Autonomous and Collaborative Robotics

Alessandro Marcelletti

alessandro.marcelletti@unicam.it

A.Y. 2025/2026

Course introduction

The course will be divided into:

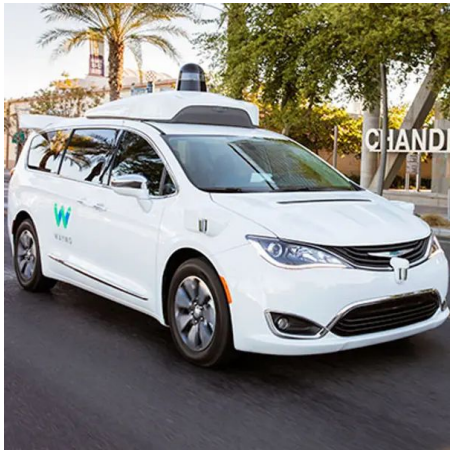
1. Theoretical module
   a. Introduction to Robotics
   b. Robotics software programming and controlling
   c. Multi-robot systems
   d. Self-adaptive and autonomous systems
   e. OMG Data Distribution Service (DDS)
   f. Robot Operating System (ROS)
2. Practical module
   a. ROS
   b. group projects

What is a robot?

“is an autonomous machine capable of sensing its environment, carrying out computations to make decisions, and performing actions in the real world.”[1]

[1] https://robotsguide.com/learn/

Spot the robot



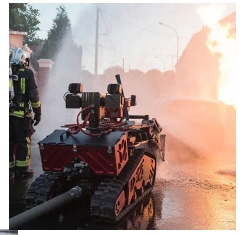
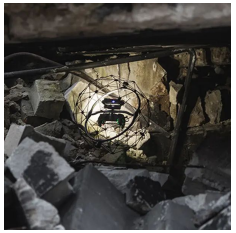
Types of robot

- Aerospace
- Aquatic
- Autonomous vehicles
- Consumer
- Delivery
- Disaster response
- Drones
- Educational
- Entertainment
- Exoskeletons
- Humanoid
- Industrial
- Medical
- Military & Security
- Research
- Service
- Social
- Telepresence

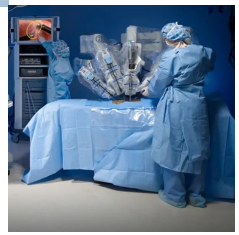
Types of robot

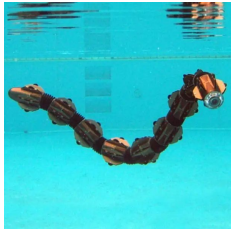




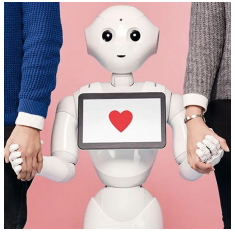









Are robot recent?

The definition of robot has been established in 1920 from the term ***robota*** (subordinate labour)

However, the idea of human-being machines can be found from ancient legends (Hephaestus, 3500 BC) to “modern” scientists (Leonardo Da Vinci, 1500 AD)

Mechanism and Actuation

Robot Mechanics

The **kinematic skeleton** of a robot is modeled as a series of **links connected by joints** forming a serial chain

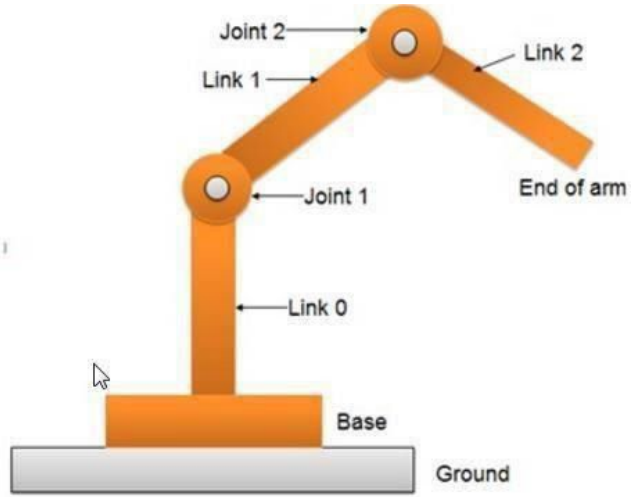
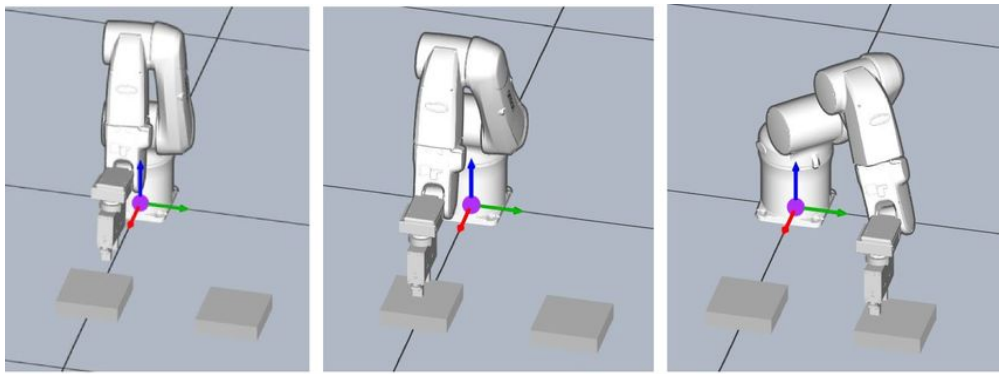


Image from: Yaseen, Syed, and Jaya Prakash. "Analysis of numerical method on inverse kinematics of robotic arm welding with artificial intelligence." Journal of Physics: Conference Series. Vol. 1964. No. 6. IOP Publishing, 2021.

Robot Mechanics

The position and orientation of a rigid body in space are collectively termed the ***pose***



(a) Initial pose (b) Pick pose (c) Place pose

Image from: Iwamura, Shintaro, et al. "Cable path optimization by fixing multiple guides on one link for industrial robot arms." The International Journal of Advanced Manufacturing Technology 129.9 (2023): 4275-4292.

Some definitions

The physical structure (e.g., beams, links, castings, shafts) of a robot that create its movable skeleton is termed the mechanical structure or ***mechanism*** of the robot

The electric, hydraulic and pneumatic motors and other elements that cause the links of the mechanism to move are called ***actuators***

The design of such components create the machine system that moves the robot starting from computer commands

Design a robot

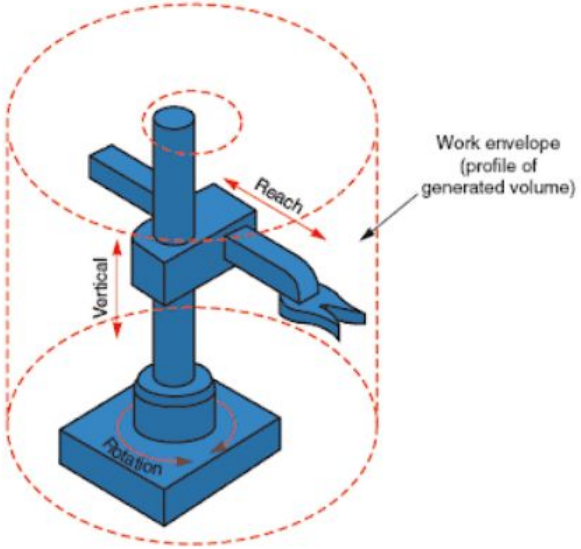
Robots were early designed to cope with a large number of tasks while they are now often designed for **specific applications** and to perform **limited sets of tasks**

This is a critical aspect to consider when designing a robot

System features: Work Envelope

Work envelope is the space in which a robot can operate

A work envelope is composed of the **workspace** (space required to accomplish a task) and the **robot volume** (space occupied from a robot)



System features: Load Capacity

Load capacity is a primary robot specification and is coupled with acceleration and speed. More important than peak velocity and it can be seen as the group of physical forces applicable to a robot

Load capacity can affect the performance of a robot during a task

System features: Kinematic Skeleton

Kinematic skeleton is the manipulatable shape and size and they strictly depends on different requirements (workspace, precision, load capacity)

Kinematics and Kinetics

Robot dynamics can be separated into:

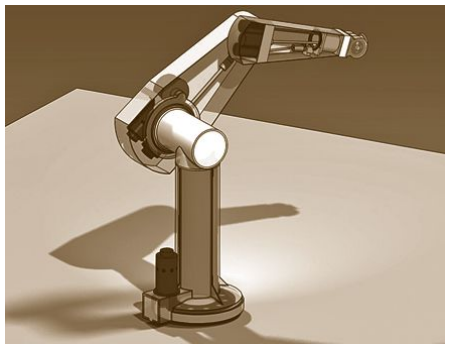
Kinematics: movement depending on robot mechanical structure

Kinetics: forces acting on the system

Kinematics and Kinetics: Robot Topology

Kinematic skeleton of a robot is the series of links connected by **hinged** or **sliding** joints forming a **serial chain**

A single serial chain characterizes a ***serial robot***



A set of serial chains characterizes a ***parallel robot***

Source: Springer Handbook of Robotics, Bruno Siciliano, Oussama Khatib (2016)

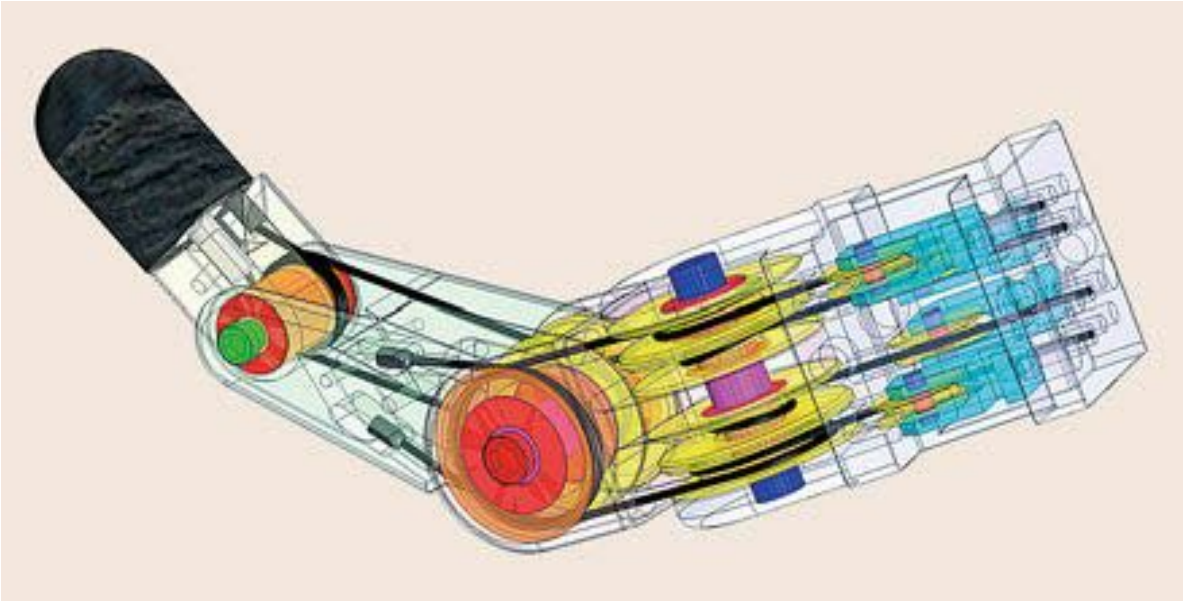
Serial Robots

A **serial chain robot** is a sequence of links and joints that begins at a base and ends with an end-effector

Usually, the first three joints position a reference point **wrist center** in the space while the last three form the **wrist** which orientate the end-effector around this point

The volume of space in which the wrist center can be places is the **reachable workspace**

Serial Robots



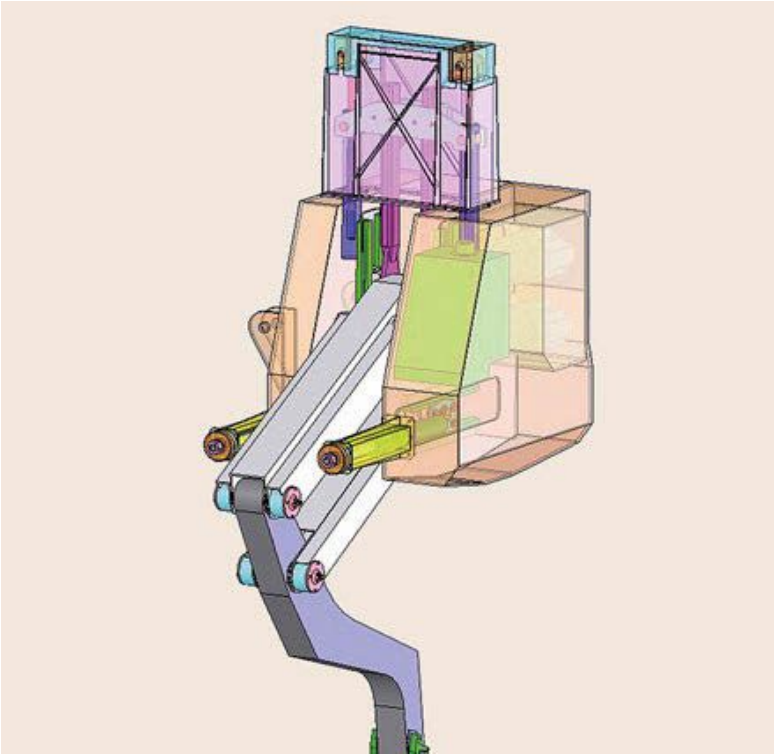
Complex finger of a robot

Parallel Robots

A **parallel robot** is a robotic system in which two or more serial chains robots support an end-effector

The **workspace** of a parallel robot is the intersection of the workspaces of the individual supporting chains

Parallel Robots



Adaptive suspension vehicle leg

Actuators

Actuators supply the motive power for robots, the most used are:

- electromagnetic (i.e., servo motors, stepper motors, DC motors)
- hydraulic
- pneumatic
- others… (thermal, shape-memory alloy, bimetallic, chemical)

Sensing and Estimation

Sensing and Estimation

Robots can **sense** the environment, capturing information to complete their knowledge about things by transforming a physical quantity into a computer representation

The sensing and estimation process can be:

Proprioception: sense and estimate to recover the state of the robot itself (PC)

Exteroception: sense and estimate to recover the state of the external world (EC)

Sensing and Perception

To sense the environment, a robot executes a **perception** process which uses (1) a **digital data** from a number of sensors/transducers, and (2) a **partial model** of the environment

Once sensory data has been matched against the world model it is possible to update the model with new information contained in the sensor data.

Sensing and Perception: Sensors

Sensors are physical components which can measure different data

- **proprioceptive**: degrees of freedom, temperature, velocity, motor current
- **exteroceptive**: distance to an object, interaction forces, vision

in different ways

- **active (A)**: emits energy into the environment, and measures properties of the environment based on the response (sonar, lidar)
- **passive (P):** sensors that are not active (camera, microphone)

Sensing and Perception: Sensors

proprioceptive measures robotic interaction with the environment

exteroceptive measures environmental data

Interoceptive measures internal robot data

Sensors: a classification

Classification	Sensor type	Sens	A/P
Tactile sensors	Switches/bumpers	EC	P
	Optical barriers	EC	A
	Proximity	EC	P/A
Haptic sensors	Contact arrays	EC	P
	Force/torque	PC/EC	P
	Resistive	EC	P
Motor/axis sensors	Brush encoders	PC	P
	Potentiometers	PC	P
	Resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacity encoders	EC	A
Heading sensors	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P
Beacon based (postion wrt an inertial frame)	GPS	EC	A
	Active optical	EC	A
	Radio frequency (RF) beacons	EC	A
	Ultrasound beacon	EC	A
	Reflective beacons	EC	A
Ranging	Capacitive sensor	EC	P
	Magnetic sensors	EC	P/A
	Camera	EC	P/A
	Sonar	EC	A
	Laser range	EC	A
	Structured light	EC	A
Speed/motion	Doppler radar	EC	A
	Doppler sound	EC	A
	Camera	EC	P
	Accelerometer	EC	P
Identification	Camera	EC	P
	Radio frequency identification RFID	EC	A
	Laser ranging	EC	A
	Radar	EC	A
	Ultrasound	EC	A
	Sound	EC	P

Robot Systems Architectures and Programming

Robot software systems are complex!

Robot software systems are complex to manage due to the need of coordinating different sensors and actuators while monitoring and reacting to unexpected situations

1. Asynchronously interaction
2. Real time interaction
3. Uncertainty
4. Dynamic environment

How to deal with complexity?

Modular decomposition of systems into simpler and independent modules

Facilitate asynchronous interaction with the environment

Hierarchical decomposition permits components to be built on top of other modular components

Reduce system complexity through abstraction

Software Development Tools

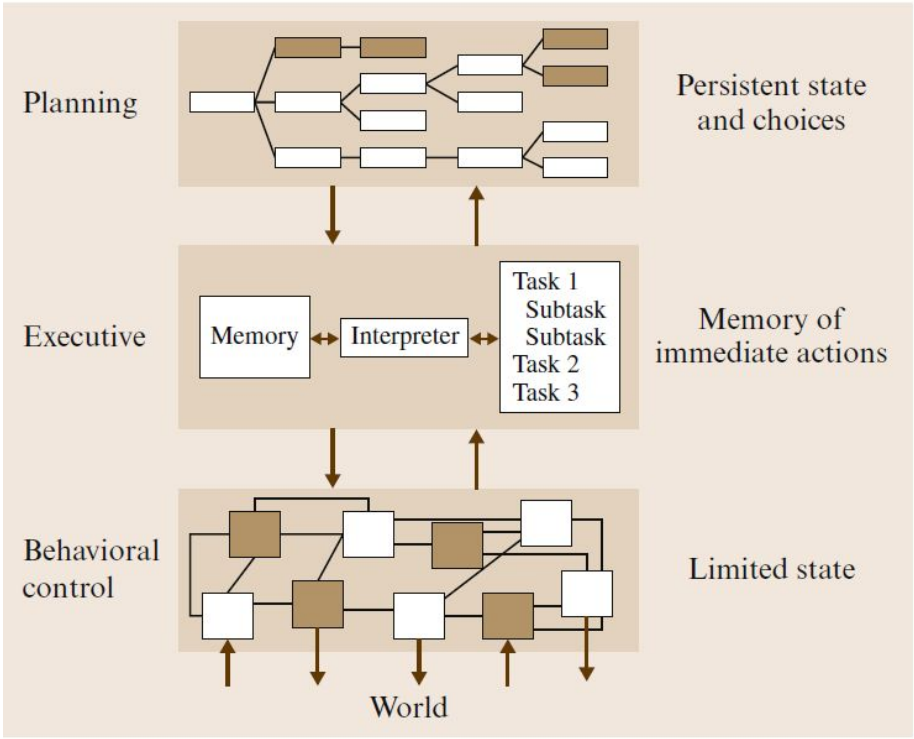
Just an idea

- CORBA and IPC libraries → message passing
- Subsumption and Skills languages → data-driven and real-time behaviours
- ESL and PLEXIL languages → high-level tasks
- ControlShell, Labview, OR-CCAD editors → assemble systems and generate code

Layered Robot Control Architectures

Three-layer-architecture (3T):

1. **behavioural control:** close to sensors and actuators
2. **executive layer:** chose the correct behaviours of the robot to achieve a task
3. **task-planning:** achieve the long-term goals of the robot



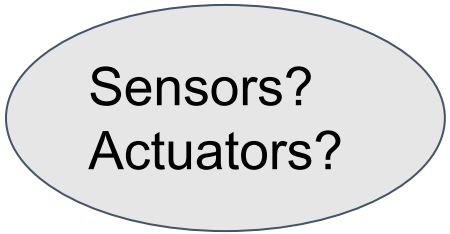
3T Behaviour Exercise

Which are the tasks of an office delivery robot that operates in a typical office building? (Office map is known)

3T Behaviour Exercise

Which are the tasks of an office delivery robot that operates in a typical office building?

1. Move to location while avoiding obstacles
2. Move down hallway while avoiding obstacles
3. Find a door
4. Find a door knob
5. Grasp a door knob
6. Turn a door knob
7. Go through door
8. Determine location
9. Find office number
10. Announce delivery



Three-layer-architecture: Behavioural Control

This is the **lowest level** in a robot architecture connecting sensors and actuators, this layer is composed of a small number of behaviors that **perceive the environment** and carry out the actions of the robot.

Specialized languages were developed during the years:

- ALFA
- Behavioural Language
- Rex
- Custom C/C++

Behavioural Layer Constraints

Speed and reactivity → algorithms must be linear in terms of time and complexity

How?

Little loop and search

Executive Layer

Interface between behavioural and planning layers, translates high-level plans into low-level invocations also taking care of **monitoring** and handling **exceptions**

Executive Layer Exercise

Our delivery robot has to deliver mail to a given office

How the layer operates?

Executive Layer Exercise

Our delivery robot has to deliver mail to a given office

1. Determine the sequence of corridors to move down
2. Determine intersections at which to turn
3. If there are doorways open and pass through the door
4. Announce that the person has mail and to concurrently monitor whether the mail has been picked up
5. Exception if mail is not picked up

Executive Layer Constraints

Hierarchical decomposition → temporal constraints based on serial, concurrent activities and more

Pick the right language for your case:

- RAP
- PRS
- ESL
- TDL
- LEXIL

Planning Layer

Determines the long-range activities of the robot based on high-level goals

Most common approaches are **Hierarchical Task Net** (HTN) planners and planners/schedules that decompose tasks into subtasks but at a higher level of abstractions and taking resources into account

Planning Layer Example

1. Look at the day's deliveries
2. Look at resources of the robot, and a map, and determine the optimal delivery routes and schedule, including when the robot should recharge.
3. Replanning when the situation changes

Planning Layer Integration

1. planning component **is invoked as needed** by the executive and returns a plan then turn silent (ATLANTIS, Remote Agent)

   → fits static environments

2. panning component **sends high-level tasks** down to the executive and monitors the progress

   → fits dynamic environments

Three-tiered Architecture

Office delivery robot example:

the **behavioral layer** is responsible for moving the robot around rooms and hallways, for avoiding obstacles, for opening doors, etc.

The **executive layer** coordinates the behavioral layer to achieve tasks such as leaving a room, going to an office, etc.

The **task-planning layer** is responsible for deciding the order of deliveries to minimize time, taking into account delivery priorities, scheduling, recharging

Three-tiered Architecture

Office delivery robot example:

Behavioral layer?

Executive layer?

Task-planning layer?

An example: GRACE (1)

Graduate **R**obot **A**ttending **C**onferenc**E** made by Carnegie Mellon, Naval Research Laboratory, Northwestern University, Metrica, and Swarthmore College

Challenge - attend the AAAI National Conference on Artificial Intelligence as a participant:

1. find the registration desk
2. register for the conference
3. find its way to a given location in time to give a technical talk about itself

An example: GRACE (2)

Techniques required:

1. localization in a dynamic environment
2. navigation in the presence of moving people
3. path planning
4. dynamic replanning
5. visual tracking of people, signs and landmarks
6. gesture and face recognition
7. speech recognition and natural language understanding
8. speech generation, knowledge representation, and social interaction with people

The GRACE architecture (1)

no! high-level plan was fixed and relatively straightforward

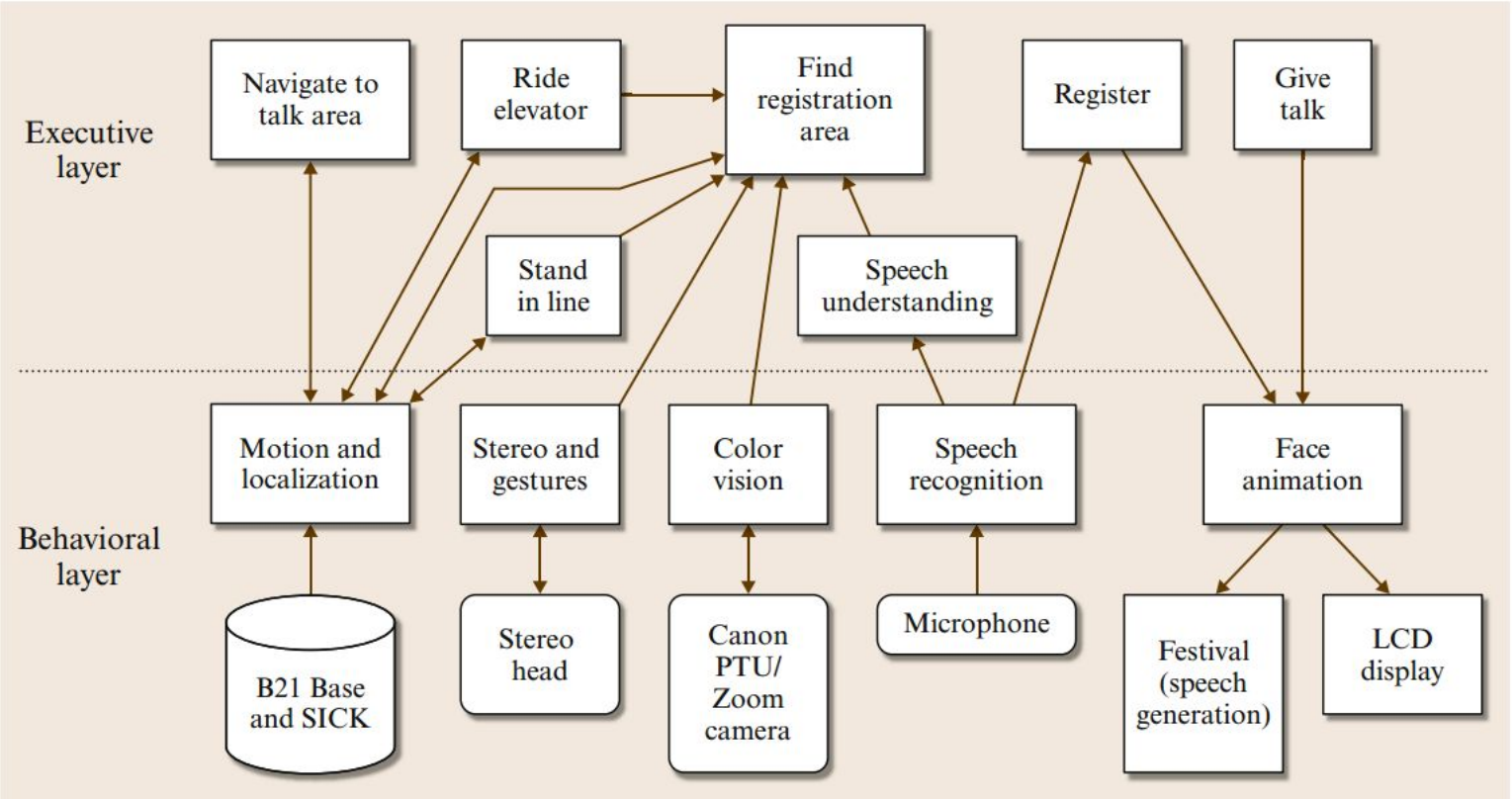
separate programs for achieving each subtask of the challenge

finding the registration desk, riding the elevator, standing in line, interacting with the person at the desk, navigating to the talk, and giving the talk

Individual processes that control pieces of hardware
synchronous, blocking and asynchronous, non blocking calls

motion and localization, speech recognition, speech generation, facial animation, color vision, and stereo vision

The GRACE architecture (2)



The Art of Robot Architectures

What computational capabilities will the robot have? What data will these computational capabilities produce

Who are the robot's users? What will they command the robot to do? What information will they want to see from

How will the robot be evaluated? What are the success criteria? What are the failure modes? What is the mitigation

What data is necessary to do the tasks? How will the robot obtain that data from the environment or from the users? What sensors will produce the data? What representations will be used for the data?

What actions are necessary to perform the tasks? How are those actions represented? How are those actions coordinated?

What are the tasks the robot will be performing? Are they long-term tasks? Short-term? User-initiated? Robot-initiated? Are the tasks repetitive or different across time?

Communication Styles

Many robot systems are designed as asynchronous processes that communicate using message passing

Client-Server: message request from the client is paired with a response from the server → deadlock if response is not received

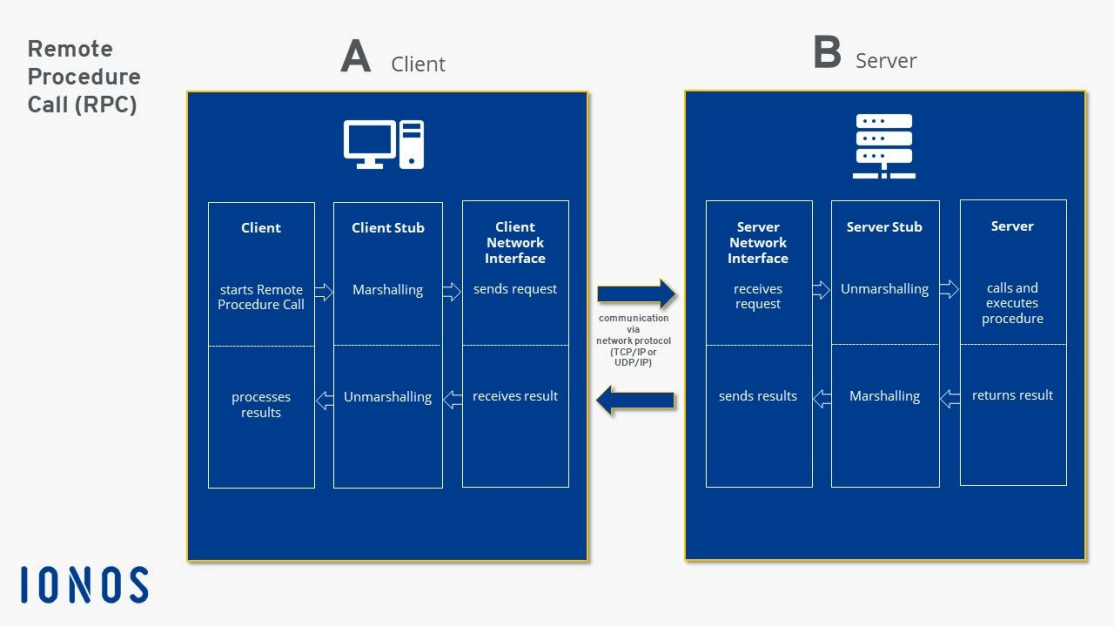
Publish-subscribe: messages are broadcast asynchronously and all modules that have previously indicated an interest in such messages receive a copy → more reliable, messages are assumed to arrive asynchronously

Client-server

Components talk directly with other components

Client-server: Remote Procedure Call

Remote Procedure Call (RPC): one component (the client) can call functions and procedures of another component (the server)



Advantages

Transparency: From the client's perspective, calling a remote function is like calling a local one

Communication: RPC handles the communication between client and server, usually over a network

Marshalling/Serialization: Data (arguments and return values) is packaged into a format suitable for network transmission

Decoupling: The client and server can be on different systems, potentially running different operating systems or languages

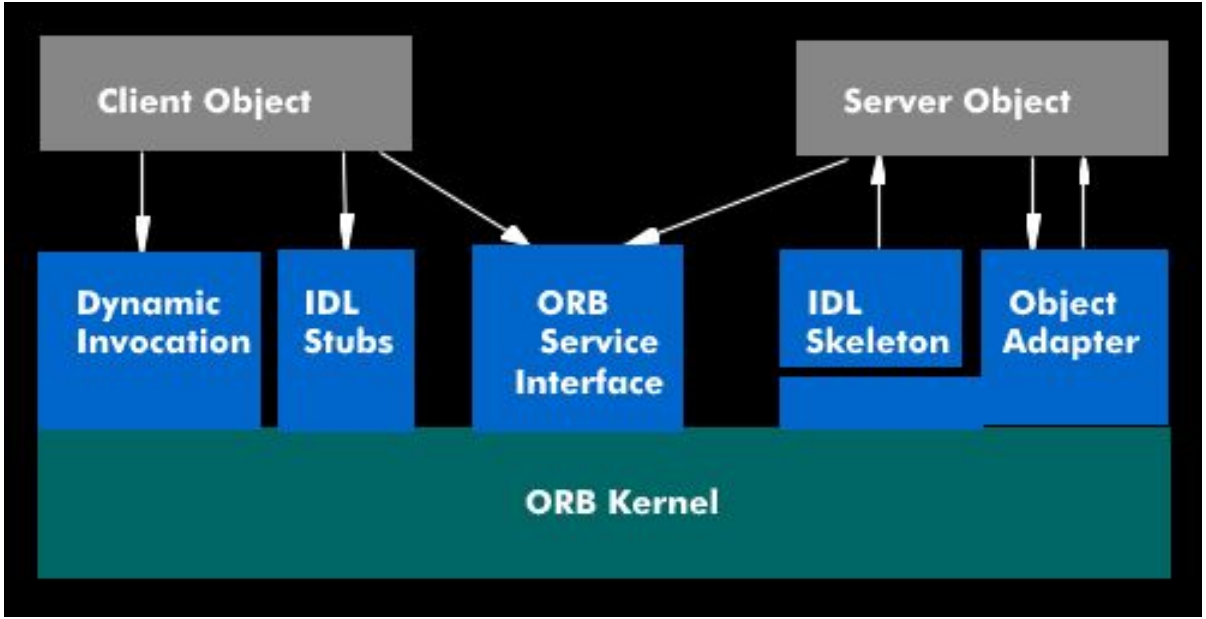
Client-server:
Common Object Request Broker Architecture

Common Object Request Broker Architecture (CORBA): one component calls object methods implemented by another component.

All method calls are defined in an **Interface Definition Language** (IDL) file that is language independent

CORBA Object Request Brokers (ORBs) are available for most major object oriented languages.

Object Request Broker Architecture (2)



source: https://www.itwissen.info/en/common-object-request-broker-architecture-OMG-CORBA-109207.html#gsc.tab=0

Invocation flow

1. Client Object calls a method
2. Stub marshals the request (method name + arguments) and passes it to the ORB Service Interface
3. ORB Service Interface locates the server object reference
4. ORB sends the marshalled request to the server-side IDL Skeleton
5. IDL Skeleton unmarshals the arguments and calls the Server Object method
6. Server Object executes the requested method and returns the result to the skeleton
7. IDL Skeleton marshals the return value and sends it back to the ORB
8. ORB delivers the marshalled response to the Client Stub
9. Client Stub unmarshals the response and returns it to the Client Object

CORBA pros and cons

Pros: when an IDL file is changed, all components that use that IDL can be recompiled automatic

Cons: Additional code into robot application

Publish-Subscribe

A component **publishes data** and any other component can **subscribe** to that **data** which is coordinated by a centralized process. Usually, components both pub and sub data.

Some examples are:

1. Data Distribution Service (DDS)
2. Inter-Process Communication (IPC)

XML is emerging as standard for defining data objects

Implementing Robotic Systems Architectures

Non-functional requirements such as **real-time performance**, **fault tolerance**, and **safety** are properties that emerge from multiple parts of a system and cannot be confined into individual modules

→born of efficient and specific implementation

Agile Robot Development Network (aRDnet)

Software suite for the development of distributed component-based system w**quirements**

Scalabile
Performance

The robot control system is structured as a **network of functional blocks and communication links** dis**work of computers connected by a fast digital bus

Decoupling

A block is a s**oftware module with multiple inp** and can be executed by individual processes or can be grou**on groups

Handling
Complexity

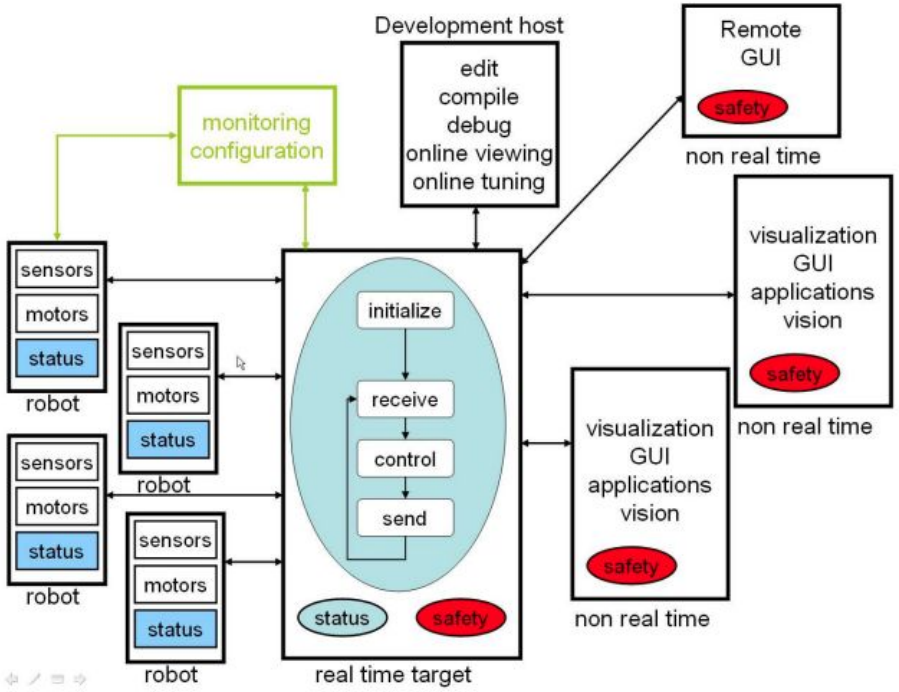
Definition

“a flexible, pragmatic and distributed software concept especially designed to support the development of complex mechatronic and robotic systems. It gives easy access to scalable computing performance (even in hard realtime) and is based on the abstract view on a robotic system as being a decentral net of calculation blocks and communication links.”

***Source**: Bauml, Berthold, and Gerd Hirzinger. "Agile robot development (ard): A pragmatic approach to robotic software." 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006.*

aRDnet complex robot system

1. **Robotic components**
   The physical end-effectors and sensing units
2. **Real time target**
   The core computation unit running hard realtime processes
3. **Non Real Time Computers**
   Networked computers handling higher-level intelligence and interaction
4. **Development Host**
   Supports the software engineering workflow
5. **Monitoring Host**
   Dedicated low-level system maintenance
6. **Remote GUI**
   Remote access for command and control



Yet Another Robot Platform (YARP)

Open-source project, same objectives of aRDnet and focuses on the development of distributed control systems for **high Degree Of Freedom** robots, such as the humanoid bodies

Lightweight software library for **concurrent and distributed programming**, data are exchanged through i**nput/output async ports** and can **manage multiple connections for a given module at different data rates** according to different protocols (Quality of Service!)

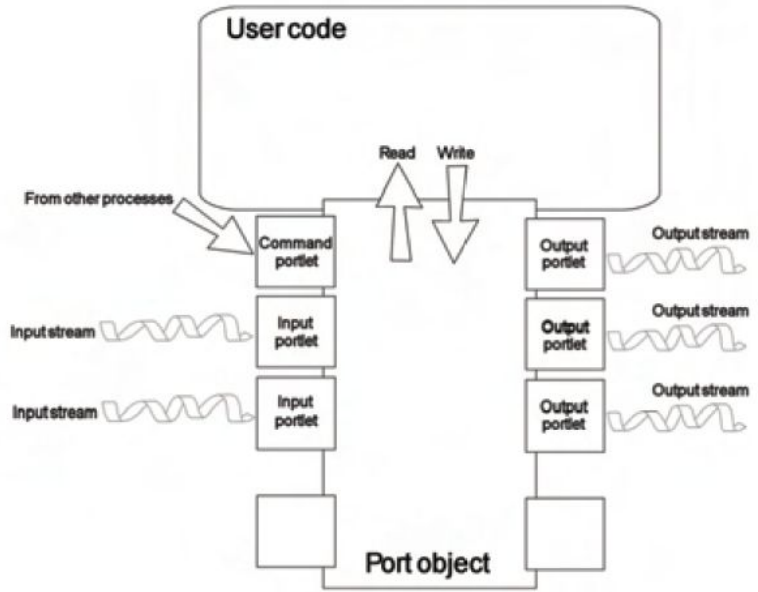
YARP system and port internal structure

1. **Ports**
   active objects managing multiple connections for input or output data. Ports are typically instantiated with a specific data type

2. **Name Server**
   Manages the network location of ports. It maps symbolic names to the required network address triplet

3. **Processes/Modules**
   Small, independent units of code that contain Ports to send or receive data

4. **Protocols**
   Ports support diverse communication protocols to optimize performance based on needs. Protocols include: TCP (reliable), UDP (faster point-to-point), Multicast (one-to-many, efficient for images)



***Source**: Metta, Giorgio, Paul Fitzpatrick, and Lorenzo Natale. "YARP: yet another robot platform." International Journal of Advanced Robotic Systems 3.1 (2006): 8.*

Open Robot Control Software (OROCOS)

One of the oldest open source framework in robotic, it focuses on hard **real-time capabilities** independent from middlewares and operating systems

OROCOS define a component model that specifies a standard behaviour for concurrent activities

OROCOS system

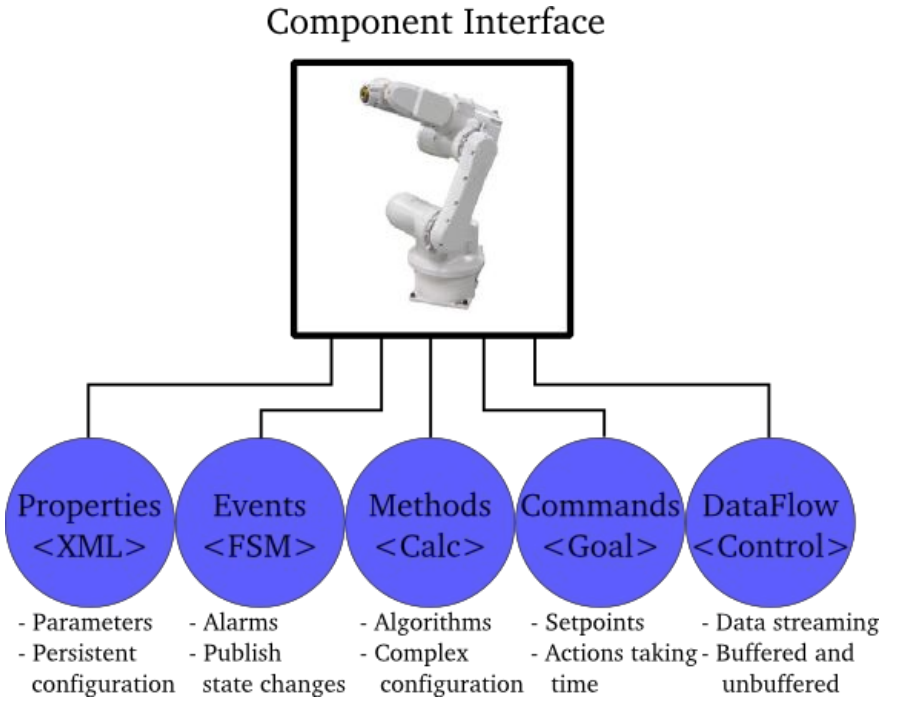
1. **Data-Flow** **Ports**
   A thread-safe mechanism used for communicating buffered or un-buffered data between components

2. **Properties**
   Parameters that are run-time modifiable, often stored in XML files

3. **Methods**
   Callable by other components to immediately calculate a result

4. **Commands**
   Instructions sent to the receiver to **'reach a goal'** generally not executed instantaneously, and the caller does not block but is informed of execution progress

5. **Events:** Functions that are executed when a change occurs within the system



Source: https://www.orocos.org/stable/documentation/rtt/current/doc-xml/orocos-overview.html

Smartsoft

Open source framework addresses issues related to **communication among software components** of a robotic control system

Defines a set of standard component interfaces (called communication patterns) with strictly defined interaction semantics

Robot Operating System (ROS)

ROS is a **message-based peer-to-peer** communication infrastructure supporting the easy integration of **independently** developed software **components**, called ROS nodes

Such **nodes wrap robotic software libraries** providing access mechanisms to the communication layer of the ROS core

Robot Operating System (ROS)

Messages are **typed data structures** exchanged through a **publish/subscribe** mechanism

When a message is received, an **handler** performs computation based on the attached data possibly **generating a new message**

Nodes can be independent entities and there is no enforced base architecture

GenoM/BIP

Component-oriented software package implementing the functional level of robot system architectures

Exchange data and events through **posters** that are **regions of shared memory**

A core feature is the support for **Behavior-Interaction-Priorities** (BIPs) framework for formally modeling and verifying complex and real-time systems with **correctness-by-construction**

Robot Control Approaches

Robotic Control

Situated robotics

Exist in a complex and challenging environment which affects the robots behaviour

Static robotics

Exist in static, unchanging environments like assembly robotics

Robot control refers to the robot architecture and decision making taking data from the environment, process it and make decisions based on it

<u>The complexity of the environment increases the complexity of the control</u>

Classes of Robotic Control Methods

1. Deliberative
2. Reactive
3. Hybrid
4. Behaviour-Based Control

Deliberative – Think, Then Act

Robots use all the available sensory information to **reason about what actions to take next**

Control system is a **decomposition of decision-making processes** permitting complex operations to be done

Planning is fundamental and requires the existence of an internal, symbolic representation of the world → internal model must be updated

Deliberative – Think, Then Act

Robots use all the available sensory information to **reason about what actions to take next**

Control system is a **decomposition of decision-making processes** permitting complex operations to be done

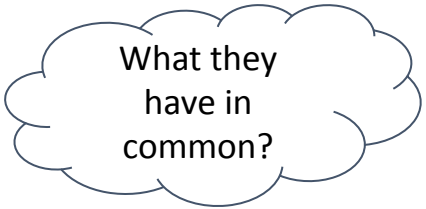
Planning is fundamental and requires the existence of an internal, symbolic representation of the world → internal model must be updated

Deliberative – Think, Then Act

Issue: **noisy, dynamic world usually makes this impossible**. Today, no situated robots are purely deliberative.

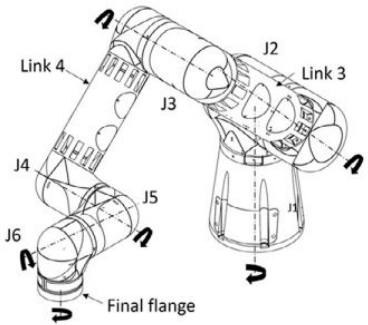
Why Deliberative?

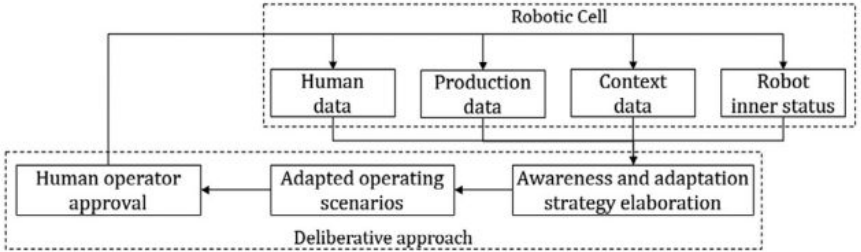
1. Path planning
2. Mission planning
3. Drone route planning



Deliberative: an example

1. comprehensive sensing infrastructure → context conscious
2. decision-making process → adaptation of tasks priorities and sequence
3. planning and control infrastructure → dynamically adapt the trajectories and motion parameters





Valente, A., et al. "Deliberative robotics–a novel interactive control framework enhancing human-robot collaboration." CIRP Annals 71.1 (2022): 21-24.

Reactive – Don’t Think, (Re)Act

Reactive control is a technique for tightly **coupling sensory inputs and effector outputs**, typically involving no intervening reasoning

Rule-based methods to react to predefined conditions, fits dynamic environments that cannot be modeled

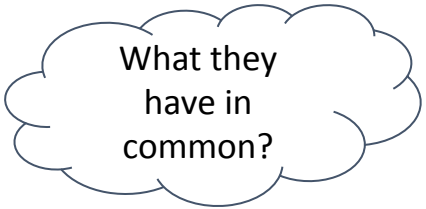
No world model, no reasoning, real-time

Reactive – Don’t Think, (Re)Act

Issue: **no internal memory or world model makes impossible to improve and learn over time**

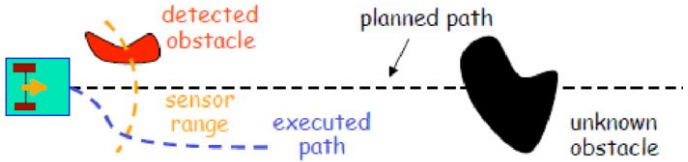
Why Reactive?

1. Obstacle avoidance
2. Wall following
3. Line following
4. Bumper sensors

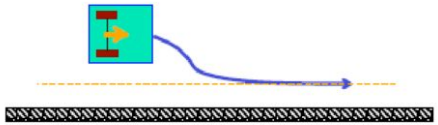


Reactive: an example

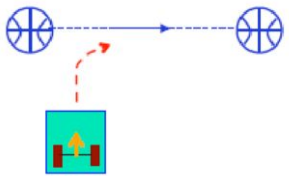
- on-line obstacle avoidance



- wall following



- target tracking



Rasam, Himanshu R. "Review on Land-Based Wheeled Robots." *MATEC Web of Conferences*. Vol. 53. EDP Sciences, 2016.

Hybrid – Think and Act Concurrently

Hybrid control aims to combine the best aspects of reactive and deliberative control: real-time response and rationality of deliberation

Two components are usually implemented:

1. **deliberative**: long-term scale, performs global and high-level decision making
2. **condition-action rules**: deals with robot immediate needs

Hybrid – Think and Act Concurrently

Interaction between reactive and deliberative components is **beneficial** for the system:

Reactive system must override the deliberative one to **react** to unexpected situations

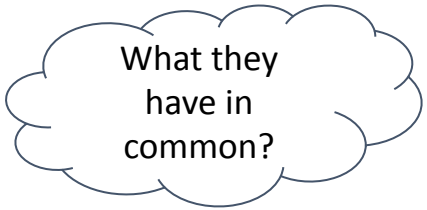
Deliberative must **inform** the reactive to guide the robot in a more efficient way

To coordinate those components, an **intermediate** one is necessary and is one of the greatest challenges in hybrid systems

3T architecture usually advocated to harness the best of both components

Why Hybrid?

1. Autonomous Vehicles
2. Robotic Manipulation
3. Search and Rescue Robots
4. Aerial Drones for Precision Agriculture



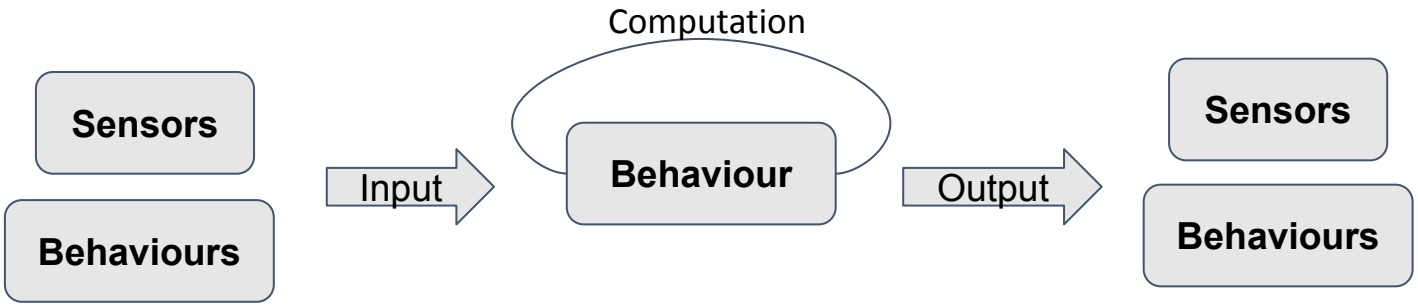
Behavior-Based Control

Developed for situated robots, allowing them to adapt to the dynamics of real-world environments **without** operating upon abstract representations of reality but also giving them **more computational capability** and expressivity than are available to reactive robots

Behavior-Based Control – Think the Way You Act

Behavior-based control employs a set of distributed, interacting modules, called behaviors, that collectively achieve the desired system-level behavior

Behaviours are control modules that cluster a set of **constraints to achieve and maintain a goal**



Behaviors

Implementation	Behaviors are implemented as control laws as procedures or processing elements
Inputs	Can receive data from sensors or from other modules
Outputs	Can send commands to effectors or to other modules
Independence	Multiple behaviors may share the same sensor inputs and actuator outputs simultaneously
Simplicity	Each behavior is relatively simple and added incrementally to the system
Execution	Behaviors run concurrently, not sequentially, enabling parallelism, faster computation, and dynamic interaction with the environment

Behavior-Based Control – Think the Way You Act

No centralized representation or control → individual behaviours can manage data independently

Why Behavior-Based Systems?

Behavior-based systems in robotics focuses on creating complex, adaptable, and robust robot behaviors through the integration of simpler behaviors:

1. Swarm Robotics
2. Multi-Agent Systems
3. Robotic Soccer Players
4. Search and Rescue Robots
5. Industrial Robots
6. Home Service Robots

When to Use What

Approaches to robot control have strengths and weaknesses and must be properly chosen

Reactive control → environments demanding immediate response, coming at the price of being myopic, not looking into the past or the future

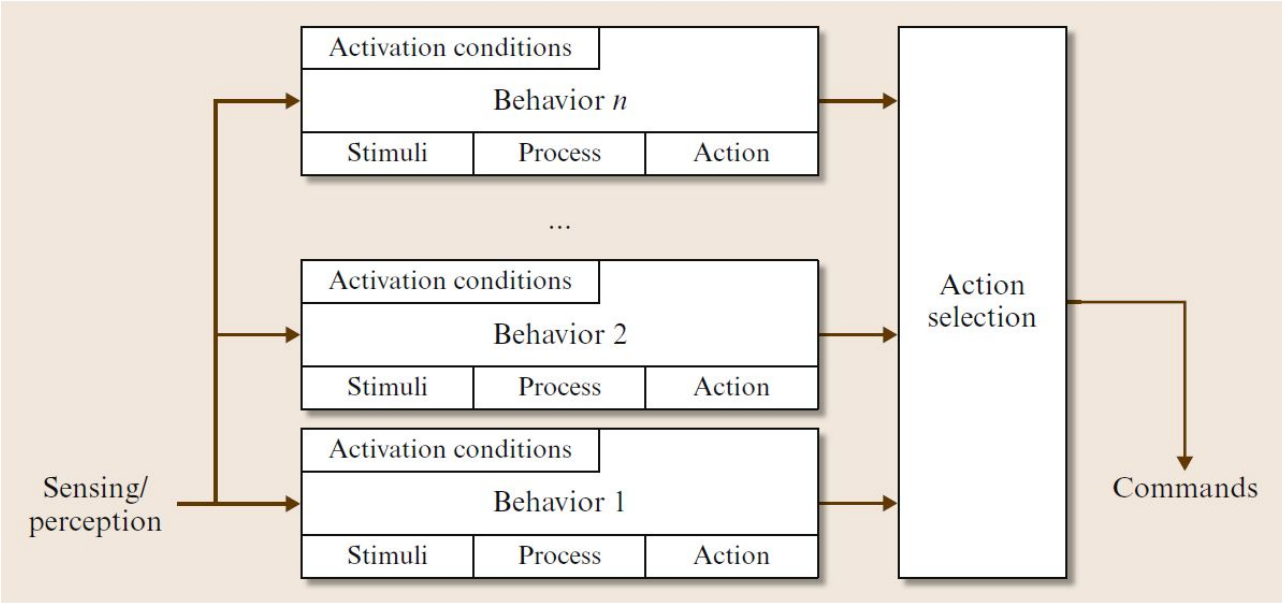
Deliberative systems → environments that require a great deal of strategy and optimization, and in turn search and planning

Hybrid systems → environments and tasks where internal models and planning are needed, and the real-time demands are few, or sufficiently independent of the higher-level reasoning

Behavior-based systems → environments with significant dynamic changes, where fast response and adaptivity are crucial, but the ability to do some looking ahead and avoid past mistakes is required

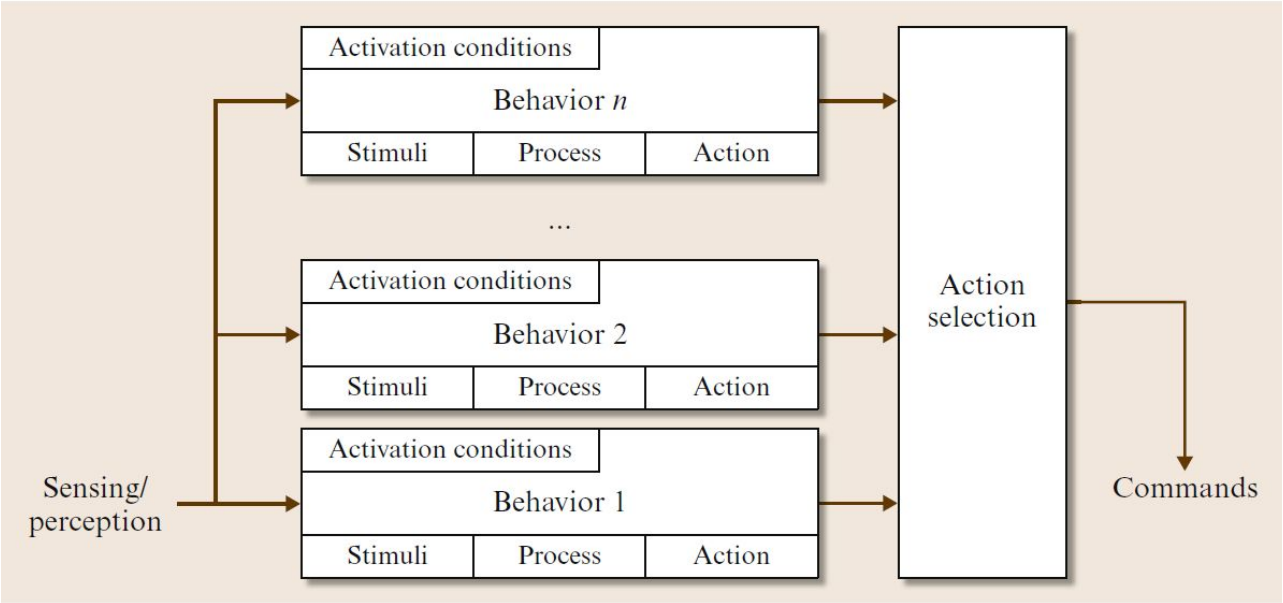
Basic Principles of Behavior-Based Systems

Allows robot to adapt in dynamic and real-world environment

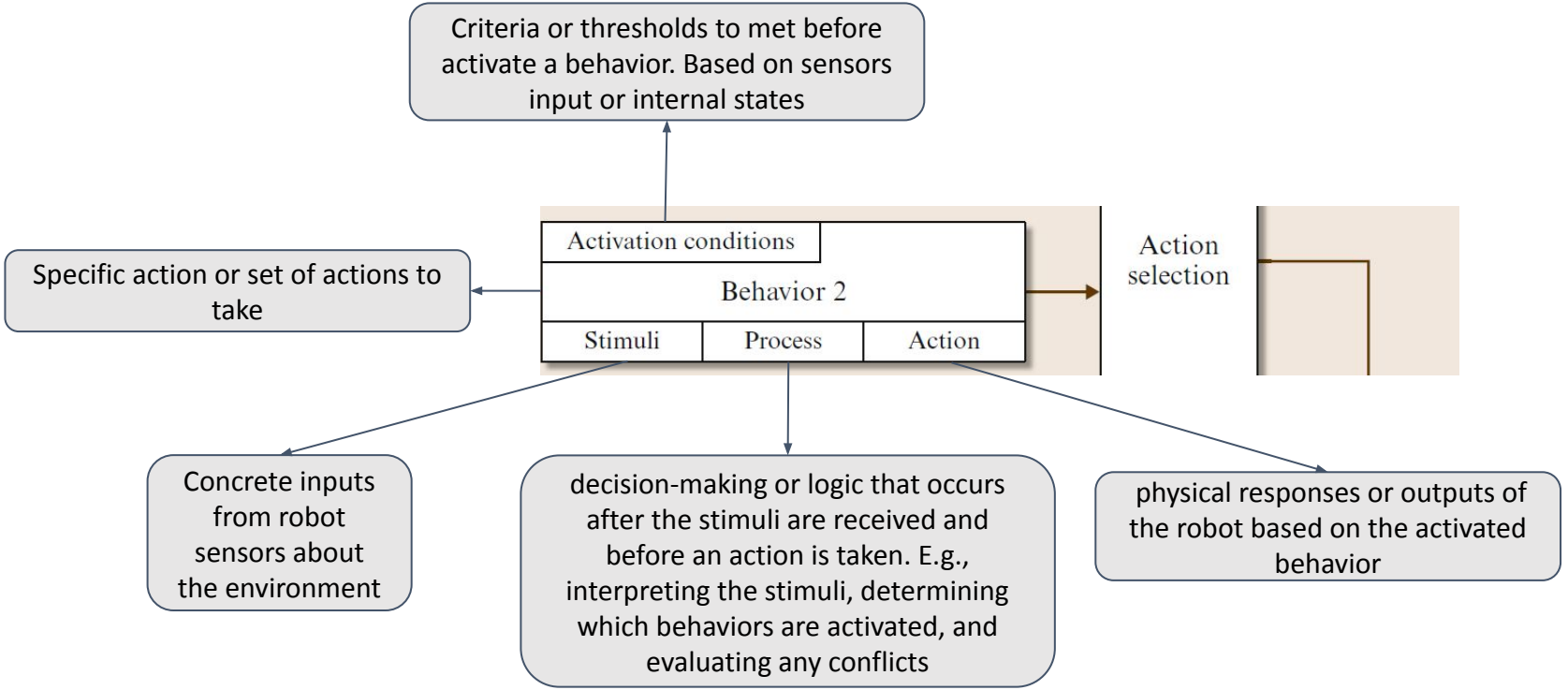


Basic Principles of Behavior-Based Systems

Can you guess the modules?



Basic Principles of Behavior-Based Systems



Behavior-Based Systems: an example

1. **Activation Conditions**: ?
2. **Stimuli**: ?
3. **Process**: ?
4. **Behavior**: ?
5. **Action**: ?

Image from *https://www.robocup.org/domains/1*

Behavior-Based Systems: an example

1. **Activation Conditions**: The robot sees the ball (1 meter away).
2. **Stimuli**: The visual sensor confirms the ball's position.
3. **Process**: The robot checks for opponents and determines that it's safe to kick.
4. **Behavior**: It activates the "Kick Ball" behavior.
5. **Action**: The robot kicks the ball towards the goal.

In contrast, if an opponent is detected within 2 meters:

1. **Activation Conditions**: An opponent is nearby.
2. **Stimuli**: Proximity sensors trigger as an opponent approaches.
3. **Process**: The robot prioritizes "Defend" over "Kick Ball."
4. **Behavior**: It activates the "Defend" behavior.
5. **Action**: The robot moves to block the opponent.

Image from *https://www.robocup.org/domains/1*

Basis Behaviors

The process of designing a set of behaviors for a robot is referred to as behavior synthesis, and is typically performed by hand

Optimal behavior?

Not realistic as it is dependent on too many specifics of a given system and environment that cannot currently be effectively formalized

Defining behaviors

Basis behaviors are a set of behaviors such that each is *necessary*, in the sense that each either achieves, or helps to achieve, a relevant goal that cannot be achieved without it by other members of that set

The basis behavior set is *sufficient* for achieving the goals mandated for the controller

Organizational principle

Two behaviors are **orthogonal** if they do not interfere with one another, each inducing no side-effects in the other. This is often achieved by:

1. having behaviors take mutually exclusive sensory inputs
2. having different behaviors control separate effectors

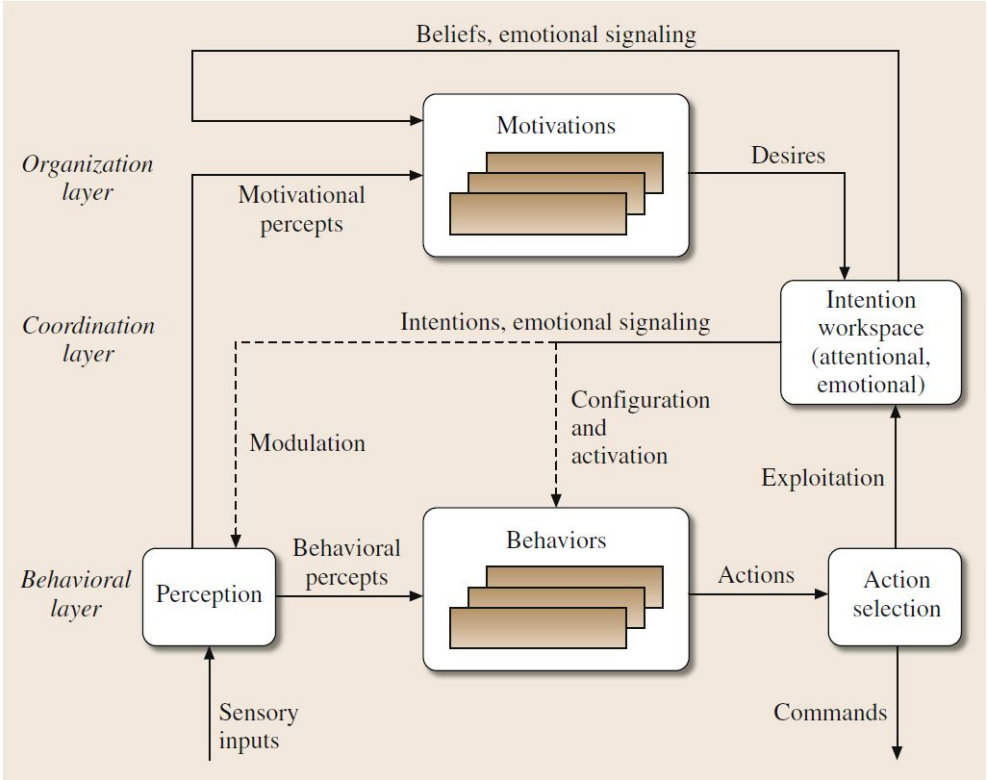

iRobot

Adaptive Behavior Selection

Designing robots capable of safely and effectively interacting with humans in real-world settings is one of the ultimate challenges in robotics

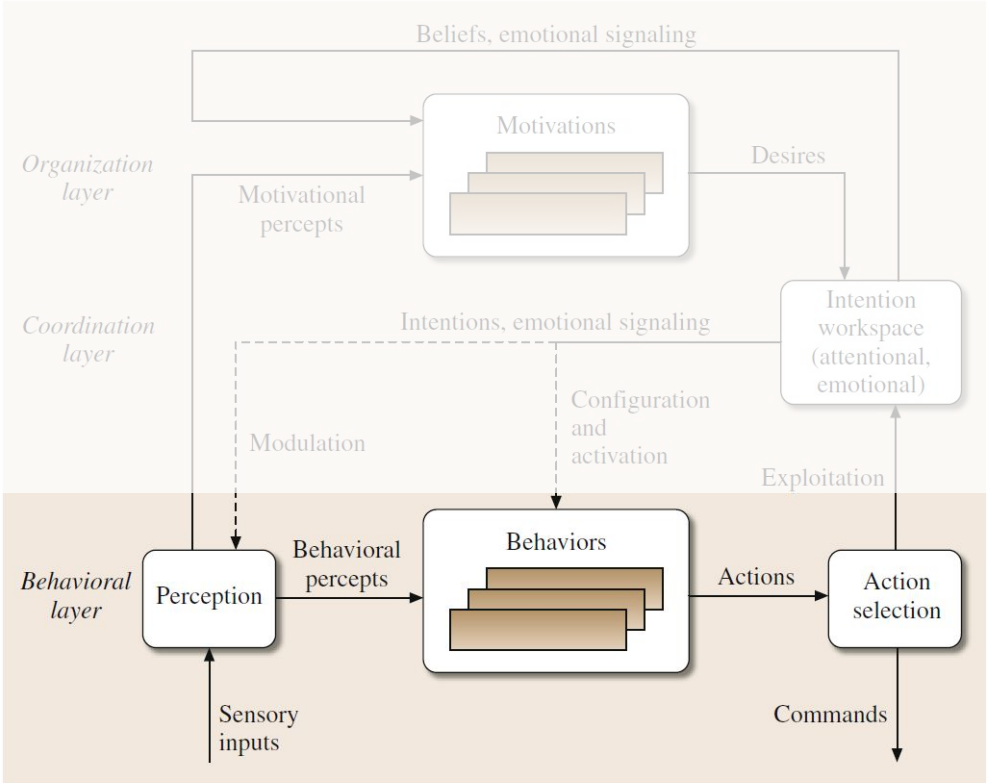
Adaptation in behavior-based systems generally consists of modifying internal parameters of the behaviors, or modifying the set of activated behaviors

Hybrid Behavior-Based Architecture



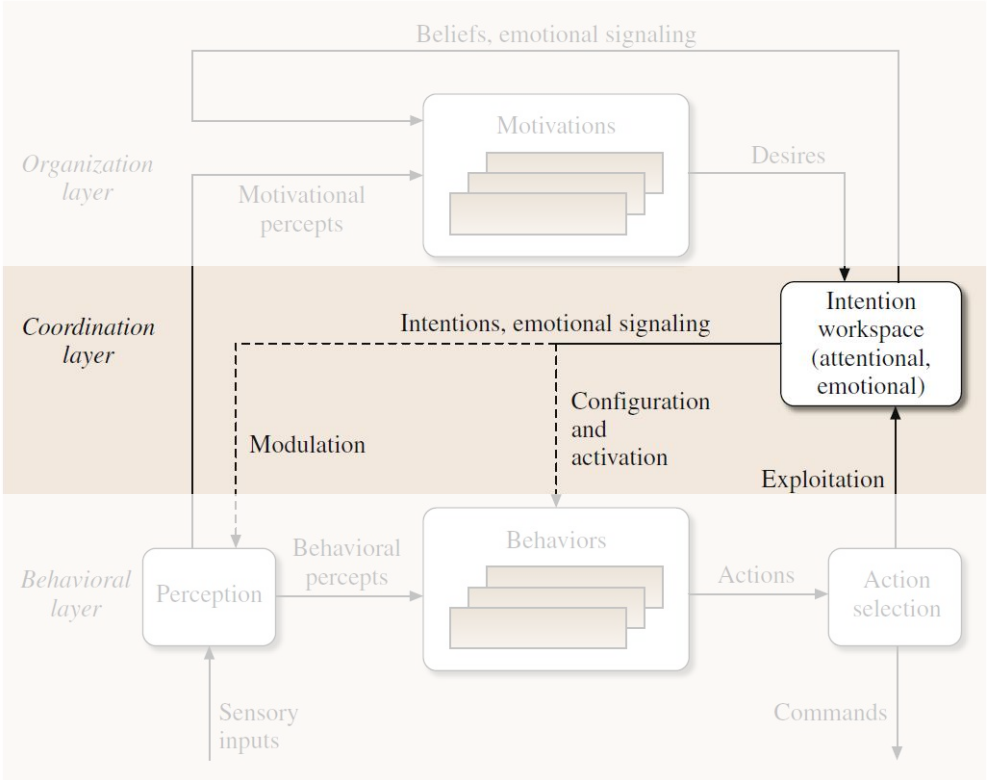
Behavioral layer

The Behavioral Layer consists of behaviors that are activated and configured according to what are referred to as the **Intentions of the system**



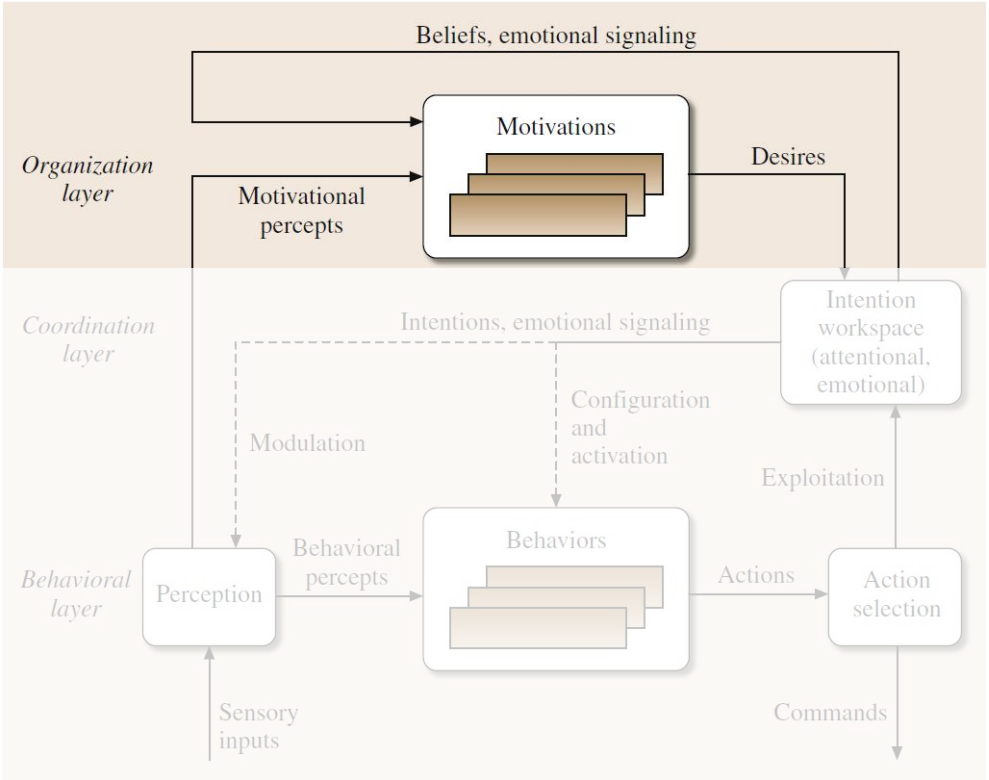
Coordination layer

Intentions are associated with the **activation and configuration of behaviors** and with the modulation of perceptual modules



Organization layer

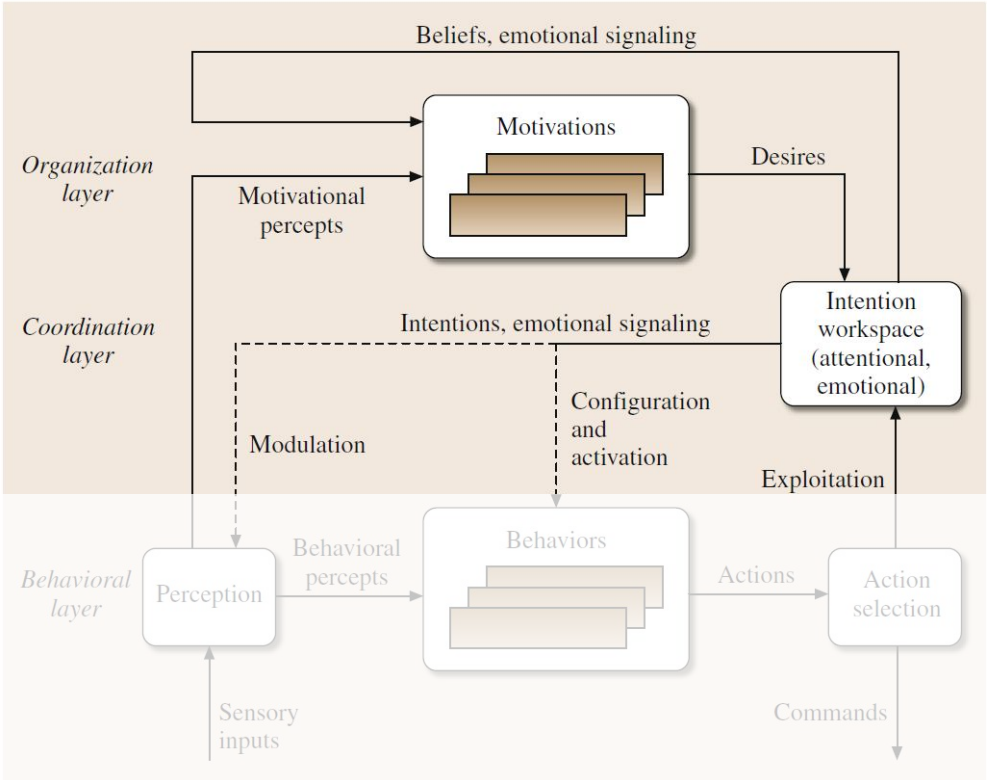
Motivations acting as distributed processes (just like Behaviors) and **manifesting Desires** for the satisfaction or inhibition of Intentions



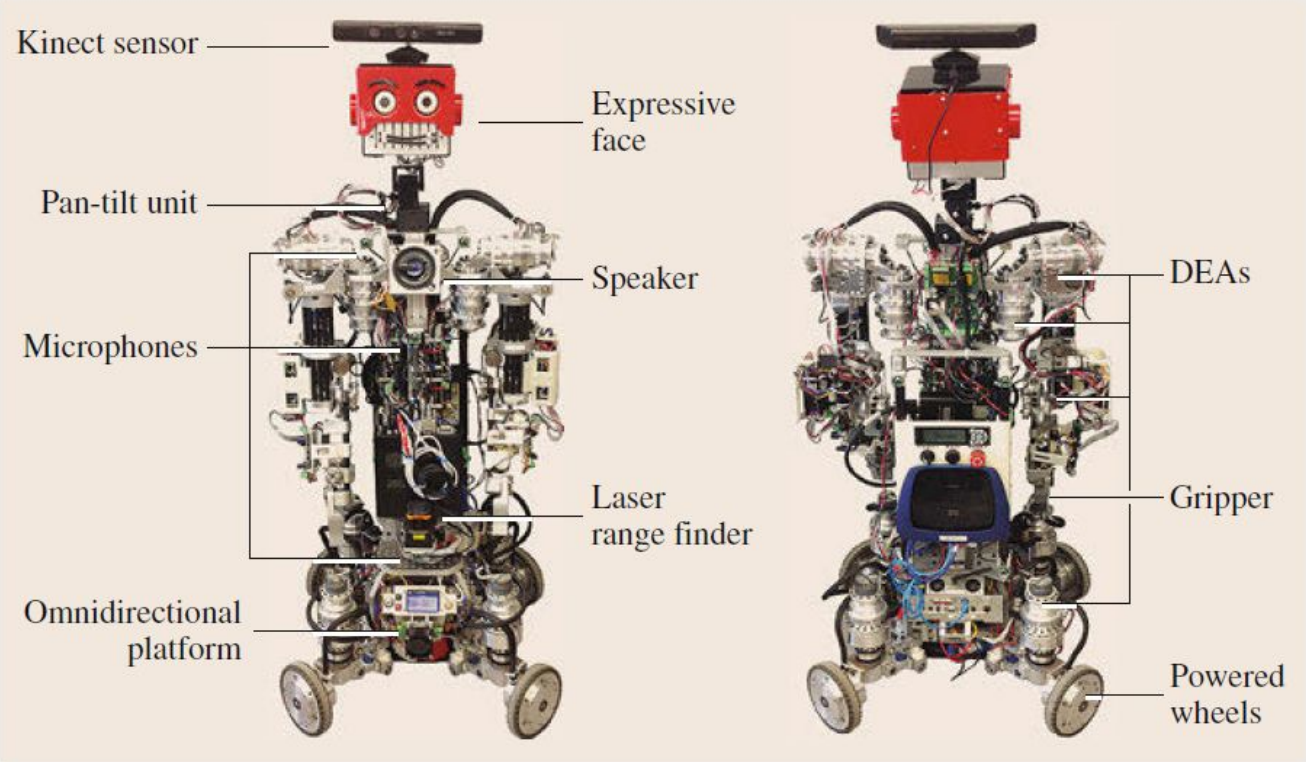
Coordination layer

The Intention Workspace processes these **Desires** to issue **Intentions**

The Intention Workspace can also provide **Emotional Signaling:** detecting situations based on models of how Behaviors are exploited over time and generates **Beliefs** which can be used to identify conflicts between Desires and their satisfaction



IRL-1 Robot


Kinect sensor
Expressive face
Pan-tilt unit
Speaker
Microphones
Laser range finder
Omnidirectional platform
DEAs
Gripper
Powered wheels

IRL-1 has behaviors for safe navigation, recharge and social interaction (with gestures, voice and vision). **Action selection is priority-based**. Motivations are implemented to provide the **intrinsic modes of operation** of the robot to localize and plan paths to desired locations, and eventually to plan and schedule tasks

Intention Workspace Roles

Motivations exchange information asynchronously, enabling flexible decision-making

Desires form a hierarchical tree, from abstract goals (e.g., deliver message) to primitive ones (e.g., avoid obstacles), **each linked to behaviors**

For example, a recharge desire may trigger docking, navigation to a station, or asking a person for guidance

Intention Workspace Roles

A representation of the history of behavior exploitations, indexed by intentions

For instance, for the recharging desire example, if being guided by people reveals not to be as reliable as using the internal map, a preference could be given to the second option

Intention Workspace Roles

An emotion-based mechanism helps robots resolve conflicts between desires and intentions by detecting when normal decision-making fails

Like in humans, emotions highlight such situations, using temporal models of intentions without needing prior knowledge of conditions or goals.

Multi-Robot Systems

Single- vs Multi-Robot

A **single robot** that has to **accomplish various tasks** in a dynamic environment is difficult to design and also a single failure can lead to errors

A **multi-robot** approach can employ a **parallel workflow**, resulting in a more efficient and versatile work

What is a Multi-Robot System?

The term Multi-Robot System (MRS) refers to a **group of robots** that **works together** to accomplish any task faster and cheaper than a single-robot system

Scalability, **reliability**, **flexibility** and **versatility** are core characteristics

A single entity has few functionalities, since the real power lies in the cooperation of multiple robots

Types of Multi-Robot Systems

The topic of multi-robotics has lead to a variety of approach and definitions which makes difficult to provide a clear classification

However, we can recognize 2/3 main areas:

1. Collective Robotics
2. Second-Order Robotics
3. Swarm Robotics

Collective Robotics

One of the first ideas of a multi-robot system (late 1980s)

In ***Collective Robotics*** a group of robots **share knowledge** and compare results to accomplish **collaborative goals** such as box pushing and coordinated motion

Example? **collective foraging**: group of robots that have to collect objects scattered in the environment, and deposit them in some particular location

Second-Order Robotics

First multi-robot system (1987, CEBOT, Fukuda and Nakagawa)

Second-order robotic unit as a robot that is formed by a number of robotic units - referred to as first-order robotic units - **physically connected** one to the other

Two main approaches can be found:

1. Self-Reconfigurable Robots
2. Self-Assembling Robots

Second-Order Robo nfigurable Robots

They MUST be connected in order to move → lack of autonomy

Groups of robotic modules nt **mobility** → very few **sensors**

Capable of **connecting among themselves** in various ways to form complex physical structures

Modules are **reconfigurable robots** that can autonomously **change their physical connections and configurations** to meet the demands of the environment

Second-Order Robotics: Self-Assembling Robots

Self-assembling robots are able of autonomous motion, recognition of other robots and assembling

They allow to **accomplish task individually** under environment conditions which obstacle the group to collaborate

Second-Order Robotics: an example

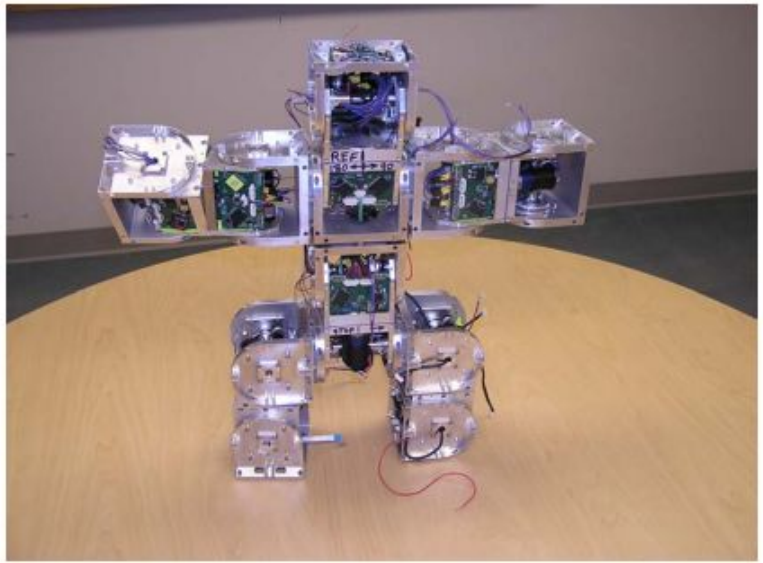
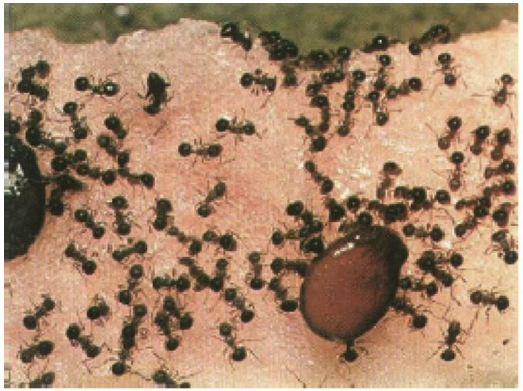


Image from: Salemi, Behnam, Mark Moll, and Wei-Min Shen. "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system." *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006.

Swarm Robotics




Source: Tan, Ying, and Zhong-yang Zheng. "Research advance in swarm robotics." Defence Technology 9.1 (2013): 18-39.

Swarm Robotics

Large numbers of mostly simple physical robots

The term **Swarm Robotics** is inspired by the behaviours of social insects

collectively intelligent systems can be generated from a large number of individuals

1. Self-organization
2. Collective decision-making

A desired collective behavior emerges from the interaction **between the robots** and the interaction of **robots with the environment**

Characteristics

Single units: normally simple, small and low cost so as to take the advantage of a large population

Communication: normally local, and guarantees the system to be scalable and robust

Architecture: swarm is distributed and decentralized (high efficiency, parallelism, scalability and robustness)

Swarm Robotics Behaviors

A plain set of **rules at individual level** can produce a large set of **complex behaviors at the swarm level**

The **rules of controlling the individuals are abstracted** from the cooperative behavior in the nature swarm

Swarm Robotics Benefits

Key Features

- **Parallel** → Large populations, multiple targets simultaneously.
- **Scalable** → Local interactions, robots can join/quit without disrupting the swarm.
- **Stable** → System keeps working even if some robots fail.
- **Economical** → Mass production cheaper than single complex robot.
- **Energy Efficient** → Small, simple robots consume less energy, ensuring longer lifetime.

Differences vs Other Multi-Agent Systems

Distinctive Traits

- **Autonomous** → Each robot acts independently in the environment.
- **Decentralized** → No central control, higher flexibility and resilience.
- **Local Sensing & Communication** → Limited range, avoids high communication cost.
- **Homogenous** → Few roles, large numbers in each role → real "swarm."
- **Flexible** → Adaptable to different tasks with same hardware and minor software changes.

Swarm Robotics: an example

Kilobots - Harvard Self-Organizing Systems Research Group

Self-Organization: Kilobots autonomously form patterns and distribute themselves using simple local rules.

Collective Decision-Making: They can collectively choose between tasks, like finding the shortest path, based on local interactions.

Task Allocation and Cooperation: Kilobots collaborate to perform tasks like object movement, achieving complex goals together.

Fault Tolerance: The swarm adapts to failures, maintaining functionality despite individual robot breakdowns.

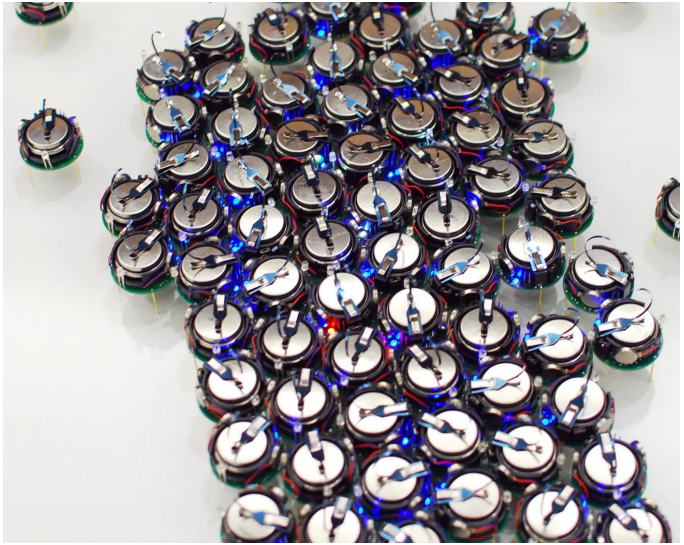


Image from: https://robotsguide.com/robots/kilobot

Characteristics of Multi-Robot Systems

A multi-robot system is distributed if:

- **Distributed control** to process gathered information and to make the decision locally while achieving the system-level goal
- **Distributed sensing** to sense itself, the environment, and other robots locally
- **Distributed actuation** to navigate freely in the environment without collision with obstacles and other robots
- **Distributed communication** to receive and transmit data from other robots in a scalable robot network

A(nother) taxonomy of multi-robot systems

Single Robots (not technically considered): the minimal number of elements of a robotic system

Pair: the smallest team you can have

Multiple: group of several members numerically lower than the size of the activity they have to carry out

Swarm: is very large team of robots that usually have limited capabilities

Types of adaptations

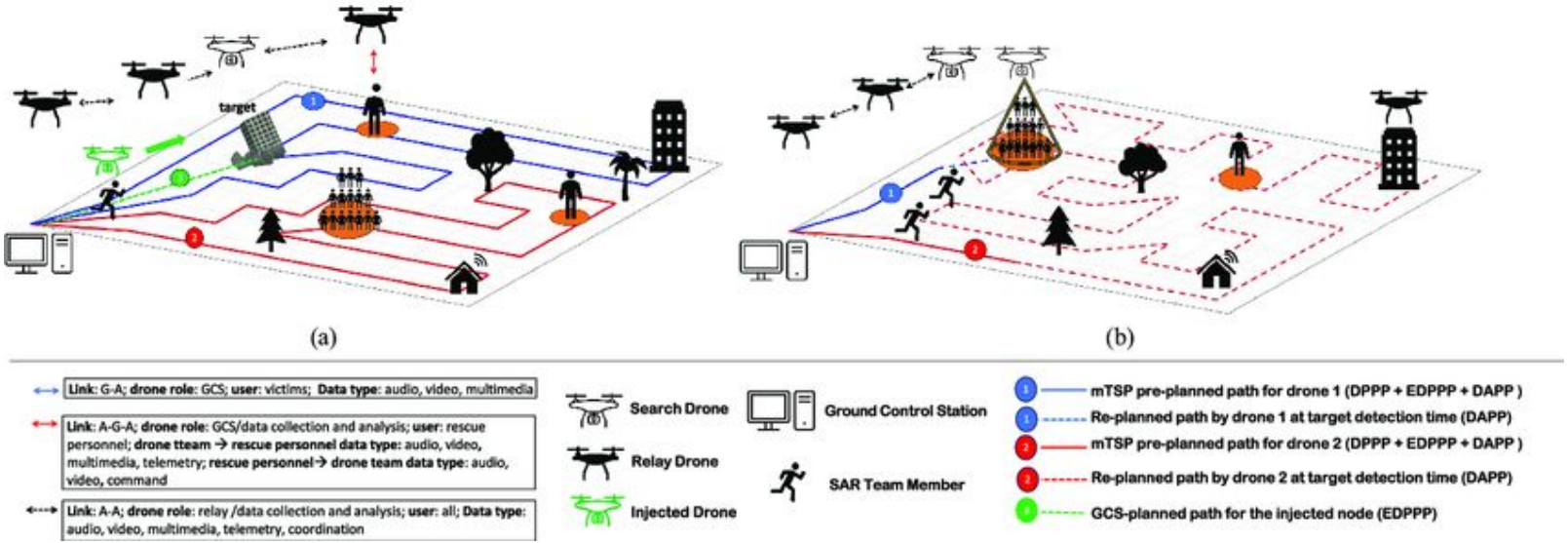
Static configuration: does not allow topology changes, so a task cannot be achieved if one agent goes down (usually pair)

Coordinated re-arrangement: allows to change the number of robots in a team, this scenario requires a reconfiguration of the mission with new tasks assigned to each element.

Dynamic configuration: the number of robots can be changed while the mission can continue without any re-assignment

Static Configuration

1) Fixed team structure, no changes allowed in number of robots
2) If one robot fails, the task cannot be completed
3) Usually based on pairs or pre-defined groups



Source: Yanmaz, Evşen, Hamid Majidi Balanji, and İslam Güven. "Dynamic multi-UAV path planning for multi-target search and connectivity." IEEE Transactions on Vehicular Technology 73.7 (2024): 10516-10528.

Coordinated Re-arrangement

1) Team composition can change, but requires reconfiguration
2) Tasks are reassigned by a central coordinator or distributed negotiation

Dynamic Configuration

1) Robots can join or leave seamlessly without halting the mission
2) No explicit re-assignment needed, tasks are taken up automatically by others

Robots Morphology

Identical team: is made up of units that are homogeneous in both hardware and software (anyone can perform a task)

Homogeneous team: refers to a group that shares physical characteristics, but not exactly the same; for example, they can be distinguished by on-board sensors.

Heterogeneous team: is composed of elements that differ in morphology, capabilities or locomotion techniques; in this case a mission can not be completed if one type of unit is removed.

Identical Team

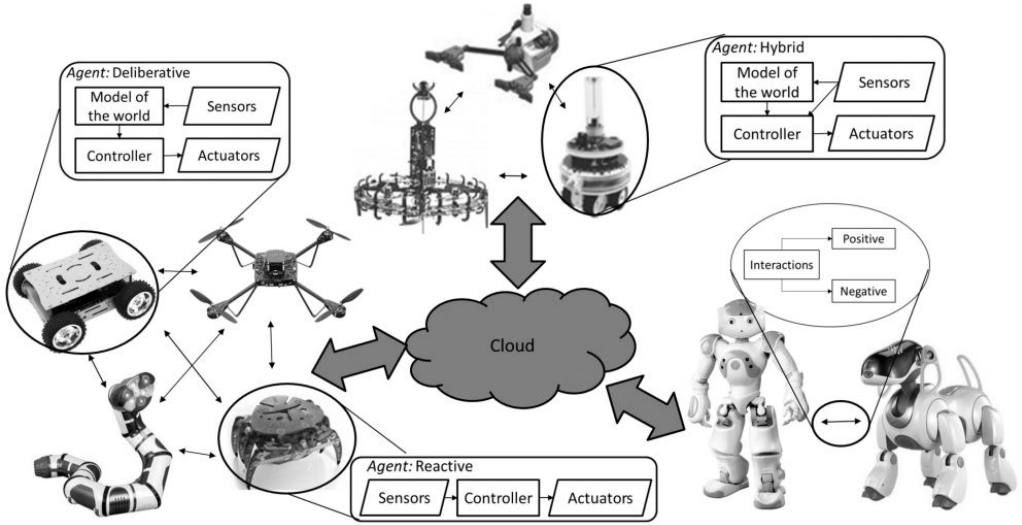
1) All robots are identical in both hardware and software
   Any robot can perform any task
2) Very robust: if one fails, another can take over

Homogeneous Team

1) Robots share the same physical structure (morphology, locomotion)
2) They differ in some components (e.g., sensors, onboard instruments)
3) Some task specialization possible

Heterogeneous Team

1) Robots differ in morphology, capabilities, or locomotion methods.
2) Mission success depends on collaboration between specialized units.
3) Losing one type may prevent completion of the mission.



Source: Rizk, Yara, Mariette Awad, and Edward W. Tunstel. "Cooperative heterogeneous multi-robot systems: A survey." ACM Computing Surveys (CSUR) 52.2 (2019): 1-31.

Robots Behaviour

Collective: entities of a team are not aware each other, even if they share goals and their actions are beneficial to teammates

Cooperative: entities are aware of others, share goals and their actions are beneficial to teammates.

Collaborative: robots are aware of their teammates, but have an individual goal, however they help each other to achieve individual, but compatible, goals; this scenario can be seen equivalent to the previous one, by considering the team goal from an higher perspective

Coordinative: systems are made by entities aware of each other, but they do not share a common goal and their actions are not helpful to other team members. This type of interaction is used by robots that share a common workspace and need to coordinate their behaviour to avoid collisions or interference

Robots Coordination

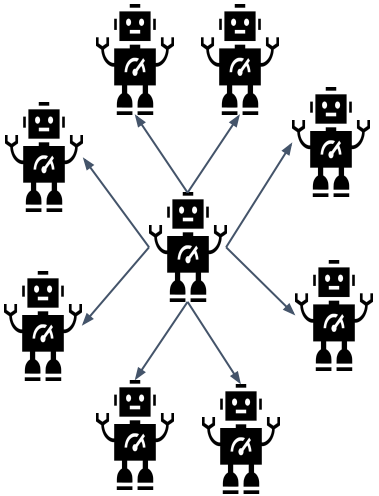
Strongly coordinated robots: relies on a protocol defining the behaviour. The coordination can be defined by:

1. a single leader (**strongly centralized**)
2. different leaders that follow a hierarchical architecture (**weakly centralized**)
3. each robot that decides autonomously its task with respect to each other (**distributed**).

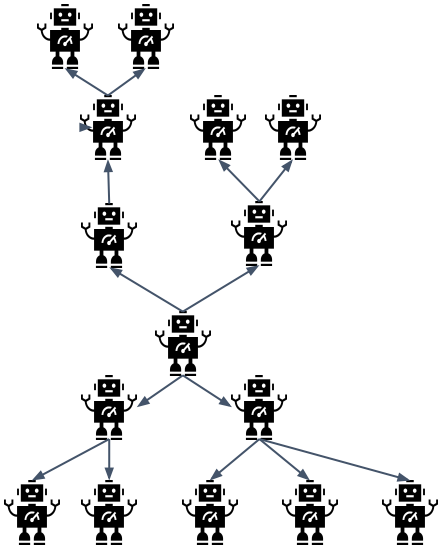
Weakly coordinated robots: does not follow any predefined protocol, robots act in parallel with some corrections of the behaviour with respect to the other robots

Not coordinated robots: act as they want, without following any protocol or caring about others status

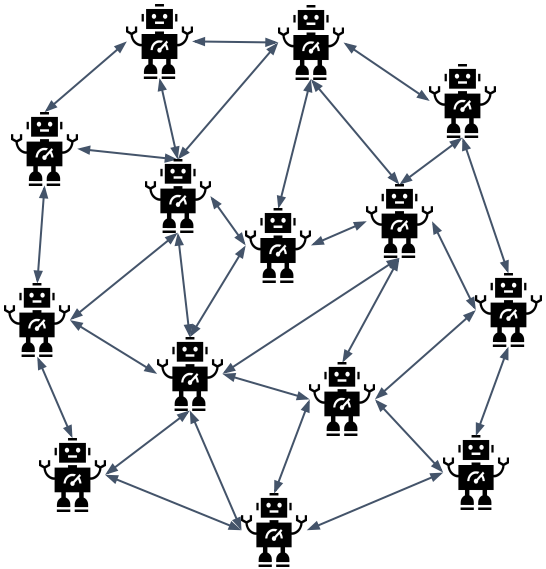
Robots Coordination



Centralized



Decentralized



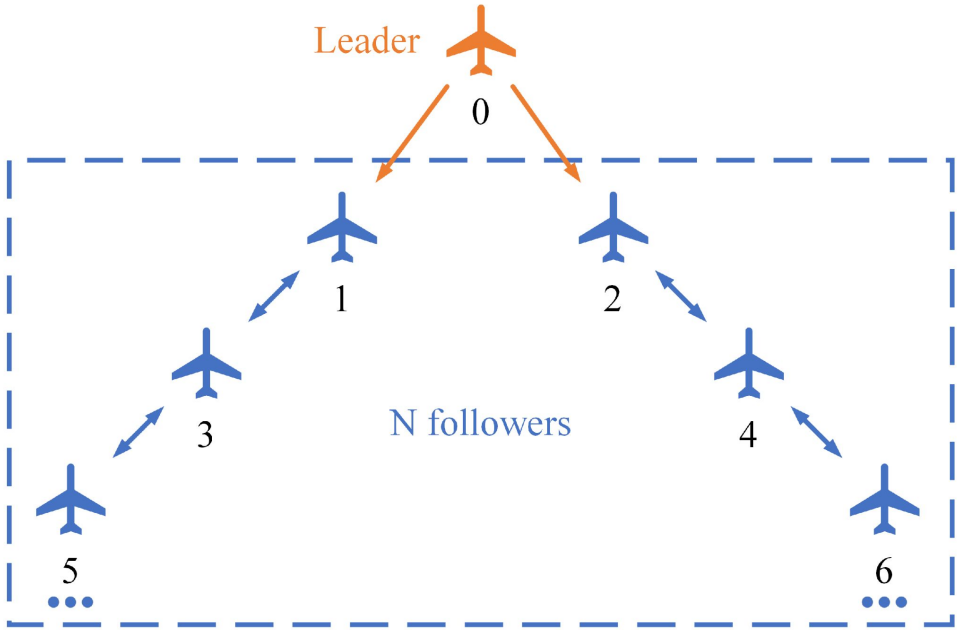
Distributed

Decentralized

No single central controller **decisions are made locally**

There may still be some **hierarchy or designated leaders**

Robots **exchange information to coordinate**, but not all decisions are fully independent



Distributed

Every robot is fully autonomous and equal

Coordination emerges from peer-to-peer communication

The system continues even if several robots fail

Robots Automation

Degree of automation for multi-robot systems

First level describes the least automated system, in which only task execution is automated

Second level adds the automation to task allocation or coalition formation, not to both

Third level allows both the task allocation and coalition formation to be automated

Fourth level is the ideal one, and lets the entire system to be fully automated

First level

Definition

1. Robots can autonomously execute tasks but do not decide what to do
2. A human operator or external system assigns tasks
3. No decision-making autonomy in allocation or teaming

Cons

1. High human workload in planning
2. Robots = “tools” that carry out predefined instructions
3. Low adaptability to unexpected changes

Second level

Definition

1. Robots can handle one layer of decision-making
   a. Either automated task allocation (deciding who does what), OR
   b. automated coalition formation (deciding which robots should work together)
2. But not both at the same time

Cons

1. Still limited flexibility if both allocation and teaming are needed simultaneously

Third level

Definition

1. Robots autonomously decide both
   a. Which tasks to perform, and
   b. Which robots should collaborate to complete them
2. Decisions may be distributed or decentralized

Cons

1. Still, humans may oversee the mission objectives

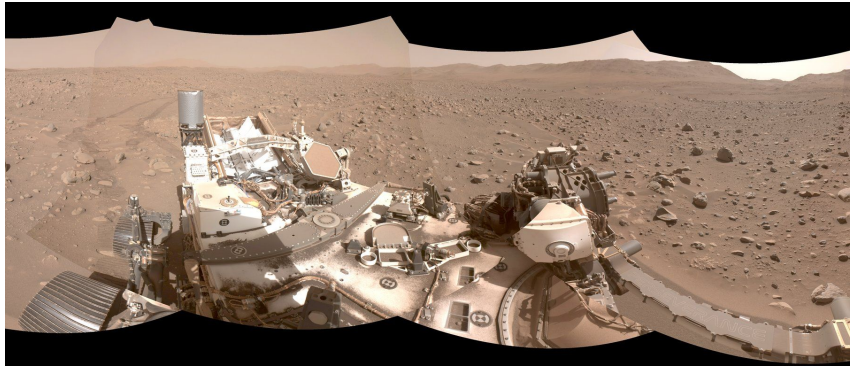
Fourth level

Definition

1. Robots autonomously assign tasks, form coalitions, and adapt dynamically.
2. The entire mission is managed without external input.
3. The system can reconfigure itself in real time under uncertainty.

Cons

1. Still a research challenge due to complexity, ethics, and trust issues



To Resume…

	Awareness of others	Shared goal	Beneficial actions	Example
Collective	*N*	*Y*	*Y*	Swarm
Cooperative	*Y*	*Y*	*Y*	Multi-robot for search and rescue
Collaborative	*Y*	*N*	*Y*	Coalition formation
Coordinative	*Y*	*N*	*N*	Path planning