Robot Software Architectures

What is an Architecture:

- When working on a complex system such as a robot, it is important to understand the elements of that system and the manner in which those interact with each other.
  - Without such an understanding, it is difficult to keep track of the system's design & its evolution.
- This understanding is typically captured by an architecture, which is a model that describes how a system works and what it needs so that it can successfully perform its operation.
- An architecture abstracts away various aspects of the system's operation, such that it can be defined at different levels of abstraction.

An architecture is a description of a system, namely its design and operation, at a given level of abstraction.

Architecture Abstraction levels:

Architectures can be observed at three different levels:

1. Operational architecture: Specifies what a system should do without describing how.
2. System architecture: Defines the components of a system and their connections.
3. Technical architecture: Describes how the system actually works at an implementation level (down to the algorithmic level)

What is a Software Architecture:

- When developing a software system, the first step is to define how the software should be organized.
- The organization of software in terms of components & connections between those is captured by a software architecture.
- An architecture is a programming language-independent representation of a system that captures the essence of what a system should do and in which particular way.
- A software architecture is a system model that defines the components of a system and the manner in which they are organised.

Objectives of System architectures:

- Common system understanding: A clear architectural representation enables all developers to unambiguously understand how a system works.
- Simplified modifiability: If the structure of a software architecture is known, the problem of how to modify or extend the system is considerably simplified.
- Fault identification: Knowledge of an architecture can help identify potential system errors and means of mitigating those.
- Simplified communication: An explicit representation of a software architecture (e.g. in terms of a visual diagram) can serve as a means of communication with stakeholders.

General Properties of (Robot) Software Architectures

- Abstraction: The architecture should be general and abstract implementation details away.
- Modularity: Components should be specialised and as uncoupled as possible.
- Change anticipation: Incremental algorithmic changes should be supported without major architectural changes.
- Niche targetability: The architecture should work well for the intended application.
- Robustness: Components should be able to handle faults and exceptions.

Robot Architecture types:

- A robot architecture considers the interplay between the hardware and software aspects of a robotic system.
- A robot architecture is a commitment to a particular paradigm on how the behavior of an intelligent system should emerge. Due to our limited understanding of intelligence, there is no universally accepted intelligence paradigm, so there is a multitude of operational robot architectures in the literature.

The development of robot architectures is an envolving process, where insights from older architectures are used to define new, improved

Sense_Plan_Act Paradigm:
- For a long Period of time, the Predominant Paradigm for developing robot Software was based on the Sense_Plan_Act Paradigm.
- It decouples the Process of Sensing the environment, Generating Plans of actions, and action execution.
- A robot based on the Sense_Plan_act Paradigm Continuously Performs deliberative Processes.
- The Sense_Plan_act Paradigm models a robot's as a Continuous loop of Perceiving its environment (sensing), interpreting the information to create Plans (Planning), and executing actions (acting)

Behaviour_Based Architectures:
- A reactive architecture, where the robot's operation is Controlled by Composing behaviorss.
- one famous example of Such architecture is the Subsumption architecture, where behaviors are able to inhibit other behaviors based on arbitration mechanism.
- Behavior_based architectures organise the operation of a robot into Specialised behaviors that Can be Composed to Perform Complex operations.

Three_Tier (3T) Architectures:
- A Common architecture in robotics that attempts to Combine the elements of both Sense_Plan_act and behaviour_based architecture is the 3T architecture.
- A 3T architecture has three hierarchically arranged elements:
  - Control: includes reactive behaviors, Performs Perceptual Processing, and takes Care of actuator Commands.
  - Executive: Manages the executions of tasks.
  - Planning: Performs task Planning based on all available information.
- A 3T architecture hierarchically decomposes different robot operation into three layers that Perform Planning, execution management, and environment interpretation & Control.

Hybrid Interactive Architecture:
- All Previous architecture types are missing aspects regarding interaction functionalities as well as Components to ensure the reliability of the operation.
- These aspects are essential to a Consider in a Contemporary robot architecture.
- A Prototypical example of Such an architecture that includes deliberative, reactive, as well as interactive elements is illustrated on the right.

Cognitive Architectures:
- Some robot architectures model aspects that are typical for biological Systems (e.g. long_ and short_term memory)
- Such architectures are called Cognitive architectures, which are directly based on Principles from natural Systems.
- A variety of Cognitive architectures exist, which typically emphasise different aspects of intelligence (e.g. affect of Continual learning)
- A Cognitive architecture is a model inspired by natural intelligence or directly models aspects thereof.

Prototypical Robot System Architecture: At a system level, most robot architectures typically include at the least the following five Components:
- Perception: Takes Care of Sensor data Processing and information extraction.
- Navigation: Performs Path Planning and trajectory execution.
- Cartographer: Collects information about the environment (maintains a world model)
- Planning: Generates high_level task Plans and monitors their execution.
- Motor Schema: Performs reactive Selection of motor actions and take Care of their execution.

System architecture Modelling :

It is usually useful to create architecture models by following certain conventions that unambiguously communicate the intent behind certain architectural decisions.

- An architecture design created using non-standard notation can confuse rather than enlighten.

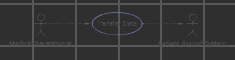
Unified Modelling Language (UML) : a standard visual representation of object-oriented systems.

UML includes different diagram types, each of which focuses on modelling a specific aspect of a system and its interaction with the external world.

- UML diagrams can be used for different purposes, such as idea brainstorming, system documentation, also for automatic code generation.

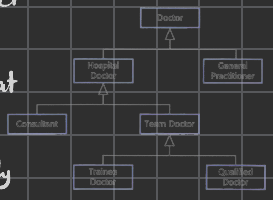
UML diagram types :

- **Use Case diagram**: the simplest UML diagram type that illustrates the interaction of a system and its users at a high level of abstraction.
  - In a use case diagram: stickmen represent actors in a system, Ellipses are processes through which interaction happens.
  - Use case diagrams can be useful during the requirement elicitation process.

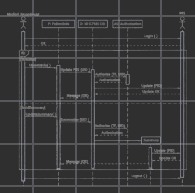


- **Activity diagram**: shows the flow of activities in a certain process.
  - In such a diagram: a full circle is the start of a process, while a full circle inside a circle is the end of a process. a round rectangle is an activity, a diamond represents a decision node, Arrows represent the direction of flow.
  - The Business Process Model and Notation (BPMN) is a similar graphical representation that is sometimes used instead of UML activity diagrams.

- **Class diagram**: shows the relation between different system components, which are modelled as classes that interact with each other.
  - A class diagram can be used to create a visual representation of a system architecture, such that it represents a system that is to be developed using object-oriented programming.
  - A class diagram hides the implementation details of classes, but includes details about how exactly classes are related to each other.



- **Sequence diagram**: To model concrete data flow in a system.
  - Sequence diagram represents how messages are passed between components so that a certain operation can be completed.
  - Sequence diagrams illustrate a complete interaction that is initiated by or involves a user.



- **State diagram**: behaviors are often represented using state-based paradigms, such as state machines or behaviour trees.
  - State diagrams can be particularly helpful in this context, as they model a system through its possible states and state transitions.
  - In a state diagram: rounded rectangles represent states, arrow labels are events that trigger particular transitions, Circles have the same semantics as in activity diagrams.