



# researchXchange

Welcome!

**Flexible programming of Industrial Robots for Agile Production environments**

Laurent Cavazzana



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

# Towards Agile Manufacturing

- ▶ Late 20th century: 3rd Industrial revolution (Digital Revolution)
  - ▶ more powerful, intelligent robots in production lines
  - ▶ production lines remained "static"
- ▶ 21st century:
  - ▶ new technologies (data exchange, cloud computing, 3D Printing, IoT, AI, etc)
  - ▶ step into the Industry 4.0 era

# Towards Agile Manufacturing

- ▶ Rapidly evolving markets
- ▶ Hyper-personalization of products

# Towards Agile Manufacturing

- ▶ Rapidly evolving markets
  - ▶ Hyper-personalization of products
-  Major challenges for a manufacturing industry

# Towards Agile Manufacturing

- ▶ Rapidly evolving markets
  - ▶ Hyper-personalization of products
-  Major challenges for a manufacturing industry
- ▶ reduce time to market to stay competitive
  - ▶ constant reprogramming of production tools and robots ?!

# Agile Manufacturing

Organization which has the processes, tools, training enabling:

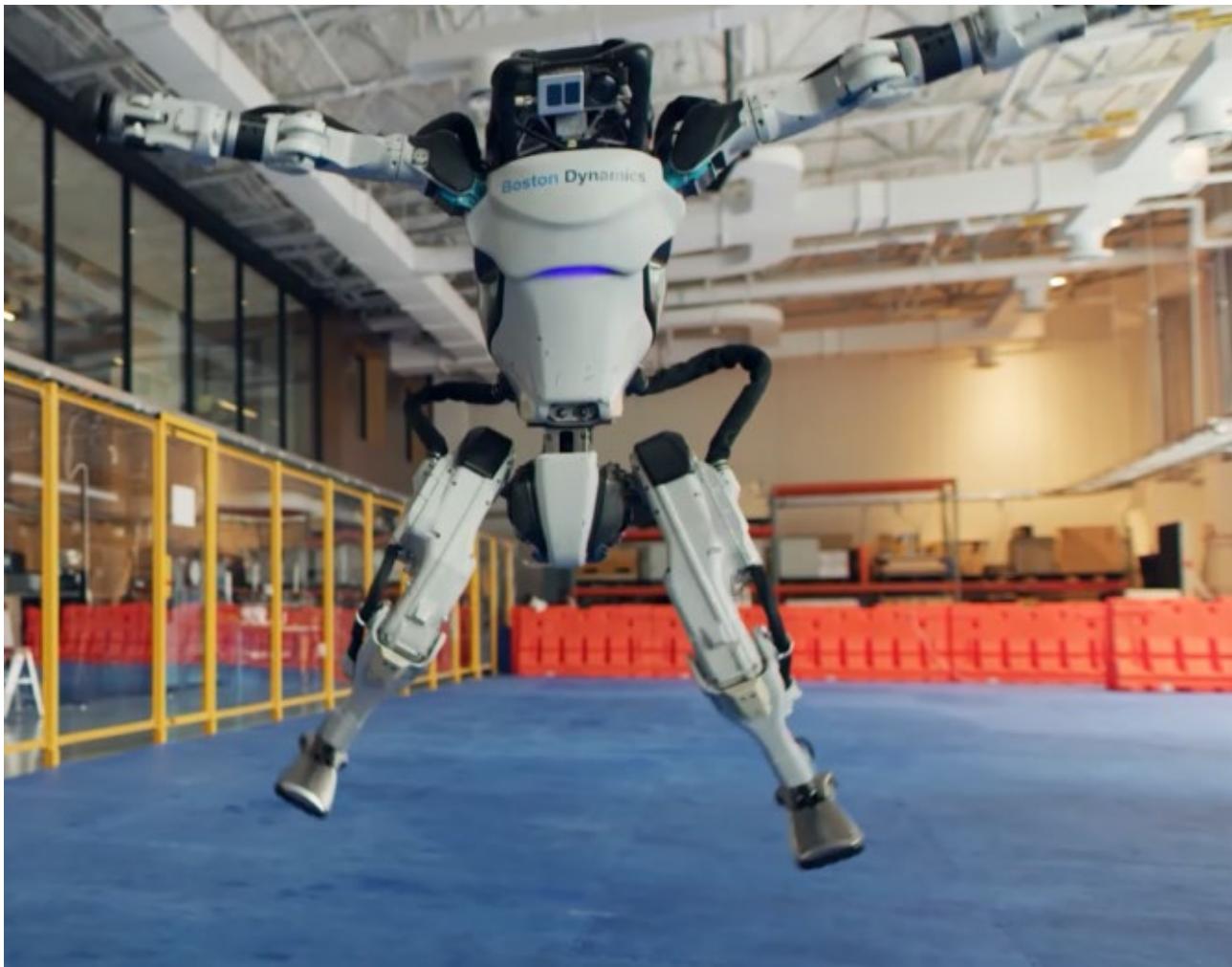
- ▶ Quick response to customer needs and market changes
- ▶ Cost control and quality

# Agile Manufacturing

Organization which has the processes, tools, training enabling:

- ▶ Quick response to customer needs and market changes
  - ▶ Cost control and quality
- 
- Fine, but concretely what does it mean to production line in practice ?

# Agile Manufacturing



# Agile Manufacturing

- ▶ **flexible production concept:**
  - ▶ Robotic system capable of self-adapting to varying production needs
  - ▶ Flexible reprogramming
  - ▶ Human Robot Collaboration (HRC) to combine:
    - ▶ Human's cognitive and creative power
    - ▶ Robots' high precision and repetitive tasks abilities
  - ▶ Digital simulation

# The ACROBA Project

## 1 large company



## 7 SMEs



## 5 research centers



## 2 clusters



## 2 universities



# The ACROBA Project

H2020  
Innovation action

Coordinator:  
BFH

17 partners  
9 countries

~8M€ budget  
~7M€ EU funding

42 months

Reference Architecture  
COPRA-AP

5 industrial pilots

2 ACROBA  
On-Site Lab

12 hackathons

# The ACROBA Project

- ▶ Design a novel concept of cognitive robotic platform
- ▶ enabling fast and cost-efficient deployment of robot systems.

Reduction of:

- ▶ robot programming time
- ▶ commissioning time
- ▶ software engineering

# The ACROBA Project

- ▶ Platform will be demonstrated by 5 industrial scenarios:
  - Lights out manufacturing
    - STERIPACK: processing of medical 3D printed parts
    - CABKA/MOSES: Defects removal and quality control
  - Collaborative assembly lines
    - IKOR: PTHs on PCBs
    - ICPE: Electric motors parts

# The ACROBA Project

► Moses:



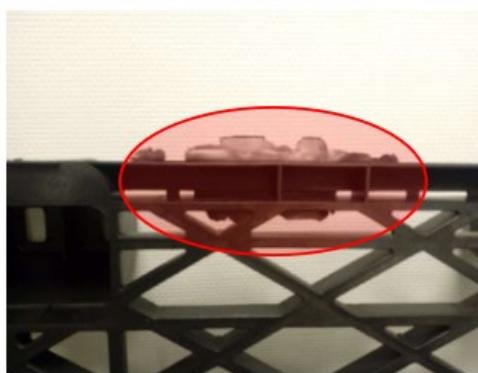
# The ACROBA Project

► Moses:



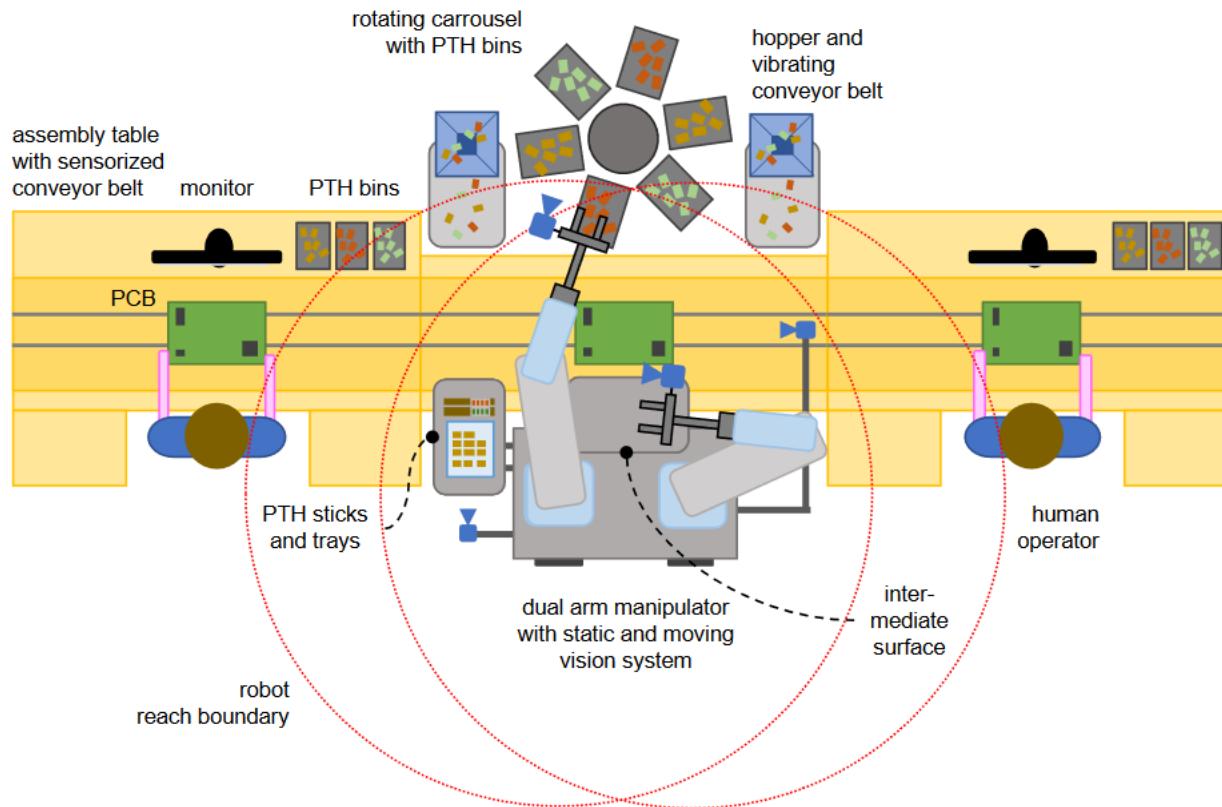
# The ACROBA Project

## ► Cabka:



# The ACROBA Project

## ► IKOR:



# The ACROBA Project



# Flexible Programming of Industrial Robots



# Flexible Programming of Industrial Robots

## User Interface

- ▶ Task design:
  - Flexible and simple enough
  - new task from scratch
  - Save, load, visualize created task(s)
  - Allow for the creation of new skills / actions
  - from collaboration (learning from Demonstration)
  - CAD (step files, etc)
- ▶ Task Execution
  - Monitoring: know actions the robot did or is going to perform
  - control the robot if needed (pause, restart, repeat some action(s), skip some, etc)

# Flexible Programming of Industrial Robots

## User Interface

### □ Typical user of the UI ?

- Lower skill workers (operator, etc): no knowledge of robotics
- Robotics experts (roboticists, robot designer, etc)

### □ What kind of user interface ?

# Flexible Programming of Industrial Robots

## User Interface

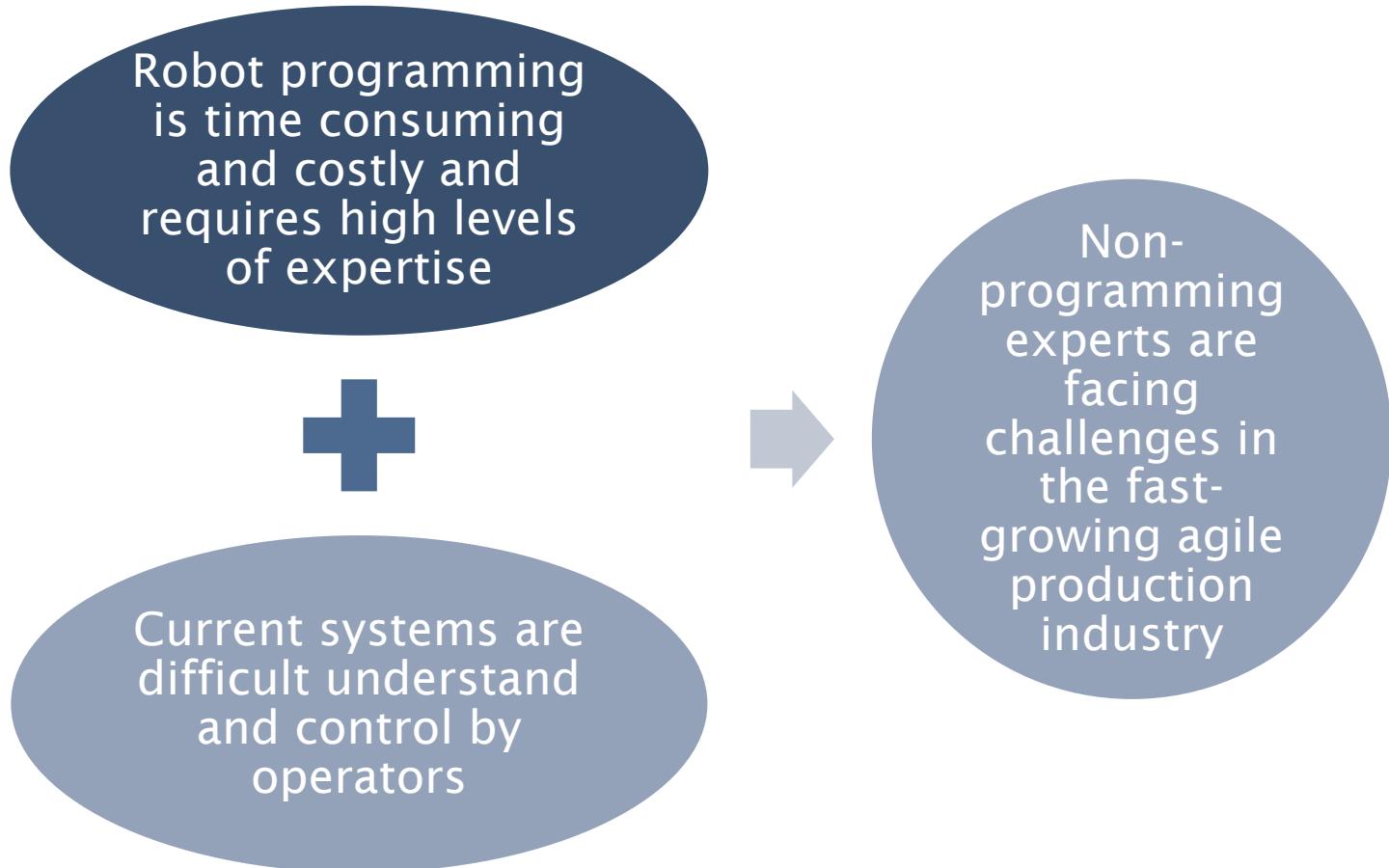
### □ Typical user of the UI ?

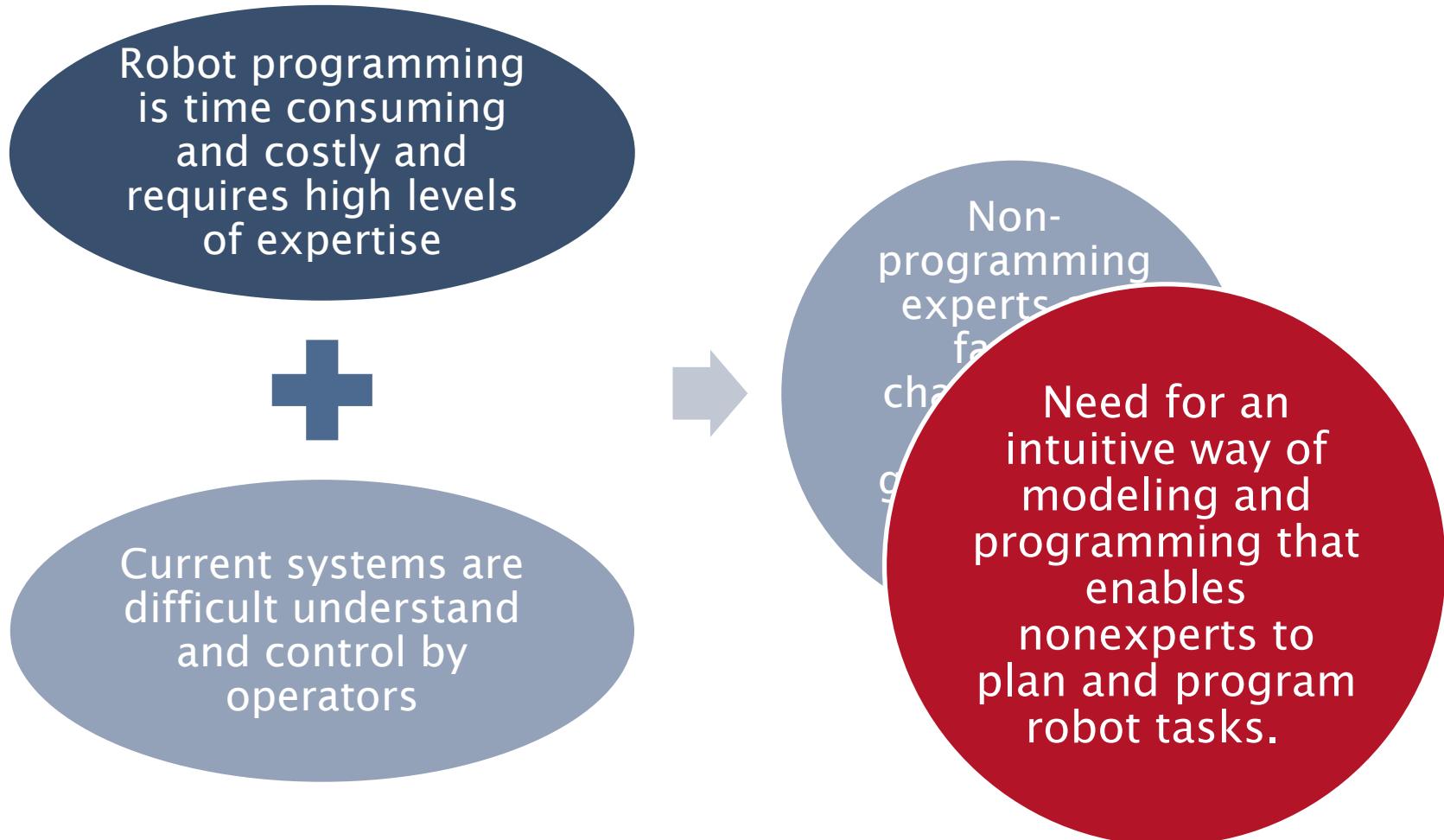
- Lower skill workers (operator, etc): no knowledge of robotics
- Robotics experts (roboticists, robot designer, etc)

### □ What kind of user interface ?



researchXchange Robot Task Model and Notation  
Congyu Zhang Sprenger

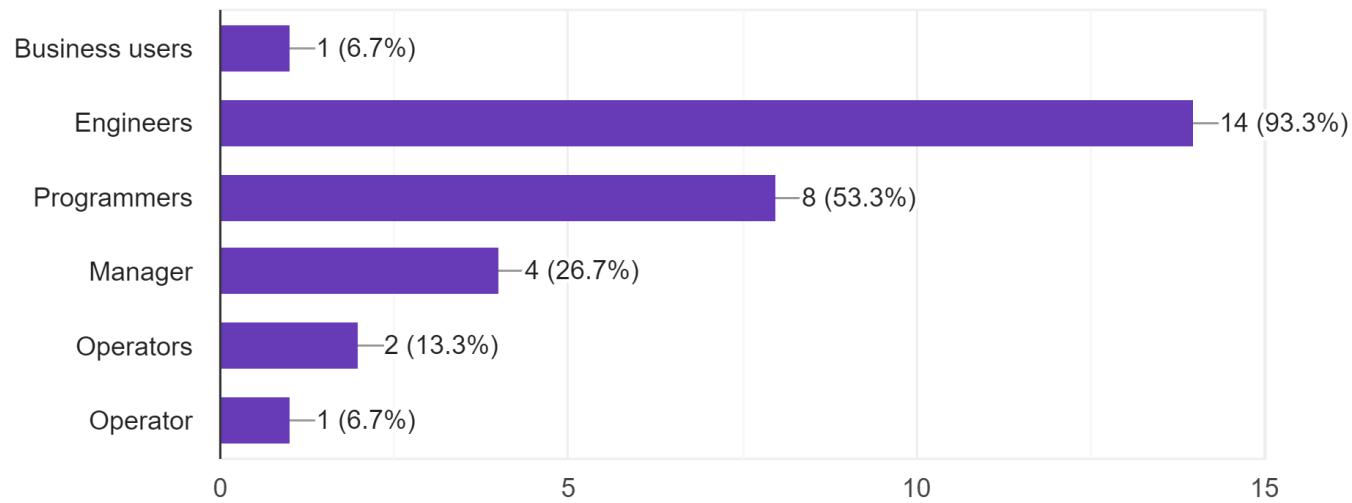




# Research - Questionnaire for ACROBA User Interface

## 1. What are the users of the ACROBA platform?

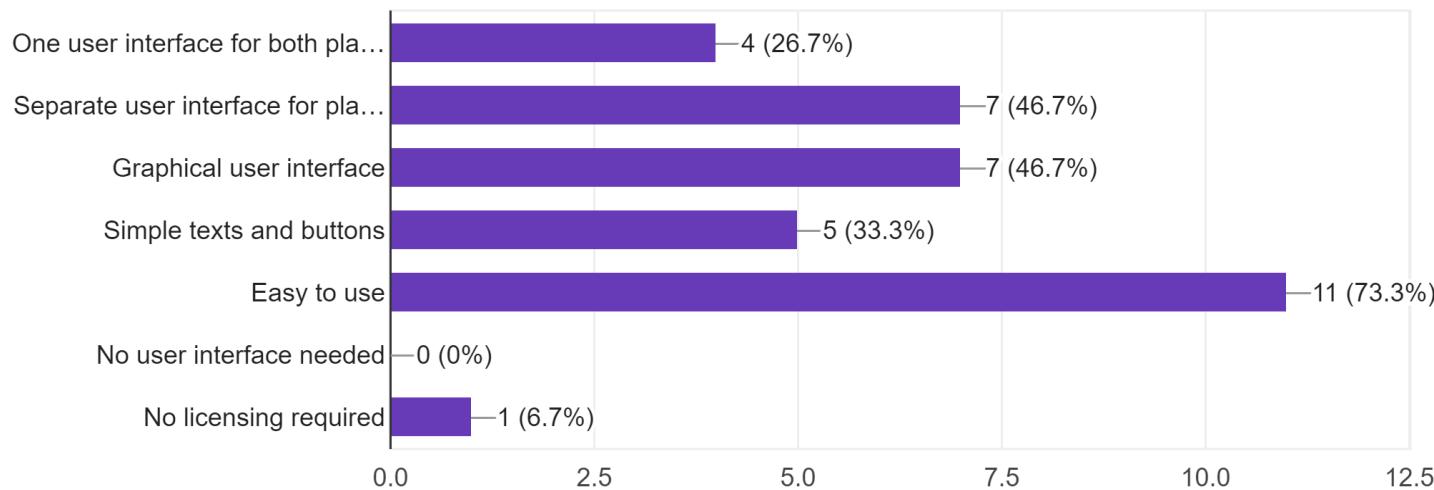
15 responses



# Research - Questionnaire for ACROBA User Interface

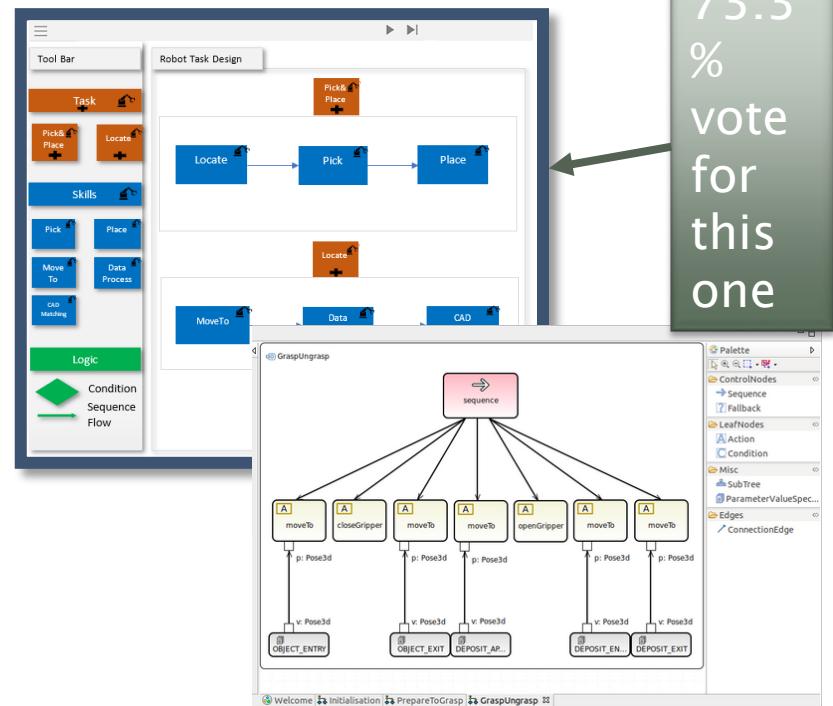
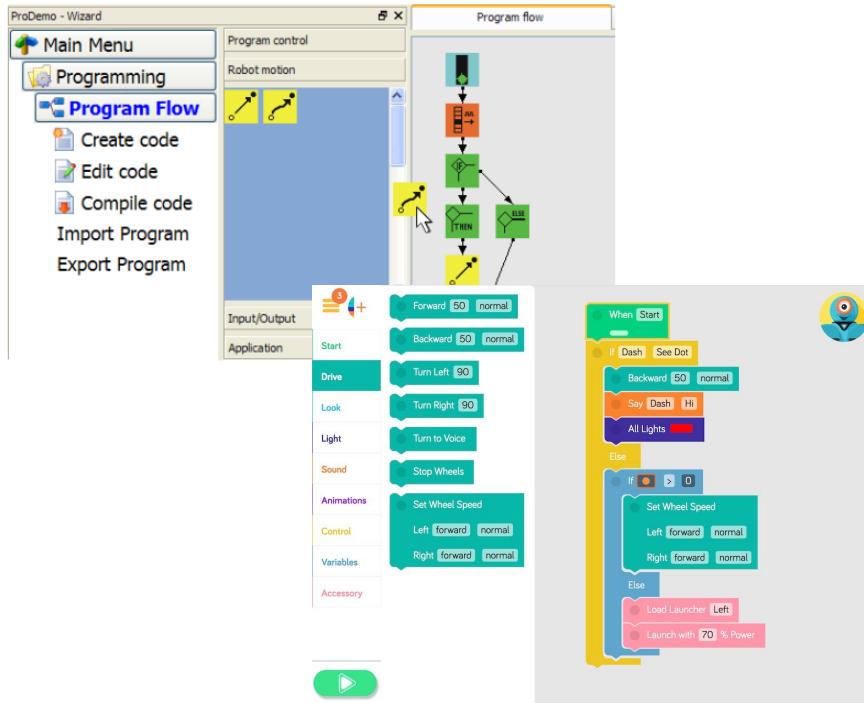
## 4. What do you expect for a user interface of ACROBA?

15 responses



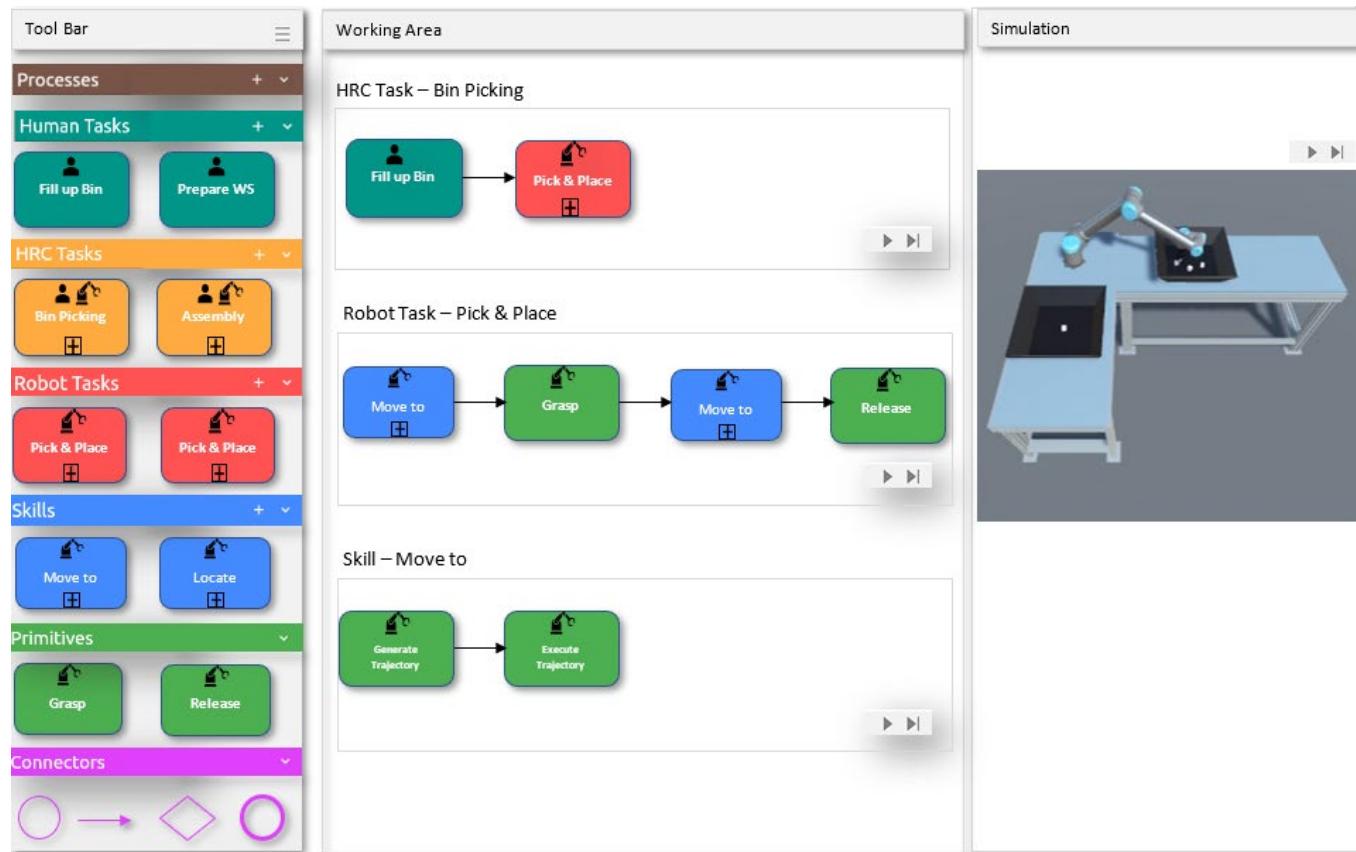
# Research - Questionnaire for ACROBA User Interface

- ▶ Which task representation do you like the most?

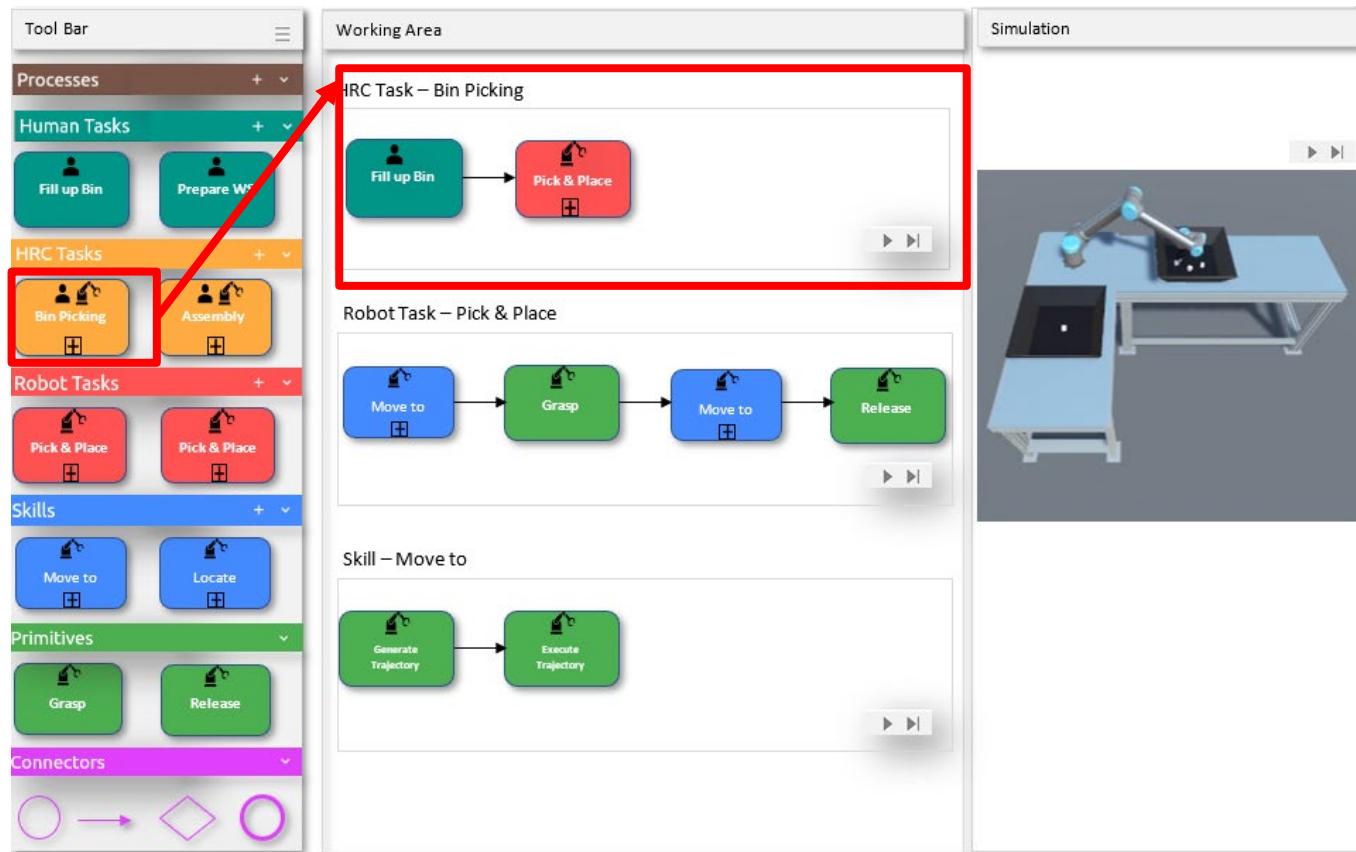


73.3%  
vote  
for  
this  
one

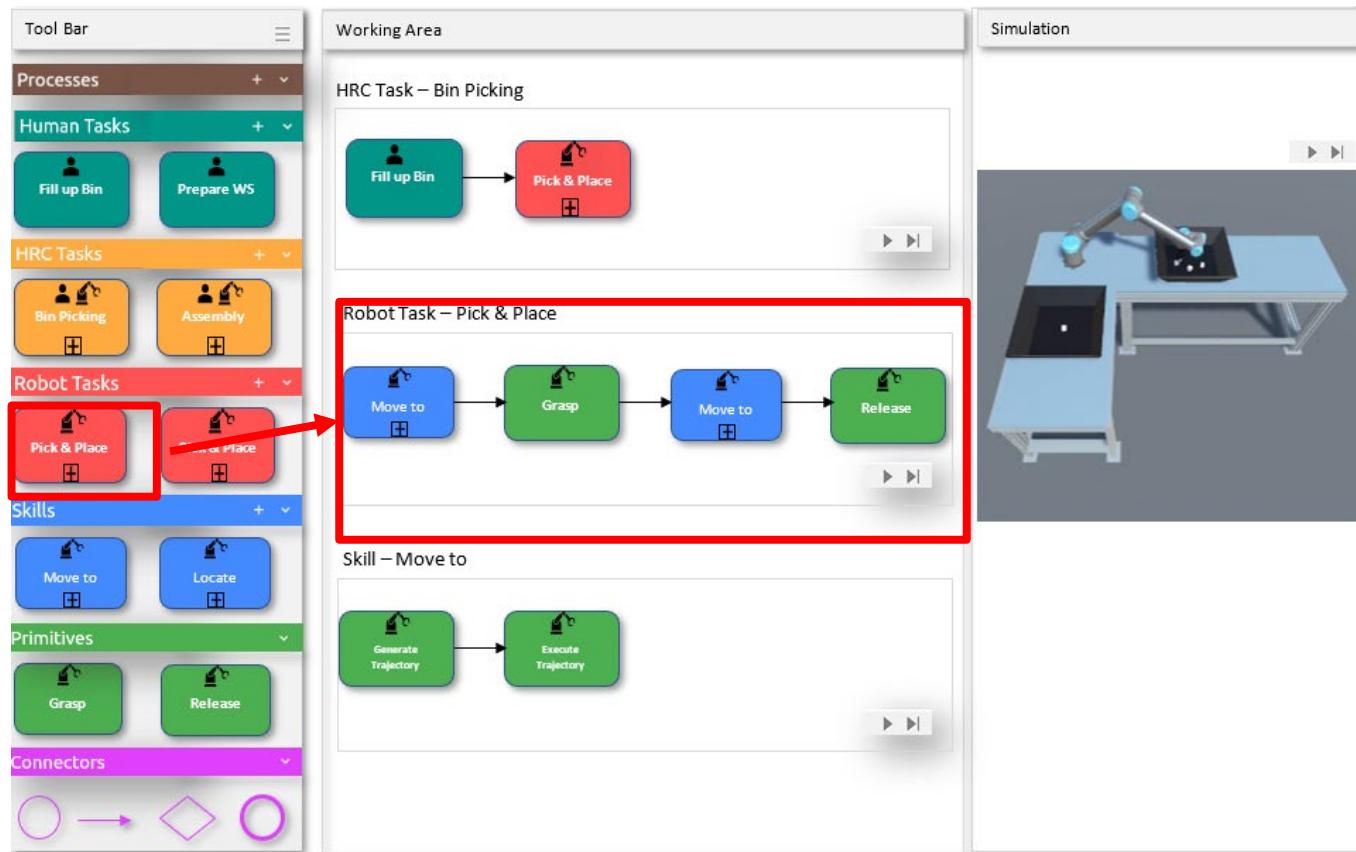
# The RTMN Model - User Interface



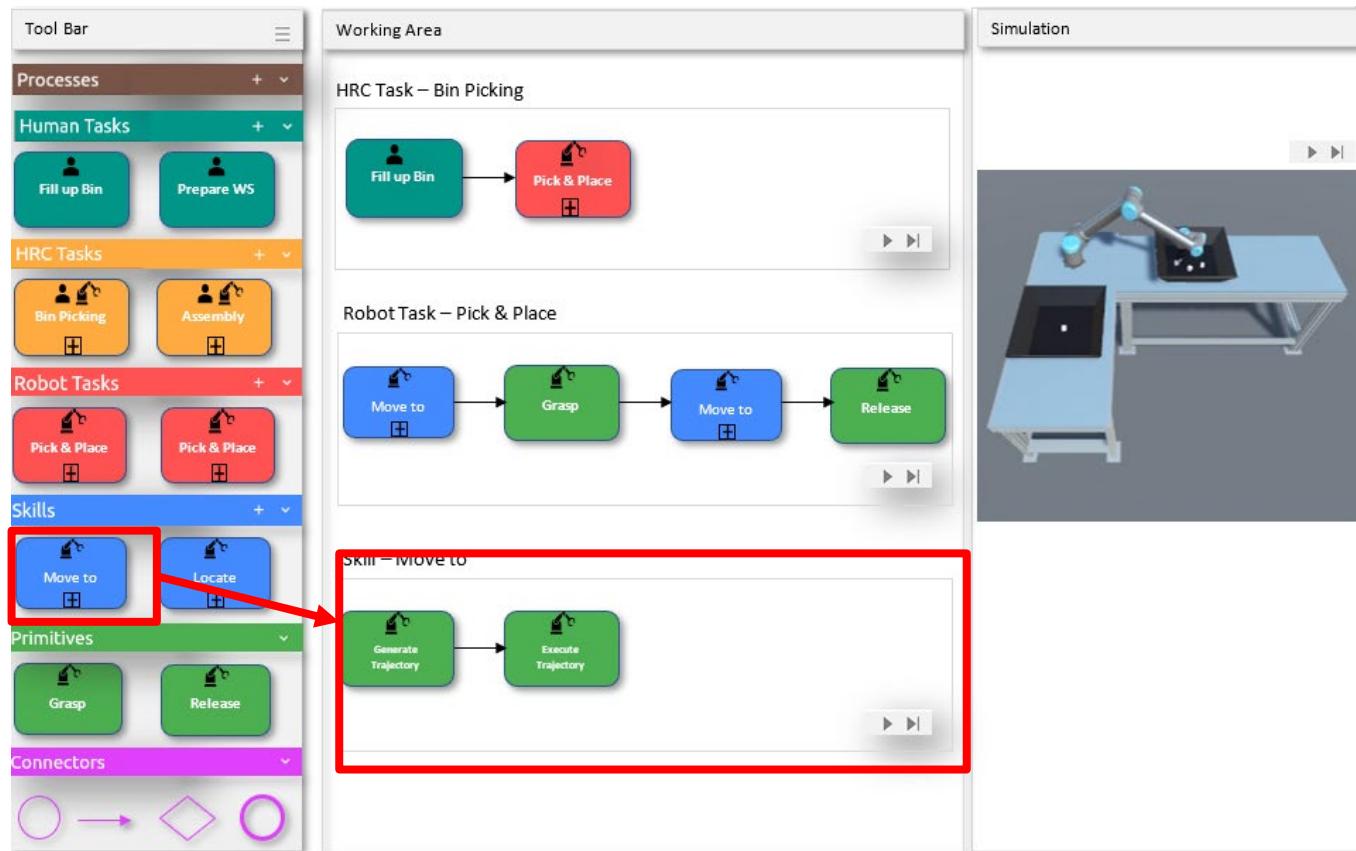
# The RTMN Model - User Interface



# The RTMN Model - User Interface

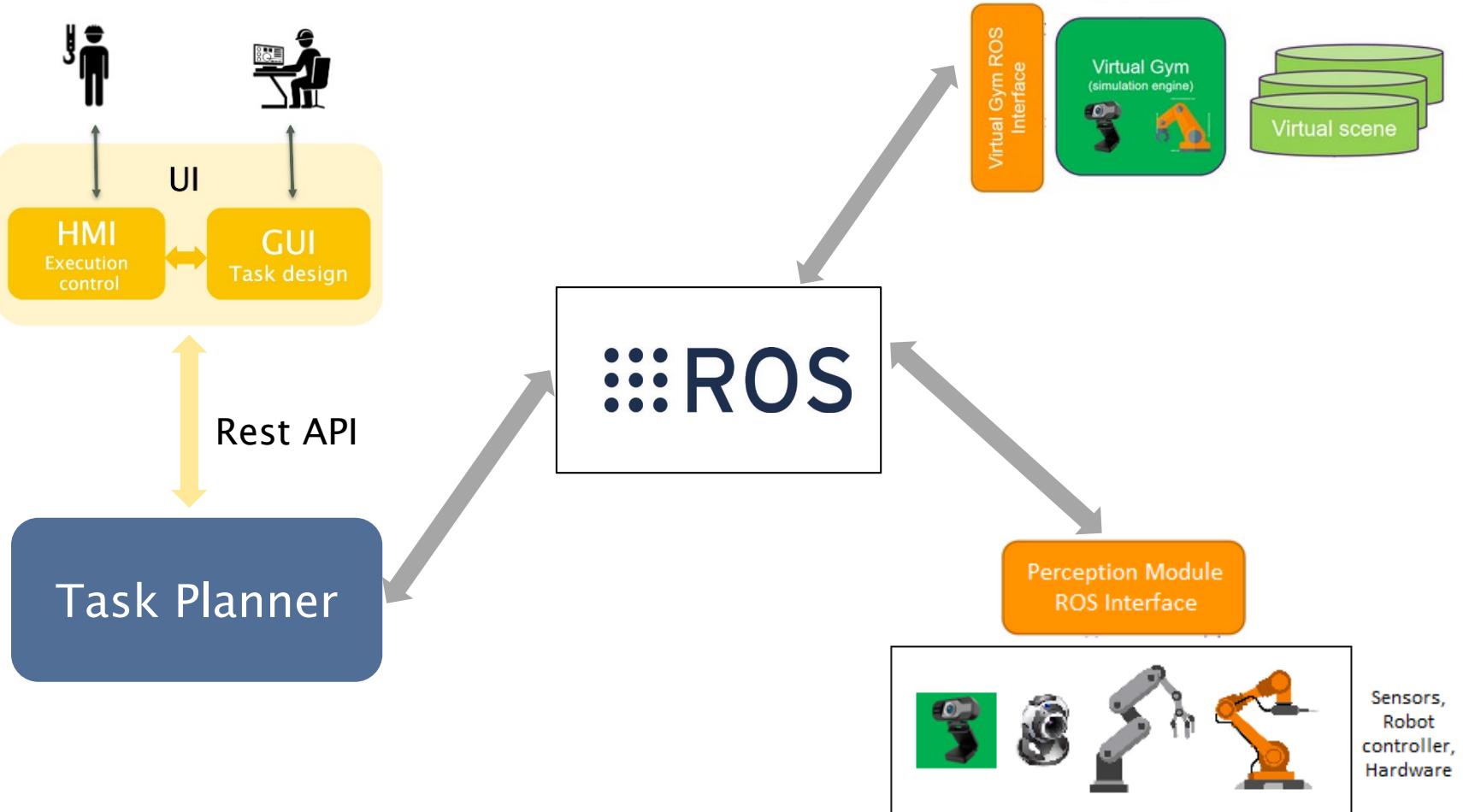


# The RTMN Model - User Interface



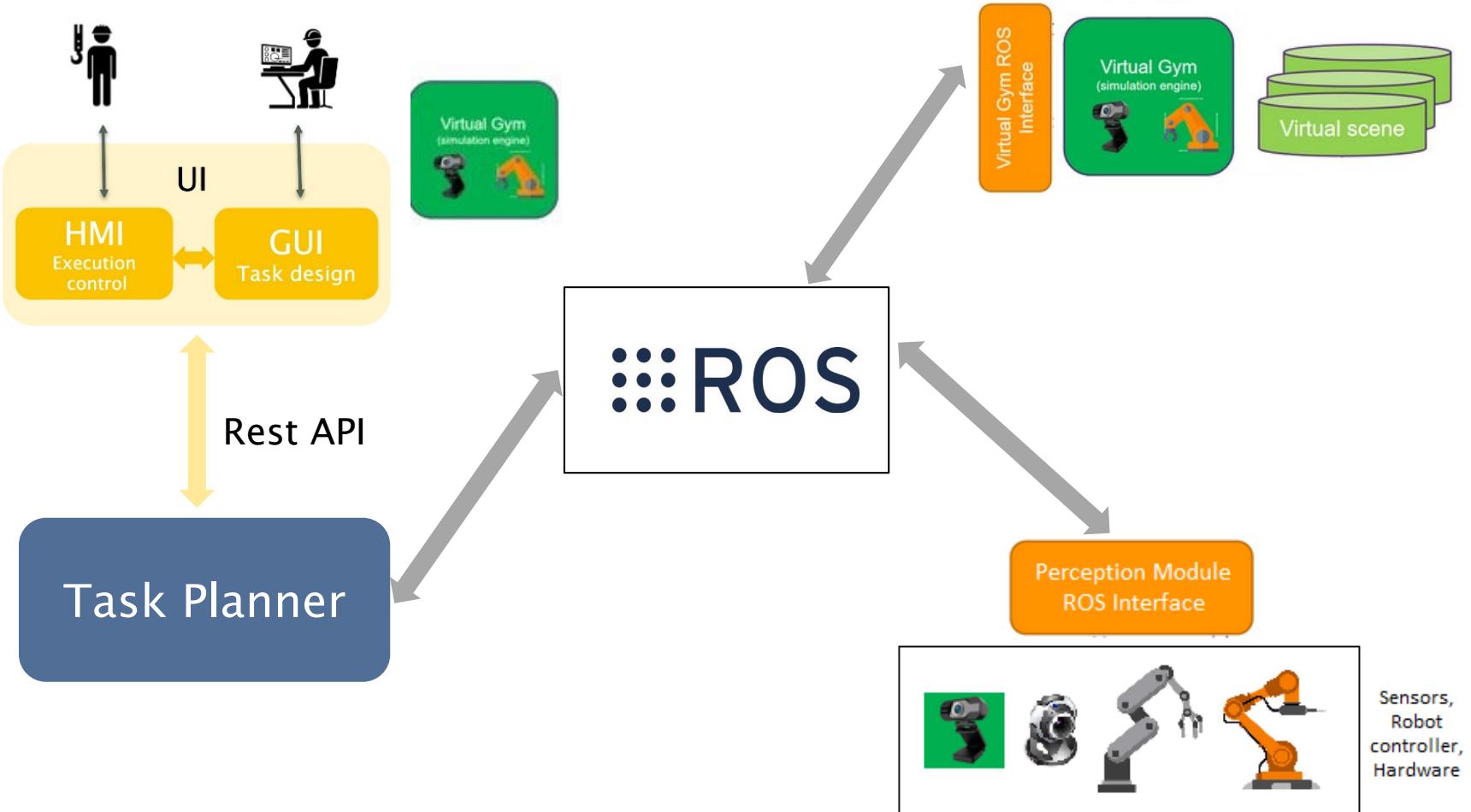
# Flexible Programming of Industrial Robots

## Task Planner



# Flexible Programming of Industrial Robots

## Task Planner



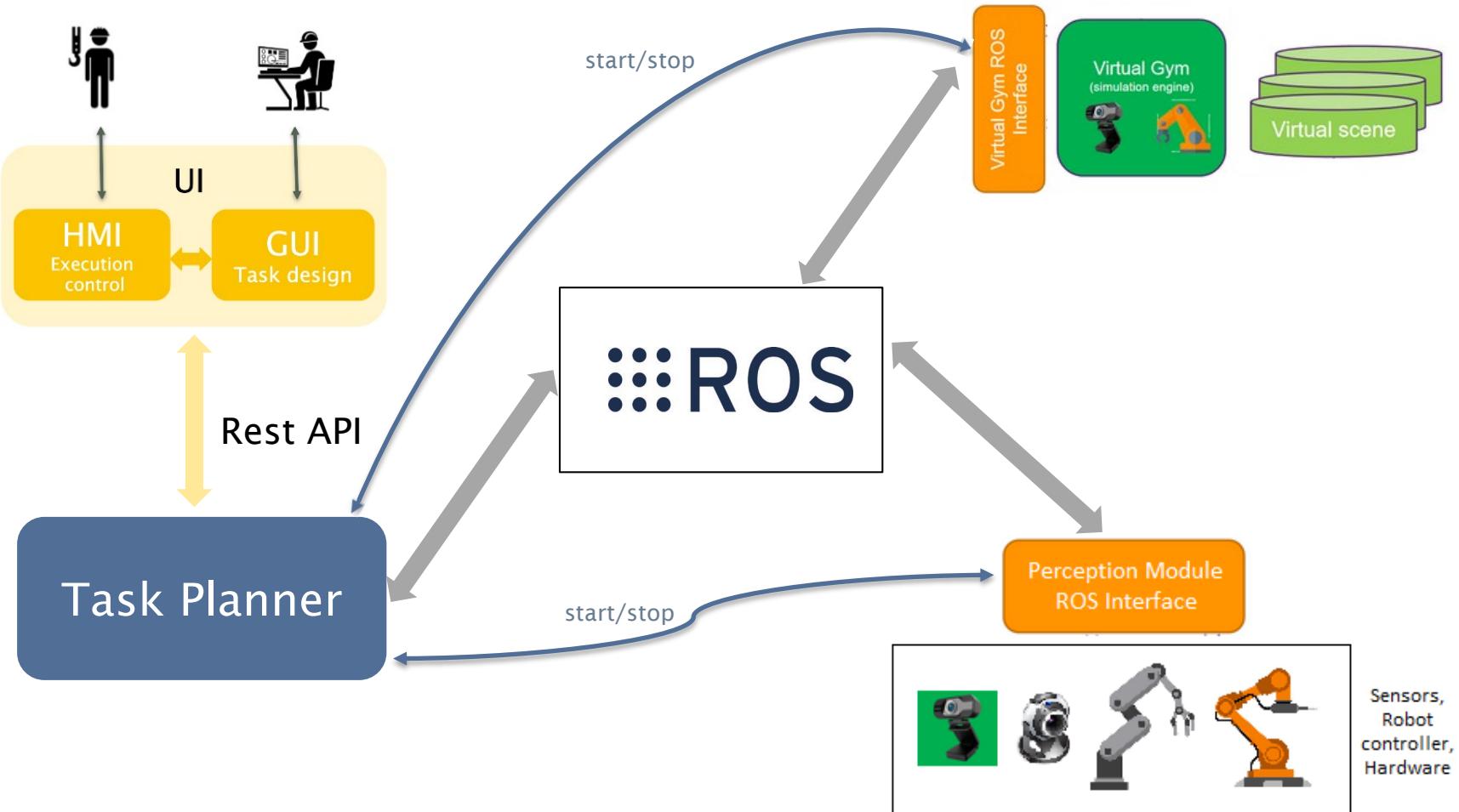
# Flexible Programming of Industrial Robots

## Task Planner

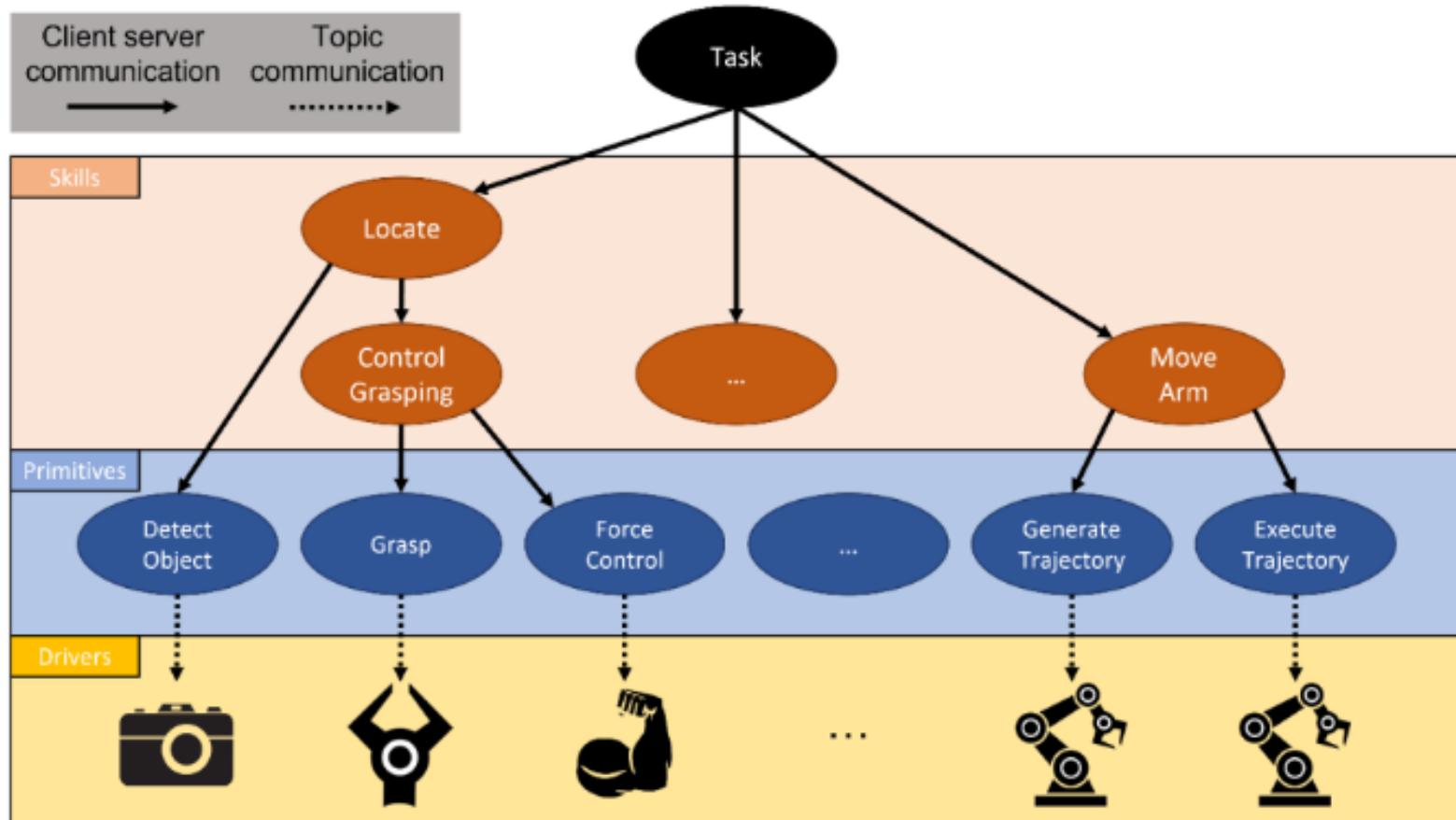
- Generate a task from the UI representation and existing skills/primitives.
- Performs the task execution/control.
- Automatic online replanning

# Flexible Programming of Industrial Robots

## Task Planner



# Flexible Programming of Industrial Robots Skills



# Flexible Programming of Industrial Robots Skills

Primitives	Input (Goals)	Outputs (Results)	Description
CADLoading	String: path String: frame_id	PointCloud2: P2	Primitive that loads a CAD file (stl, .STL, .obj, .pcd or .ply). Then converts it into a point cloud.
Detect Markers	None	None	Primitive that detects specific markers (ARUCO, CHARUCO, QRcode, datamatrix..) to outputs its associated data.
ExecuteTrajectory	RobotTrajectory: tr	None	Primitive that execute a given trajectory and check the proper execution
GenerateGraspPose	bool: visualize (optional)	Pose: id_frame	Primitive that outputs a optimal grasping position, of an object, from a rgbd image.
GenerateTrajectory	string: id_frame Pose[]: pose Float[]: articular_pose bool: cartesian	RobotTrajectory: tr	Primitive that generates a robot trajectory between actual robot state and one or several given points, with a cartesian or articular movement. <small>Primitive that outputs the biggest distance between 2 points of a pointcloud.</small>
GetMaxDist	PointCloud2: point_cloud_in	Float: max_dist	<small>Primitive that outputs the biggest distance between 2 points of a pointcloud.</small>
Grasp	Int: gap=255 (by default)	Bool: finish	Primitive that controls the closure of a Robotiq 2f or Robotiq 3F (pinch) gripper, with object detection. More gripper could be added later on.
Matching	PointCloud2: P4 PointCloud2: P3	Pose: P string: id_frame	Primitive that matches a source pointcloud to an object in the target scene.
OutlierRemoval	PointCloud2: P4 Float: Deviation Int: Mean	PointCloud2: P5	Primitive that removes outliers from a pointcloud. Based on remove_statistical_outlier from open3d library.
PlaneFiltering	PointCloud2: P3 Float: Threshold	PointCloud2: P4	Primitive that filters a plane from a pointcloud. It identifies the biggest possible planes that fits the pointcloud using ransac method ( <a href="https://pypi.org/project/pyransac3d/">https://pypi.org/project/pyransac3d/</a> ). It then filters all points that are at a given distance of this plane.
Point3DTo2Dpixels	Pose: pose camera_info	int: x int: y	Primitive that takes a 3D point in the camera Frame and convert it to 2D pixel coordinates in the image.
Release	Int: gap=0 (by default)	Bool: finish	Primitive that controls the opening of a Robotiq 2f or Robotiq 3F gripper. More gripper could be added later on.
ROISelection	PointCloud2: P1 Float[]: Min Float[]: Max	PointCloud2: P2	Primitive that crop the pointcloud to a given region of interest
Subsampling	PointCloud2: P2 Float: size_leaf	PointCloud2: P3	Primitive to downsample the pointcloud using open3d.geometry.voxel_down_sample

# Flexible Programming of Industrial Robots Skills

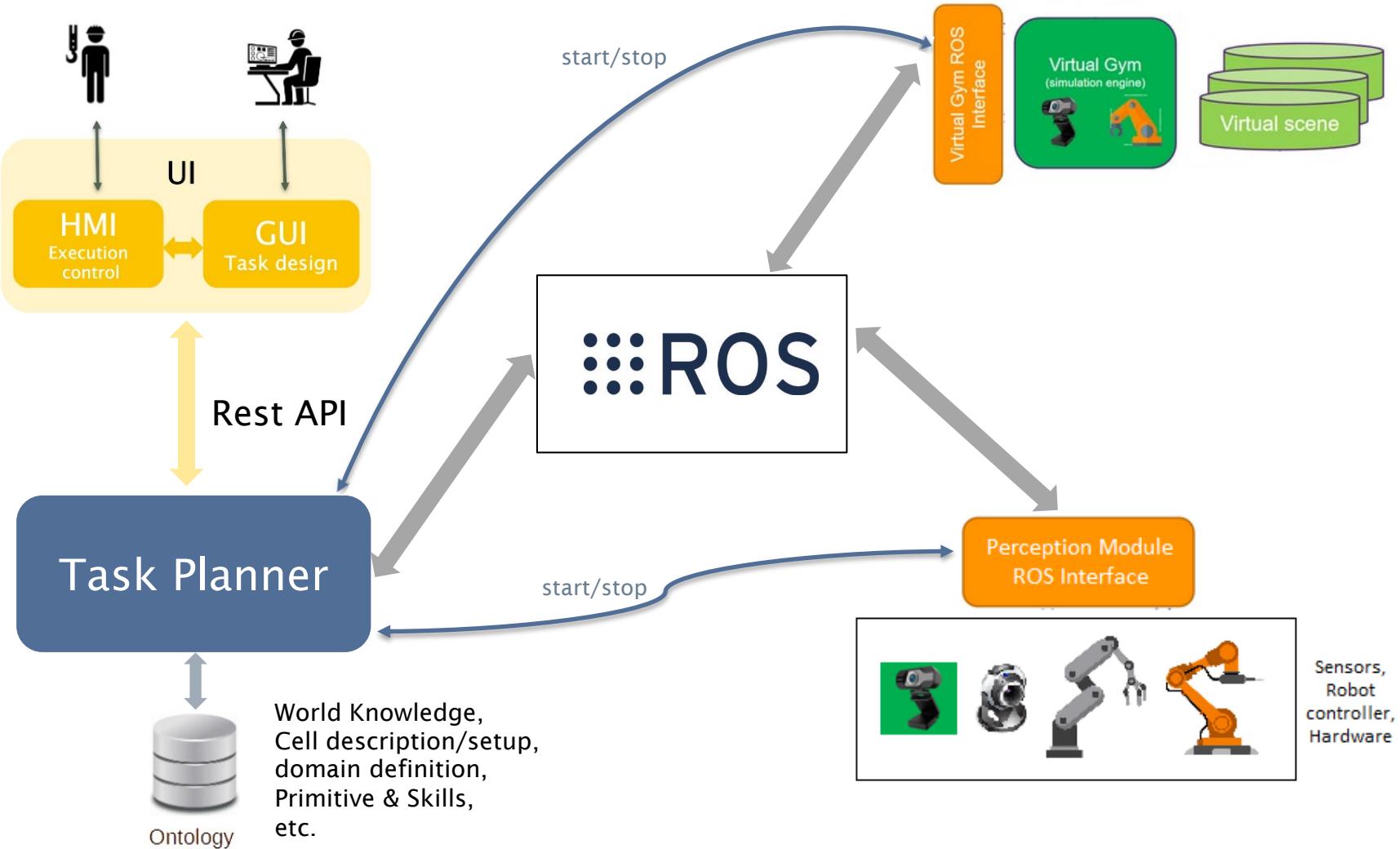
Skills	Goals	Results
PointCloudProcessing	PointCloud2: P1 Float[]: Min Float[]: Max Float: size_leaf Float: Th Float: Deviation Int: Mean	PointCloud2: P5
Pick	string: id_frame Pose[]: pose Float[]: articular_pose bool: cartesian	None
Place	string: id_frame Pose[]: pose Float[]: articular_pose bool: cartesian	None
CADMatching	String: path String: frame_id PointCloud2: P4	Pose: P string: id_frame
MoveTo	string: id_frame Pose[]: pose Float[]: articular_pose bool: cartesian	None
Locate	Float: timeout String: rgb_image_topic_name String: depth_image_topic_name String: pointcloud_topic_name String: camera_info_topic_name String: robot_frame String: depth_frame String: color_frame Pose: points Float: articular_pose Bool: cartesian PointCloud2: point_cloud_in Float: std Int: mean Float: thresh Float: leaf_size Float: min Float: max String: path Float: offset Bool: visualize	Pose: grasping_pose String: frame_id Bool: cartesian

# Flexible Programming of Industrial Robots Skills

Skills	Input (Goals)	Outputs (Results)	Sub-Program (Primitive or Skill)	Type	Input (Goals)	Outputs (Results)
Locate	Float: timeout String: rgb_image_topic_name String: depth_image_topic_name String: pointcloud_topic_name String: camera_info_topic_name String: robot_frame String: depth_frame String: color_frame Pose: points Float: articular_pose Bool: cartesian PointCloud2: point_cloud_in Float: std Int: mean Float: thresh Float: leaf_size Float: min Float: max String: path Float: offset Bool: visualize	Pose: grasping_pose String: frame_id Bool: cartesian	MonitoringCameraInfo	Primitive	String: topic_name Float: timeout	None
	MonitoringDepthImage		Primitive	String: topic_name Float: timeout	None	
	MonitoringPointCloud		Primitive	String: topic_name Float: timeout	None	
	MonitoringRGBImage		Primitive	String: topic_name Float: timeout	None	
	MoveTo		Skill	string: id_frame Pose[]: pose Float[]: articular_pose bool: cartesian	None	
	WorkspaceDetection		Primitive	None	None	
	DataProcessing		Skill	PointCloud2: P1 float[]: plane_eq float: th float[]: leaf_size float[]: min float[]: max	PointCloud2: P2	
	CADMatching		Skill	String: path String: frame_id PointCloud2: P4	Pose: P string: id_frame	
	MoveTo		Skill	string: id_frame Pose[]: pose Float[]: articular_pose bool: cartesian	None	
	GenerateGraspPose		Primitive	bool: visualize (optional)	Pose: id_frame	

# Flexible Programming of Industrial Robots

## Task Planner



# Flexible Programming of Industrial Robots

## Task Planner

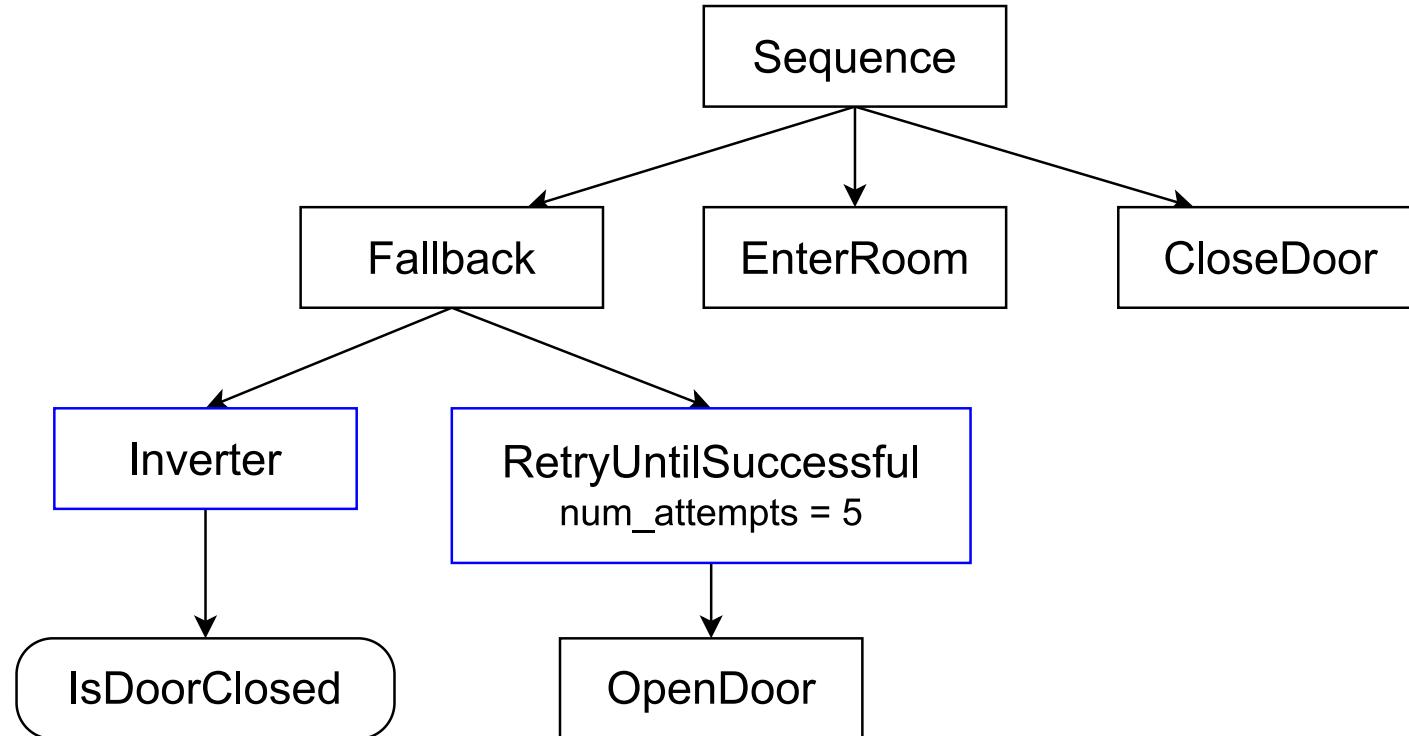
- Generate a task from the existing skills/primitives
- Performs the task execution/control
- Automatic online replanning

# Flexible Programming of Industrial Robots Behavior Trees

- Generalization of Finite State Machine
- Tree of hierarchical nodes control the flow of decision making
- Flexible and powerful
- Used in games to emulate AI, intelligent agent/behavior

# Flexible Programming of Industrial Robots

## Behavior Trees



Source: [https://www.behaviortree.dev/docs/learn-the-basics/BT\\_basics](https://www.behaviortree.dev/docs/learn-the-basics/BT_basics)

# Flexible Programming of Industrial Robots

## Behavior Trees

- ▶ Tick signal sent to the root of the tree
- ▶ Propagate it to through the tree till it reaches a leaf node
- ▶ Tree node receiving a tick executes a callback, which returns either:
  - ▶ SUCCESS
  - ▶ FAILURE
  - ▶ RUNNING
- ▶ If a node has multiple children, it is responsible to propagate to its children

# Flexible Programming of Industrial Robots

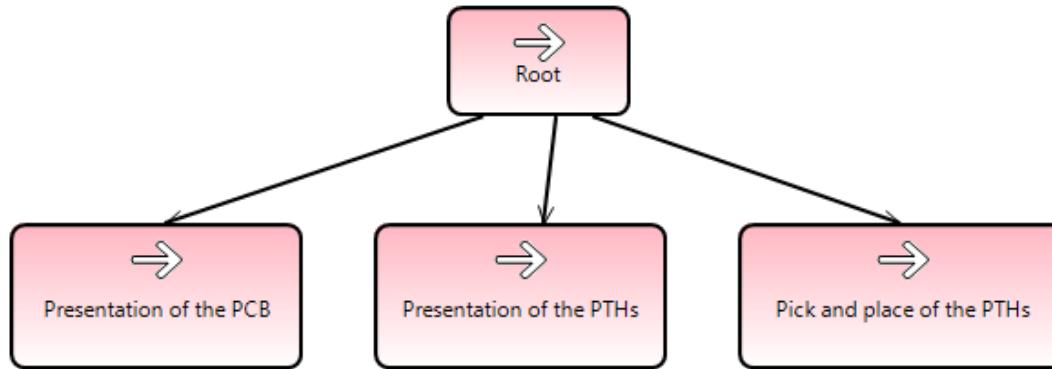
## Behavior Trees

Type of TreeNode	Children Count	Notes
ControlNode	1...N	Usually, ticks a child based on the result of its siblings or/and its own state.
DecoratorNode	1	Among other things, it may alter the result of the children or tick it multiple times.
ConditionNode	0	Should not alter the system. Shall not return RUNNING.
ActionNode	0	This is the Node that "does something"

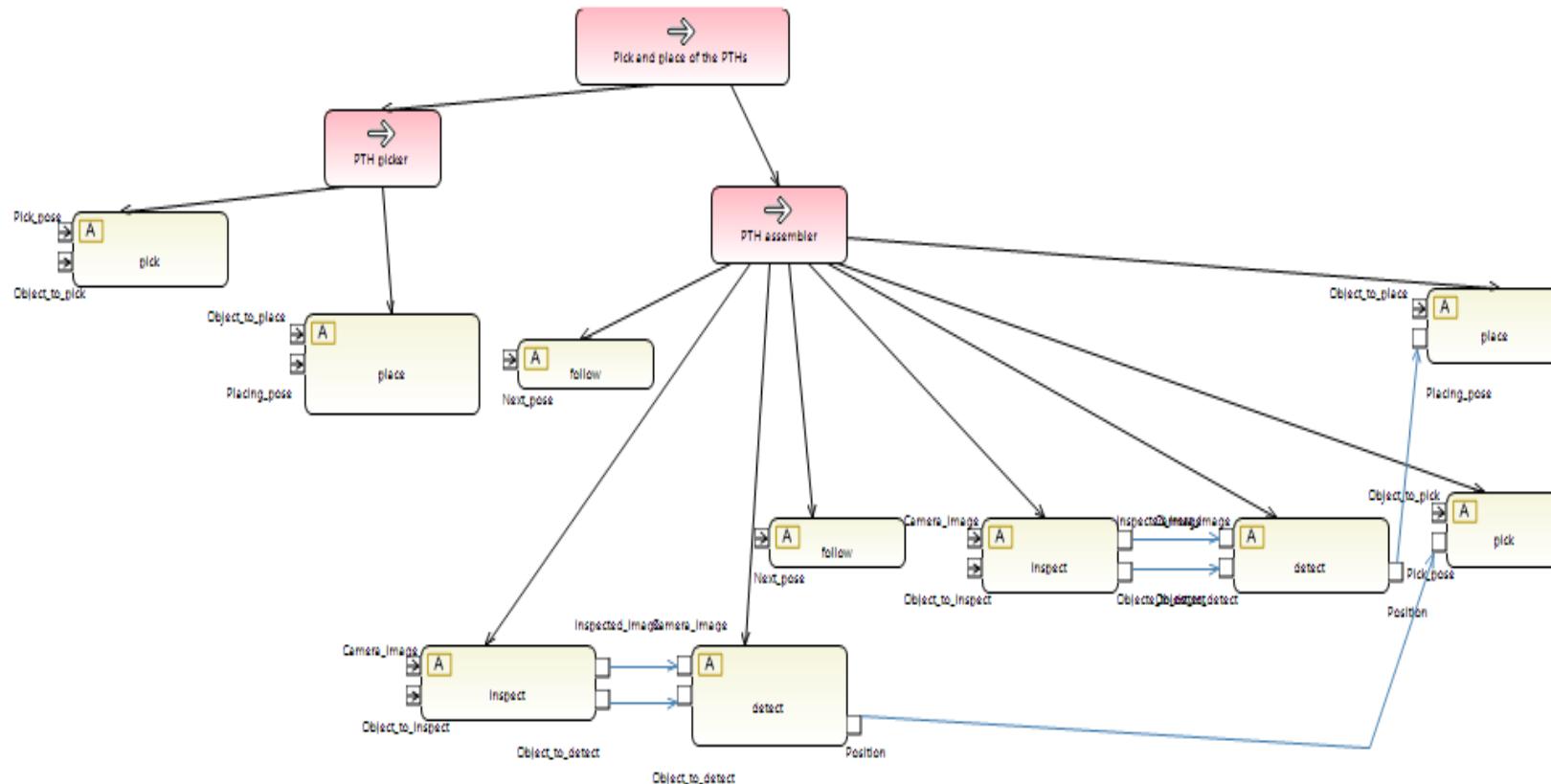
Source: [https://www.behaviortree.dev/docs/learn-the-basics/BT\\_basics](https://www.behaviortree.dev/docs/learn-the-basics/BT_basics)

# Flexible Programming of Industrial Robots

## Behavior Trees



# Flexible Programming of Industrial Robots Behavior Trees



# Flexible Programming of Industrial Robots

## Task Planner

- Generate a task from the existing skills/primitives
- Performs the task execution/control
- **Automatic online replanning**

# Flexible Programming of Industrial Robots

## AI Planning

- Planning Problem:
  - Initial starting state, I
  - Goal state, G
  - Set of actions, A.  
(Each action has Preconditions, Effects)
- From this, a planner generates a plan:
  - Series of actions from A that turn I into G

# Flexible Programming of Industrial Robots

## AI Planning

### PDDL (Planning Language definition)

- Introduced in 1998, extended in 2002, 2004, 2008
- Lisp-like syntax
- Two parts:
  - Domain: abstract predicate definitions, actions
  - Problem: initial state, goal state
- Standardized !
  - write one model, solve it with any planner
  - Easy benchmarking across solvers (same input)
- Domain independent planning

# Flexible Programming of Industrial Robots

## AI Planning

Example:

Domain:

Two rooms, four balls and two robot arms

Actions:

- robot can move between rooms
- pick up a ball
- drop a ball

Problem:

Initial State: All balls and the robot are in the first room

Goal: All balls must be in the second room

# Flexible Programming of Industrial Robots

## AI Planning

- Domain definition:

```
(define (domain <domain name>)
  <PDDL code for predicates>
  <PDDL code for first action>
  [...]
  <PDDL code for last action>
)
```

```
(:predicates (ROOM ?x) (BALL ?x) (GRIPPER ?x)
             (at-roddy ?x) (at-ball ?x ?y)
             (free ?x) (carry ?x ?y))
```

# Flexible Programming of Industrial Robots

## AI Planning

- Domain definition (actions):

```
(:action move :parameters (?x ?y)
  :precondition (and (ROOM ?x) (ROOM ?y)
                      (at-roddy ?x) )
  :effect      (and (at-roddy ?y)
                      (not (at-roddy ?x)) ) )

(:action pick-up :parameters (?x ?y ?z)
  :precondition (and (BALL ?x) (ROOM ?y) (GRIPPER ?z)
                      (at-ball ?x ?y) (at-roddy ?y) (free ?z) )
  :effect      (and (carry ?z ?x)
                      (not (at-ball ?x ?y)) (not (free ?z)) ) )
```

# Flexible Programming of Industrial Robots

## AI Planning

- Problem definition:

```
(define (problem <problem name>)
  (:domain <domain name>)
  <PDDL code for objects>
  <PDDL code for initial state>
  <PDDL code for goal specification>
)
```

```
(:objects rooma roomb
          ball1 ball2 ball3 ball4
          left right)
```

# Flexible Programming of Industrial Robots

## AI Planning

- Problem definition:

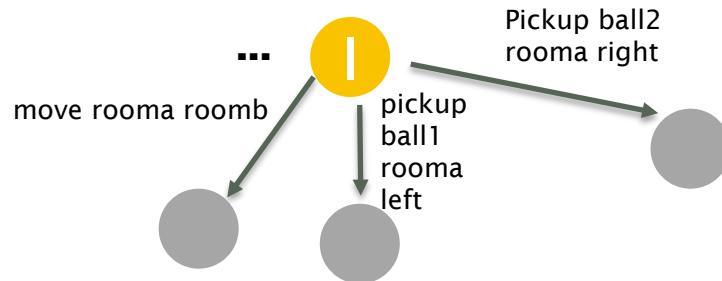
```
(:init (ROOM rooma) (ROOM roomb)
       (BALL ball1) (BALL ball2) (BALL ball3) (BALL ball4)
       (GRIPPER left) (GRIPPER right) (free left) (free right)
       (at-roby rooma)
       (at-ball ball1 rooma) (at-ball ball2 rooma)
       (at-ball ball3 rooma) (at-ball ball4 rooma))

(:goal (and (at-ball ball1 roomb)
            (at-ball ball2 roomb)
            (at-ball ball3 roomb)
            (at-ball ball4 roomb))))
```

# Flexible Programming of Industrial Robots

## AI Planning

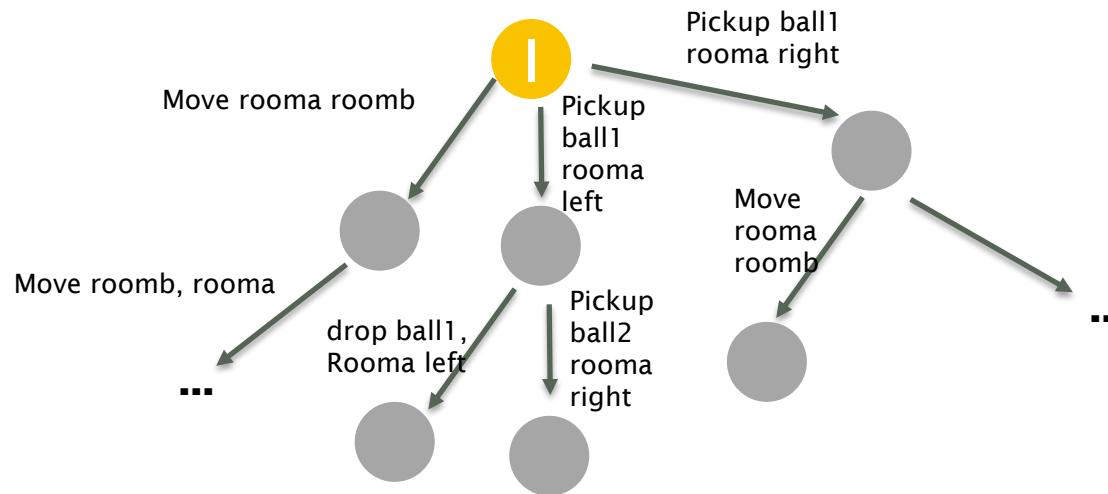
- Solving: Forward chain planning



# Flexible Programming of Industrial Robots

## AI Planning

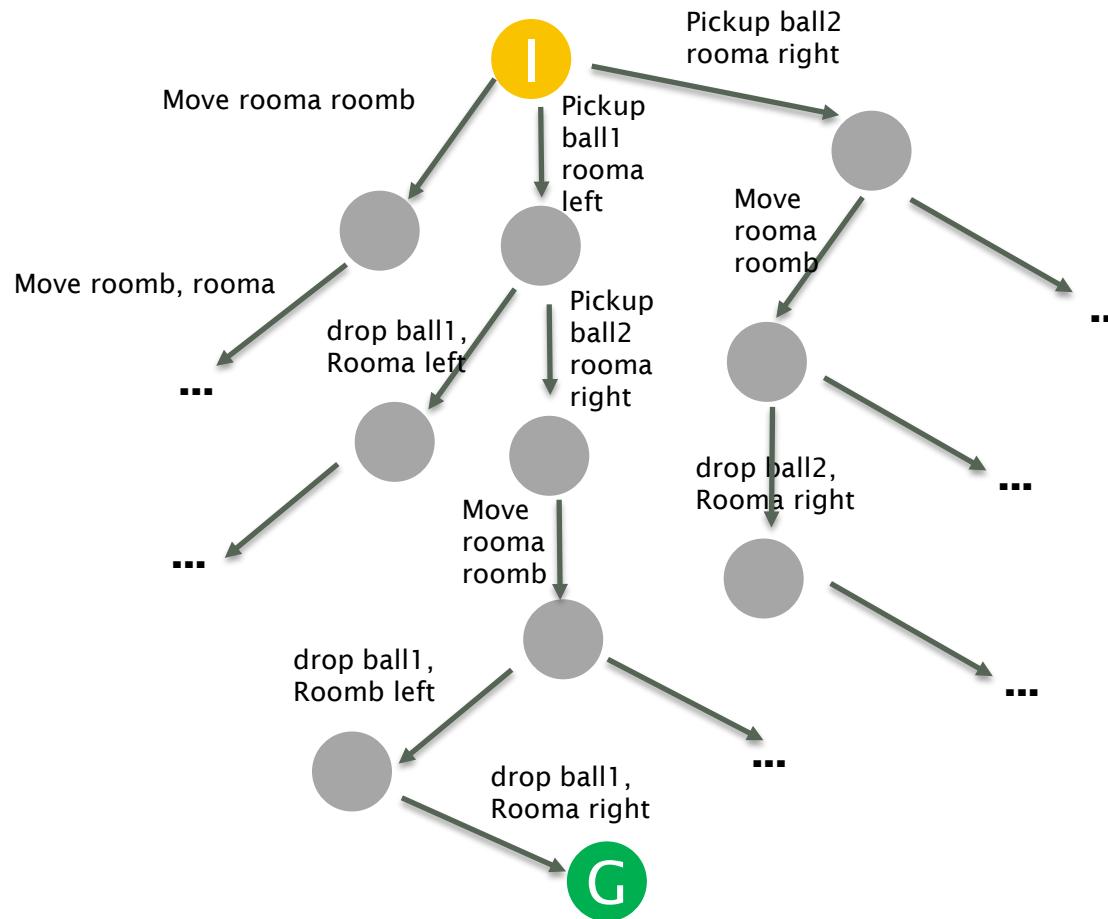
- Solving: Forward chain planning



# Flexible Programming of Industrial Robots

## AI Planning

- Solving: Forward chain planning



# Flexible Programming of Industrial Robots

## AI Planning

- Solving: Forward chain planning
- Solution plan:
  - pickup ball1 rooma left
  - pickup ball4 rooma right
  - move rooma roomb
  - drop ball1
  - drop ball4
  - move roomb rooma
  - pickup ball3 rooma left
  - pickup ball2 rooma right
  - move rooma roomb
  - drop ball3
  - drop ball2

# Flexible Programming of Industrial Robots

## AI Planning

Search strategies:

- Breadth First Search
- Depth First Search

Planning problem is Hard:

Suppose:

20 actions to choose from

- Solution plan is 20 steps long.
- Worst case search  $20^{20} = 1e26$   
1 million iterations per second, ETA  $1e20$  seconds  
Age of the universe in seconds =  $4.35e+17$  seconds!

# Flexible Programming of Industrial Robots

## AI Planning



Key challenge: search guidance!

Use of heuristics:

- function giving an estimate of how far we are from a goal.
- use simplified problem
- use heuristics that never over-estimates

# Flexible Programming of Industrial Robots

## AI Planning

Initial state



Goal state



Source: Introduction to AI Planning, Part 1 (Amanda Coles, EASSS 2013)

# Flexible Programming of Industrial Robots

## AI Planning

Initial state



Goal state



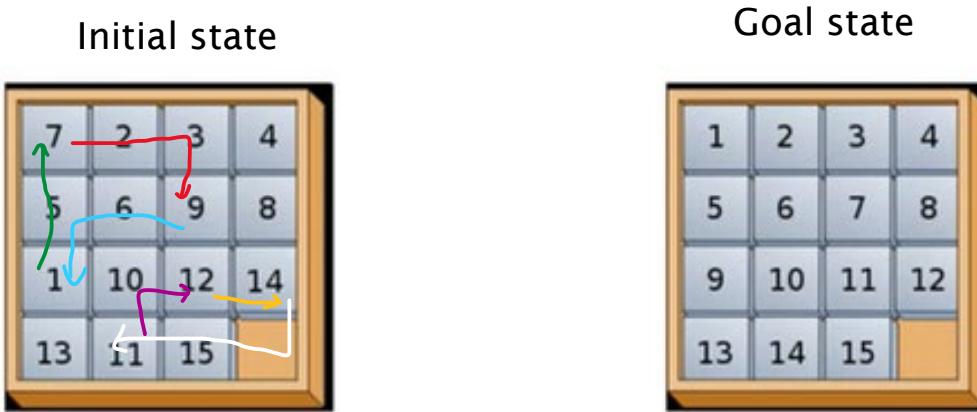
Heuristic: number of misplaced tiles

$$H(\text{initial state}) = 6$$

Source: Introduction to AI Planning, Part 1 (Amanda Coles, EASSS 2013)

# Flexible Programming of Industrial Robots

## AI Planning



Heuristic: sum of distance to correct position of misplaced tiles

$$h(\text{initial state}) = 2 + 3 + 3 + 2 + 1 + 3 = 14$$

Source: Introduction to AI Planning, Part 1 (Amanda Coles, EASSS 2013)

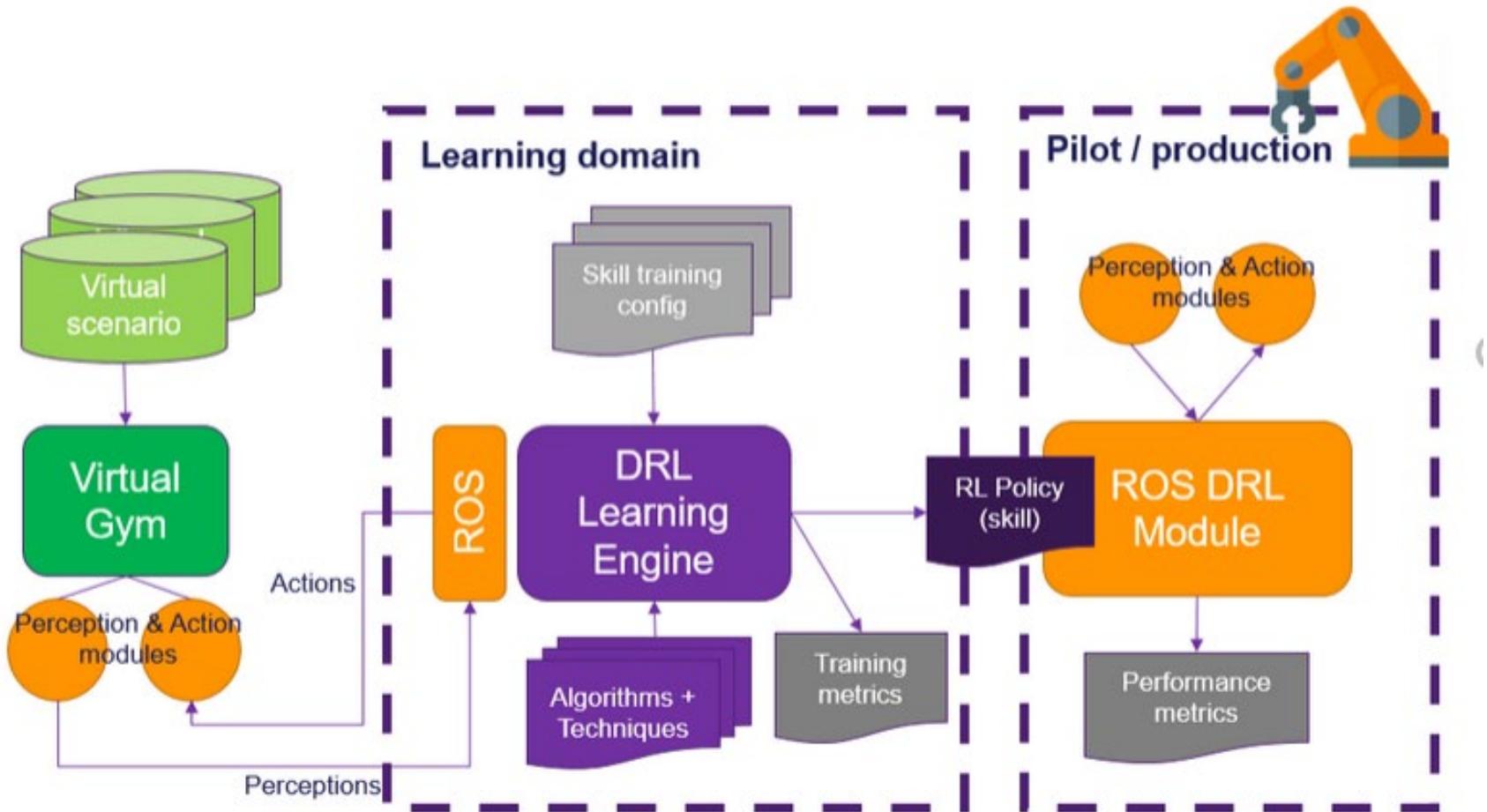
# Flexible Programming of Industrial Robots

## AI Planning

And then ?

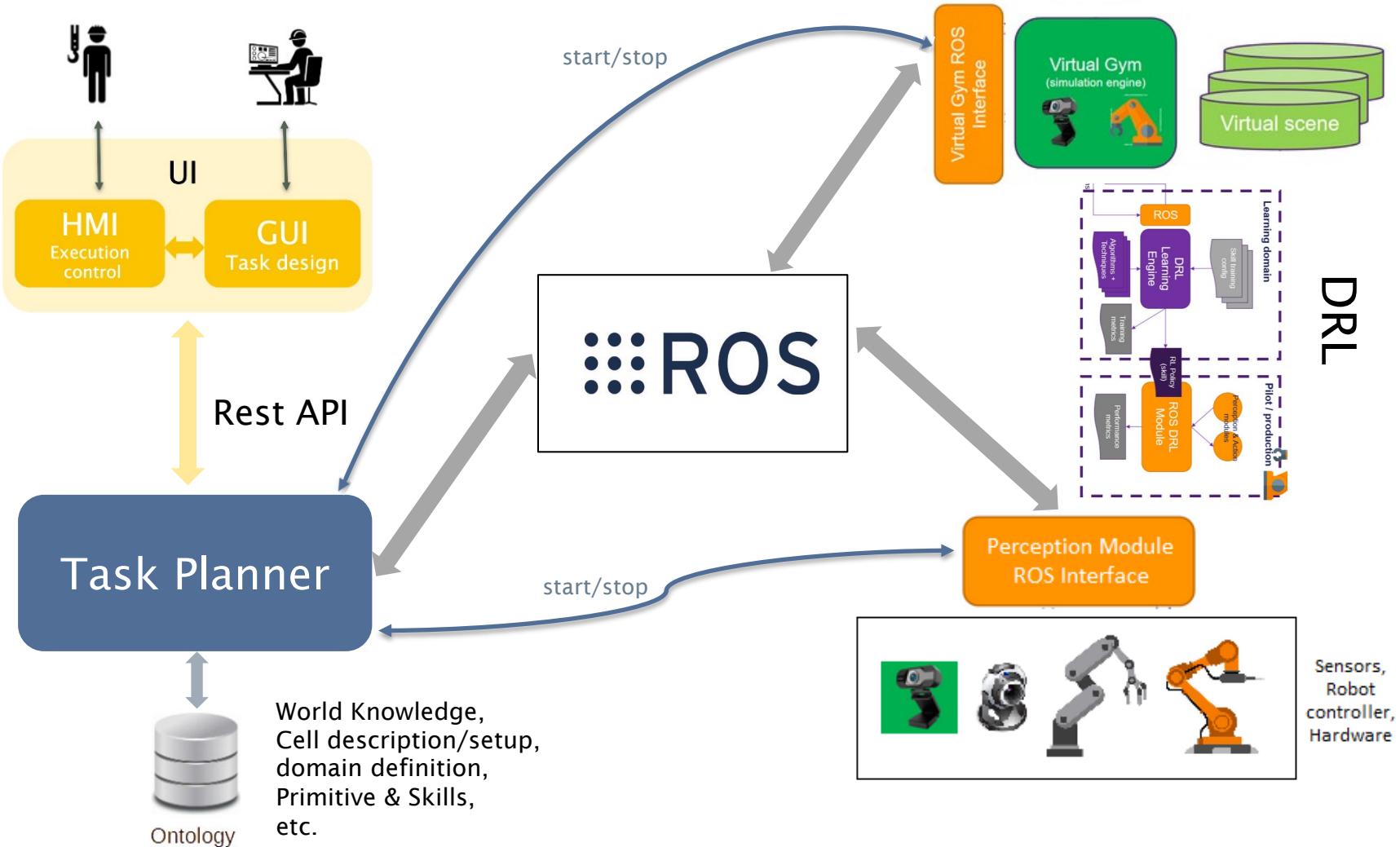
- Solution plan:
    - pickup ball1 rooma left
    - pickup ball2 rooma right
    - move rooma roomb
    - drop ball1
    - drop ball2
-  Behavior Tree generation, then task Execution
- Alternative? Reinforcement Learning.

# Flexible Programming of Industrial Robots Deep Reinforcement Learning



# Flexible Programming of Industrial Robots

## How Flexible?



# Flexible Programming of Industrial Robots

## How Flexible?

- ▶ Platform shipped with a set of skills & compatible hardware
- ▶ UI & BTs gives great flexibility.
- ▶ Challenges:
  - ▶ Ontology definitions (robotics engineer)
  - ▶ Virtual scene definition (templates?)
  - ▶ Hardware abstraction?

# Flexible Programming of Industrial Robots



# Next seminars

## Biel/Bienne

Quellgasse 21, Aula

**25.11.22 Experimental heart rate variability characterization** Lars Brockmann, Assistant, Institute for Human Centered Engineering HuCE, BFH-TI

**09.12.22 Parylene-based encapsulation technology for wearable or implantable electronic devices** Dr. Andreas Hogg, CEO, Coat-X AG, La Chaux-de-Fonds

**13.01.23 Care@Home mit technischer Unterstützung** Prof. Dr. Sang-II Kim, Professor, Institute for Medical Informatics I4MI, BFH-TI

## Burgdorf/Berthoud

Pestalozzistrasse 20, E 013

**02.12.22 Wie gefährlich ist ein Unfall mit einem Cabriolet?** Prof. Raphael Murri, Institutsleiter IEM, Institut für Energie- und Mobilitätsforschung IEM, BFH-TI

**16.12.22 Systemtechnologie für die Mikrobearbeitung mit Hochleistungs-UKP-Lasern** Prof. Dr. Beat Neuenschwander, Institutsleiter ALPS, Institute for Applied Laser, Photonics and Surface Technologies ALPS, BFH-TI