

# HOW TO BUILD A SOFTWARE AGENT?

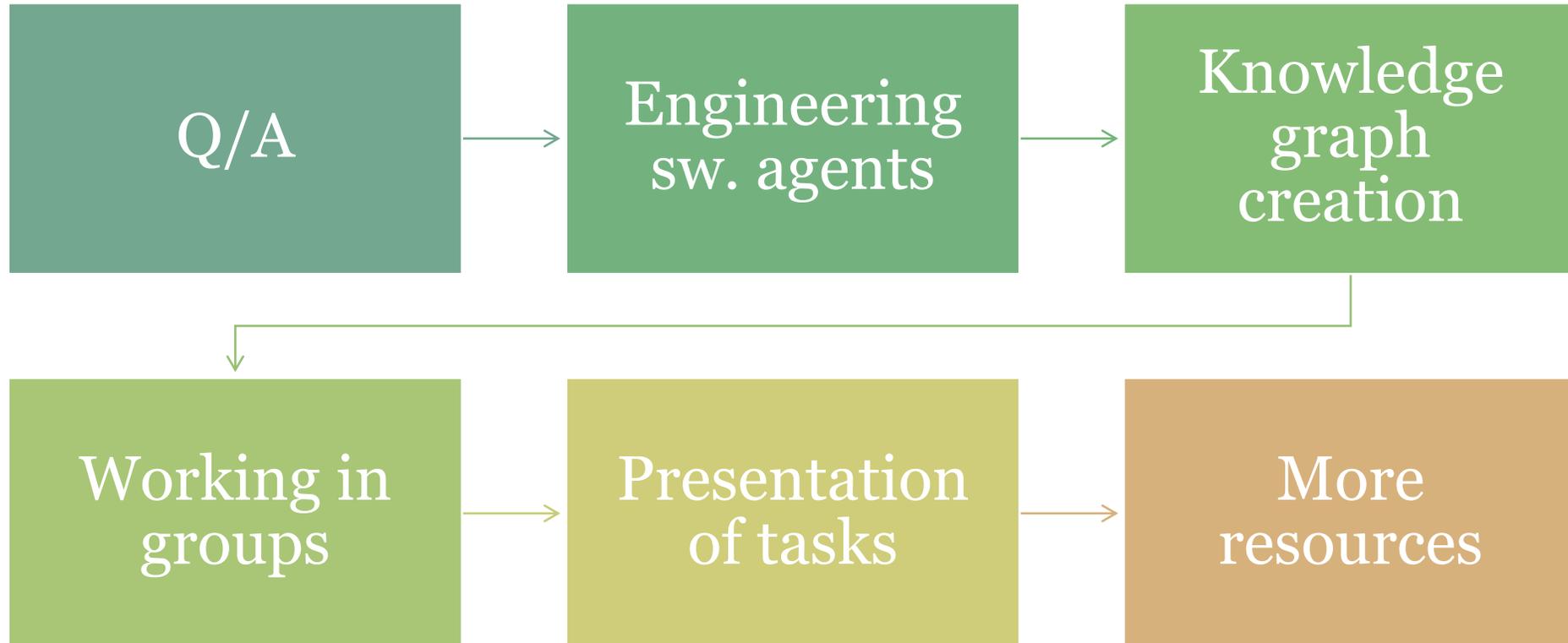
Lab. session 2 – November 5 2020.

Ph.D. Esteban Guerrero  
[esteban@cs.umu.se](mailto:esteban@cs.umu.se)



UMEÅ UNIVERSITY

# CONTENT



UMEÅ UNIVERSITY

INTERACTIVITY IN SMART ENVIRONMENTS 2020

**QUESTIONS?**



UMEÅ UNIVERSITY

# BEFORE START...

- Any question about the lecture?
- Any suggestion or petition?

Go to [www.menti.com](http://www.menti.com) and use the code 63 46 25 7



UMEÅ UNIVERSITY

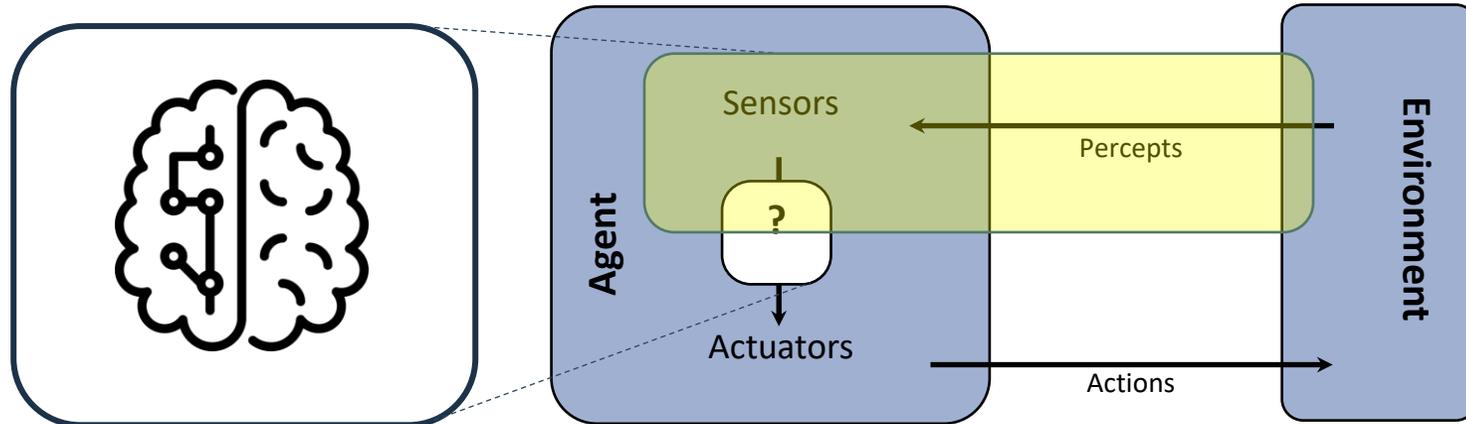
INTERACTIVITY IN SMART ENVIRONMENTS 2020

# ENGINEERING AGENTS



UMEÅ UNIVERSITY

# TOOLS FOR BUILDING AGENT'S MODULES



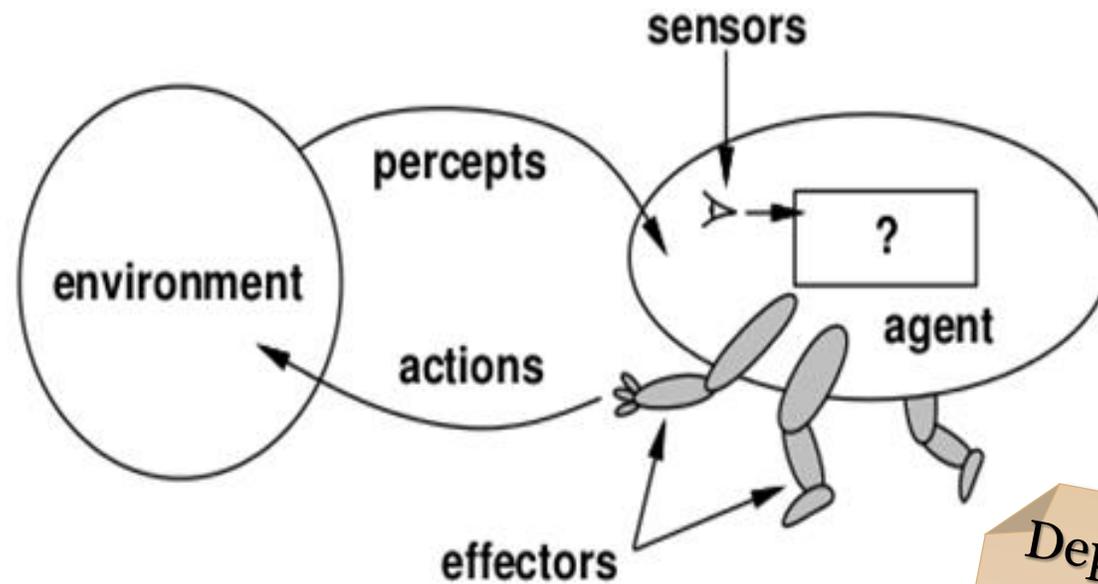
This lecture focus



UMEÅ UNIVERSITY

INTERACTIVITY IN SMART ENVIRONMENTS 2020

# TOOLS FOR BUILDING AGENT'S MODULES



Depends on the problem.  
Tools selection depends on ways to solve that problem



# TOOLS

## BDI Frameworks

- JADE (Java Agent DEvelopment)
- JaCaMo (Jason, Cartago, Moise)
- Jason
- Jack
- JS-son
- ROS (Robot Operating System)

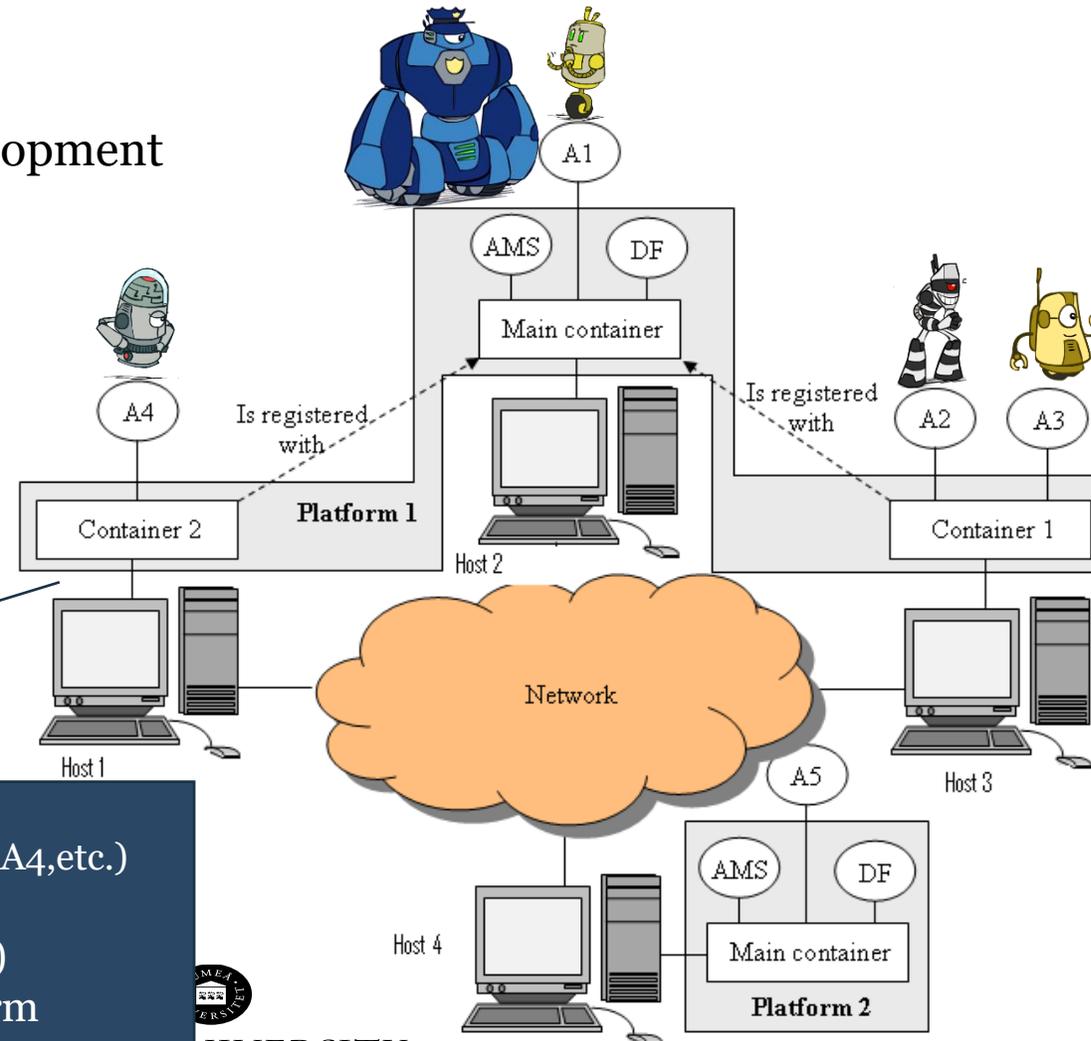
## Specialized tools/sw

- Commercial AI-based tools: Amazon aws, Google AI, Facebook AI, IBM Watson, Microsoft Azure AI, etc.
- Open source
  - **Python**: nlTK, OpenCV, pandas, OpenAI Gym, etc.
  - **Java** libraries: Caffe, Deeplearning4J, etc.
  - **R** packages: scikit-learn, DataExplorer, MLR, parsnip, Ranger, purrr, etc.



# TOOLS (BDI FRAMEWORKS)

- JADE (Java Agent DEvelopment Framework)
- Jason
- JaCaMo
- JS-son
- ...



*Platform*: message delivery  
*Containers*: agents home (e.g. A1,A4,etc.)  
*Communication protocol*: Agent Communication Language (ACL)  
*AMS*: the authority of the platform  
*DF*: yellow pages (public board of services)

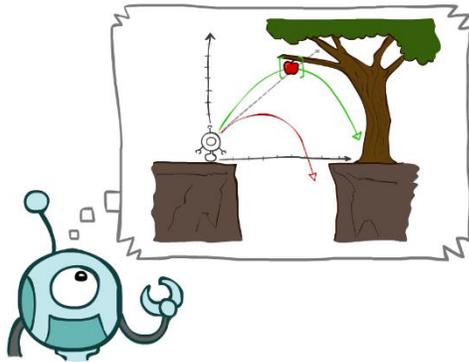


UNIVERSITY

# TOOLS (BDI FRAMEWORKS)

JaCaMo=**J**ason for programming autonomous agents +  
**C**artago for programming environment artifacts +  
**M**oise for programming multi-agent  
organisations

<http://jacamo.sourceforge.net/>



**J**ason



**C**artago



**M**oise

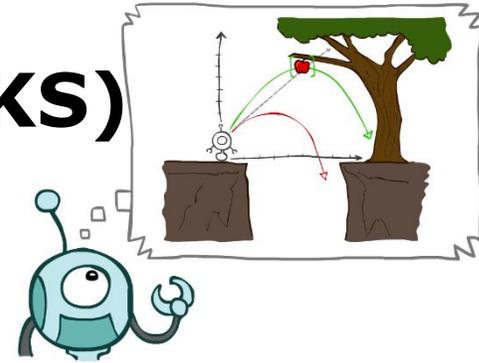
**CARTAgO:** Common ARTifact infrastructure for AGents Open environments

**Moise:** Model of Organization for multi-agent SystEms

**Jason:** Java-based Agentspeak interpreter used with Saca for multi-agent distribution Over the Net

# TOOLS (BDI FRAMEWORKS)

## Jason



### by perception

beliefs annotated with `source(percept)` are automatically updated accordingly to the perception of the agent

### by intention

the operators `+` and `-` can be used to add and remove beliefs annotated with `source(self)`

```
+lier(alice); // adds lier(alice)[source(self)]
-lier(john); // removes lier(john)[source(self)]
-+lier(john); // updates lier(john)[source(self)]
```

### Types of goals

- Achievement goal: goal *to do*
- Test goal: goal *to know*

### Syntax

Goals have the same syntax as beliefs, but are prefixed by `!` (achievement goal) and `?` (test goal)

### Example (initial goal of agent Tom)

```
!write(book).
```

An AgentSpeak plan has the following general structure:

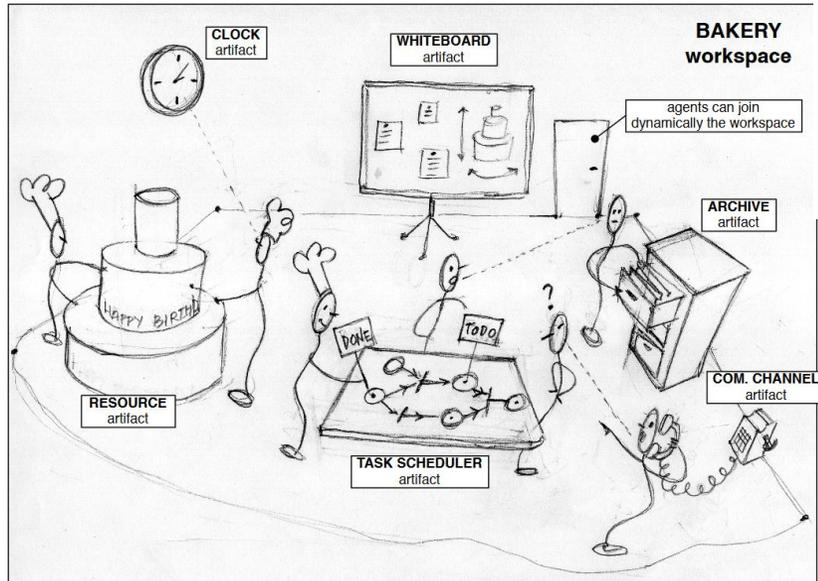
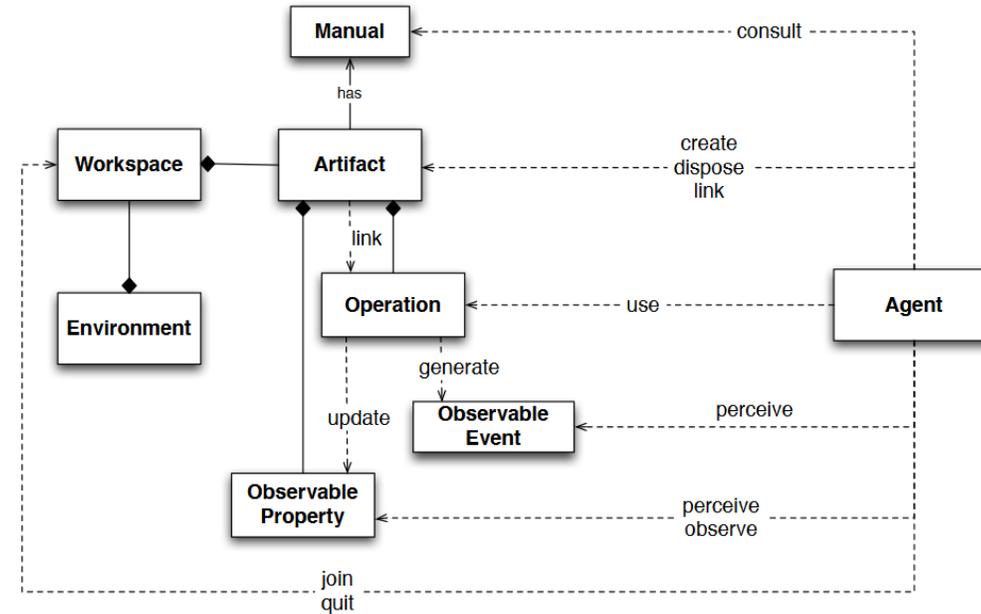
```
triggering_event : context <- body.
```



UMEÅ UNIVERSITY

# TOOLS (BDI FRAMEWORKS)

Cartago



UMEA UNIVERSITY

# JACAMO DEMO

```
eclipse-workspace - SmartEnv2019-2agents/main-container.jcm - Eclipse IDE
File Edit Navigate Search Project Run Window Help

bob.asl
1 /* Initial beliefs and rules */
2
3 /* Initial goals */
4 !start.
5 !sayagain.
6
7 /* Plans */
8 +!start : true <- .send(alice,tell,hello); .wait(3000).
9
10 +!sayagain : true <- .send(alice,tell,hello); .wait(3000).
11
12
13
14
15
16 { include("$jacamoJar/templates/common-cartago.asl") }
17 { include("$jacamoJar/templates/common-moise.asl") }
18
19
20

alice.asl
1 /* Initial beliefs and rules */
2
3 /* Initial goals */
4 !greetings.
5
6 /* Plans */
7 +hello <- .print(hello).
8 +helloagain <- .print(helloagain).
9
10 +!greetings : true <- .print("hello world.").
11
12
13
14
15
16 { include("$jacamoJar/templates/common-cartago.asl") }
17 { include("$jacamoJar/templates/common-moise.asl") }
18
19
20
21

container.jcm
1 /*
2
3 Run main-container.jcm first
4 */
5
6
7 mas simple_container { // not a main container
8
9     agent bob
10
11     asl-path: src/agt, src/agt/inc
12
13     platform: jade("-container -host localhost -container-nam
14
15 }

main-container.jcm
1 mas main_container_jade {
2
3     agent alice
4
5     asl-path: src/agt, src/agt/inc
6
7     platform: jade("-gui -sniffer")
8 }
9
10
```



Demo: JaCaMo multi-agents

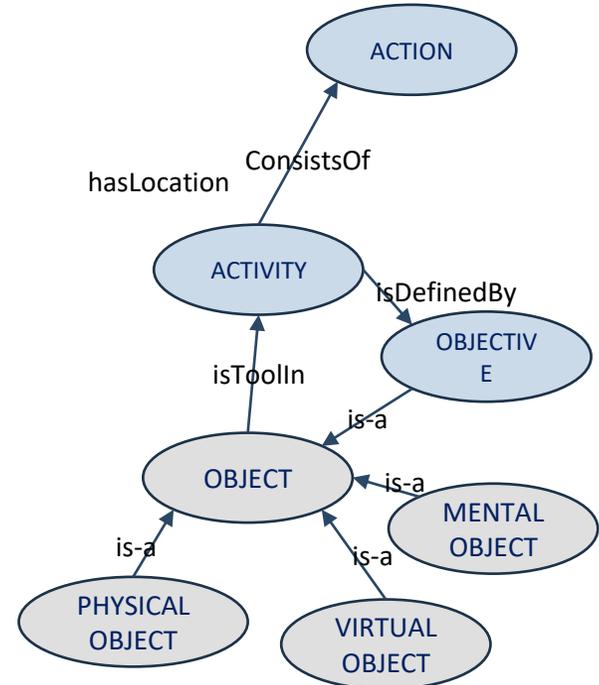
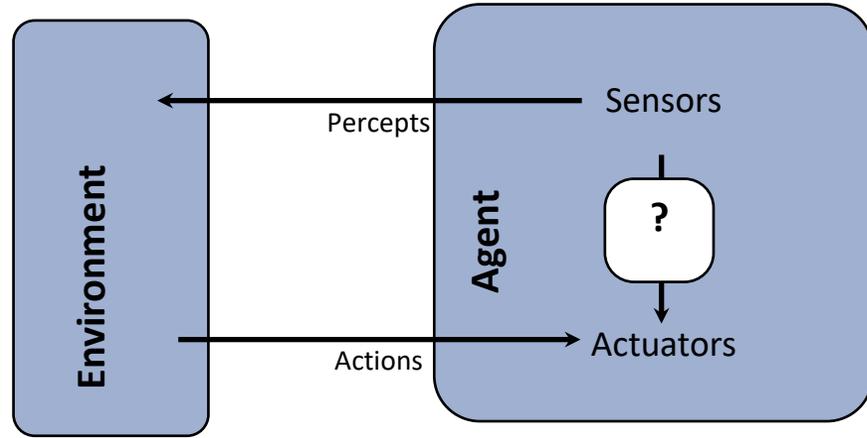
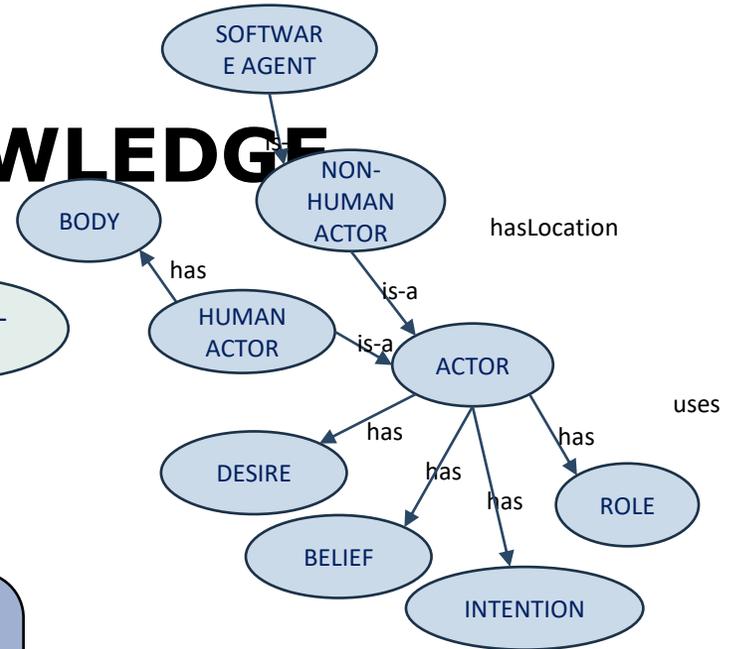
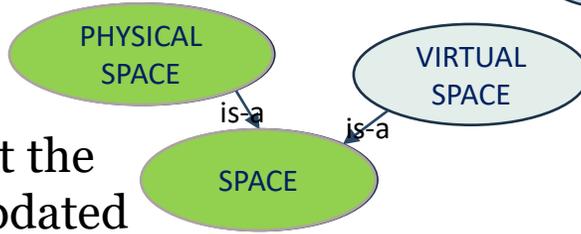
# KNOWLEDGE GRAPH CREATION



UMEÅ UNIVERSITY

# THE AGENT'S KNOWLEDGE

Information about the environment is updated





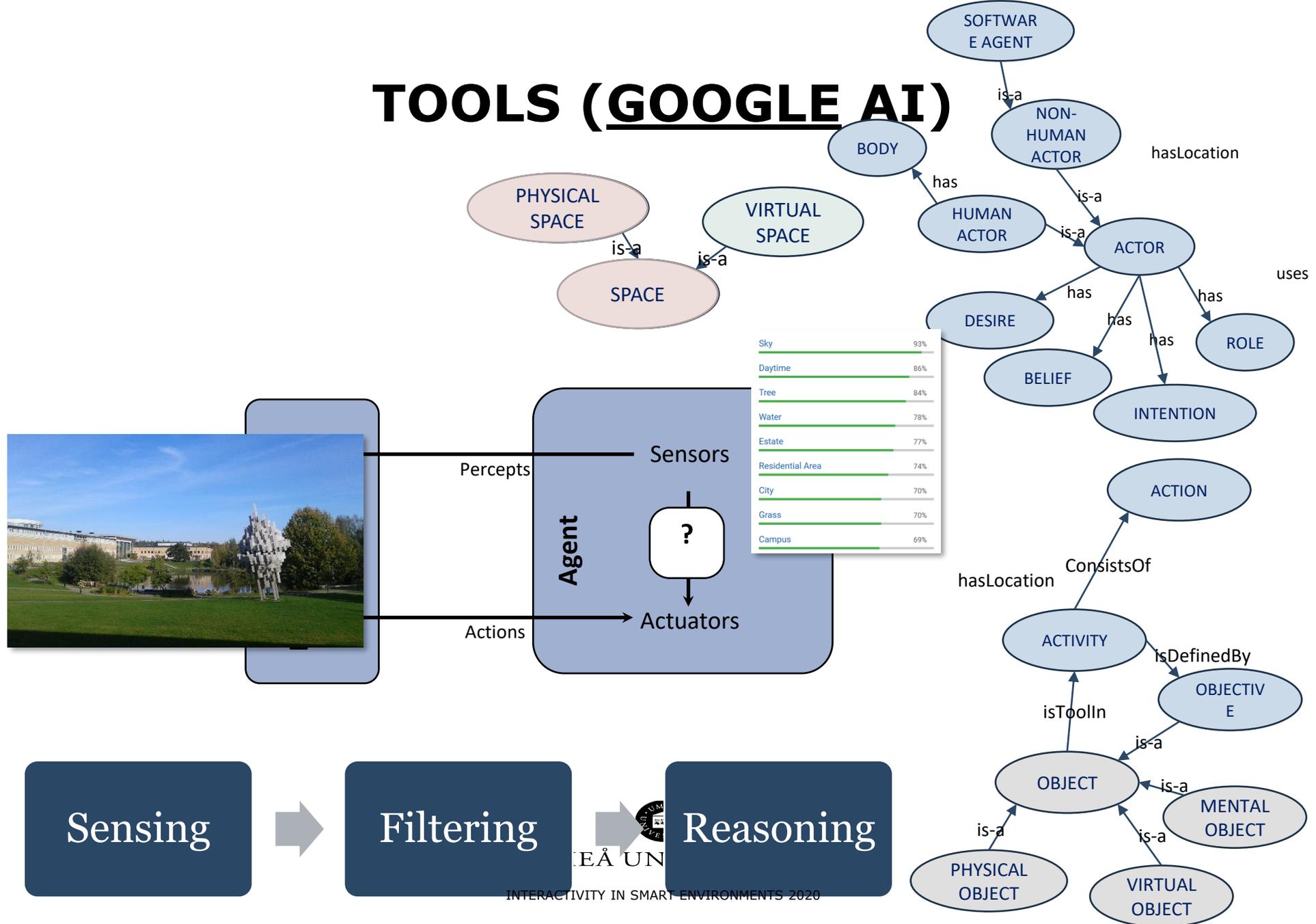
Demo: Google API vision

# TOOLS (GOOGLE AI)

The image displays two screenshots of the Google Cloud Vision API interface. The top screenshot shows the 'Sentiment/emotion analysis' and 'Object analysis' results for a group of people. The bottom screenshot shows the 'Environment analysis' results for a landscape image, with a list of labels and their confidence percentages.

Label	Confidence
Sky	93%
Daytime	86%
Tree	84%
Water	78%
Estate	77%
Residential Area	74%
City	70%
Grass	70%

# TOOLS (GOOGLE AI)



**WEB PROTÉGÉ**



UMEÅ UNIVERSITY

# PROTÉGÉ VERSIONS

people (http://owl.man.ac.uk/2006/07/sssw/people) : [E:\umu\teaching\material\ontologies\peopleOntology1.owl]

File Edit View Reasoner Tools Refactor Window Ontop Mastro Help

people (http://owl.man.ac.uk/2006/07/sssw/people)

Active Ontology x Entities x Individuals by class x OWLViz x DL Query x OntoGraf x

Annotation properties Datatypes Individuals owl:Thing — http://www.w3.org/2002/07/owl# Annotations Usage

Classes Object properties Data properties

Class hierarchy: owl:Thing

Annotations: owl:Thing

Annotations +

owl:Thing

- adult
- animal
- bone
- brain
- company
- dog
- female
- 'haulage worker'
- leaf
- male
- pet
- plant
- publication
- vehicle
- 'white thing'
- young

Description: owl:Thing

Equivalent To +

SubClass Of +

General class axioms +

- person and (has\_pet min 3 owl:Thing) SubClassOf 'anim
- is\_pet\_of some owl:Thing SubClassOf pet
- eats some bone SubClassOf eats some owl:Thing
- eats some (brain and (part\_of some sheep)) SubClassOf eats som

SubClass Of (Anonymous Ancestor)

Instances +

- 'Daily Mirror'
- Pete
- Spike
- Tom

people ontology - WebProtégé

https://webprotege.stanford.edu/#projects/a683153e-44f1-4b19-a07f-8254cad61e...

people ontology Home

Classes Properties Individuals Comments Changes by Entity History Query

Class Hierarchy

- owl:Thing
  - adult
  - animal
  - bone
  - brain
  - company
  - dog
  - female
  - haulage worker
  - leaf
  - male
  - pet
  - plant
  - publication
  - vehicle
  - white thing
  - young

Nothing selected

# **TASK 2**

## **BUILDING A KNOWLEDGE BASE COLLABORATIVELY**

**Adding extra information about sensing**  
**Make the first query**



UMEA UNIVERSITY

# COLLABORATIVE TASK

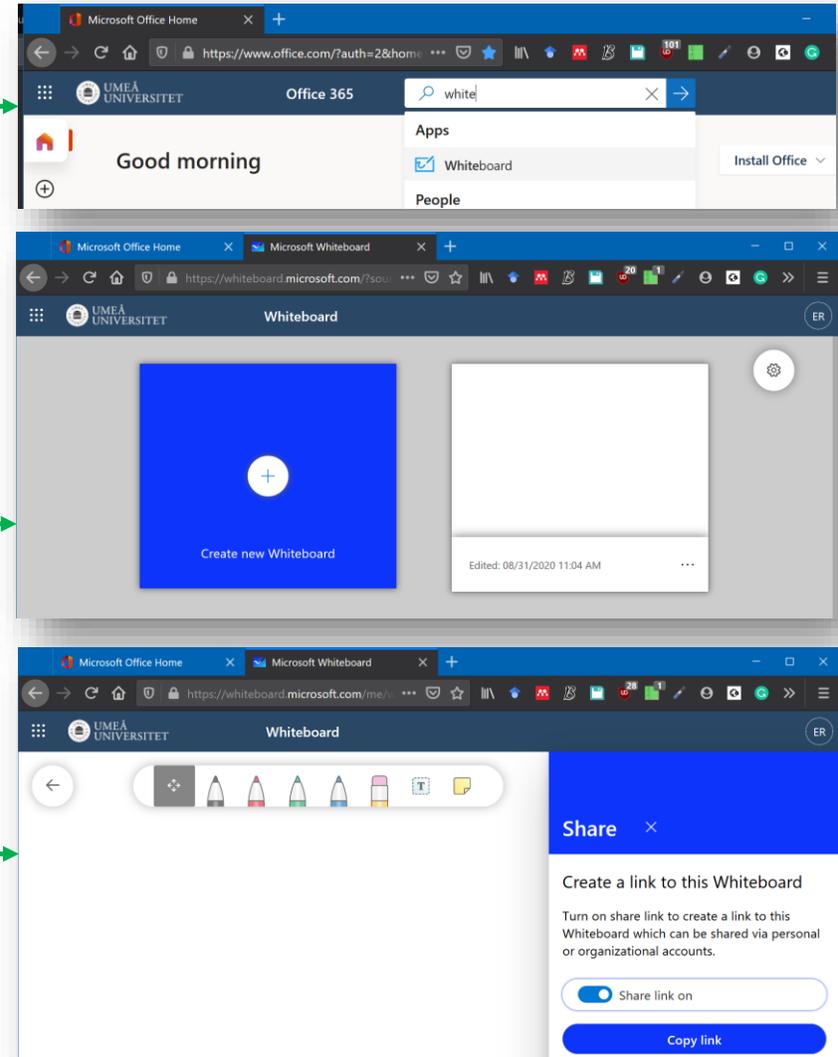
## Procedure:

- Split the group in Zoom rooms to form teams –**randomly**-
- Select a specific running example of a smart environment
  - 1.Examples: an older adult living at home with some specific needs (smart home environment), a tourist looking for specific city places with particular needs (smart city), a mixed reality pet that work as a companion of a person with specific needs (smart virtual environment), etc. –**please imagine other scenarios**
  1. **Re-use the previous example - maybe**
- Knowledge graph collaboratively building based on the example
  - 2.In the Zoom room one person opens the Zoom Whiteboard (Share Screen->Whiteboard-> Share)
  - 3.Create the taxonomy of entities (nodes) with relationships (**is-a**) using a color
  - 4.Create semantic relationships (e.g. **has\_pet, is\_pet\_of**) with other color
  - 5.Add some *individuals* in other words, examples or instances of those entities with other color
  - 6.Add names of the group member in the top-left of the drawing
  - 7.Add a title to the graph, example: older adult smart environment graph, tourist smart city graph, etc.
- Save the drawing graph locally
  - 8.In Zoom click in Save on top of the Whiteboard
- Answer questions (next slide), take notes about those answers. Then present those answers.
- **Time 10 minutes**

# COLLABORATIVE TASK

**Alternatively, for sharing if Annotations and Whiteboard in Zoom is limited:**

1. All the team members go to [o365.umu.se/](https://o365.umu.se/)
2. Login with UmU credentials
3. One person creates and opens a new Whiteboard
4. Share the link of the new whiteboard with the other members via Zoom
5. Make the knowledge graph collaboratively
6. Save the graph



# COLLABORATIVE TASK

## **Procedure:**

- Open Web protégé: <https://webprotege.stanford.edu/>
- Create an account
- Create the taxonomy (the families of knowledge entities)

**•Time 10 minutes**

# COLLABORATIVE TASK

## Questions

- What information of the graph is *necessary*?
- What parts of the graph *change on time* (dynamic smart environment variables)?
- What information may be *uncertain or incomplete*?
- What sensors can be used to capture that information?

# PRESENTATION OF THE RUNNING EXAMPLE AND KNOWLEDGE GRAPH



UMEÅ UNIVERSITY

# PRESENTATION OF EXAMPLE AND KNOWLEDGE GRAPH

## Procedure:

- Join the main Zoom session
- Different members of the group present the running example.
- Others present the graph.
- Other the reflections

**Time for every presentation 5-10 minutes**



UMEÅ UNIVERSITY

INTERACTIVITY IN SMART ENVIRONMENTS 2020

# MORE EXAMPLES AND RESOURCES



UMEÅ UNIVERSITY

# EXAMPLES



Demo: KNIME example of an ontology, querying using SPARQL

The screenshot displays the KNIME Analytics Platform interface. On the left, the 'KNIME Explorer' pane shows a project structure with 'LOCAL (Local Workspace)' containing 'Example Workflows' and '12\_Exploring-pizza-ontology-with-semantic-web'. Below it, the 'Node Repository' lists various categories like IO, Manipulation, Views, Analytics, DB, etc.

The main workspace shows a workflow diagram. It starts with a 'Memory Endpoint' and a 'Triple File Reader' (loading 'pizza.owl'). Both feed into a 'SPARQL Insert' node. This node branches into two 'SPARQL Query' nodes: 'Query for veggie toppings' and 'query for all pizza types plus toppings + spiciness'. The 'Query for veggie toppings' also feeds into a 'Constant Value Column' node. The outputs of the 'SPARQL Query' nodes and the 'Constant Value Column' node are joined by a 'Joiner' node. This is followed by two 'Column Filter' nodes, a 'Missing Value' node, a 'RowID' node, and a 'Column Resorter' node. The final output is an 'Interactive View' node with the text 'Let's explore pizzas here!'.

A yellow-bordered text box above the workflow contains the text: 'This workflow shows how an OWL file can be read in KNIME Analytics Platform using the Triple File Reader node. An interactive view has been created where the extracted data can be explored.'

On the right side, the 'Description' pane for the workflow '12\_Exploring-pizza-ontology-with-semantic-web' is visible. It includes a title, a description, tags (ontology, OWL, triples, n-triples, ontologies, semantic web, linked data), and a link to 'Will They Blend? KNIME Meets the Semantic Web'. The creation date is listed as 2019-10-15 and the author as Martyna.

**THANK YOU**