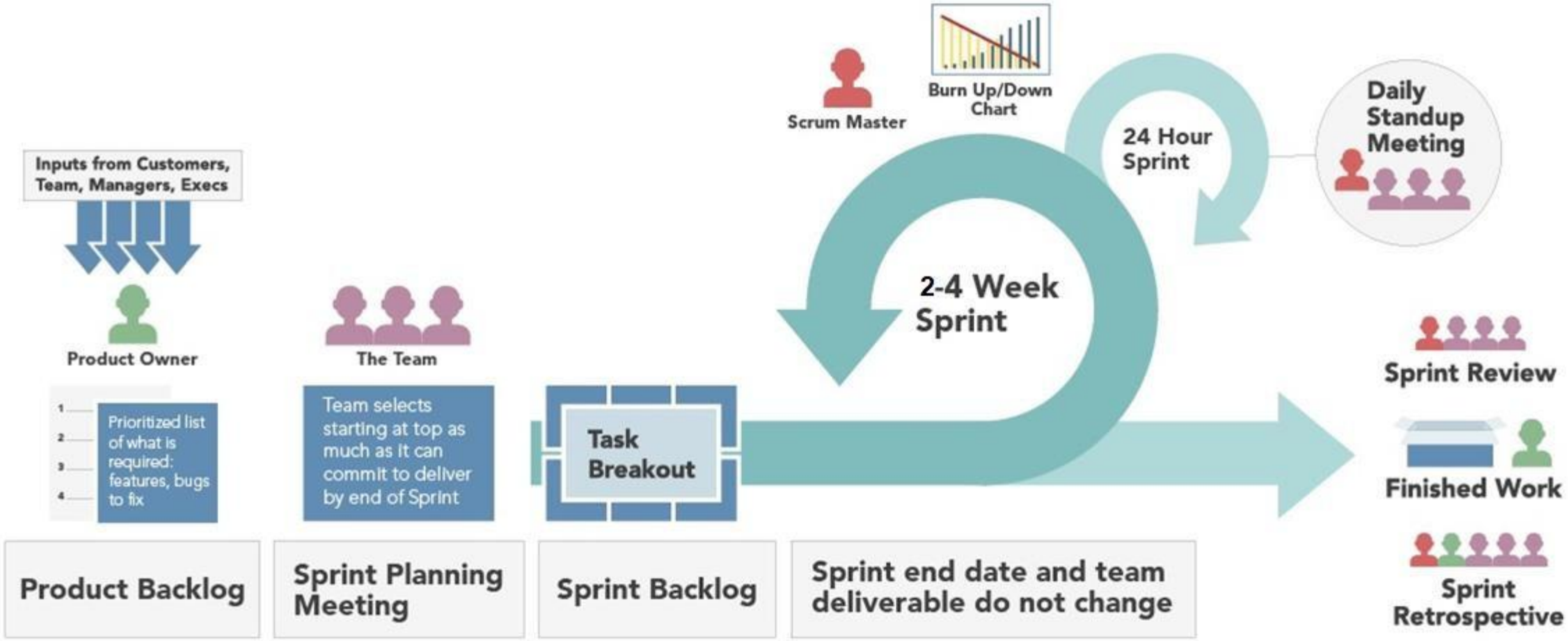


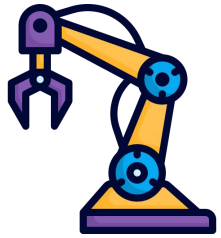
The Sprint Cycle

- Once these are agreed, the team organize themselves to develop the software.
- During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called 'Scrum master'.
- The role of the Scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.



Agile Scrum Framework

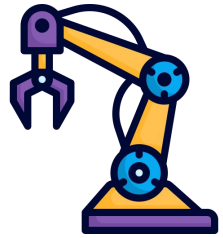




JIRA tool

- JIRA is a **project management** tool used for issues and bugs tracking system. It is widely used as an issue-tracking tool for all types of testing.This tutorial introduces the readers to the fundamental features, usage, and advantages of JIRA.This tutorial will guide the users on how to utilize this tool to track and report bugs in different applications.
- JIRA is developed by **Atlassian Inc**., an Australian Company
- JIRA is a **commercial** tool and available as aTrial version for a **limited time.**
- To utilize JIRA services, a license is required.
- JIRA provides **free license** for **academic** projects.
- A 15-day trial version is available for an individual person to use.

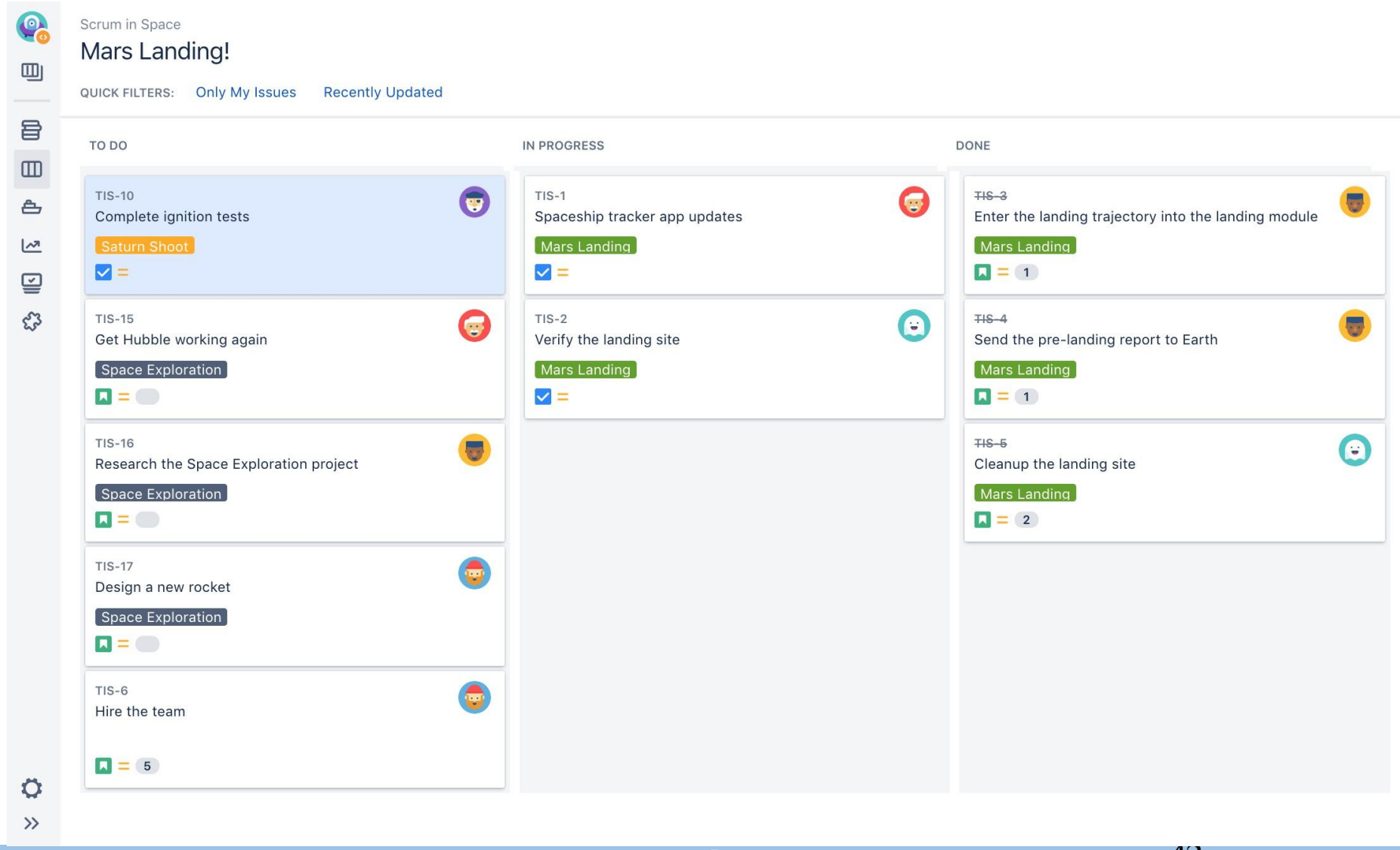
ROBOTICS CORNER

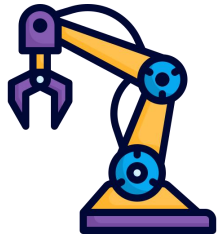


Use of JIRA

- Following are some of the most significant uses of JIRA.
- JIRA is used in Bugs, Issues and Change Request Tracking.
- JIRA can be used in Help desk, Support and Customer Services to create tickets and track the resolution and status of the created tickets.
- JIRA is useful in Project Management, Task Tracking and Requirement Management.
- JIRA is very useful in Workflow and Process management.

JIRA Boards


Scrum in Space
Mars Landing!
QUICK FILTERS: Only My Issues Recently Updated
TO DO
IN PROGRESS
DONE
TIS-10
Complete ignition tests
Saturn Shoot
TIS-15
Get Hubble working again
Space Exploration
TIS-16
Research the Space Exploration project
Space Exploration
TIS-17
Design a new rocket
Space Exploration
TIS-6
Hire the team
5
TIS-1
Spaceship tracker app updates
Mars Landing
TIS-2
Verify the landing site
Mars Landing
~~TIS-3~~
Enter the landing trajectory into the landing module
Mars Landing
1
~~TIS-4~~
Send the pre-landing report to Earth
Mars Landing
1
~~TIS-5~~
Cleanup the landing site
Mars Landing
2



Markdown language for documentation

1. Markdown Language is a modern way of documentation for the web which you might need for documenting your project or your Repository on **GitHub** we always find the readme.md when creating a repo. But did you know how to make the title in **Bold** or add the link, That's when you need Markdown language.

2. Let's **Get Started : create a readme.md for your own repo.**

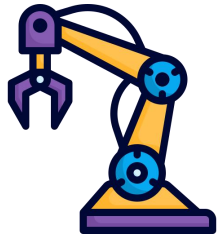
   Try some cool stuff of markdown language

1. You can use an online editor or Offline tools such as visual studio code, Dillinger is one of the best online editors for the Markdown language

Lab Time

ROBOTICS CORNER

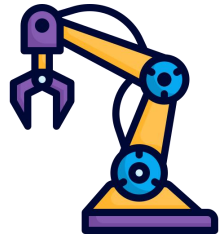
References



- Most of Diagrams and Software Engineering Concepts Are quoted from Sommerville https: / / iansommerville.com/software-engineering-book
- Thanks to the open-source world
- Markdown language guide https://www.markdownguide.org/
- More tutorials

Scrum vs Kanban https://www.youtube.com/watch?v=rIaz-l1Kf8w

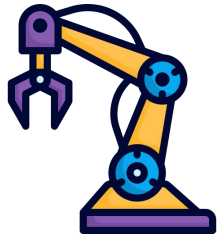
ROBOTICS CORNER



Thank You

Do you have any questions?

ROBOTICS CORNER



ROBOTICS CORNER

01211626904

www.roboticscorner.tech

Robotics Corner

Robotics Corner

Robotics Corner

Robotics Corner