Embedded Agents

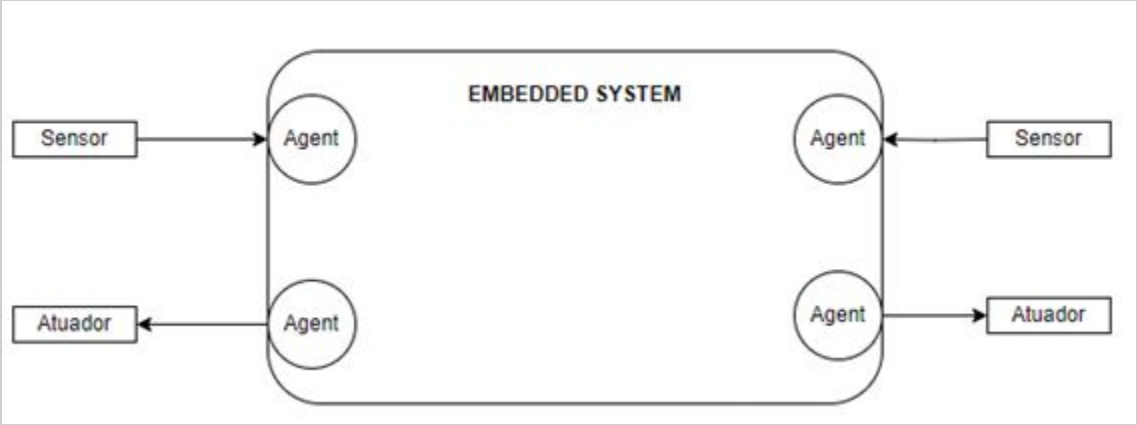
Felipe da Cunha Calegari
PPGEAS| DAS | CTC

- What is an Embedded Agent?
- Difficulties
- Related Work
- Frameworks
- Example - ARGO architecture
- Conclusions

What is an Embedded Agent?

- **Embedded Systems**
  - Computer systems that run on custom hardware.
- **Embedded Agents**
  - Computer systems that sense and act on their environment.
  - Monitor dynamic conditions.
  - Goal-directed ways.

1 - Embedded Agent model



Difficulties

- Embedded system - resource-constrained characteristic.
- Metrics: energy consumed, usage of execution time, code size, weight and cost.
- JADE and Jason:
  - Java based - requires large memory and processing capabilities.
  - Possible solution: heterogeneous systems where agent reasoning and acting are split in two systems - increases the cost of the system.

Related Work

- Raspberry Pi and Arduino boards for agent reasoning and acting/perceiving the environment, respectively - Barros et al. (2014)/ Lazarin and Pantoja (2015).
- AgentSpeak translator to embed agents in UAVs - allow programming agents for UAVs with specific characteristics - Bucheli et al. (2015).
- **Directions for implementing BDI agents in embedded systems with limited hardware resources. - Santos et al. (2021).**

Table 1 - Main requirements to implement an Agent framework

Requirement	Proposal
Development Methodology	Incremental
Platform Support	Multi-Platform
Programming Language for Framework Development	C and/or C++
Programming Language for Agent Implementation	AgentSpeak
Main AgentSpeak Features Supported in Initial Versions	Propositions Simple Unification Algorithms
Memory Allocation	Static
Size of Internal Data Structures	Configurable in the Framework
Internal Methods	No Internal Methods
Belief Update and Action Functions	Provided by the Agent Programmer
Hardware Interruptions	Not Supported

Table 1. Summary of Framework Requirements

Source: Santos et al. (2021)

Frameworks

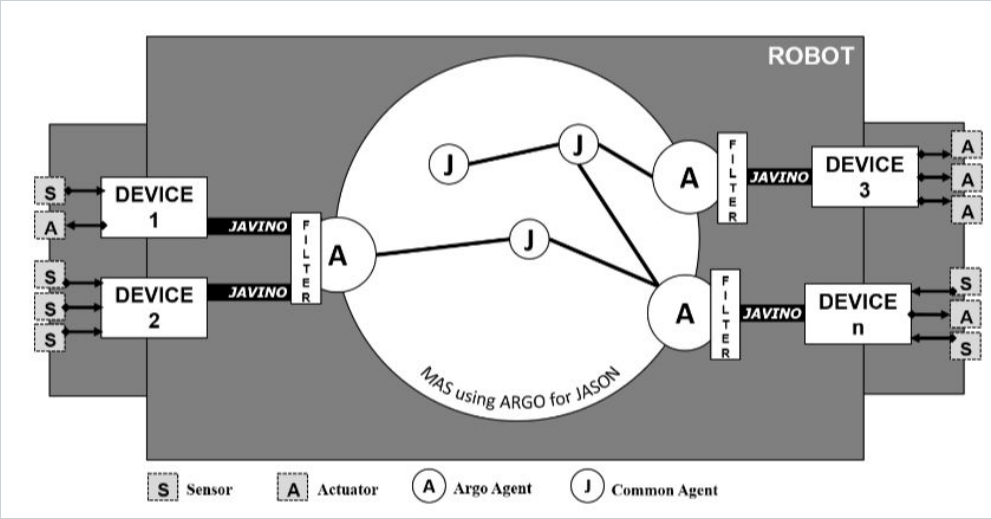
- **ARGO**
  - Middleware that enables Jason's agents to communicate with low-level hardware by using Javino.
- **GOAL**
  - Middleware that connects the high-level layer (agent) with a low-level (hardware control).
- **Jason** (Santos):
  - Use Jason on a BeagleBone board in UAV context.
  - Advantages of using AOP compared to imperative paradigm (C language).

ARGO - Overview

- Reasoning layer - adopt an AOPL responsible for the cognitive reasoning of the robot.
- BDI - high cost of processing the perceptions.
- Plans may be added in running time and a quite large intention stack is generated - issue with time limit.
- Perception filters would fix it.
- Customized Jason architecture
    - Employ perceptions filters.
    - Javino - Jason agent’s reasoning cycle.

ARGO - Overview

2 - ARGO Illustration



ARGO - agents

- Two types of agents: Common Jason agents and ARGO agents.
- **ARGO agents**:
  - Control actuators at runtime and receive perceptions from the sensors.
  - Only ARGO agents can control devices and receive data from real world.
  - Can delegate for Jason agents the reasoning about perceptions or process all by themselves.

ARGO - agents

- **ARGO agents**:
  - Needs to be defined in the MAS design by defining the ARGO architecture.
  - Connect to one or more devices by choosing a serial port.
  - Only use one port at a time - sensing and acting.

ARGO - Internal Actions

- **Limit(x)** - sensing interval.
- **Port(y)** - serial port to be used by the agent.
- **Percepts(open–block)** - decides if it will perceive the real world.
- **Act(w)** - action to be executed by the microcontroller.
- **Change_filter(filterName)** - which filter to constrain perceptions in runtime.

ARGO - Customizing Jason architecture

- Reasoning cycle of Jason - extended.
- Javino - responsible for getting percepts coming from low-level layers and send them to the perceive step.
- ARGO architecture - customize Jason (extending ***AgArch*** and *TransitionSystem* classes).
- Modifications do not change Jason's original functionality.

ARGO - Customizing Jason architecture

- **AgArch class**
  - Responsible for the Jason's native architecture and provides a list of perceptions sent by the Jason's environment in Java and the communication with other agents.
  - Custom architecture: Javino is inserted as a communication bridge to the sensors and actuators.
  - Serial port identification had to be added to the AgArch class - Javino communication.

ARGO - Customizing Jason architecture

- **TransitionSystem class**
  - New attribute *Blocked* - perceptions.
  - New attribute *Limit* - time interval for perceiving the real world (sensors).
  - New function *realWorldPerceptions* - verifies if the percepts are blocked or if the time limit for the next perception has been reached.
    - If percepts are not blocked and the time limit was reached, Javino requests the percepts from sensors and sends them to the perceive method in Agent class.

ARGO - Case study

- **Robot** - four distance sensors, four light sensors, four temperature sensors, an Arduino board and an Arduino 4wd chassis.
- Robot: perceive the environment and move forward at a constant speed until the distance to the wall was less than a specific value.
- As soon as it perceived that the distance was smaller, the robot should stop.

ARGO - Case study

- **Filters:**
  - No filter - no perception filters.
  - Front side - removed all perceptions except the ones from the sensors on the front side.
  - Front distance - removed all perceptions except the ones from the distance sensors on the front side of the robot.

ARGO - Case study

Table 2 - Factors and filters of the experiment

Factor	**Levels**		
Distance	40 cm	80 cm	120 cm
Perception interval	20 ms	35 ms	50 ms
Filter	No filter	Front Side	Front Distance

3 - Agent code

```
1  value(40).
2  !config.
3
4  +!config: true <-
5          .port(COM5);
6          .limit(20);
7          .filter(byValue);
8          .percepts(open);
9          !start.
10
11 +!start : true <-
12         .act(front);
13         +status(front);
14         !moving.
15
16 +!moving: dist(f, X) &
17                 value(J) & X>J &
18                 status(front) <-
19         .act(front);
20         !moving.
21
22 +!moving: dist(f, X) &
23                 value(J) &
24                 X<=J & status(front) <-
25         .act(stop).
26
```

```
26
27 -!moving <-
28         !!moving.
29
30 +light(X,Y) : Y>100 <-
31         .act(ledLightOn).
32
33 +light(X,Y) : Y<=100 <-
34         .act(ledLightOn).
35
36 +temp(X,Y) : Y>25 <-
37         .act(ledTempOff).
38
39 +temp(X,Y) : Y<=25 <-
40         .act(ledTempOn).
```

ARGO - Case study

- Results:
  - Agent without perception filter: collided with the wall.
  - With ARGO architecture with perception filter, it helped reducing processing time.

Conclusions

- Embedded agents
  - Main difficulties found on developing systems.
  - Related work.
  - Different frameworks to try to solve some of the difficulties.

References

- BARROS, R. S. et al. An Agent-oriented Ground Vehicle's Automation using Jason Framework. **Proceedings of the 14th International Conference on Agents and Artificial Intelligence**, 1 jan. 2014.
- BUCHELI, S. et al. From AgentSpeak to C for Safety Considerations in Unmanned Aerial Vehicles. **Towards Autonomous Robotic Systems**, p. 69–81, 2015.
- KAELBLING, L. P.; ROSENSCHEIN, S. J. Action and planning in embedded agents. **Robotics and Autonomous Systems**, v. 6, n. 1-2, p. 35–48, jun. 1990.
- Menegol, Marcelo S, et al. "Evaluation of Multi-Agent Coordination on Embedded Systems." *Lecture Notes in Computer Science*, 1 Jan. 2018, pp. 212–223, https://doi.org/10.1007/978-3-319-94580-4_17. Acesso em: 4 Dez. 2024.
- PANTOJA, C. E.; LAZARIN, N. M. A Robotic-agent Platform for Embedding Software Agents Using Raspberry Pi and Arduino Boards. **9th Software Agents, Environments and Applications School (WESAAC)**, 1 jun. 2015.
- PANTOJA, C. E. et al. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. Lecture notes in computer science, p. 136–155, 1 jan. 2016.
- SANTOS, Matuzalem M.; HÜBNER, Jomi F.; BRITO, Maiquel. Directions for implementing bdi agents in embedded systems with limited hardware resources. In Anais do XV Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações, pages 148–156, 2021. doi: http://dx.doi.org/10.5281/zenodo.5774181.

Contato

E-mail: felipecunhacalegari@gmail.com

UNIVERSIDADE FEDERAL
DE SANTA CATARINA