An Agent-based Cyber-Physical Production System using Lego Technology

Metehan Mustafa Yalçın, **Burak Karaduman**, Geylani Kardaş, Moharram Challenger

MICSS Lab

Modelling Intelligent Complex Software and Systems Lab

AnSyMo (Antwerp Systems and Software Modeling) is a research group in the Department of Computer Science investigating foundations, techniques, methods, and tools for the design, analysis, and maintenance of software-intensive systems.

Content

1. Introduction

- **Cyber-Physical Systems:**
  - New paradigm and new design challenges.
  - The information processing and computation are merged with communication and control .
- **Cyber-physical Production Systems (CPPS)**
  - Autonomous and cooperative systems.
  - Subsystems, production and process.
- **Smart Manufacturing**
  - Considers adapting the embedding SW and HW tech. to the CPS,
  - Includes intelligent methodologies.
  - Aims to increase efficiency for production.

Universiteit Antwerpen

1. Introduction

CPS: Cyber-Physical Systems (CPS) integrate **computing** and **communication** capabilities with **monitoring** and **control** of entities in the physical world. These systems are usually **composed by a set of networked agents**, including: **sensors**, **actuators**, **control processing units**, and **communication devices** [1].

sCPS: sCPS adds the features of computational **intelligence** on CPS that makes them capable to building **awareness**, **reasoning** about the objectives, states of **operations**, and **adaptation** (to obtain knowledge out of the collected information by monitoring the environment) [2-3] .

sCPSoS: A sCPSoS is a sCPS which exhibit the features of SoS including [4, 5]:

- Distributed (heterogeneous) physical systems (sensors and actuators)

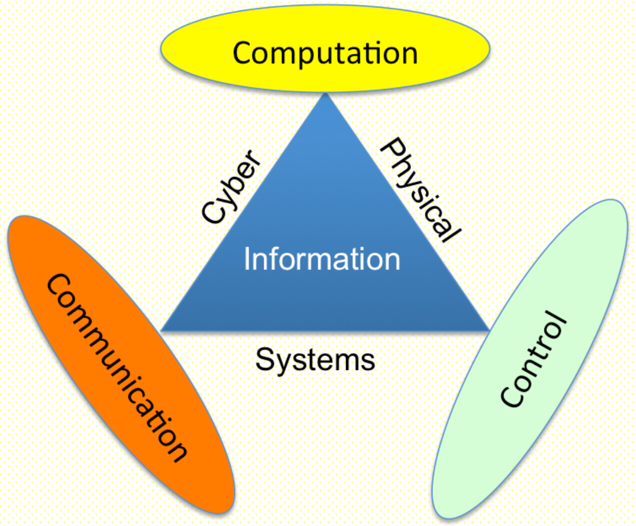
- Distributed control, supervision and management

- (Partial) Autonomy of the subsystems

- Dynamic reconfiguration of the system

- Emerging behaviours

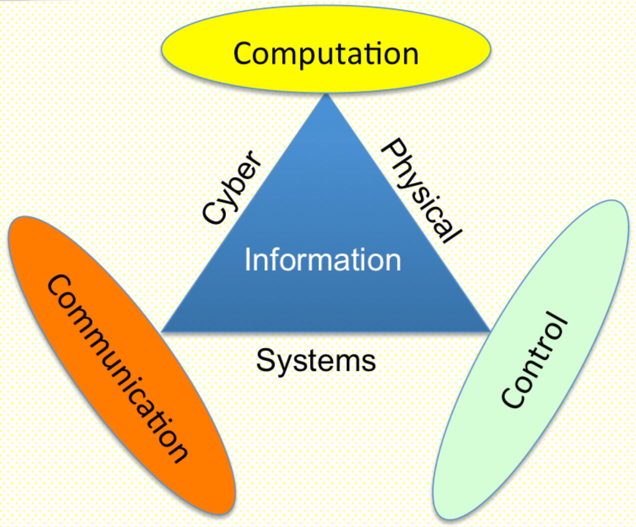
- Continuous evolution of the system during its operation



Universiteit Antwerpen

Challenges in sCPSoS (Why?)

- **Cyber-Physical Systems:**
  - Structural complexity (composed of various components).
  - Resource-constraint (cyber part).
  - Uncertainty (physical part).
- **Cyber-Physical System-of-Systems:**
  - Heterogeneous (more complex).
  - Distributed multiple CPS instances (with different stakeholders).
  - Autonomy of subsystems.
  - Emergent and dynamic behaviors.
- **Smart CPSoS:**
  - Reasoning mechanisms.
  - Awareness and Adaptability.



Universiteit Antwerpen

2. Challenges (Why?)

Production Problems

- Flexibility
- Safety
- Complexity
- Lack of intelligence

Reasons

- Frequent interactions.
- Co-operative requirements
- between machines.
- Lack of human experts.
- No intelligent mechanism to reason unpredictable behaviors.

Universiteit Antwerpen

2. Intelligence for CPS: next generation CPPS

Why Multi-Agent Systems?

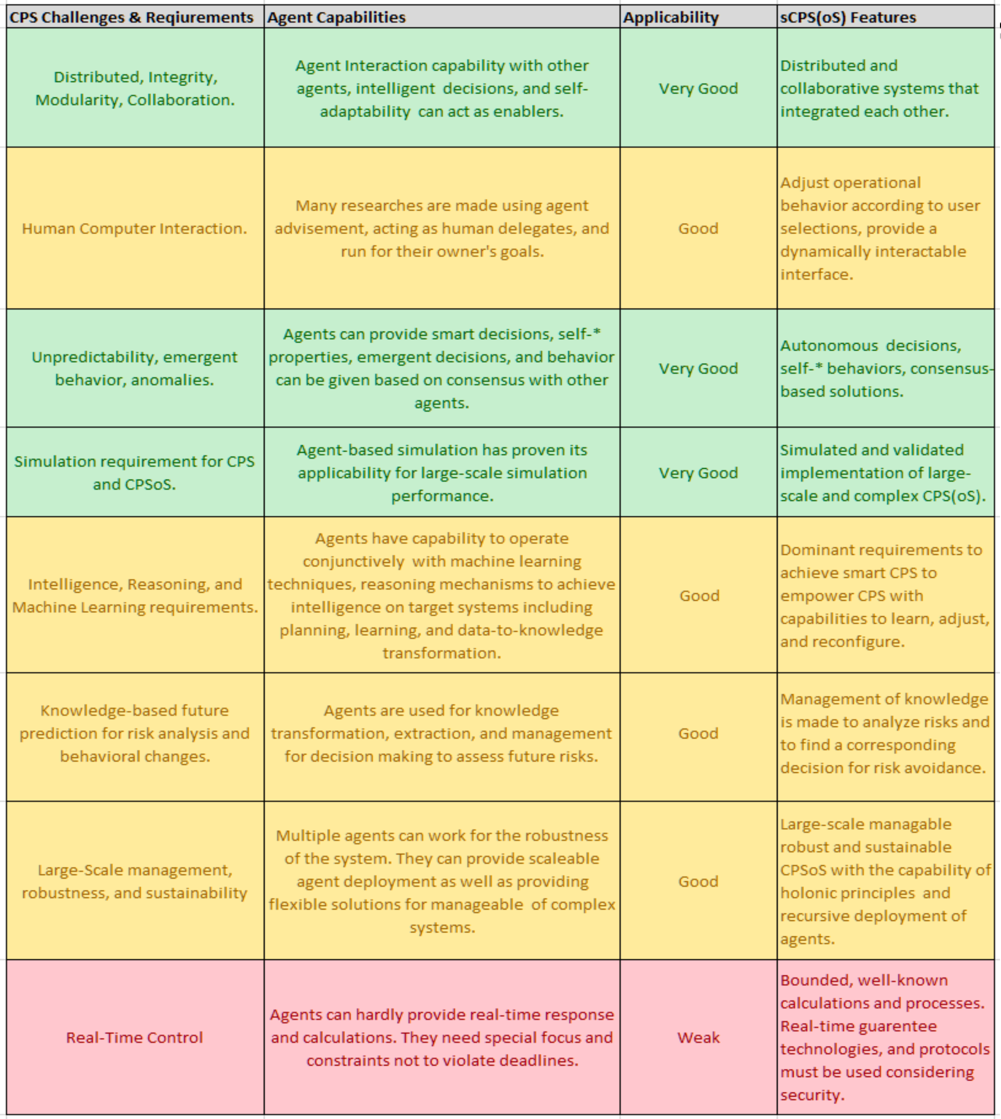
❖ To cope with the challenges of CPS, many studies propose benefiting from the features of multi-agent systems (MAS)
❖ MAS provide:

1. Smartness
2. Decentralization
3. Autonomy and socialization of CP(P)S.

❖ They increase the effectiveness of CPS, enhanced functionalities for production and automation.
❖ The software agents:

1. Control functions/parameters of the system.
2. Can monitor transition between processes.
3. Observe the human-errors.

2. Intelligence for CPS : MAS Capabilities and CPS Challenges



CPS Challenges & Reqirements	Agent Capabilities	Applicability	sCPS(oS) Features
Distributed, Integrity, Modularity, Collaboration.	Agent Interaction capability with other agents, intelligent decisions, and self-adaptability can act as enablers.	Very Good	Distributed and collaborative systems that integrated each other.
Human Computer Interaction.	Many researches are made using agent advisement, acting as human delegates, and run for their owner's goals.	Good	Adjust operational behavior according to user selections, provide a dynamically interactable interface.
Unpredictability, emergent behavior, anomalies.	Agents can provide smart decisions, self-* properties, emergent decisions, and behavior can be given based on consensus with other agents.	Very Good	Autonomous decisions, self-* behaviors, consensus-based solutions.
Simulation requirement for CPS and CPSoS.	Agent-based simulation has proven its applicability for large-scale simulation performance.	Very Good	Simulated and validated implementation of large-scale and complex CPS(oS).
Intelligence, Reasoning, and Machine Learning requirements.	Agents have capability to operate conjunctively with machine learning techniques, reasoning mechanisms to achieve intelligence on target systems including planning, learning, and data-to-knowledge transformation.	Good	Dominant requirements to achieve smart CPS to empower CPS with capabilities to learn, adjust, and reconfigure.
Knowledge-based future prediction for risk analysis and behavioral changes.	Agents are used for knowledge transformation, extraction, and management for decision making to assess future risks.	Good	Management of knowledge is made to analyze risks and to find a corresponding decision for risk avoidance.
Large-Scale management, robustness, and sustainability	Multiple agents can work for the robustness of the system. They can provide scaleable agent deployment as well as providing flexible solutions for manageable of complex systems.	Good	Large-scale managable robust and sustainable CPSoS with the capability of holonic principles and recursive deployment of agents.
Real-Time Control	Agents can hardly provide real-time response and calculations. They need special focus and constraints not to violate deadlines.	Weak	Bounded, well-known calculations and processes. Real-time guarentee technologies, and protocols must be used considering security.

MAS Capabilities and CPS Challenges (Adapted from [1]).

[1] Leitao, Paulo, et al. "Smart agents in industrial cyber–physical systems." Proceedings of the IEEE 104.5 (2016): 1086-1101.

An Agent-based Cyber-Physical Production System using Lego Technology

Universiteit Antwerpen

3. Motivation (How?)

Multi-agent Systems (MAS)

- Smartness, decentralization, autonomy, and socialization for CPS.
- Decide reconfiguration of the control functions/parameters.
- Monitor transition between processes and observe the human errors.

MAS and CPS

- Once agents controls CPS' components, the developer can focus on higher-level solutions (e.g intelligence mechanisms).
- Industrial systems might be placed in dangerous environment
- It may be a burden to prototype an actual industrial system.
- Therefore, we need a technology to miniaturize any physical system, sustaining its functionality, accuracy and goal-orientedness.

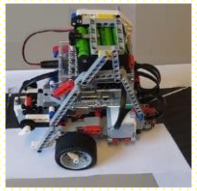
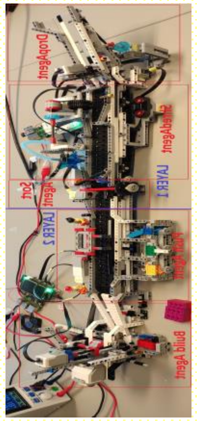
Universiteit Antwerpen

3. Motivation

What is needed?

1. A composable and concrete technology.
2. One of the technologies commonly used for imitating such systems is LEGO.
3. However, it is not possible to integrate software agents and any intelligence mechanism easily.
4. Hence, in this study, we introduce an agent-based implementation of a system using Lego technology.
5. We focus on the low-level integration requirements of CPS and software agents.

3. Motivation - LEGO and Embedded Systems

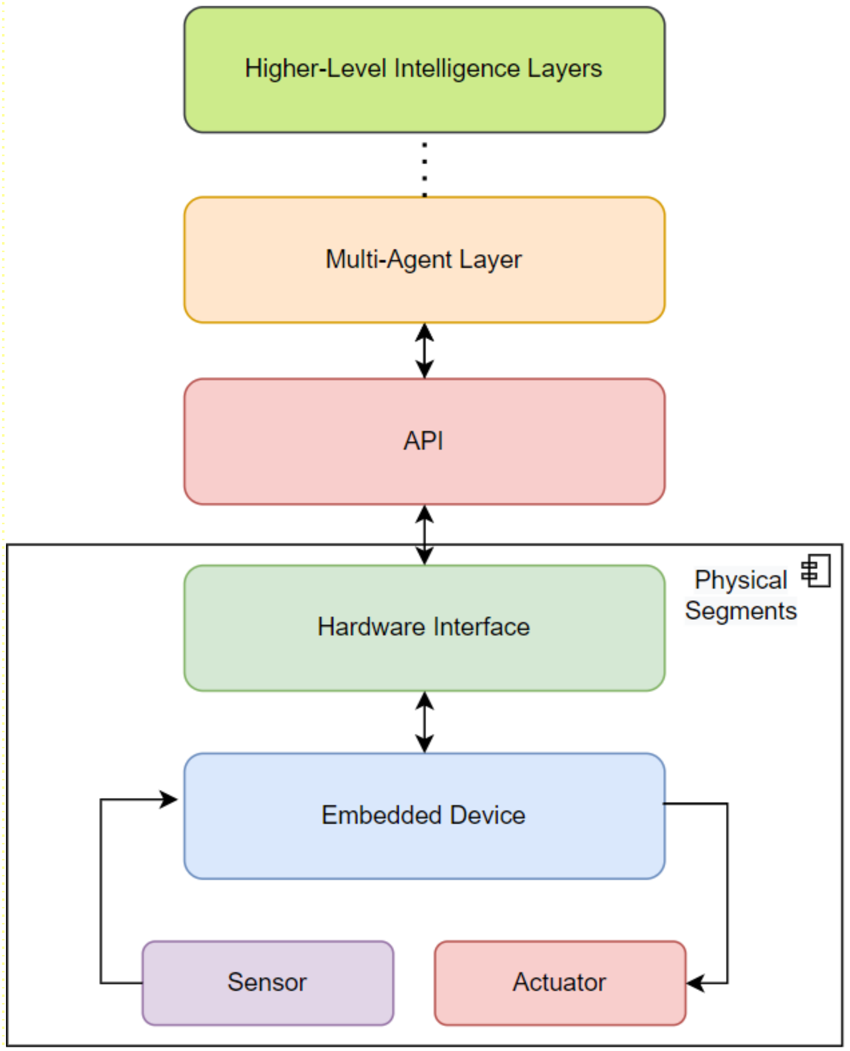
Hardware Options	OS Options	Software Options	Plant
BrickPI + RaspberryPI 3	Ev3Dev + Debian Linux	Java + Jade Java Agent DEvelopment Framework	
PiStorms + RaspberryPI 3	Ev3Dev + Raspbian Linux	python™ + SPADE Smart Python Agent Development Environment	
EV3Dev (Standalone)	Ev3Dev + Debian Linux	Java + Jade Java Agent DEvelopment Framework	

4. System Software Analysis and Design

Layer 0: Physical Segments This layer includes all physical layers and components of a CPS. The segments can consist of many passive and physical type entities. They conform to physics laws. A set of passive components create a segment. Our study addressed this layer using LEGO technology and composable parts made of assembled plastic bricks.

Layer 1: Sensor and Actuator Layer The sensor and actuator layer describes the physical environment where the CPS was located. A CPS changes its environment using its physical capabilities. These changes create events via actuators around that environment and events which were created by the environment are perceived via sensors and actions.

Layer 2: Embedded Device Layer The embedded device layer contains a microcontroller development board where the sensors and actuators are connected and controlled. Sensors are used to gather data from the environment, while actuators are used to create motion and/or movement.



Karaduman, B., Kardas, G., & Challenger, M. (2023). Development of Autonomous Cyber-Physical Systems Using Intelligent Agents and LEGO Technology. In *Cyber-Physical Systems for Industrial Transformation* (pp. 193-211). CRC Press.

Universiteit Antwerpen

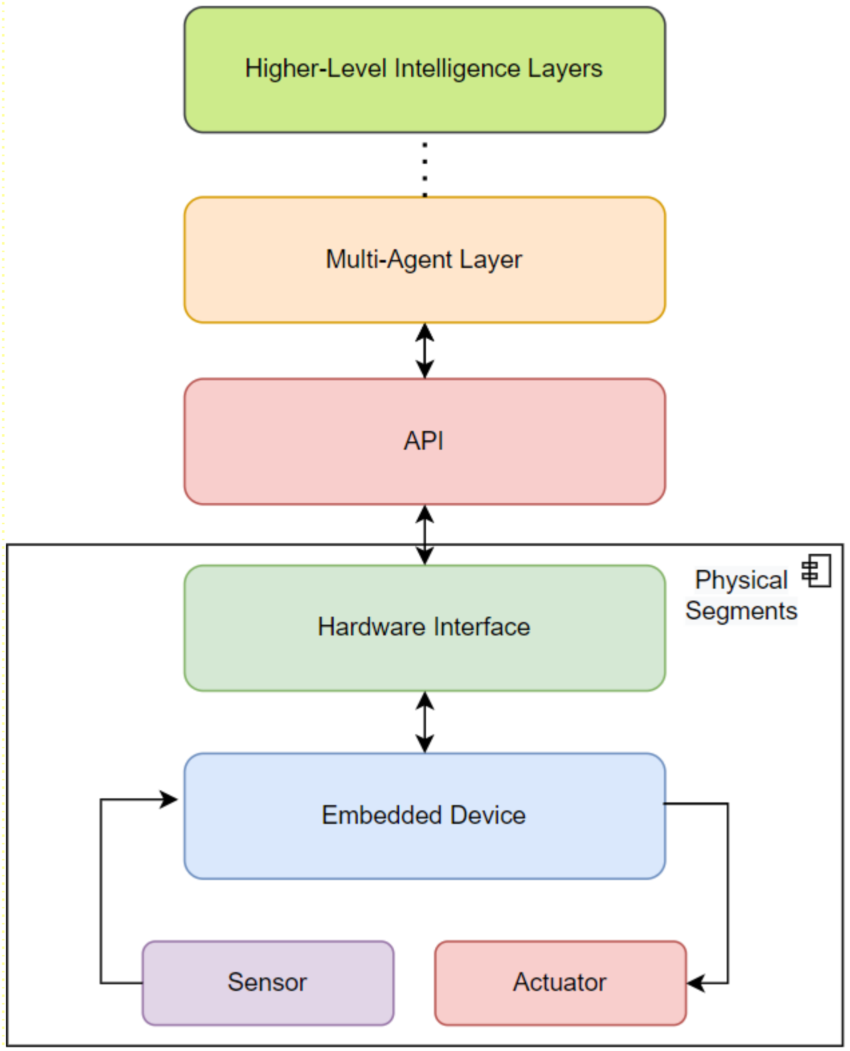
4. System Software Analysis and Design

Layer 3: Hardware Interface This layer is preferred to increase the capabilities of the embedded device. It can be an extension board, HAT or expansion interface to adapt the embedded device to the system's requirements. It is an optional layer but has importance in augmenting the hardware's capabilities.

Cyber Layers

Layer 4: API layer describes any device-specific software. This API should provide device functions to set, configure and control the sensors and actuators. It should also abstract away the hardware level programming details such as device registers, bit shifting, bit-wise operations and instruction-based operations.

Layer 5: Multi-Agent Layer This layer is a middle-ware where the agent framework runs. The MAS layer controls the device-specific functions. In this way, agents' perceptions are bound to the sensors and agents' actions are bound to the actuators of the embedded device. Agents can be reactive, cognitive or hybrid.



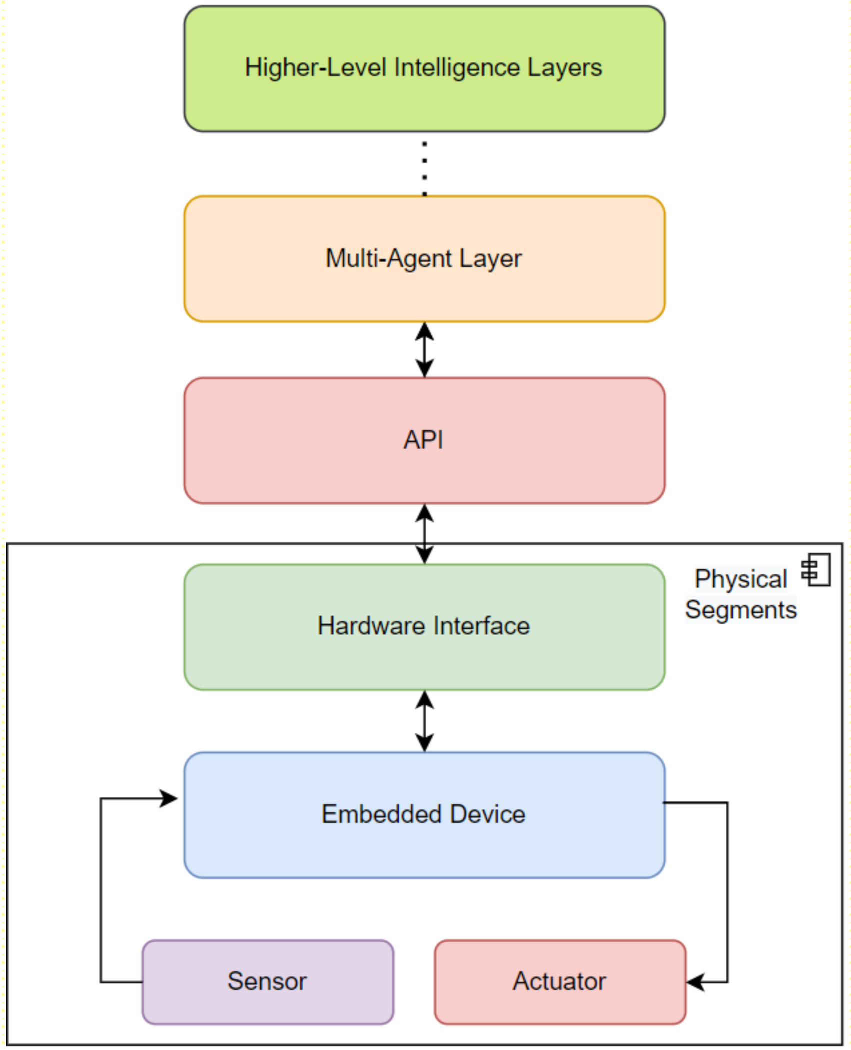
Karaduman, B., Kardas, G., & Challenger, M. (2023). Development of Autonomous Cyber-Physical Systems Using Intelligent Agents and LEGO Technology. In *Cyber-Physical Systems for Industrial Transformation* (pp. 193-211). CRC Press.

Universiteit Antwerpen

4. System Software Analysis and Design

Cyber Layers

Layer 6: Higher-Level Intelligence Layers This layer represents where the higher-level intelligence is contained. This layer applies logic-based approaches, machine-learning techniques, probabilistic approaches, and run-time simulations. Generally, these higher level methodologies are computationally heavy and provide long-term results.



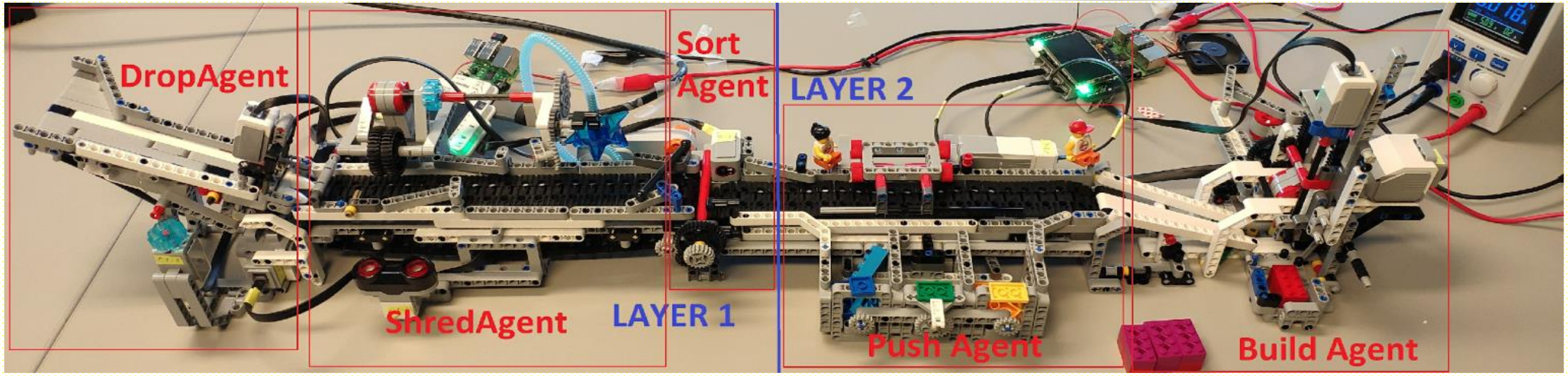
Karaduman, B., Kardas, G., & Challenger, M. (2023). Development of Autonomous Cyber-Physical Systems Using Intelligent Agents and LEGO Technology. In *Cyber-Physical Systems for Industrial Transformation* (pp. 193-211). CRC Press.

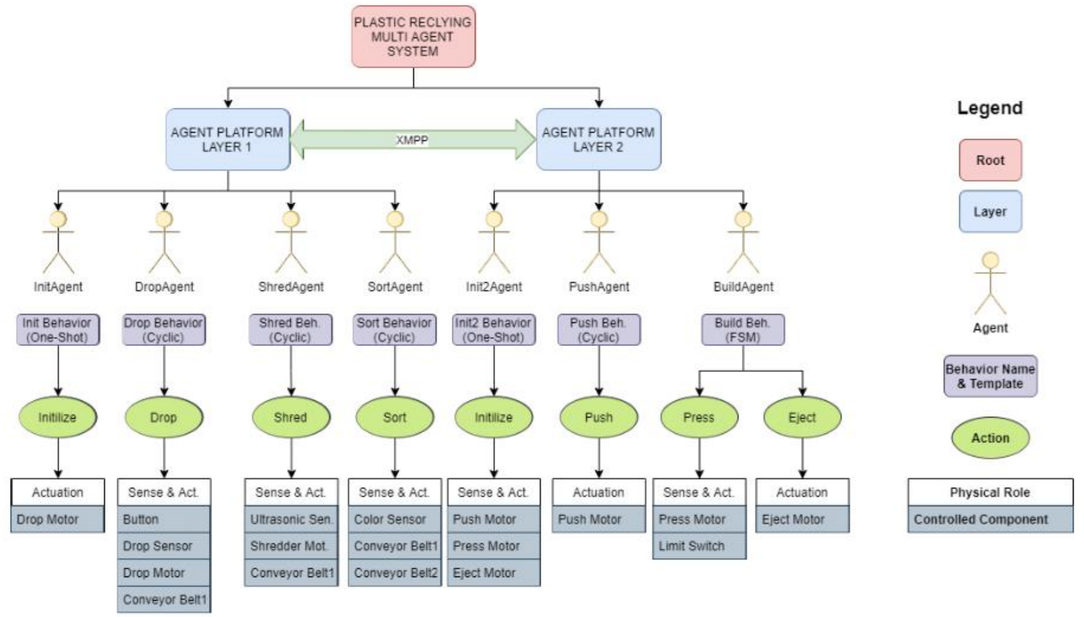
Demonstration

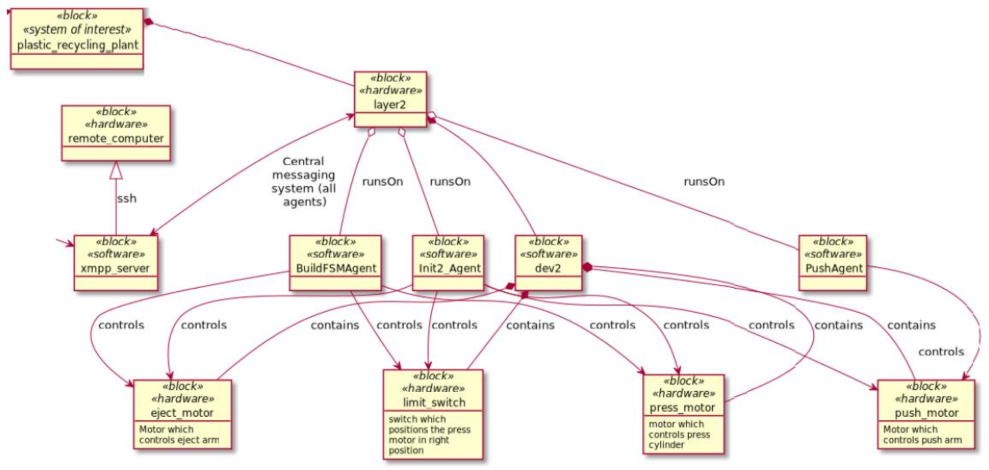
https://www.youtube.com/watch?v=dRUyXYuDPlY&list=PLMqG2GkQQo1C2BBRdMPKqojvOhGo2f2iP&ab_channel=MICSSLab

Universiteit Antwerpen

4. System Software Analysis and Design

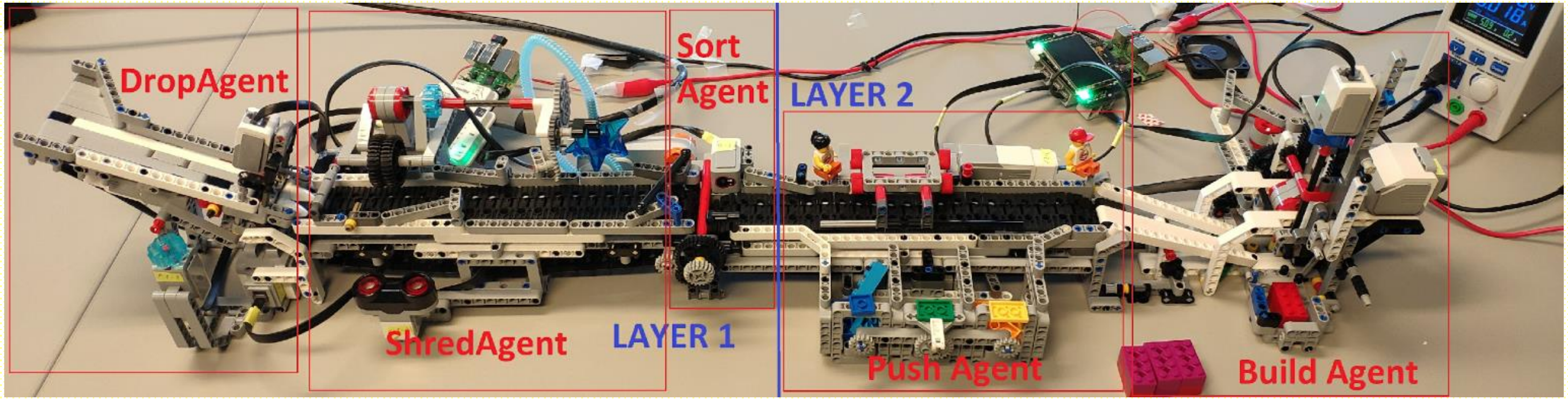


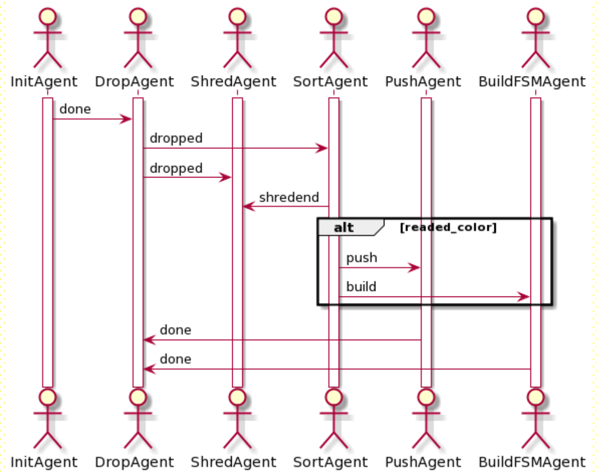


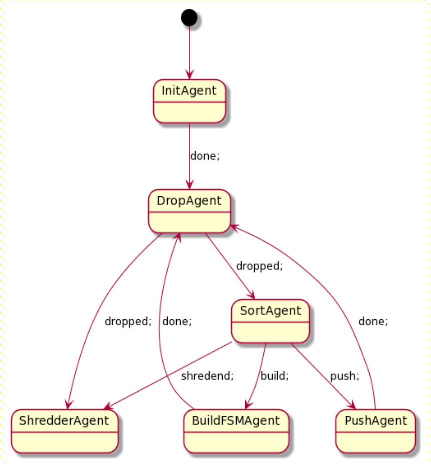

«block» «system of interest» plastic_recycling_plant
«block» «hardware» layer2
«block» «hardware» remote_computer
ssh
Central messaging system (all agents)
runsOn
runsOn
runsOn
«block» «software» xmpp_server
«block» «software» BuildFSMAgent
«block» «software» Init2_Agent
«block» «software» dev2
«block» «software» PushAgent
controls
controls
contains
controls
controls
contains
controls
controls
controls
contains
contains
controls
«block» «hardware» eject_motor
Motor which controls eject arm
«block» «hardware» limit_switch
switch which positions the press motor in right position
«block» «hardware» press_motor
motor which controls press cylinder
«block» «hardware» push_motor
Motor which controls push arm

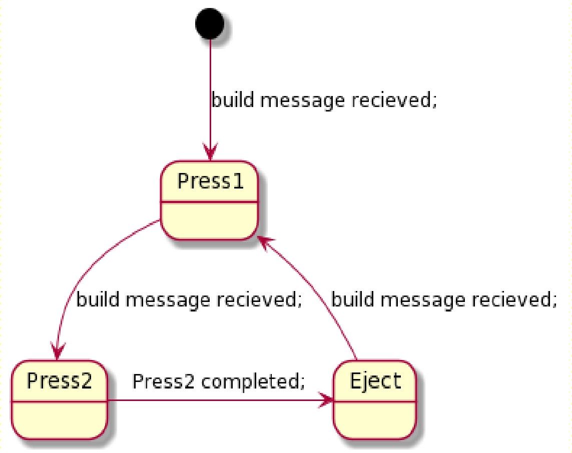
4. System Software Analysis and Design

- Production system is designed using SysML.









4. System Software Analysis and Design

B. Karaduman, G. Kardas, and M. Challenger

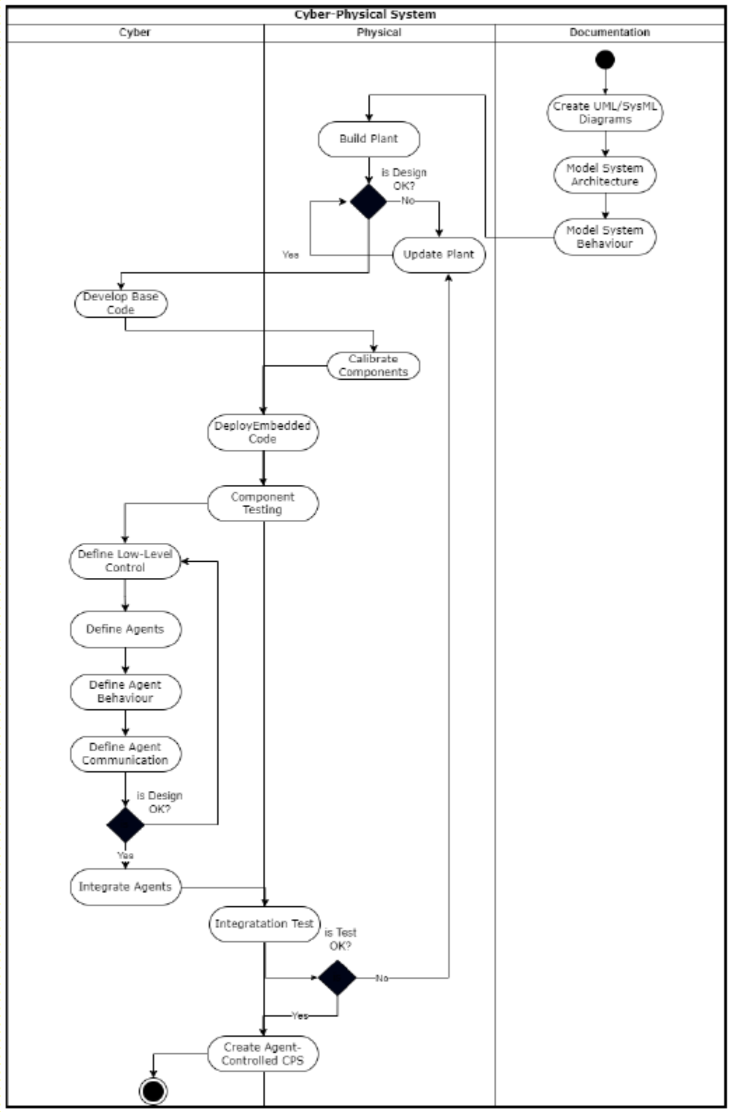
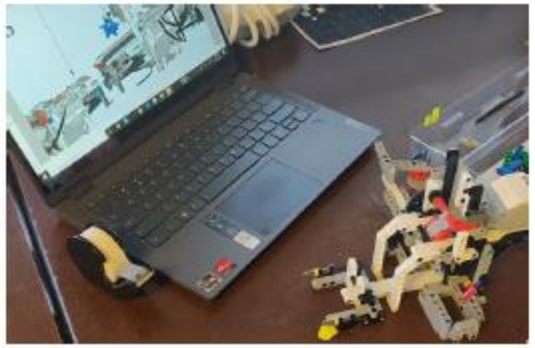
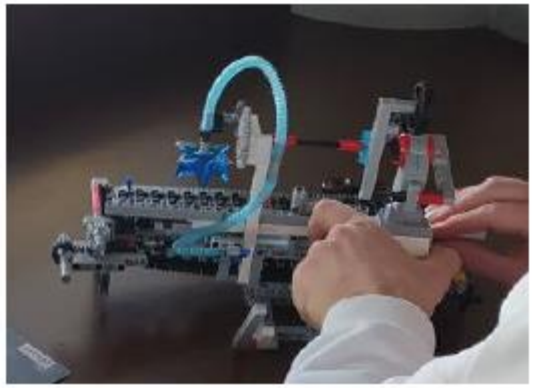


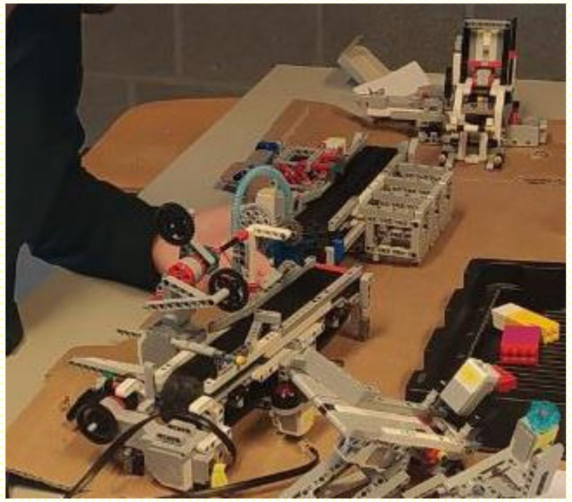
Fig. 7: Workflow for implementing the agent-based CPSs.



A figure that describes the construction of the Build Section.



A figure that describes the construction of the Shred Section.



An Agent-based Cyber-Physical Production System using Lego Technology

Universiteit Antwerpen

4. System Software Analysis and Design

B. Karaduman, G. Kardas, and M. Challenger

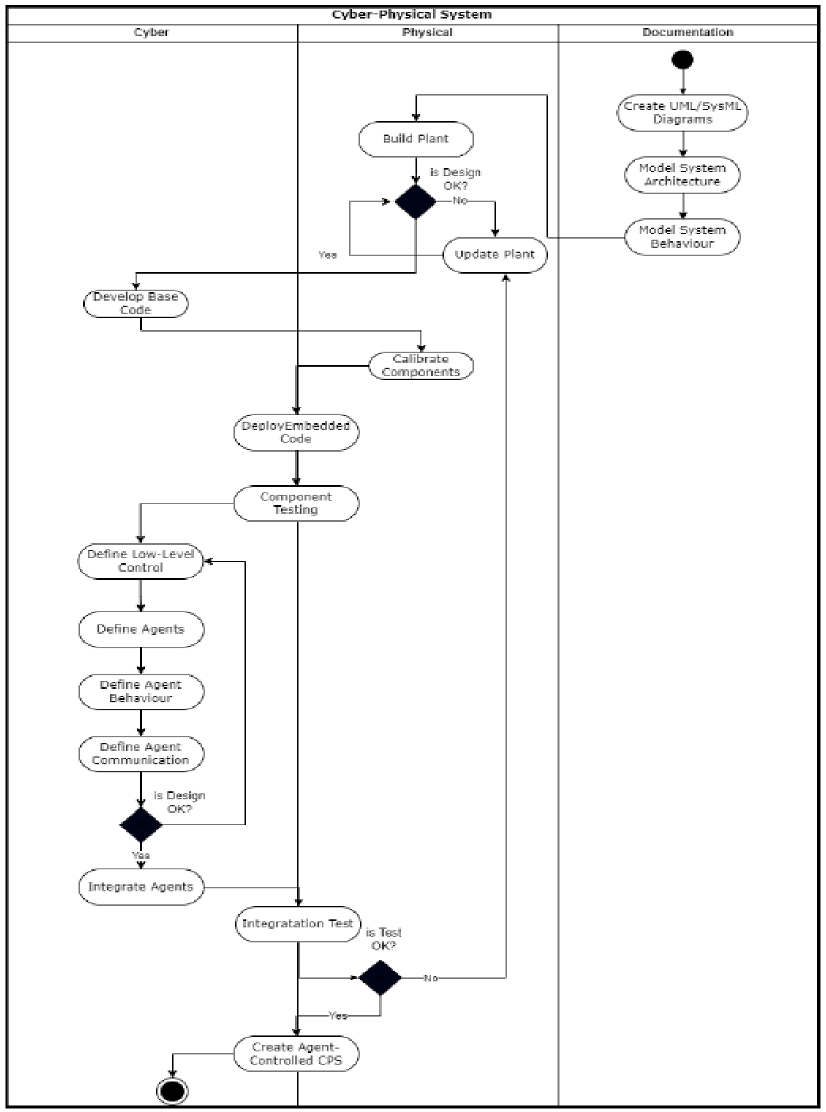
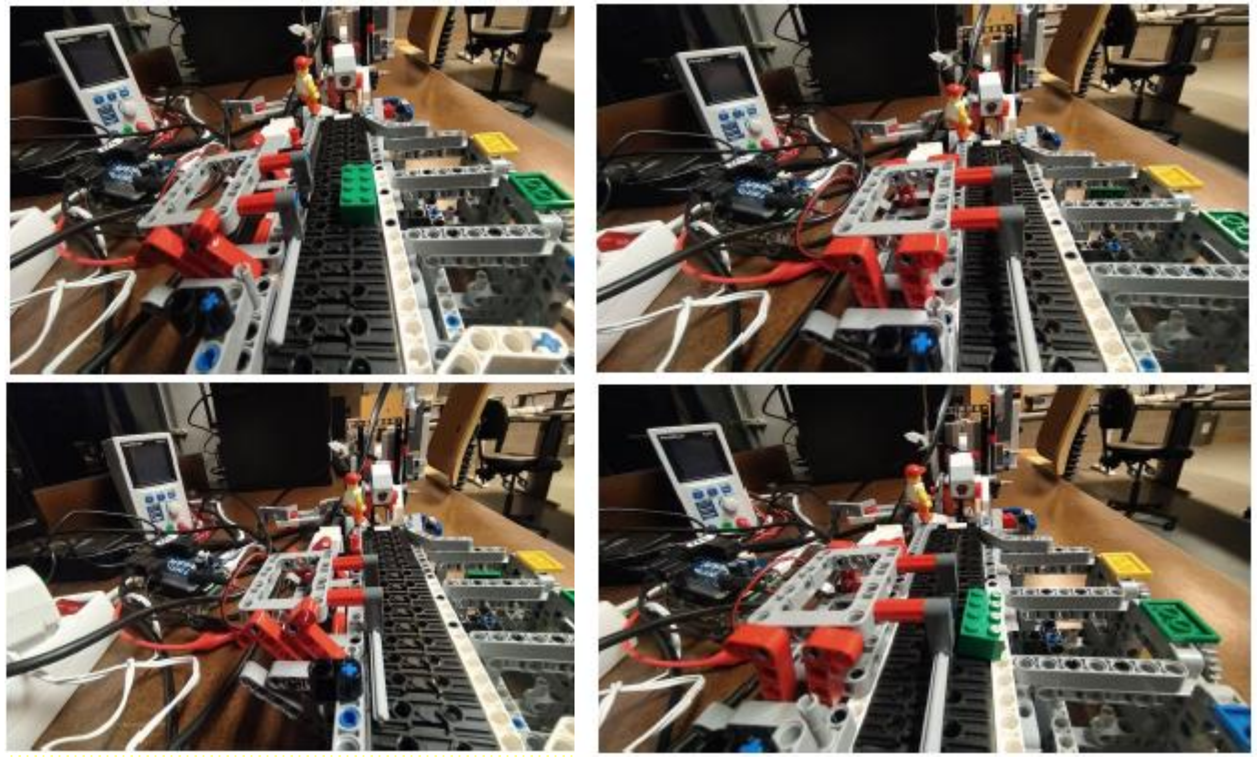
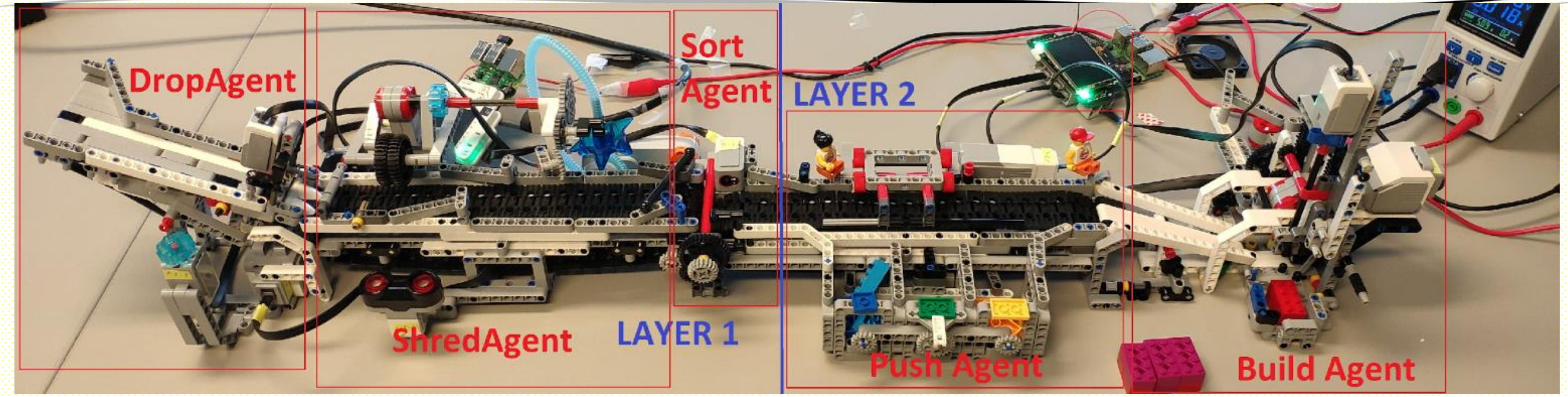


Fig. 7: Workflow for implementing the agent-based CPSs.



5. System and Agent Software



PiStorms v2

Listing 1. ConveyorMotor inner class

```
class ConveyorMotor:
    def start ( self ):
        dev1.psm.BBM2.setSpeed(−100)
        print (f'Conveyor Started')
    def startSlow ( self ):
        dev1.psm.BBM2.setSpeed(−20)
        print (f'Conveyor Slow Started')
    def stop( self ):
        dev1.psm.BBM2.setSpeed(0)
        print (f'Conveyor Stopped')
    def brickStucked( self ):
        dev1.psm.BBM2.runDegs(200, 100, True, False)
        print (f'Brick stucked')
    def runDegs( self ,degree ,speed):
        dev1.psm.BBM2.runDegs(degree, speed, True, False)
        motorState = dev1.psm.BBM2.isBusy()
        print (f'Motor rotated {degree} degree on {speed}
            speed')
        return motorState
```

Listing 2. Threading mechanism to avoid blocking I/O operations

```
threading .Thread( target =dev1.ColorSensor.waitBrick , args
    =(dev1.ColorSensor,) )
async def setup( self ):
        print ("SortAgent :: started ")
        b = self .SortBeh()
        template = Template()
        template . set_metadata ("performative", "inform")
        self .add_behaviour(b, template )
        print ("SortAgent :: running")
        t . start ()
```

Listing 3. LimitSwitch innerclass

```
class LimitSwitch:
    def isPressed ( self ):
        touch = dev2.psm.BBS1.isTouchedEV3()
        if touch != True:
            return False
        else :
            print (f'LS Pressed')
            return True
```

Listing 4. Color Sensor Sampling

```
def waitBrick( self ):
    readedcolorlist = [0]∗15
    index = 0
    readSensor = True
    count = 0
    print (" Starting to wait brick")
    dev1. retVal = 0.0
    while readSensor:
        color = dev1.psm.BBS2.colorSensorEV3()
        readedcolorlist [index] = color
        now = datetime.now()
        index = index + 1
        x=sum( readedcolorlist )/15
        if index ==15:
            index=0
            if x==2 or x==3 or x==4 or x==5 or x==6:
                print ( str (index) + " −> " + "
                    ReadedColor:" + str ( color ) )
                print ("RETVAL:",str(x))
                dev1. retVal = x
            else :
                dev1. retVal = 0.0
        time. sleep (0)
```

Button and Motor Control

```
public static boolean check_button() {

   boolean touch= touchSensor.isPressed();

   return touch;
}

public static void main(String[] args) {
   while(true) {
      boolean value = check_button();
      if (value==true){

         motor2.setSpeed(500);
         Delay.msDelay(1);
         motor2.backward();
         Delay.msDelay(1);
         // motor2.stop();
         // motor2.getPosition();

      }
      else{
         motor2.stop();
         Delay.msDelay(1);
      }
```

Java- Ultrasonic sensor-based Motor Control

```
public static int check_Emergency() {
   Delay.msDelay(1);
   SampleProvider sp = ultrasonicSensor.getDistanceMode();
   int distanceValue = 0;
   float[] sample = new float[sp.sampleSize()];
   sp.fetchSample(sample, 0);
   distanceValue = (int) sample[0];
   System.out.println("Distance: " + distanceValue);
   return distanceValue;
}
public static void main(String[] args) {

while(true) {

      int value = check_Emergency();
     if (value>1000){
        value=1000;
     }
     motor2.setSpeed(value*20);
     Delay.msDelay(1);
     motor2.backward();
     Delay.msDelay(1);


  }
```

Universiteit Antwerpen

6. Discussion

- **Agent- based approaches:**
  - ➢ Adaptiveness
  - ➢ Awareness
  - ➢ Intelligence
  - ➢ Abstraction

- However, an integration of the industrial systems with the agents is still a significant issue.
- Even though MAS technology has already been integrated into several industrial applications acceptance and standardization of industrial agents is still under debate.

Universiteit Antwerpen

6. Discussion

- Top-level methodologies cannot be evaluated properly and show without low-level integration.
- To achieve CPS and agent integration, device specific libraries are mostly required.
- These libraries can be merged with agent development environments.
- The library can be wrapped to provide behavioral structures.
- The integration process can be facilitated by using software engineering design principles (e.g. design patterns)
- Physical construction of the target system is still required. Lego technology can be a way for miniaturization.
- We need physically easy-to-construct and easy-to-modify technologies integrated with easy-to-deploy and easy-to-run programming paradigms

7. Lessons Learnt

- CPS has a wide umbrella, so courses should consider, agent-based programming, embedded technology and the requirement of intelligence, next generation systems.
- We need physically easy-to-construct and easy-to-modify technologies integrated with easy-to-deploy and easy-to-run programming paradigms.
- LEGO provides modular and modifiable structures while agent-oriented approaches present higher-level abstraction of programming.
- Multi-disciplinary studies can be taught to the future's engineers and students using our proposed approach.

Technical Notes:

- Limit switches, the system was fed with two power supplies with 9.8 Volts and 3 amps. Alternatively, Li-Po for short-term tests and mobility.
- If power requirements cannot be fed, the motors fail, and the system shuts down.
- If the motors get heated, cold gels are required.
- To reduce the friction between Lego bricks and moving parts, machine oil is used.

Universiteit Antwerpen

8. Conclusion and Future Work

– A system to integrate software agents and CPS is proposed.
– The design and implementation of the system line are discussed including agent software and low-level integration.
– Software agents helped to develop heterogeneous components in the system.
– Lego technology may also assist the education activities especially considering how automation on CPPS can be supported via software agents.

In the future:

- We aim to improve the current reasoning and planning capabilities of the system using belief-desire-intention (BDI) agents.
- We intend to create a CPSoS by benefiting from the IoT paradigm where the system also works with the same system instances and incorporate with different type systems.

Universiteit Antwerpen

Thank you for your attention