Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

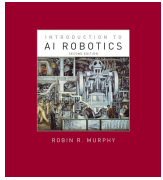
Institute for AI and Autonomous Systems

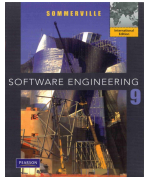
(Robot) Software Architectures

An Overview

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure





- Software architecture fundamentals
- Robot architecture types
- System architecture modelling

Software Architecture Fundamentals

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 and its evolution

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 and 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**

What is an Architecture?

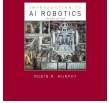
- When working on a complex system such as a robot, it is **important to understand the elements of that system and the manner in which those interact with each other**
  - Without such an understanding, it is difficult to keep track of the system's design and its evolution
- This understanding is typically 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

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 and 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

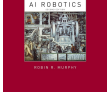
An architecture is a description of a system, namely its design and operation, at a given level of abstraction

Architecture Abstraction Levels



Architectures can be observed at three different levels:

Architecture Abstraction Levels



Architectures can be observed at three different levels:

1. **Operational architecture**: Specifies what a system should do without describing how

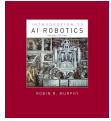
Architecture Abstraction Levels

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

Architecture Abstraction Levels



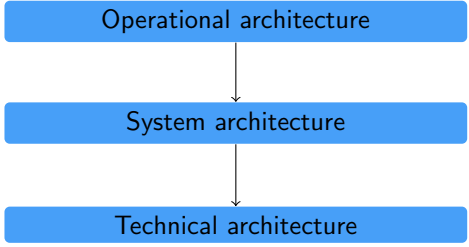
Architectures can be observed at three different levels:

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)

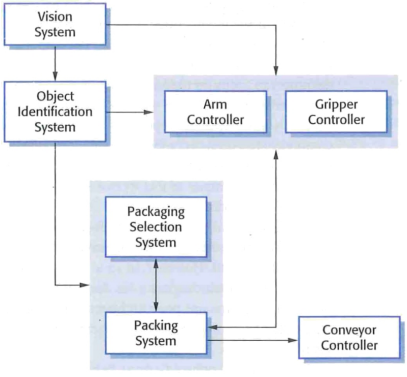
Architecture Abstraction Levels

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)

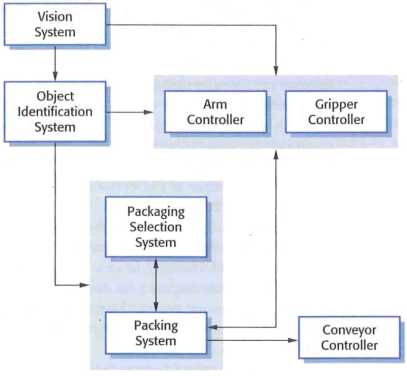


What is a Software Architecture?



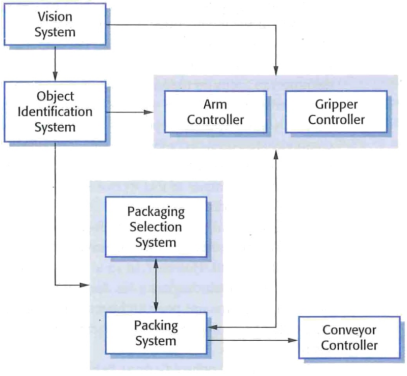
- When developing a software system, the first step (once the system's purpose is known) is to **define how the software should be organised**

What is a Software Architecture?



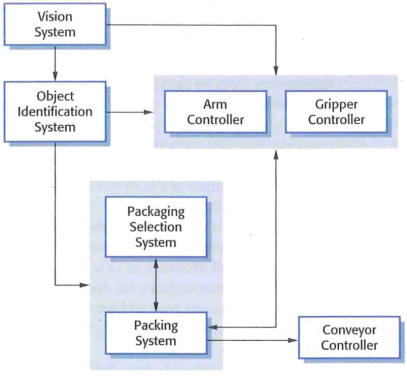
- When developing a software system, the first step (once the system's purpose is known) is to **define how the software should be organised**
- The organisation of software in terms of components and connections between those is captured by a **software architecture**

What is a Software Architecture?



- When developing a software system, the first step (once the system's purpose is known) is to **define how the software should be organised**
- The organisation of software in terms of components and 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**

What is a Software Architecture?



- When developing a software system, the first step (once the system's purpose is known) is to **define how the software should be organised**
- The organisation of software in terms of components and 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

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

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

Objectives of System Architectures

Common system understanding

A clear architectural representation enables all developers to unambiguously understand how a system works

Fault identification

Knowledge of an architecture can help identify potential system errors and means of mitigating those

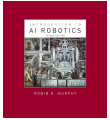
Simplified modifiability

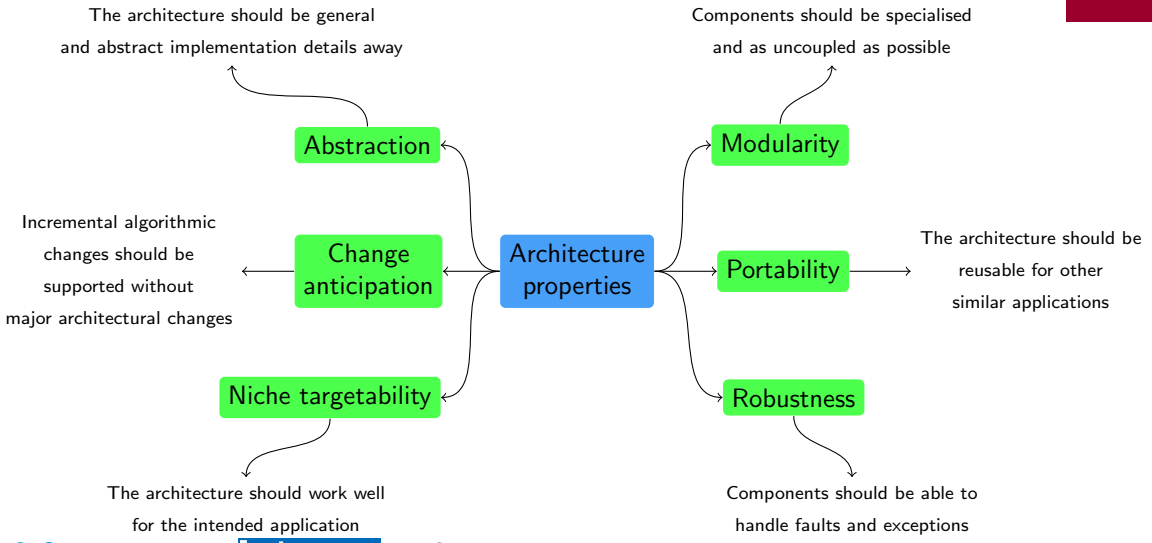
If the structure of a software architecture is known, the problem of how to modify or extend the system is considerably simplified

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





Robot Architecture Types

Robot Architectures

- A robot is an embodied software system; thus, **a robot architecture considers the interplay between the hardware and software aspects of a robotic system**

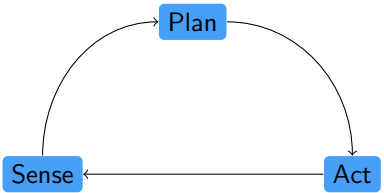
Robot Architectures

- A robot is an embodied software system; thus, **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 behaviour 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

Robot Architectures

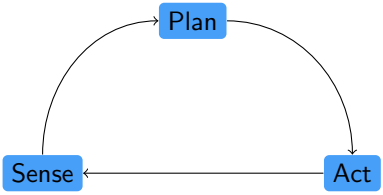
- A robot is an embodied software system; thus, **a robot architecture considers the interplay between the hardware and software aspects of a robotic system**
- **A robot architecture is a commitment to a particular paradigm on how the behaviour of an intelligent system should emerge**
  - Due to our 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 evolving process, where **insights from older architectures are used to define new, improved architectures**

Sense-Plan-Act Paradigm



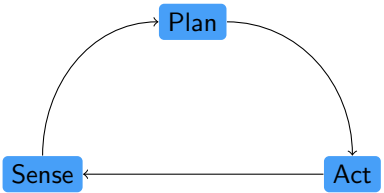
- For a long period of time, the predominant paradigm for developing robot software was based on the **sense-plan-act** paradigm

Sense-Plan-Act Paradigm



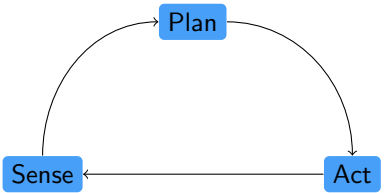
- For a long period of time, the predominant paradigm for developing robot software was based on the **sense-plan-act** paradigm
- As the name suggests, **sense-plan-act decouples the processes of sensing the environment, generating plans of actions, and action execution**

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
- As the name suggests, **sense-plan-act decouples the processes of sensing the environment, generating plans of actions, and action execution**
- A robot based on the sense-plan-act paradigm continuously performs **deliberative processes**

Sense-Plan-Act Paradigm

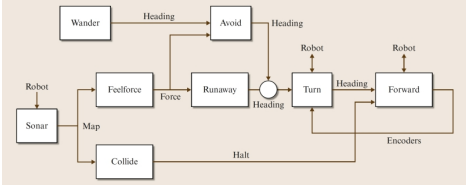


- For a long period of time, the predominant paradigm for developing robot software was based on the **sense-plan-act** paradigm
- As the name suggests, **sense-plan-act decouples the processes of sensing the environment, generating plans of actions, and action execution**
- A robot based on the sense-plan-act paradigm continuously performs **deliberative processes**

The sense-plan-act paradigm models a robot's decision-making process as a continuous loop of perceiving its environment (sensing), interpreting the information to create plans (planning), and executing actions (acting)

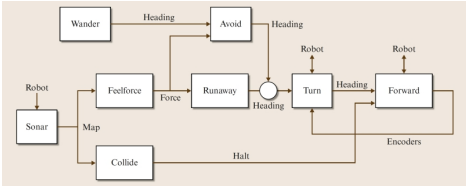
Behaviour-Based Architectures

- In the sense-plan-act paradigm, the action execution process is performed without taking sensor information into account — **problematic if quick reactions are needed**



Subsumption example

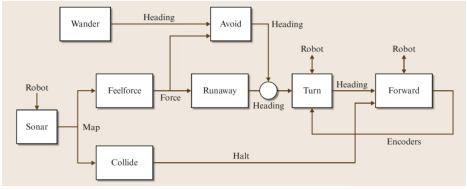
Behaviour-Based Architectures



Subsumption example

- In the sense-plan-act paradigm, the action execution process is performed without taking sensor information into account — **problematic if quick reactions are needed**
- An alternative to deliberative architectures is a reactive architecture, where **the robot's operation is controlled by composing behaviours**

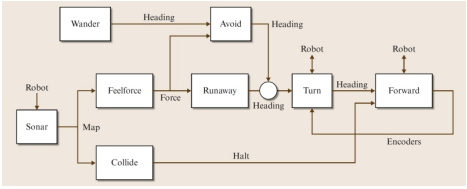
Behaviour-Based Architectures



Subsumption example

- In the sense-plan-act paradigm, the action execution process is performed without taking sensor information into account — **problematic if quick reactions are needed**
- An alternative to deliberative architectures is a reactive architecture, where **the robot's operation is controlled by composing behaviours**
- One famous example of such an architecture is the **subsumption** architecture, where behaviours are able to inhibit other behaviours based on an **arbitration mechanism**

Behaviour-Based Architectures



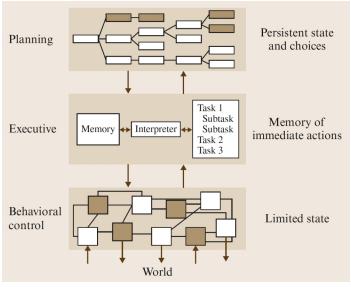
Subsumption example

- In the sense-plan-act paradigm, the action execution process is performed without taking sensor information into account — **problematic if quick reactions are needed**
- An alternative to deliberative architectures is a reactive architecture, where **the robot's operation is controlled by composing behaviours**
- One famous example of such an architecture is the **subsumption** architecture, where behaviours are able to inhibit other behaviours based on an **arbitration mechanism**

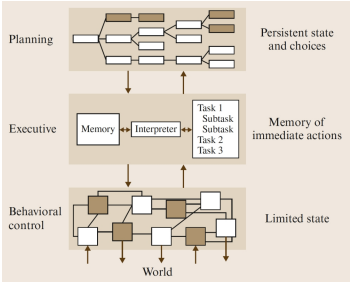
Behaviour-based architectures organise the operation of a robot into specialised behaviours that can be composed to perform complex operations

Three-Tier (3T) Architectures

- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight

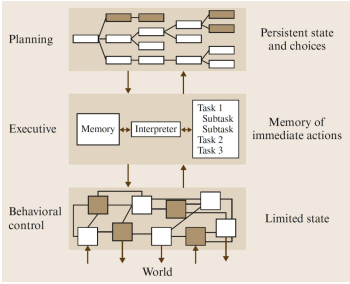


Three-Tier (3T) Architectures



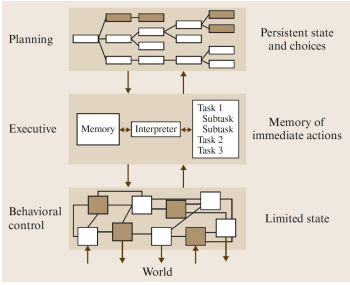
- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight
- A common architecture in robotics that attempts to combine the elements of both sense-plan-act and behaviour-based architectures is the 3T architecture

Three-Tier (3T) Architectures



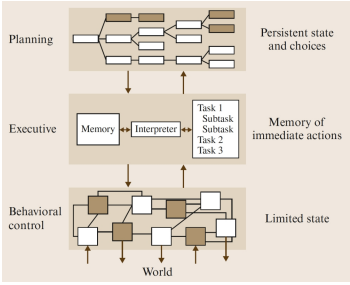
- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight
- A common architecture in robotics that attempts to combine the elements of both sense-plan-act and behaviour-based architectures is the 3T architecture
- As the name implies, a 3T architecture has three hierarchically arranged elements:

Three-Tier (3T) Architectures



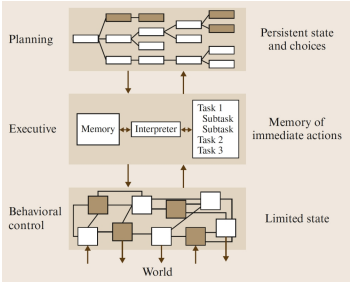
- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight
- A common architecture in robotics that attempts to combine the elements of both sense-plan-act and behaviour-based architectures is the 3T architecture
- As the name implies, a 3T architecture has three hierarchically arranged elements:
  - **Control**: Includes reactive behaviours, performs perceptual processing, and takes care of actuator commands

Three-Tier (3T) Architectures



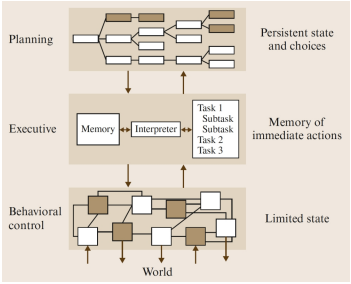
- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight
- A common architecture in robotics that attempts to combine the elements of both sense-plan-act and behaviour-based architectures is the 3T architecture
- As the name implies, a 3T architecture has three hierarchically arranged elements:
  - **Control**: Includes reactive behaviours, performs perceptual processing, and takes care of actuator commands
  - **Executive**: Manages the execution of tasks

Three-Tier (3T) Architectures



- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight
- A common architecture in robotics that attempts to combine the elements of both sense-plan-act and behaviour-based architectures is the 3T architecture
- As the name implies, a 3T architecture has three hierarchically arranged elements:
  - **Control**: Includes reactive behaviours, performs perceptual processing, and takes care of actuator commands
  - **Executive**: Manages the execution of tasks
  - **Planning**: Performs task planning based on all available information

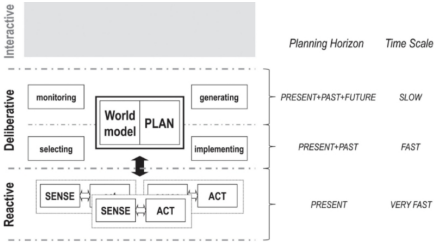
Three-Tier (3T) Architectures



- By missing a deliberation mechanism, behaviour-based architectures are challenging to use for processes that require foresight
- A common architecture in robotics that attempts to combine the elements of both sense-plan-act and behaviour-based architectures is the 3T architecture
- As the name implies, a 3T architecture has three hierarchically arranged elements:
  - **Control**: Includes reactive behaviours, performs perceptual processing, and takes care of actuator commands
  - **Executive**: Manages the execution of tasks
  - **Planning**: Performs task planning based on all available information

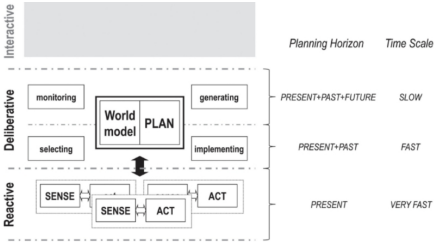
A 3T architecture hierarchically decomposes different robot operation into three layers that perform planning, execution management, and environment interpretation and control

Hybrid Interactive Architecture



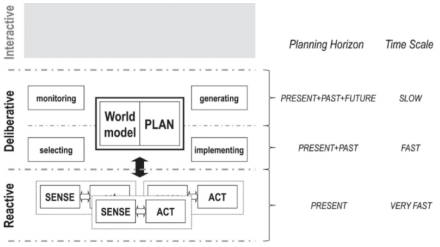
- All previous architecture types are missing aspects regarding **interaction functionalities** as well as **components to ensure the reliability of the operation**

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 consider in a contemporary robot architecture**

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 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 left

Cognitive Architectures

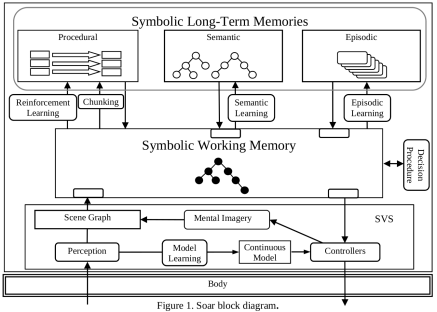


Figure 1. Soar block diagram.

J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," in *Cognitive Robotics Workshop at the 26th AAAI Conf. Artificial Intelligence*, 2012.

- Unlike typical agent architectures that are not concerned with biological plausibility, some robot architectures model aspects that are typical for biological systems (e.g. long- and short-term memory)

Cognitive Architectures

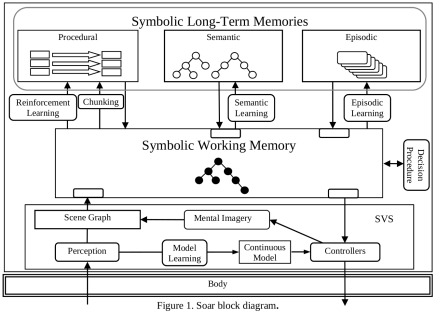


Figure 1. Soar block diagram.

J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," in *Cognitive Robotics Workshop at the 26th AAAI Conf. Artificial Intelligence*, 2012.

- Unlike typical agent architectures that are not concerned with biological plausibility, 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**

Cognitive Architectures

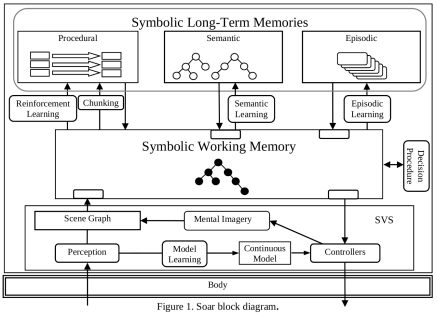


Figure 1. Soar block diagram.

J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," in *Cognitive Robotics Workshop at the 26th AAAI Conf. Artificial Intelligence*, 2012.

- Unlike typical agent architectures that are not concerned with biological plausibility, 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 or continual learning)

Cognitive Architectures

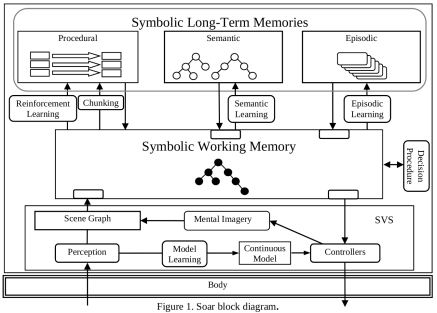


Figure 1. Soar block diagram.

J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," in *Cognitive Robotics Workshop at the 26th AAAI Conf. Artificial Intelligence*, 2012.

- Unlike typical agent architectures that are not concerned with biological plausibility, 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 or continual learning)
- Cognitive architectures are treated in more detail in my "Cognitive Robotics" elective course

Cognitive Architectures

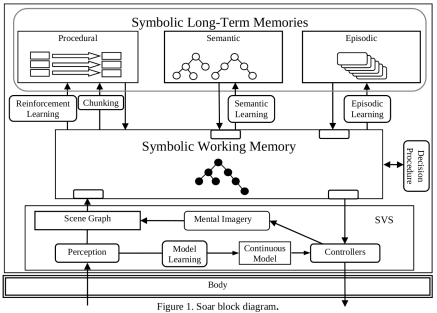


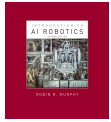
Figure 1. Soar block diagram.

J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," in *Cognitive Robotics Workshop at the 26th AAAI Conf. Artificial Intelligence*, 2012.

- Unlike typical agent architectures that are not concerned with biological plausibility, 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 or continual learning)
- Cognitive architectures are treated in more detail in my "Cognitive Robotics" elective course

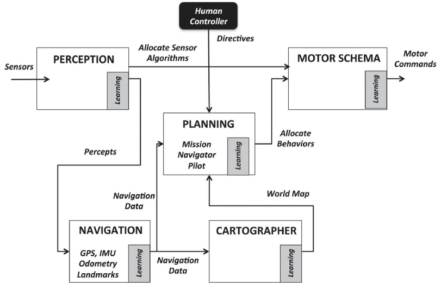
A cognitive architecture is a model inspired by natural intelligence or directly models aspects thereof

Prototypical Robot System Architecture



At a system level, most robot architectures typically include at 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 action and takes care of their execution



System Architecture Modelling

Architectural Modelling Standards

Not all architectural models are equally useful

- Architectures are a useful abstraction concept, but it is usually useful to create architectural 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

Architectural Modelling Standards

Not all architectural models are equally useful

- Architectures are a useful abstraction concept, but it is usually useful to create architectural 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
- Architecture modelling paradigms and tools have evolved over many years of developing complex systems; through these efforts, certain **de facto discipline-agnostic modelling standards have been accepted**

Architectural Modelling Standards

Not all architectural models are equally useful

- Architectures are a useful abstraction concept, but it is usually useful to create architectural 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
- Architecture modelling paradigms and tools have evolved over many years of developing complex systems; through these efforts, certain **de facto discipline-agnostic modelling standards have been accepted**
- On the following few slides, we will take a closer look at one widely accepted modelling paradigm — the Unified Modelling Language (UML)

Unified Modelling Language (UML)

- The Unified Modelling Language (UML) is a **standard visual representation of object-oriented systems**
  - But aspects of UML can also be used to represent systems that are not based on object-oriented programming

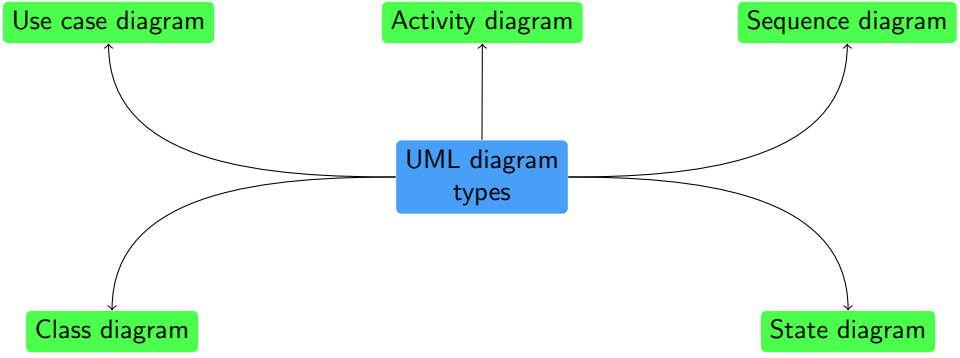
Unified Modelling Language (UML)

- The Unified Modelling Language (UML) is a **standard visual representation of object-oriented systems**
  - But aspects of UML can also be used to represent systems that are not based on object-oriented programming
- UML includes **different diagram types**, each of which focuses on modelling a specific aspect of a system and its interaction with the external world

Unified Modelling Language (UML)

- The Unified Modelling Language (UML) is a **standard visual representation of object-oriented systems**
  - But aspects of UML can also be used to represent systems that are not based on object-oriented programming
- 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 with stakeholders, system documentation**, but also for **automatic code generation** (in the context of model-driven engineering)

UML Diagram Types



Use Case Diagram

- A use case diagram is the simplest UML diagram type that **illustrates the interaction of a system and its users at a high level of abstraction**

Use Case Diagram

- A use case diagram is 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:

Use Case Diagram

- A use case diagram is 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 (this can be people interacting with the system, or the system itself)

Use Case Diagram

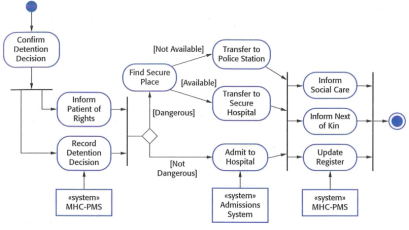
- A use case diagram is the simplest UML diagram type that **illustrates the interaction of a system and its users at a high level of abstraction**
- In a use case diagram:
  - Stickmen represent actors in a system (this can be people interacting with the system, or the system itself)
  - Ellipses are processes through which interaction happens

Use Case Diagram

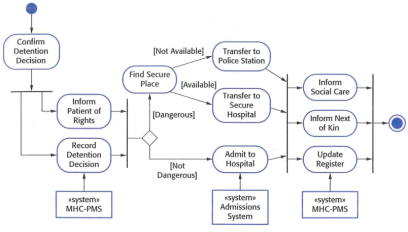
- A use case diagram is the simplest UML diagram type that **illustrates the interaction of a system and its users at a high level of abstraction**
- In a use case diagram:
  - Stickmen represent actors in a system (this can be people interacting with the system, or the system itself)
  - Ellipses are processes through which interaction happens
- Use case diagrams can be useful during the requirement elicitation process

Activity Diagram

- The objective of an activity diagram is to show **the flow of activities in a certain process**

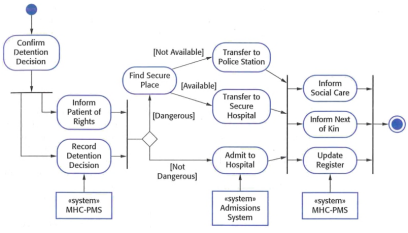


Activity Diagram



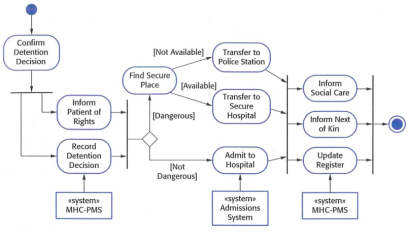
- The objective of an activity diagram is to show **the flow of activities in a certain process**
- In such a diagram:

Activity Diagram



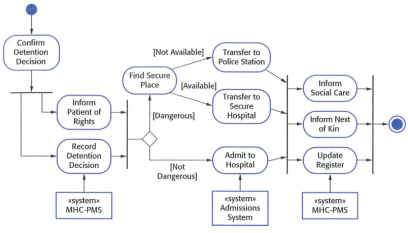
- The objective of an activity diagram is to show **the flow of activities in a certain process**
- In such a diagram:
  - A full circle represents the start of a process, while a full circle inside a circle is the end of the process

Activity Diagram



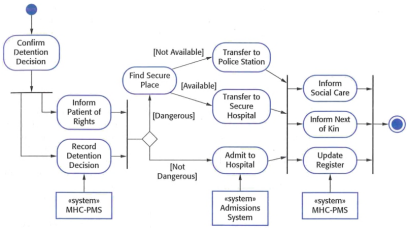
- The objective of an activity diagram is to show **the flow of activities in a certain process**
- In such a diagram:
  - A full circle represents the start of a process, while a full circle inside a circle is the end of the process
  - A round rectangle is an activity

Activity Diagram



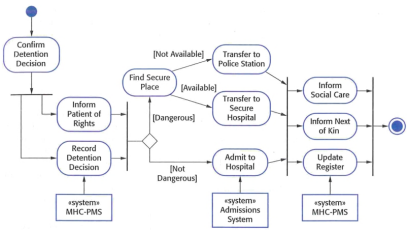
- The objective of an activity diagram is to show **the flow of activities in a certain process**
- In such a diagram:
  - A full circle represents the start of a process, while a full circle inside a circle is the end of the process
  - A round rectangle is an activity
  - A diamond represents a decision node

Activity Diagram



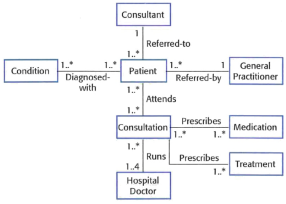
- The objective of an activity diagram is to show **the flow of activities in a certain process**
- In such a diagram:
  - A full circle represents the start of a process, while a full circle inside a circle is the end of the process
  - A round rectangle is an activity
  - A diamond represents a decision node
  - Arrows represent the direction of flow

Activity Diagram

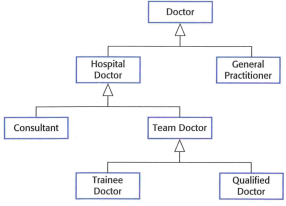


- The objective of an activity diagram is to show **the flow of activities in a certain process**
- In such a diagram:
  - A full circle represents the start of a process, while a full circle inside a circle is the end of the process
  - A round rectangle is an activity
  - A diamond represents a decision 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



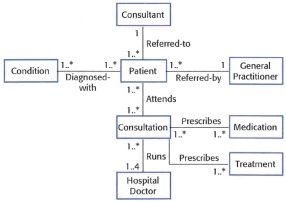
Class diagram illustrating associations between classes



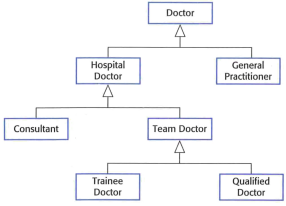
Class diagram representing generalisation relations between classes

- The purpose of a class diagram is to show **the relation between different system components**, which are modelled as classes that interact with each other

Class Diagram



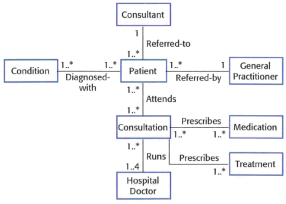
Class diagram illustrating associations between classes



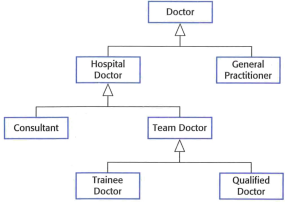
Class diagram representing generalisation relations between classes

- The purpose of a class diagram is to show **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**

Class Diagram



Class diagram illustrating associations between classes

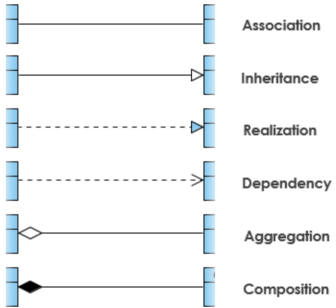


Class diagram representing generalisation relations between classes

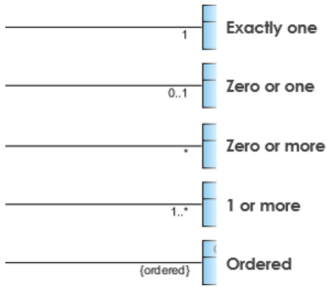
- The purpose of a class diagram is to show **the relation between different system components**, which are modelled as classes that interact with each other
- A class diagram can be used to create a visual representation of a system architecture, such that it **represents a system that is to be developed 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**

Class Diagram Relation Types

- UML class diagrams allow showing different types of relations between classes through various relation types
- Relationships between classes can also have explicit cardinalities (as illustrated on the previous slide)



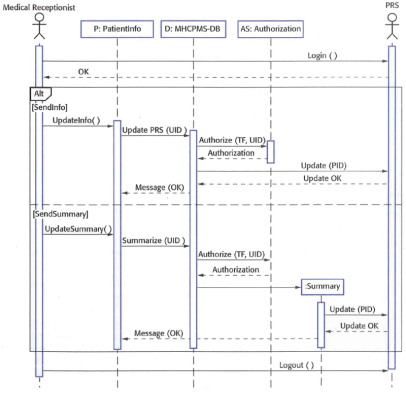
Relation types



Cardinality types

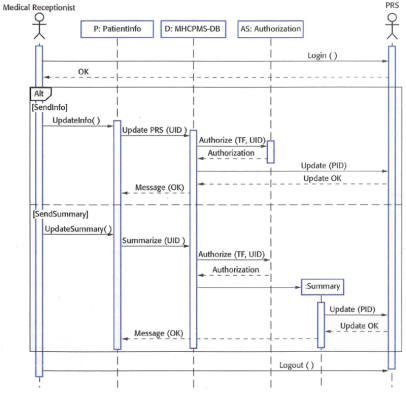
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

Sequence Diagram



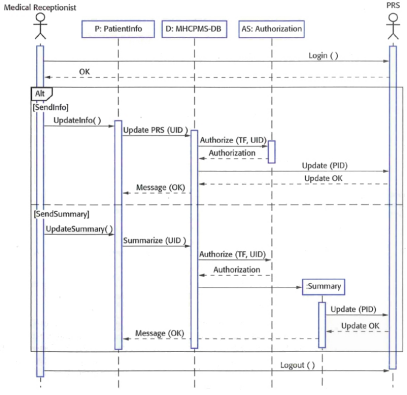
- To model concrete data flow in a system, sequence diagrams can be used

Sequence Diagram



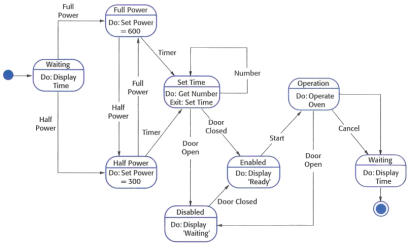
- To model concrete data flow in a system, sequence diagrams can be used
- Sequence diagrams represent **how messages are passed between components so that a certain operation can be completed**

Sequence Diagram



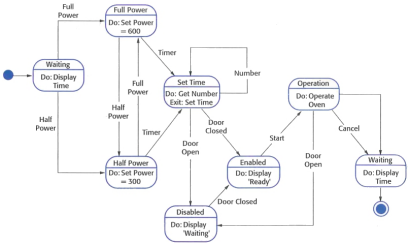
- To model concrete data flow in a system, sequence diagrams can be used
- Sequence diagrams represent **how messages are passed between components so that a certain operation can be completed**
- Typically, sequence diagrams illustrate a complete interaction that is initiated by or involves a user

State Diagram



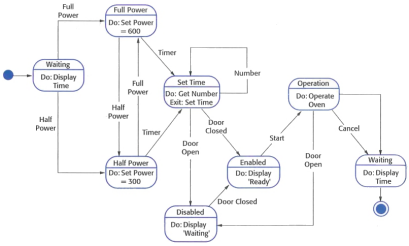
- In robotics, we typically care about **the current state of the robot and the transitions as a result of the robot's actions**; thus, behaviours are often represented using state-based paradigms, such as state machines or behaviour trees

State Diagram



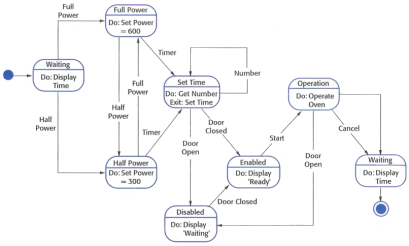
- In robotics, we typically care about **the current state of the robot and the transitions as a result of the robot's actions**; thus, behaviours are often represented using state-based paradigms, such as state machines or behaviour trees
- State diagrams can be particularly helpful in this context, as they **model a system through its possible states and state transitions**

State Diagram



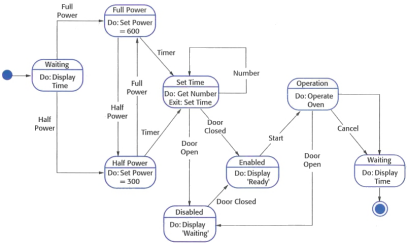
- In robotics, we typically care about **the current state of the robot and the transitions as a result of the robot's actions**; thus, behaviours are often represented using state-based paradigms, such as state machines or behaviour trees
- State diagrams can be particularly helpful in this context, as they **model a system through its possible states and state transitions**
- In a state diagram:

State Diagram



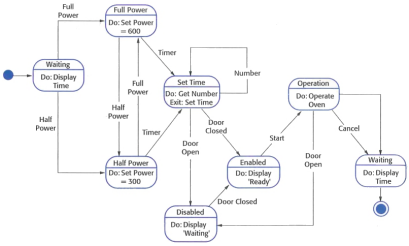
- In robotics, we typically care about **the current state of the robot and the transitions as a result of the robot's actions**; thus, behaviours are often represented using state-based paradigms, such as state machines or behaviour trees
- State diagrams can be particularly helpful in this context, as they **model a system through its possible states and state transitions**
- In a state diagram:
  - Rounded rectangles represent states

State Diagram



- In robotics, we typically care about **the current state of the robot and the transitions as a result of the robot's actions**; thus, behaviours 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

State Diagram



- In robotics, we typically care about **the current state of the robot and the transitions as a result of the robot's actions**; thus, behaviours 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

Summary

- Architectures represent an abstraction of the design and operation of a system, and can be done at different abstraction levels
- Software architectures model the component design of a system as well as the interaction between system components
- There are various operational architectures used in robotics, such as sense-plan-act, behaviour-based, three-tier, hybrid, and cognitive architectures
- UML is a de facto standard notation for system and component modelling, which includes multiple diagram types, such as use case, activity, class, sequence, and state diagrams