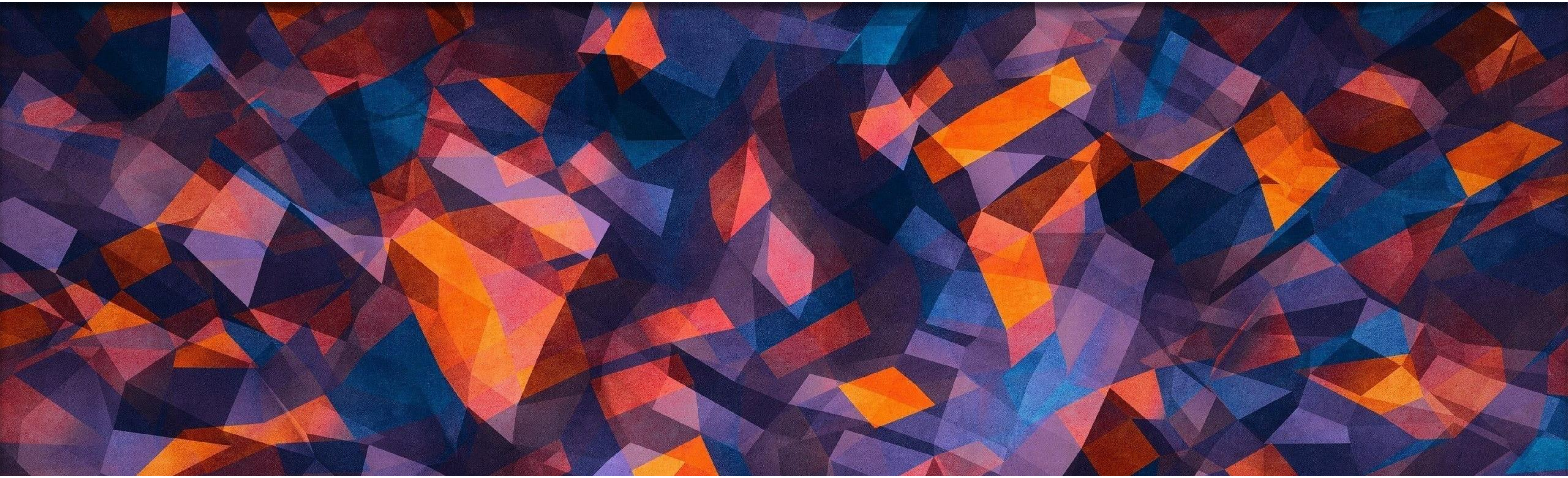


Web-based Autonomous Systems

Coordination I: Communication and Interaction

Chair for Interaction- and Communication-based Systems (ICS-HSG)

Universität St.Gallen

Our Journey

Prerequisites:
- **ASSE**
- **(…)**

Week 1:
Introduction

Week 2:
A Web for Machines

Week 3:
Knowledge Representation and Reasoning for the Web

Week 4:
Linked Data and Distributed Knowledge Graphs

Week 5:
Hypermedia Agents (Arch. and Programming)

Week 6 (Coordination I):
Agent Communication and Interaction

Week 7 (Coordination II):
Self-Organization and Stigmergy

Week 8 (Coordination III):
Normative MAS and Organizations

Week 9 (Coordination IV):
Trust & Reputation

Week 10:
Game Theory and Social Choice

Week 11:
Reinforcement Learning and Multi-Agent Learning

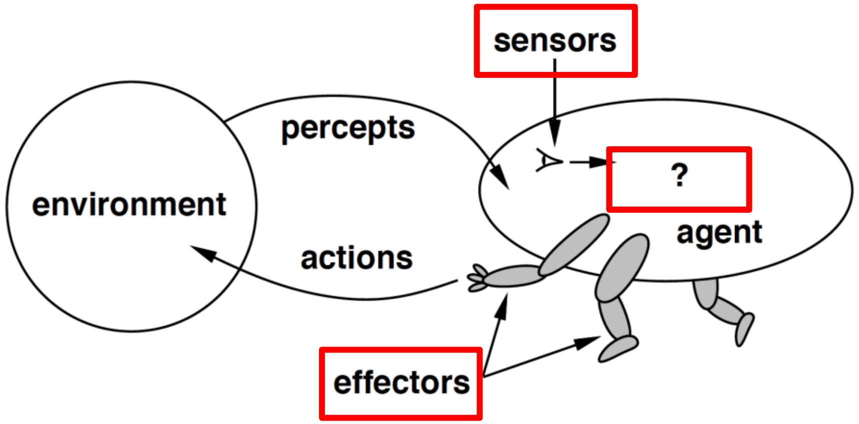
Week 12:
An Industry Perspective

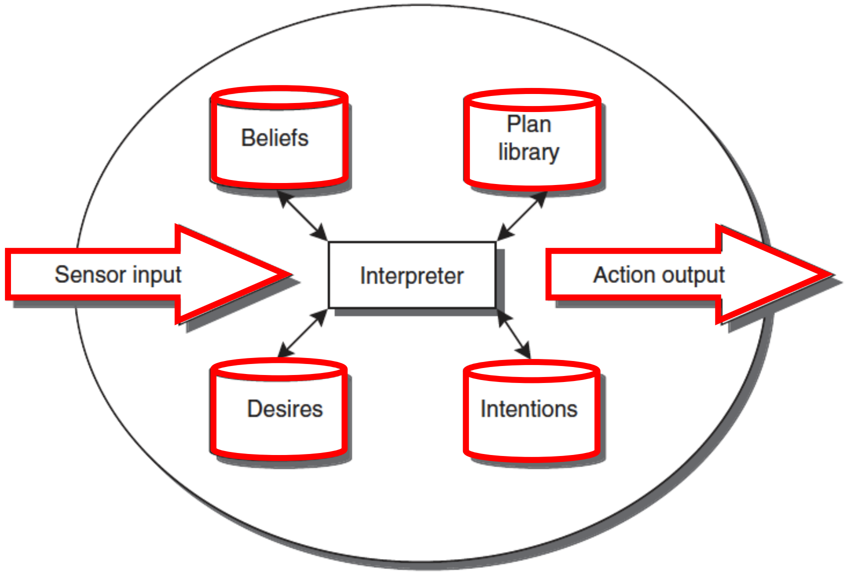
Exercises

Ex1: Writing Your First Agent(s)!
Ex2: Automated Planning
Ex3: Web Ontologies
Ex4: Operating on Linked Data
Ex5: BDI Agent
Ex6: Interacting Agents on the Web
Ex7: Ant Colony Optimization
Ex8: Organized Agent
Ex9: Trustworthy Agents
Ex10: Axelrod’s Agents
Ex11: Reinforcement Learning Agents
Course Review and Q&A

Last Week: Agent-Oriented Programming

A formal, “human-oriented” level of abstraction for programming systems of artificial agents [Shoham, 1993]





Lecture #1: An **agent function** maps any given percept sequence to an action

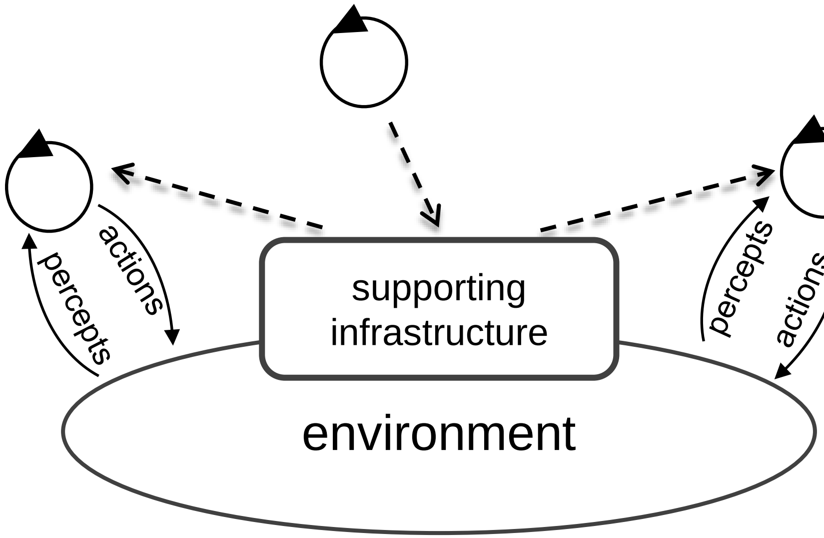
Lecture #5: The function of **BDI agents** is implemented based on Beliefs-Desires-Intensions.

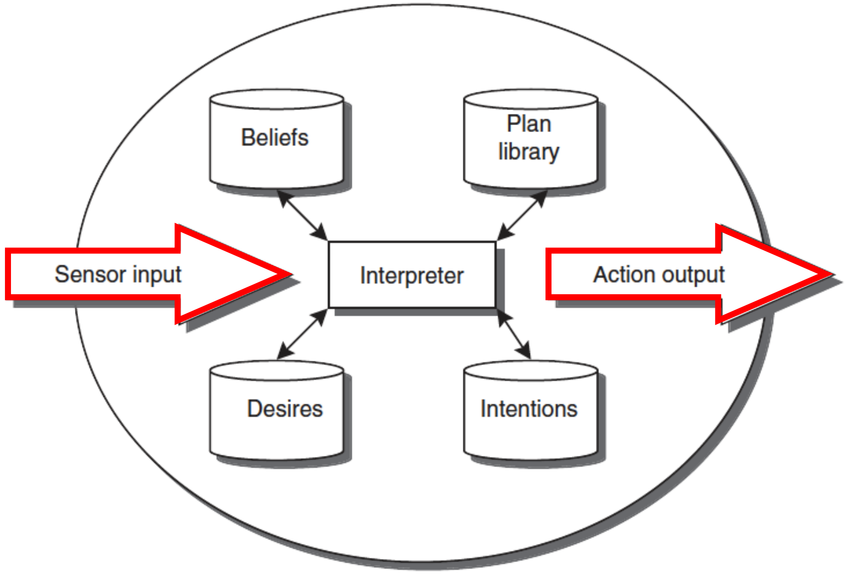
Lecture #6: Interaction-Oriented Programming deals with abstractions for defining interactions among entities in the system

Yoav Shoham, *Agent-oriented programming*, Artificial Intelligence, Volume 60, Issue 1, 1993.
Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Last Week: Agent-Oriented Programming

A formal, “human-oriented” level of abstraction for programming systems of artificial agents [Shoham, 1993]



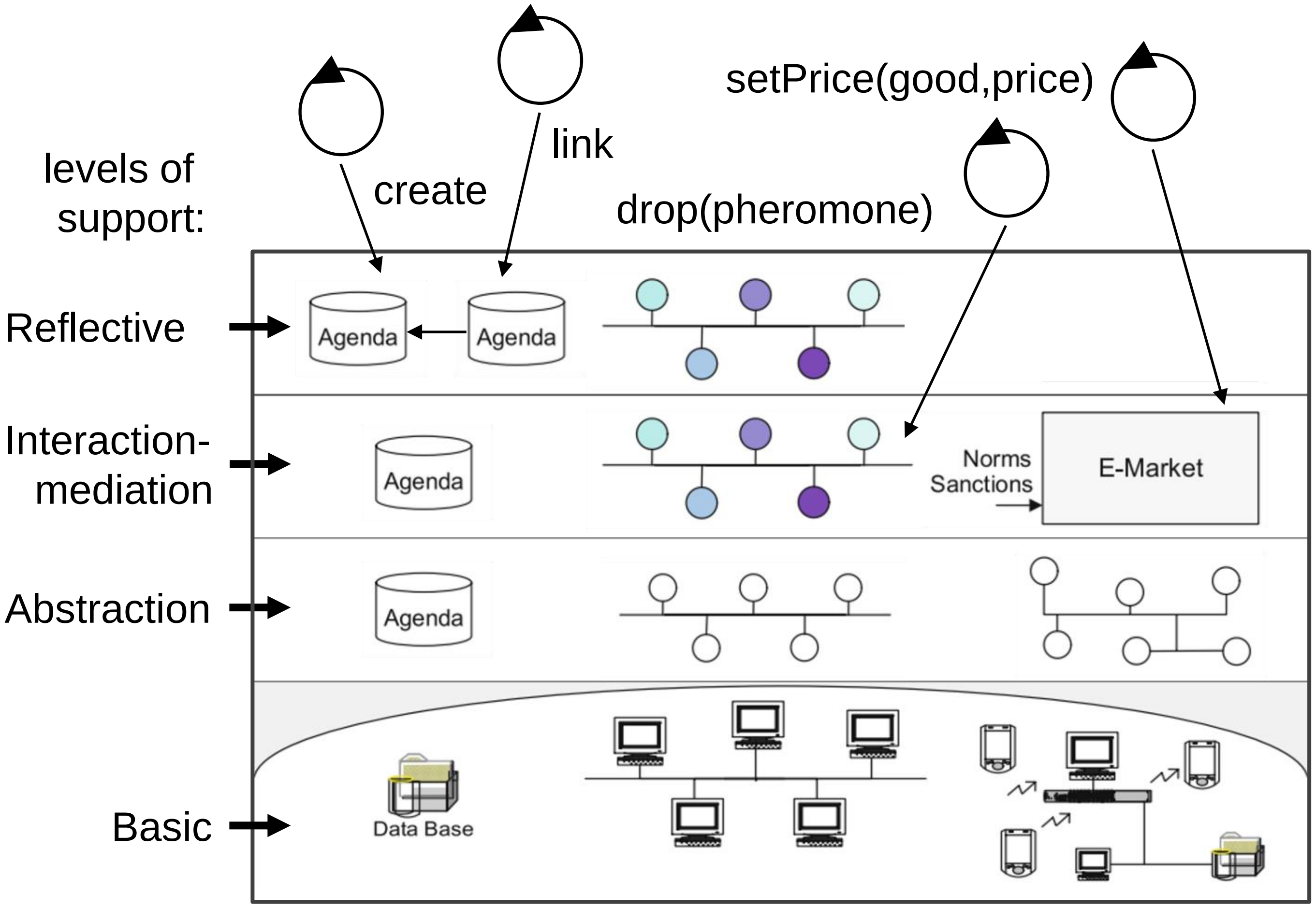


Lecture #6: Environment-Oriented Programming deals with abstractions for defining and structuring resource/processing entities shared among agents

Yoav Shoham, *Agent-oriented programming*, Artificial Intelligence, Volume 60, Issue 1, 1993.
Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Environment as a Design Abstraction

The **environment is a first-class abstraction** that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources [Weyns et al., 2007].



Reflection support: mechanisms to modify the functional behavior of the environment

- Example: creating and destroying artifacts

Interaction-mediation support: mechanisms to mediate, enact, and regulate interactions

- Example: pheromone infrastructure

Abstraction support: conceptual bridge between abstractions used to design and program agents and the deployment context domain specific

- Example: semantic models context, e.g. WO3 Farm

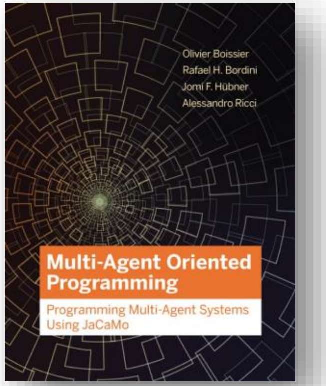
Basic interface support: raw access to the deployment context

- Example: Web APIs

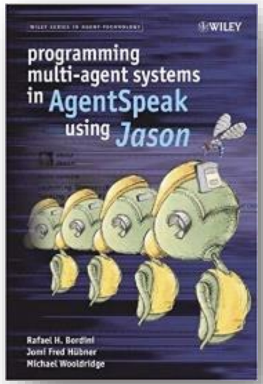
D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. JAAMAS 14, 5–30, 2007.

Today’s Agenda

- Agent to Environment Interaction
  - The Agents & Artifacts Meta-Model
  - Hands-on: Programming Artifacts in JaCaMo
- Agent to Agent Interaction
  - A Theory of Speech Acts
  - Hands-on: Communication actions in Jason
  - Agent Interaction Protocols



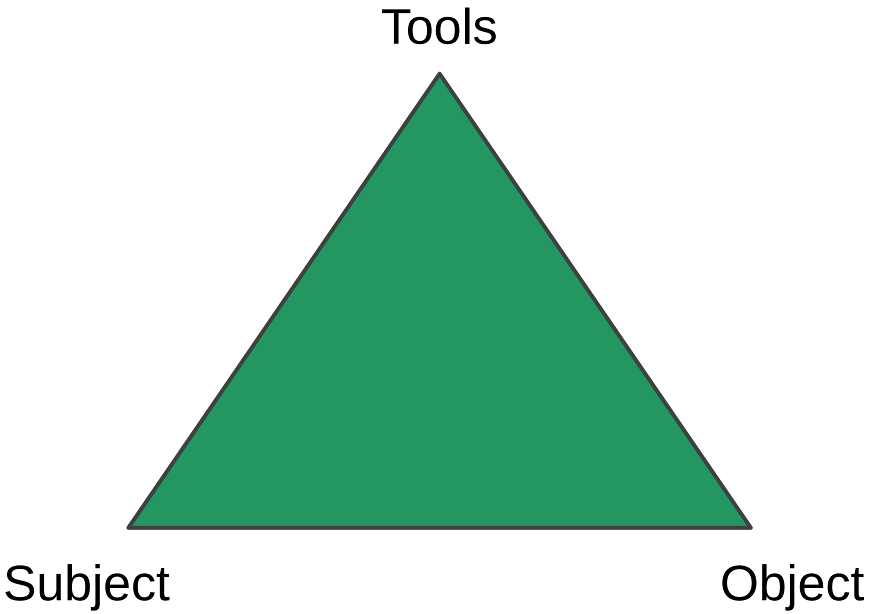
Chapters 5-7



Chapters 6

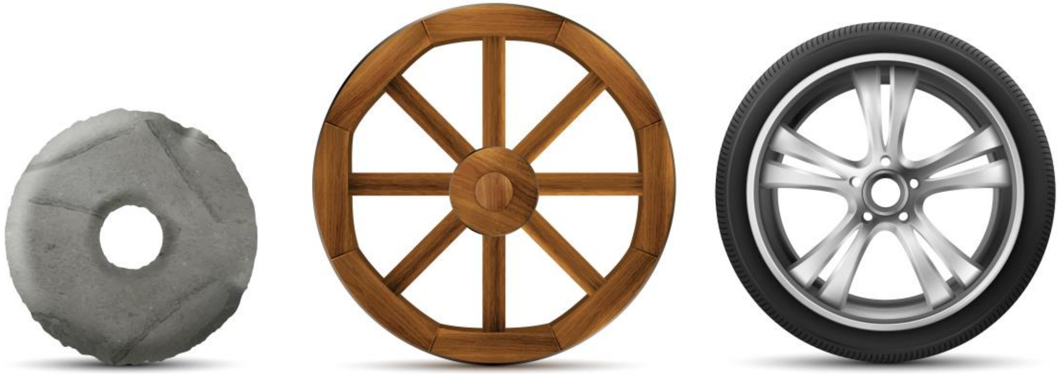
Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Activity Theory



Roots in cultural-historical psychology (1920s and 1930s)

Brought to CS and Human-Computer Interaction in the 1990s

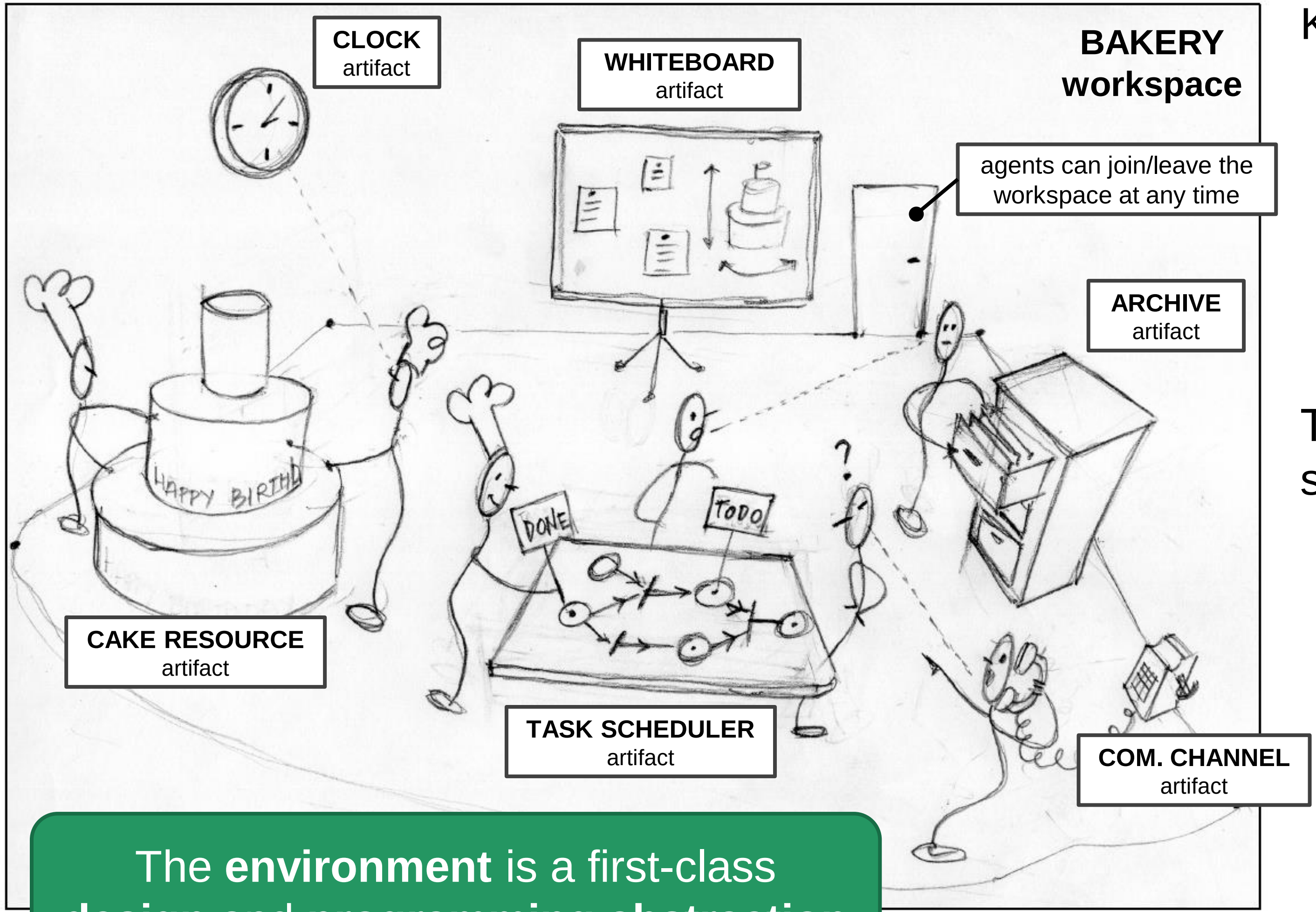


Activity (basic unit of analysis) is a **goal-directed interaction** with the world

The activity is mediated through **tools** (or **artifacts**), which evolve over time based on the experience of subjects

Bonni A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.
Bonnie A. Nardi. *My Life as a Night Elf Priest: An Anthropological Account of World of Warcraft*. University of Michigan Press, 2010.

The Agents & Artifacts Meta-Model



The **environment** is a first-class **design** and **programming abstraction**

Key idea: **separation of concerns**

- **agents** encapsulate **autonomous** behavior
- **artifacts** encapsulate **non-autonomous** behavior

Programming MAS = Programming **Agents**
+ Programming the **Environment**

The agents' environment is modelled as a **dynamic** set of **artifacts** grouped into **workspaces**

- the **actions** provided to agents are determined by the artifacts **discovered at run time**
- agents **construct**, **share**, and **use** artifacts to support their working activities

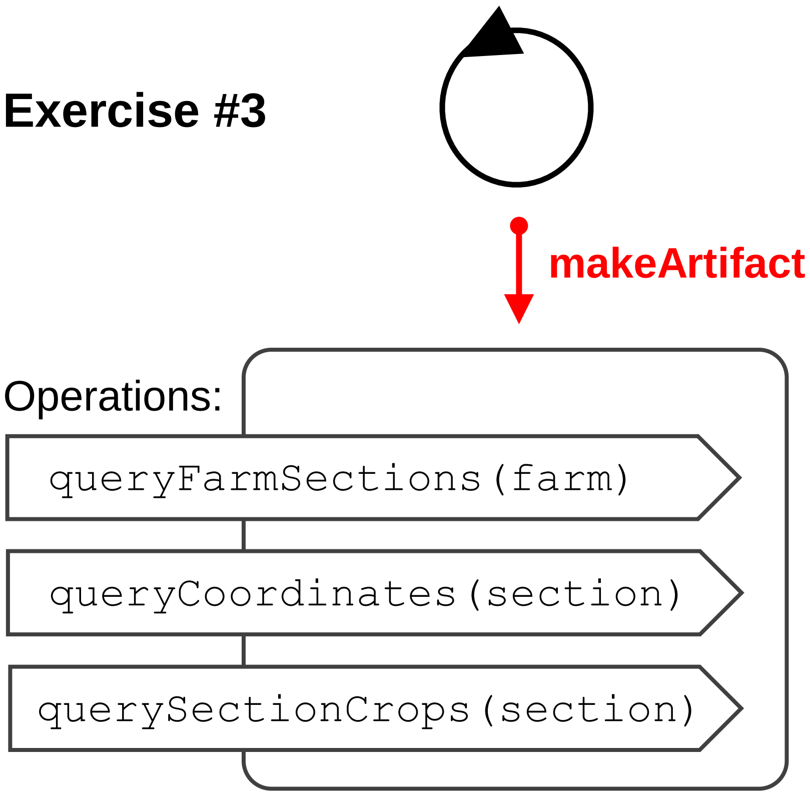
⇒ artifacts are **mediating tools** for goal-directed agents

⇒ agents can **modify** the **functional behavior** of the environment to meet their needs

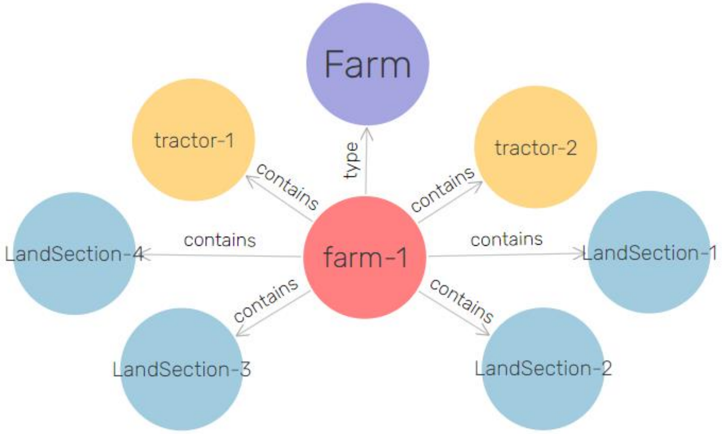
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Artifacts

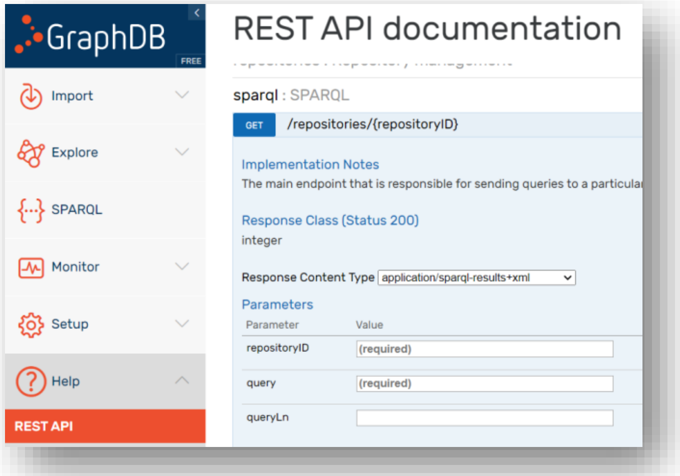
Exercise #3



Farm KG Artifact



Farm KG



GraphDB HTTP endpoint

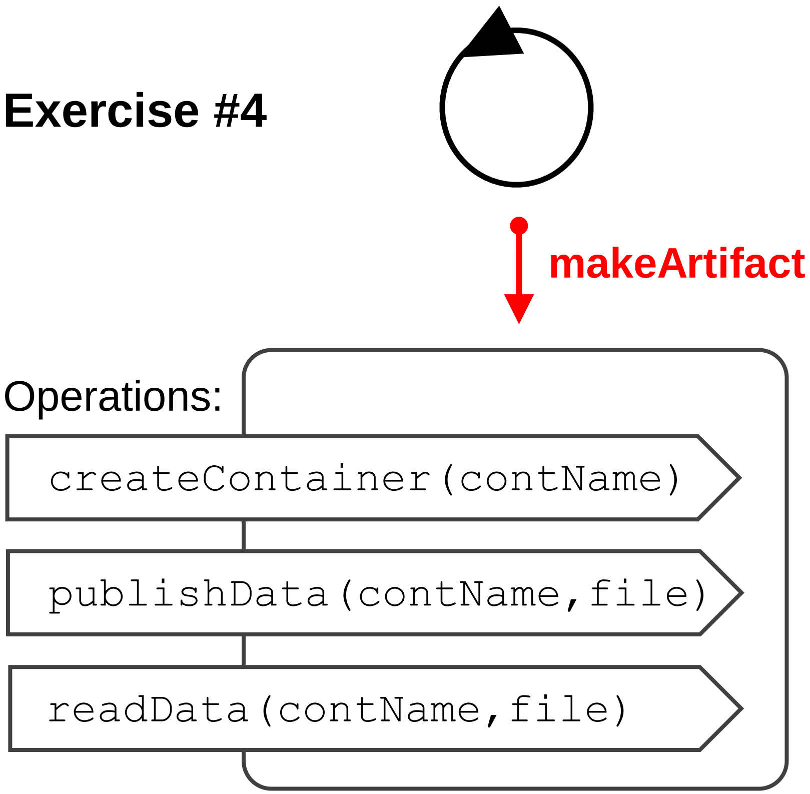
Reflective level

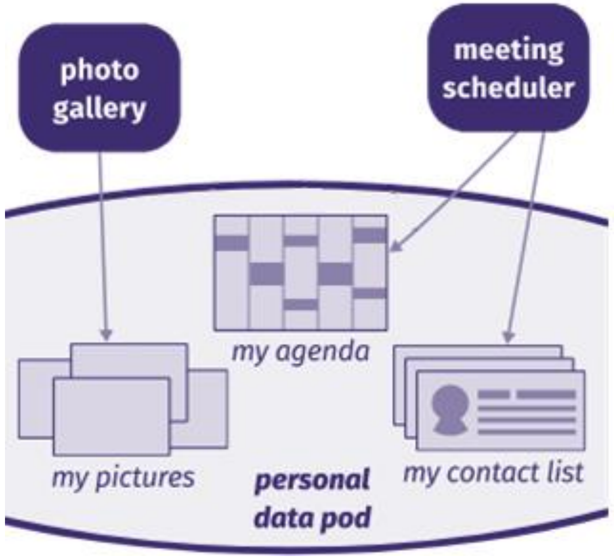
Abstraction level

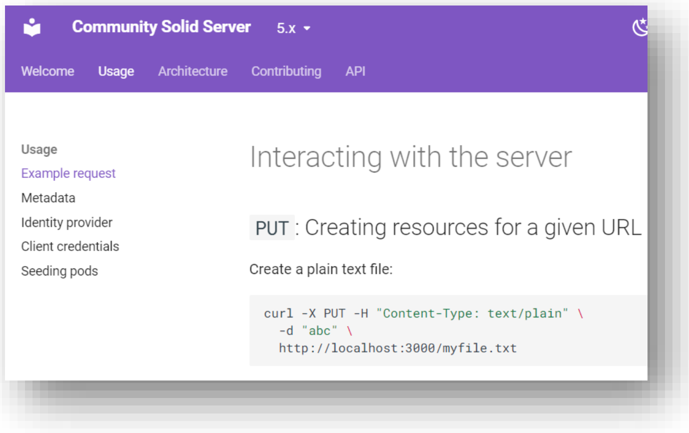
Basic interface level

Artifacts

Exercise #4







Solid Pod Artifact

Solid Pod

Solid Community Server
HTTP endpoint

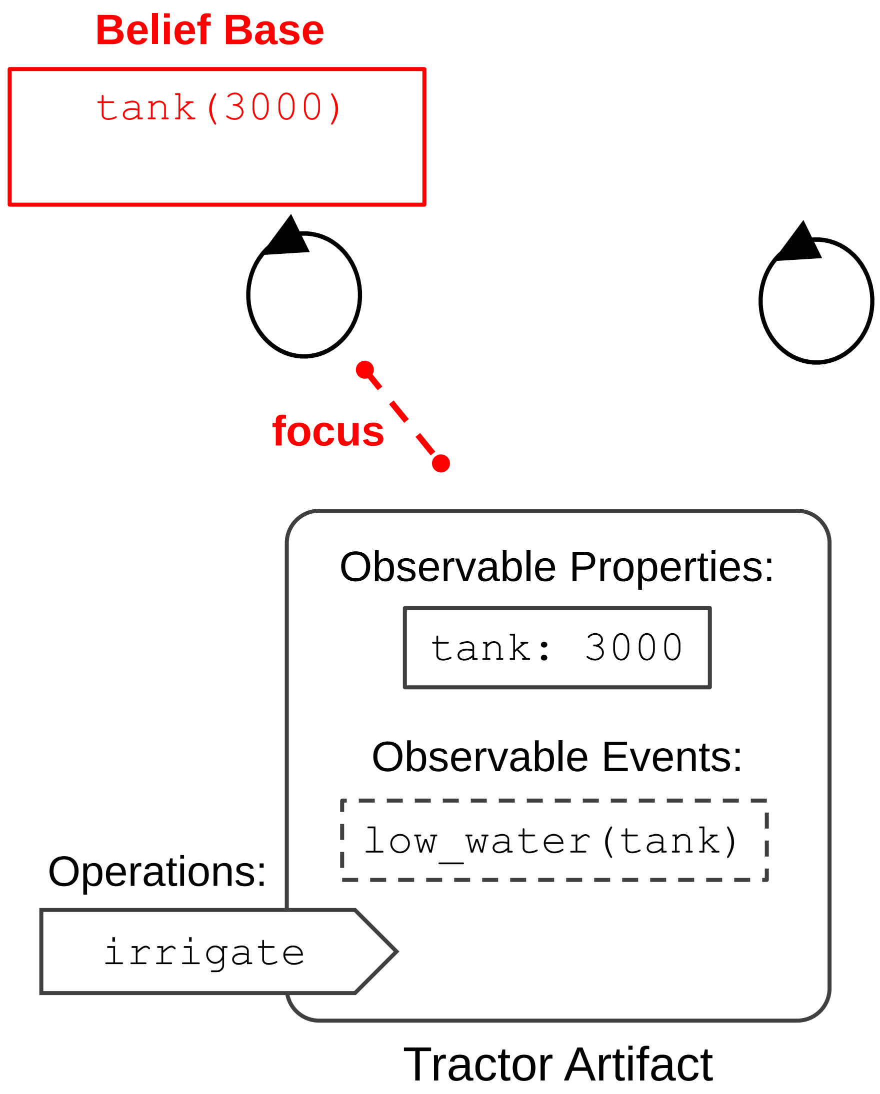
not just RDF triples
but more usable interfaces for agts

Reflective level

Abstraction level

Basic interface level

Artifacts



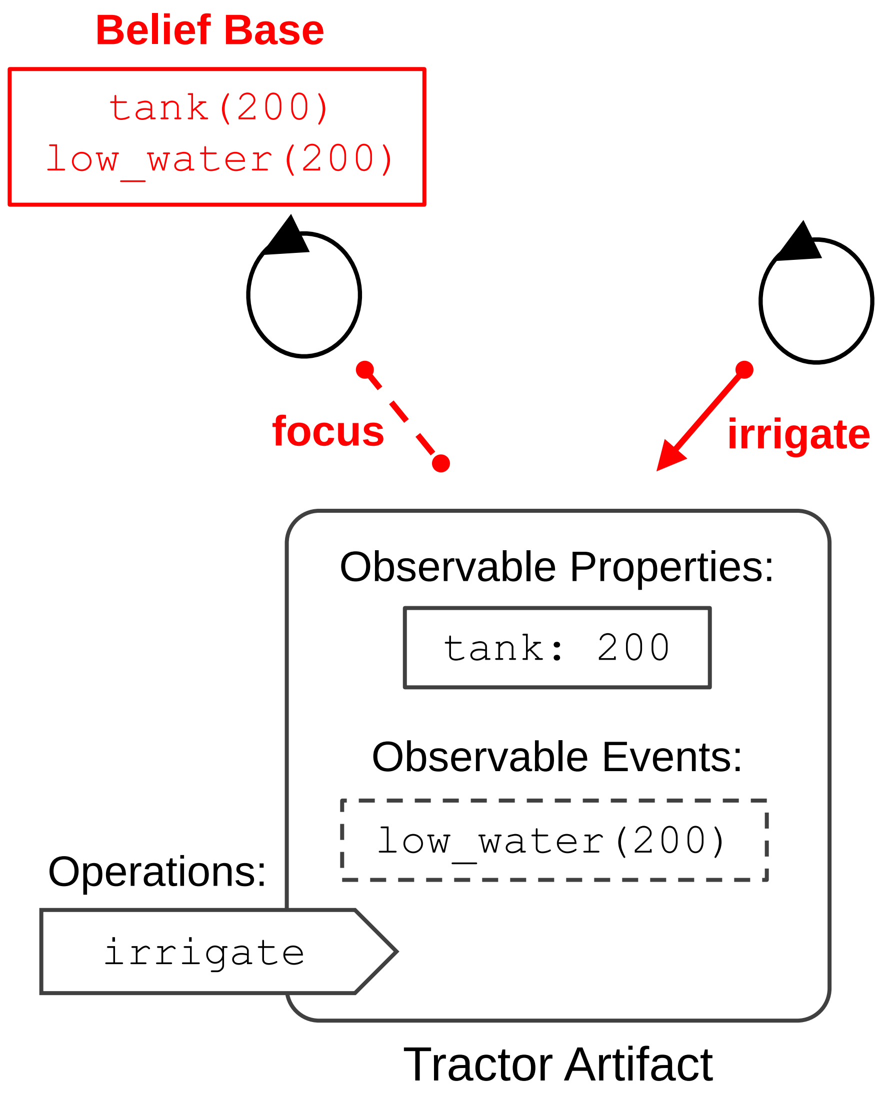
Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can change the values of observable properties or can trigger events

Agents can **focus** on artifacts to perceive observable properties and events

Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Artifacts



Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can change the values of observable properties or can trigger events

Agents can **focus** on artifacts to perceive observable properties and events

Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Artifacts

Why is **intentional focus** useful?

Allows agents to **select** the parts of the environment that are relevant to their goals

- promotes **scalability**
  - agents can cope with larger environments
  - the environment infrastructure can serve more agents
- promotes **autonomy** from the environment

Artifacts as computational objects

– *usage interface*:

- **observable properties**: state variables that can be perceived by agents
- **observable events**: non-persistent signals that carry information and can be perceived by agents
- **operations**: environmental actions provided to the agent
  - operations can change the values of observable properties or can trigger events

Agents can **focus** on artifacts to perceive observable properties and events

⇒ what parts of web are relevant to achieve my goals? -- can't observe whole web

Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

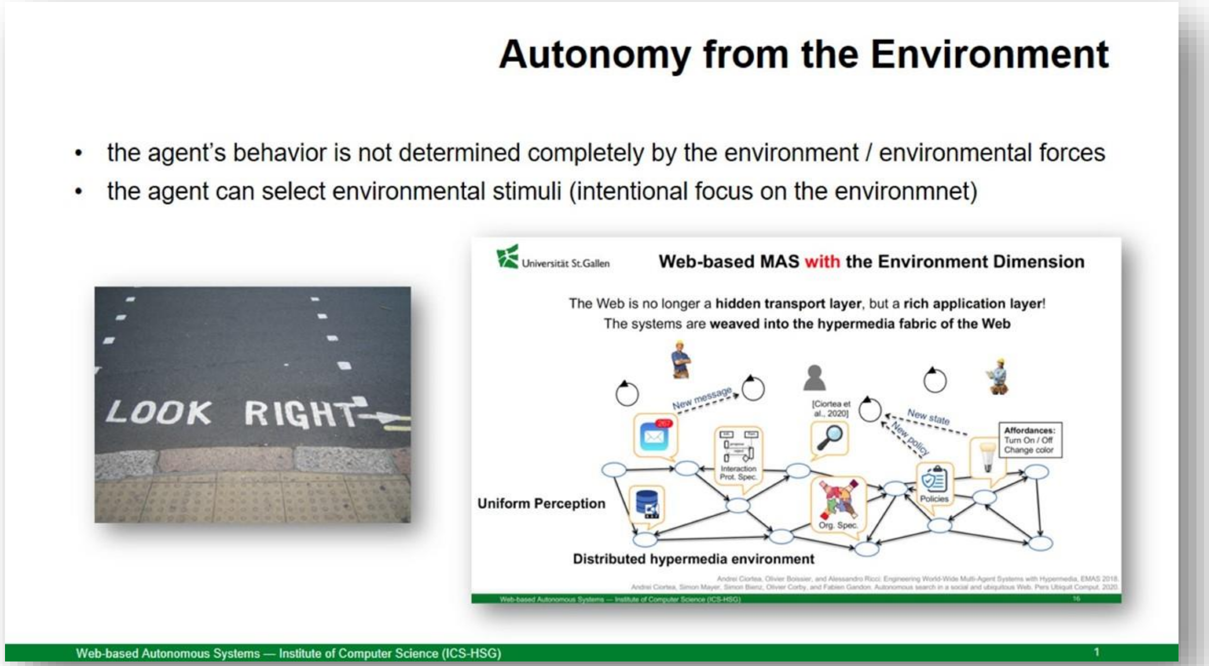
Artifacts

Why is **intentional focus** useful?

Allows agents to **select** the parts of the environment that are relevant to their goals

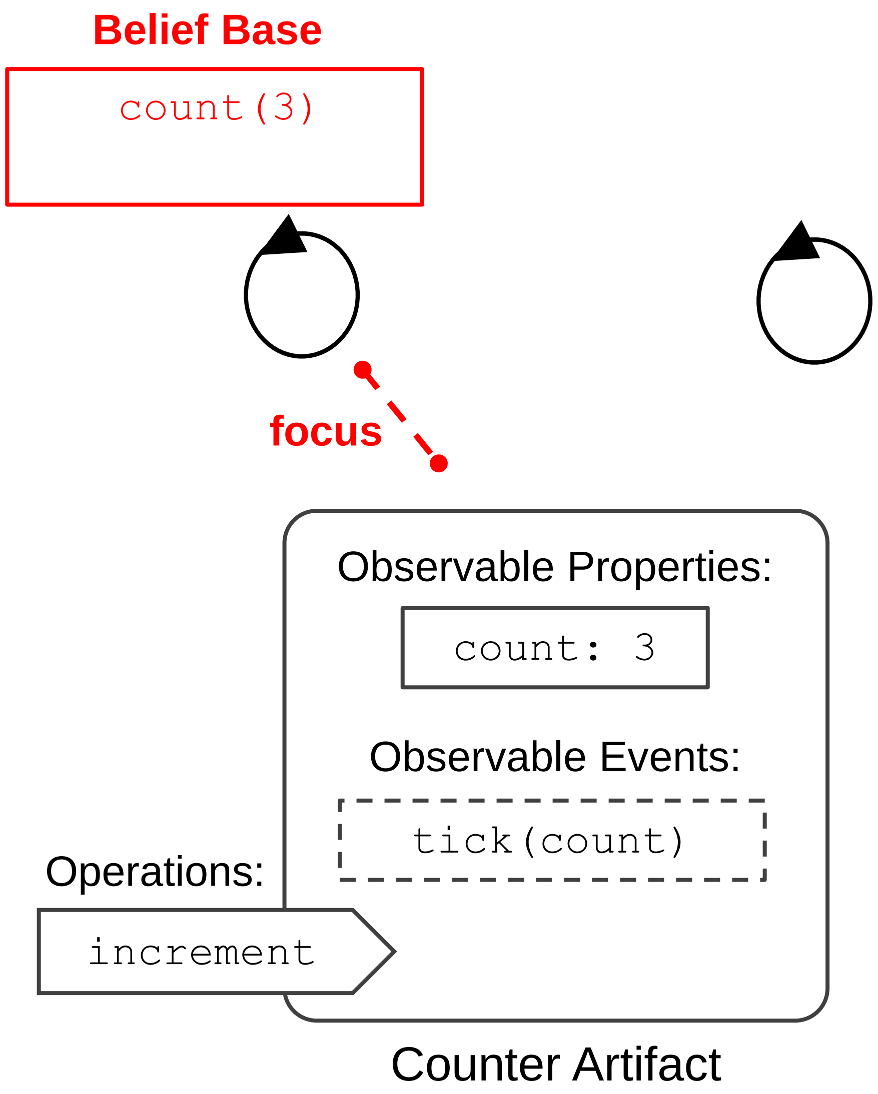
- promotes **scalability**
  - agents can cope with larger environments
  - the environment infrastructure can serve more agents
- promotes **autonomy** from the environment

Lecture #1:



Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Artifacts



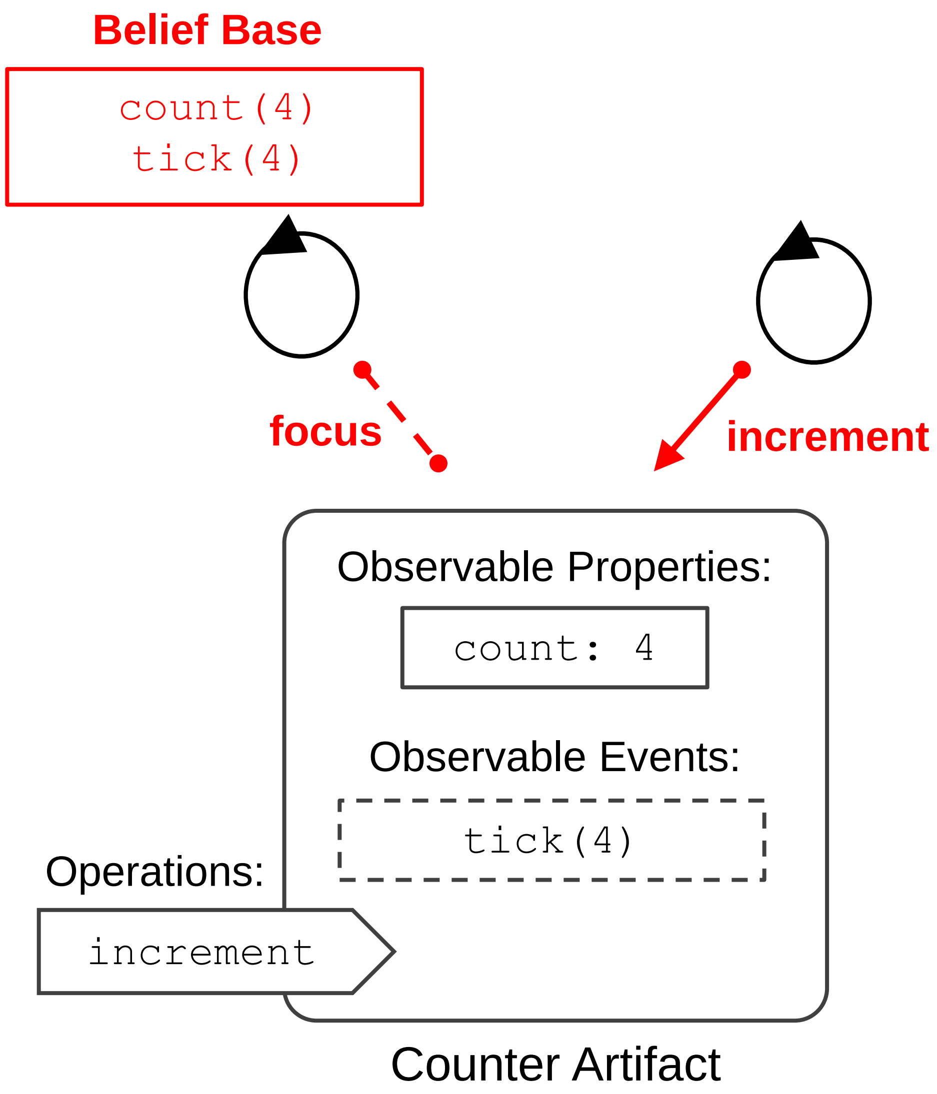
Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can change the values of observable properties or can trigger events

Artifacts can be used a programming construct for **coordination**

Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Artifacts



Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can change the values of observable properties or can trigger events

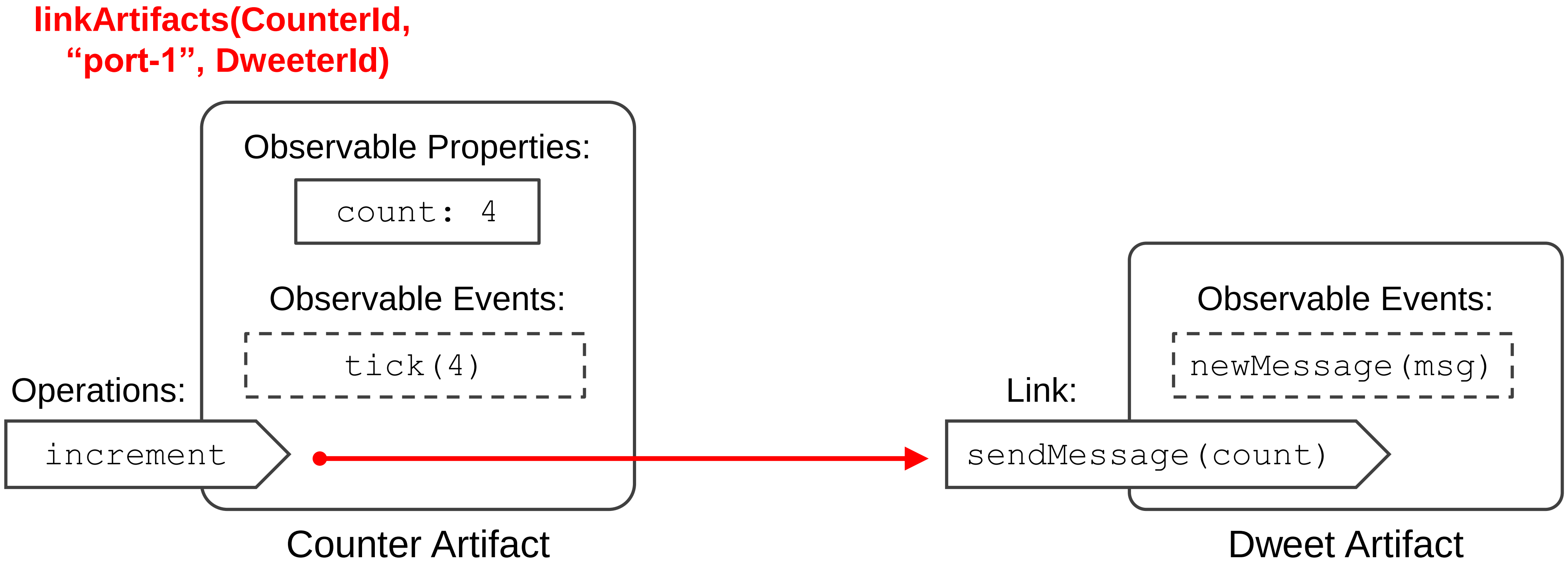
Artifacts can be used a programming construct for **coordination**

Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Artifacts

Artifacts as computational objects

- *link interface*:
  - used to connect artifacts
- *manual*:
  - what functionalities and how to use them

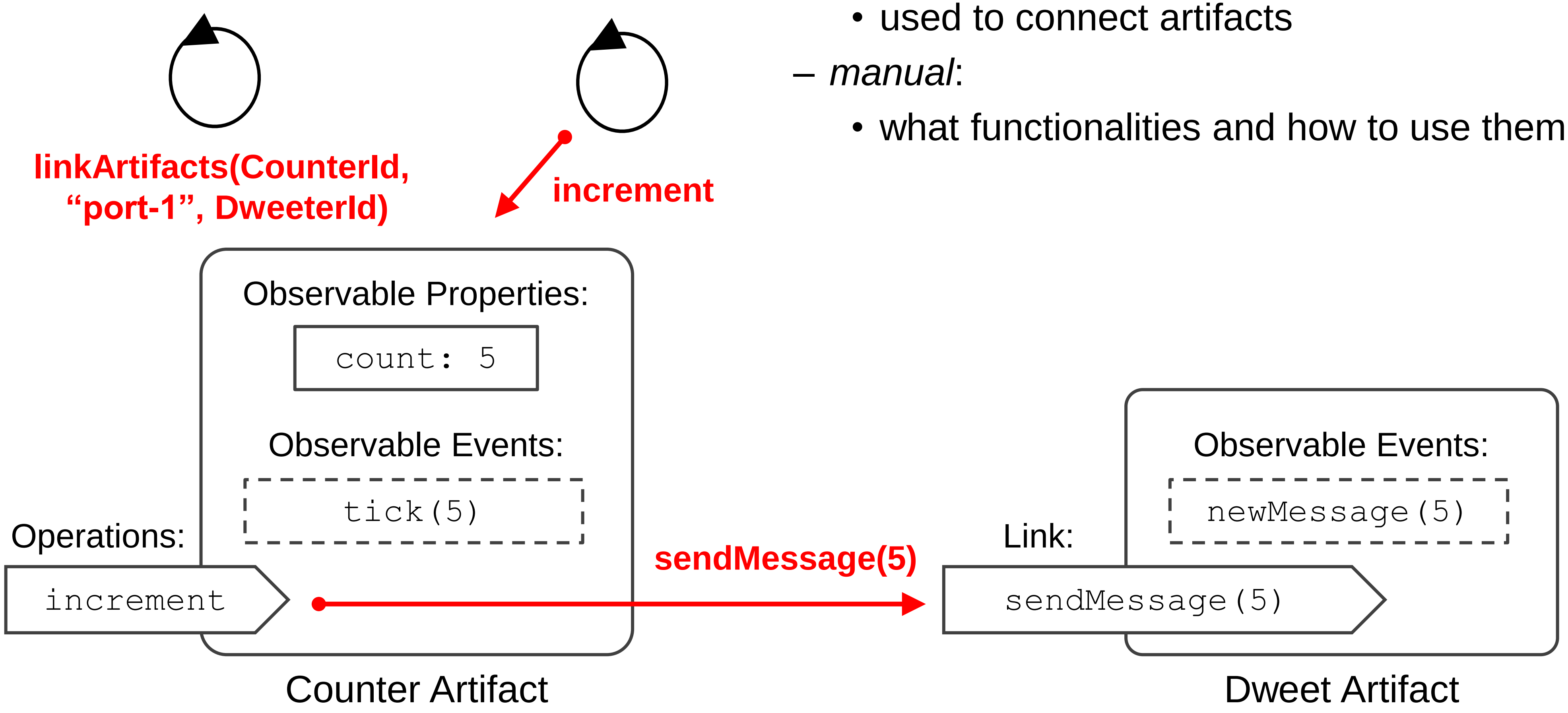


Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Artifacts

Artifacts as computational objects

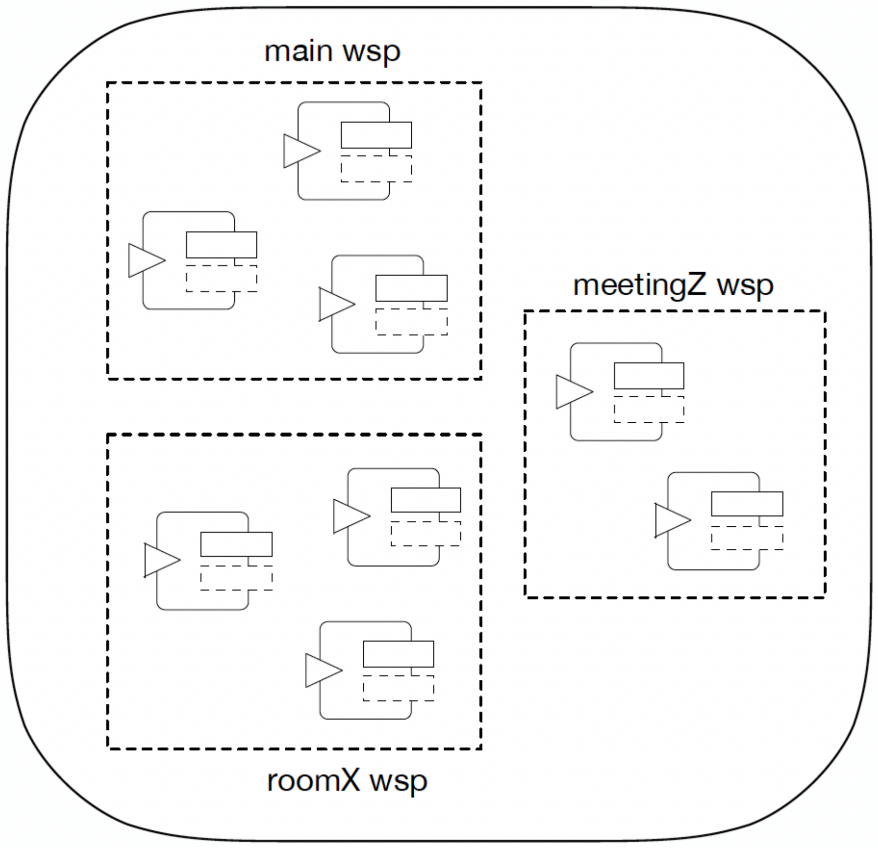
- *link interface*:
  - used to connect artifacts
- *manual*:
  - what functionalities and how to use them



Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

Workspaces

Lecture #1:
situatedness and **embodiement**



Containers for agents and artifacts

- allow to **structure** complex/distributed environments
- provide a notion of **locality** and **situatedness**

Agents can **join**, **leave**, and **work in** multiple workspaces (at the same time)

- agents are **embodied** and interact within the workspace through **body artifacts**

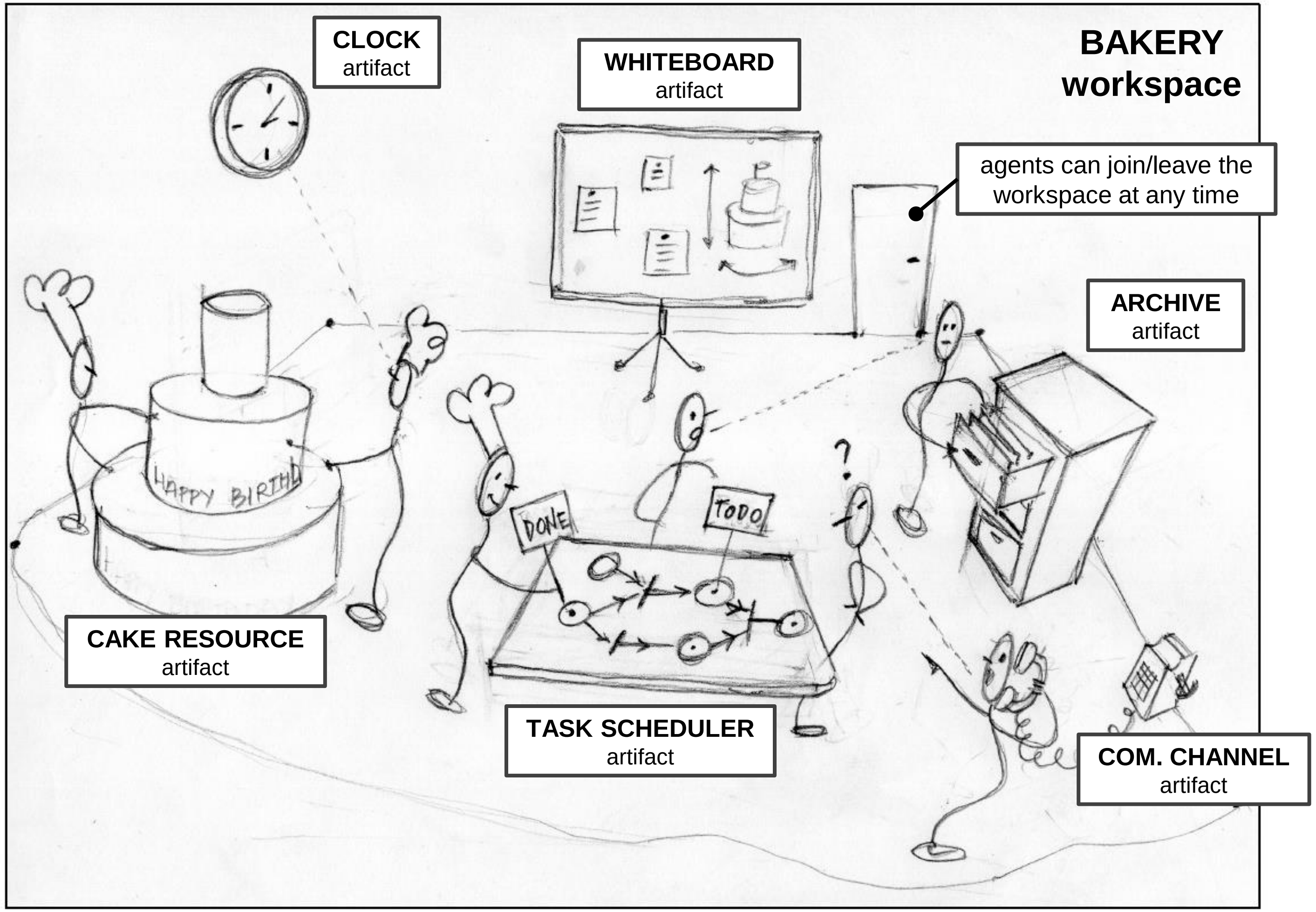
⇒ separation of concerns between the **agent's mind** and the **agent's body**

⇒ allows **heterogeneous agents** (implementing different architectures) to *join* and *work in* the same environment

Workspaces can be distributed over a network

Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

The Agents & Artifacts Meta-Model



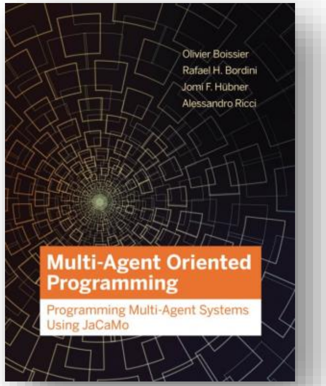
The **environment** is a first-class **design** and **programming abstraction**

Programming MAS = Programming **Agents** + Programming the **Environment**

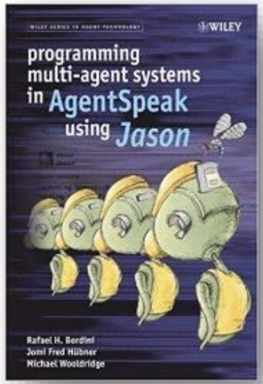
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Today's Agenda

- Agent to Environment Interaction
  - The Agents & Artifacts Meta-Model
  - Hands-on: Programming Artifacts in JaCaMo
- Agent to Agent Interaction
  - A Theory of Speech Acts
  - Hands-on: Communication actions in Jason
  - Agent Interaction Protocols



Chapters 5-7



Chapters 6

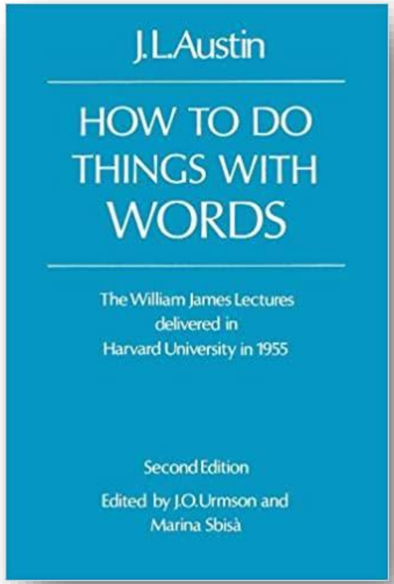
Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

A Theory of Speech Acts

Communication as **action** [Austin, 1962]

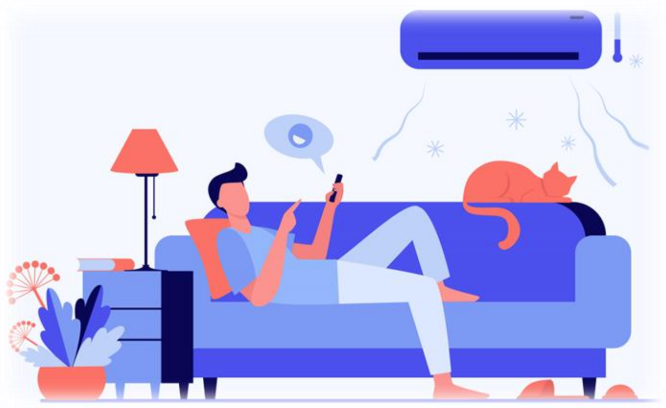
- **constatives** (true/false utterances that describe something): “Switzerland is in Europe.”
- **performatives** (utterances that represent actions): “Would you please shut the door?”

Speech Acts



The implied action is not always obvious

- **locutionary** act (physical utterance):
  “It’s getting chilly!” — when the AC is raging
- **illocutionary** act (the actual action):
  “She requested me to turn down the AC.”
- **perlocutionary** act (effect of the action):
  “She got me to turn off the AC.” — hopefully!



John L. Austin, How to Do Things With Words. Oxford University Press, Oxford, 1962.
John R. Searle. A classification of illocutionary acts. Language in Society, 5(1). 1976.

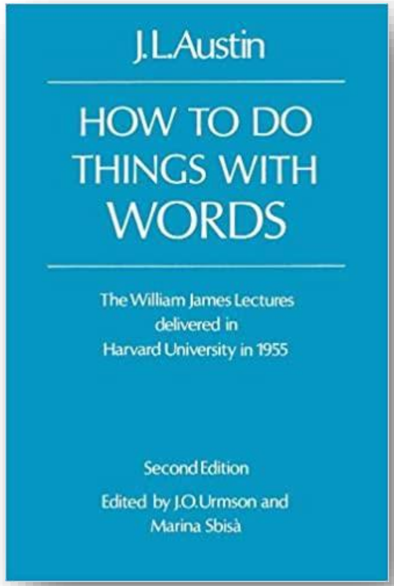
A Theory of Speech Acts

Communication as **action** [Austin, 1962]

– **constatives** (true/false utterances that describe something): “Switzerland is in Europe.”

– **performatives** (utterances that represent actions): “Would you please shut the door?”

Speech Acts



A taxonomy of **speech acts** [Searle, 1976]

- **Representatives/Assertives** (informing the *hearer*): The door is shut.
- **Directives** (attempts to get the *hearer* to do something): Shut the door!
- **Commissives** (promises—commit the *speaker* to doing something): I will shut the door.
- **Expressives** (express a pshychological state of the *speaker*): Thank you for shutting the door! (gratitude)
- **Declaratives** (effect institutional changes): Your employment is hereby terminated.

More precise **classification**

Defines **speech acts** in terms of the **mental states** of the speaker and a hearer

John L. Austin, How to Do Things With Words. Oxford University Press, Oxford, 1962.
John R. Searle. A classification of illocutionary acts. Language in Society, 5(1). 1976.

Agent Communication Languages

An **agent communication language (ACL)** is a language for constructing messages that encode **speech acts**, where a **message** is the individual unit of communication between two or more agents (definition adapted from [FIPA, 2002]).

- provides the basis of communication between independently designed and developed agents

Foundation for Intelligent Physical Agents (FIPA)
founded in Geneva in 1996

Two well-known ACLs:

- **Knowledge Query and Manipulation Language (KQML)**, developed as part of the DARPA Knowledge Sharing Effort
- **FIPA ACL** (based on KQML)

Jason uses a variant of **KQML**

KQML introduced a separation of concerns between:

- the semantics of **illocutionary acts** (**performative verbs**), which are independent of an application domain
- the semantics of the **message content**, which is domain-dependent; the content is usually represented using domain ontologies in a formal knowledge representation language

FIPA Abstract Architecture, 2002: http://fipa.org/specs/fipa00001/SC00001L.html

Agent Communication in Jason

Send message (non-blocking)

```
.send(bob, tell, forecast(rainy)[certainty(0.6)]); // sent by jane to bob
```

Send ask message and wait for reply (blocking action)

```
.send(bob, askOne, forecast(Forecast), Answer); // sent by bob to jane
```

Broadcast message

```
.broadcast(tell, forecast(rainy)[certainty(0.6)]); // sent by jane to everyone
```

The Jason interpreter equips every agent with default plans for handling received messages

- message handling follows the Jason-defined semantics of KQML performative verbs

Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

KQML Performatives in Jason

An **action** on the **mental states** of *r*

Sender: *s*, Receiver: *r*

tell: *s* intends *r* to believe (that *s* believes) the literal in the message's content to be true

Semantics of Speech Acts

If utterances are actions, a formalism for reasoning about actions can be applied to utterances as well [Cohen & Perrault, 1979]

⇒ STRIPS-style approach to define the semantics of speech acts

Semantics of KQML performatives in terms of [Labrou & Finin, 1994]:

- Preconditions: describe necessary conditions for an agent to send a performative and for the receiver to accept and process it
- Postconditions: describe the state of the sender after the utterance of a performative and the state of the receiver after the receipt of a message
- Completion conditions: final state of the sender

P. Cohen and R. Perrault. Elements of a plan based theory of speech acts. Cognitive Science, 3, 1979.
Y. Labrou and T. Finin. A semantics approach for KQML—a general purpose communication language for software agents. CIKM 1994.
Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

KQML Performatives in Jason

An **action** on the **mental states** of *r*

Sender: *s*, Receiver: *r*

tell: *s* intends *r* to believe (that *s* believes) the literal in the message's content to be true

Semantics of Speech Acts

Semantics of **tell** [Labrou & Finin, 1994]:

– Preconditions on the states of sender s and receiver r:
  - Pre(*s*): $bel(s,x) \wedge know\left(s, want\left(r, know(r, bel(s,x))\right)\right)$
  - Pre(*r*): $intend\left(r, know(r, bel(s,x))\right)$

– Postconditions on the states of sender s and receiver r:
  - Post(*s*): $know\left(s, know(r, bel(s,x))\right)$
  - Post(*r*): $know\big(r, bel(s,x)\big)$

– Completion condition: $know\big(r, bel(s,x)\big)$

Mental atitudes:
- belief (bel)
- knowledge $(know)$
- desire $(want)$
- intention $(intend)$

Y. Labrou and T. Finin. A semantics approach for KQML—a general purpose communication language for software agents. CIKM 1994.
Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

KQML Performatives in Jason

Sender: *s*, Receiver: *r*

tell: *s* intends *r* to believe (that *s* believes) the literal in the message's content to be true

untell: *s* intends *r* not to believe (that *s* believes) the literal in the message's content to be true

askOne: *s* wants to know if the content of the message is true for *r* (i.e., if there is an answer that makes the content a logical consequence of *r*'s belief base, by appropriate substitution of variables)

askAll: *s* wants all of *r*'s answers to a question

achieve: *s* requests *r* to try and achieve a state of affairs where the literal in the message content is true (i.e., *s* is delegating a goal to *r*)

unachieve: *s* requests *r* to drop the goal of achieving a state of affairs where the message content is true

tellHow: *s* informs *r* of a plan (*s*'s know-how)

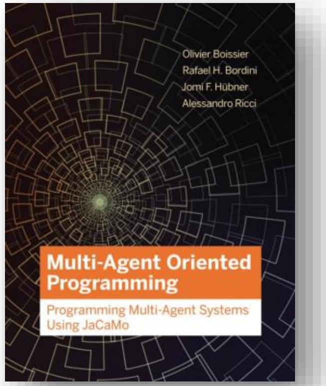
untellHow: *s* requests that *r* disregard a certain plan (i.e., delete that plan from its plan library)

askHow: *s* wants all of *r*'s plans that are relevant for the triggering event in the message content

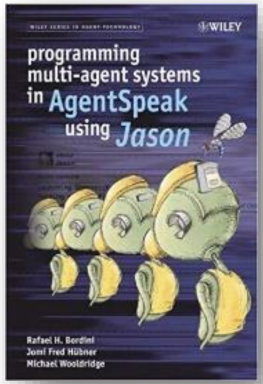
Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Today's Agenda

- Agent to Environment Interaction
  - The Agents & Artifacts Meta-Model
  - Hands-on: Programming Artifacts in JaCaMo
- Agent to Agent Interaction
  - A Theory of Speech Acts
  - Hands-on: Communication actions in Jason
  - Agent Interaction Protocols



Chapters 5-7



Chapters 6

Rafel Bordini et al., *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

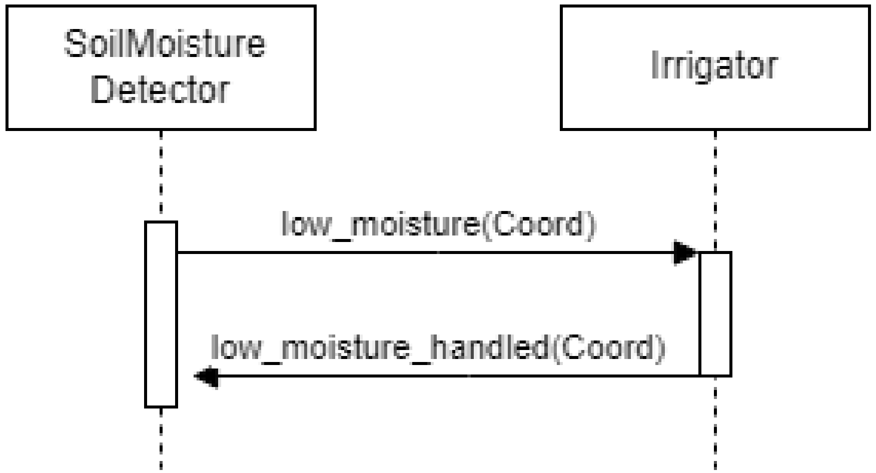
Agent Interaction Protocols

An agent communication language allows to **construct messages** with well-defined semantics

An **agent interaction protocol** specifies **who** can say **what** to **whom**, and what are **possible reactions** to received messages

- structure conversations as sequences of speech acts — and thus restrict the use of speech acts
- enable interaction-oriented engineering in MAS (protocols as **first-class abstractions**)

Exercise #3



SoilMoistureDetector

```
+!initiate_protocol(Coord) <-
   .send(irrigator,tell,low_moisture(Coord)).

+moisture_sufficient(Coord) : true <-
  .print("Protocol completed").
```

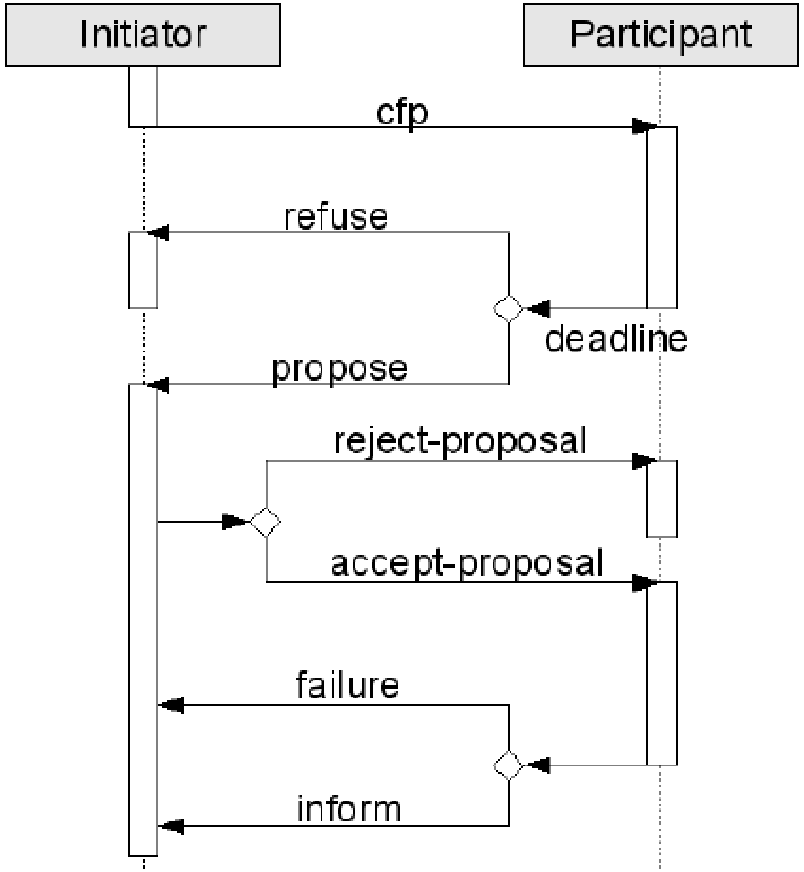
Irrigator

```
+low_moisture(Coord): true <-
   !irrigate(Coord); // not specified by the protocol
   .send(detector,tell, low_moisture_handled(Coord)).
```

Agent Interaction Protocols

FIPA standardized several agent interaction protocols:
http://fipa.org/repository/standardspecs.html

Identifier	Title
SC00026	FIPA Request Interaction Protocol Specification
SC00027	FIPA Query Interaction Protocol Specification
SC00028	FIPA Request When Interaction Protocol Specification
SC00029	FIPA Contract Net Interaction Protocol Specification
SC00030	FIPA Iterated Contract Net Interaction Protocol Specification
XC00031	FIPA English Auction Interaction Protocol Specification
XC00032	FIPA Dutch Auction Interaction Protocol Specification
SC00033	FIPA Brokering Interaction Protocol Specification
SC00034	FIPA Recruiting Interaction Protocol Specification
SC00035	FIPA Subscribe Interaction Protocol Specification
SC00036	FIPA Propose Interaction Protocol Specification



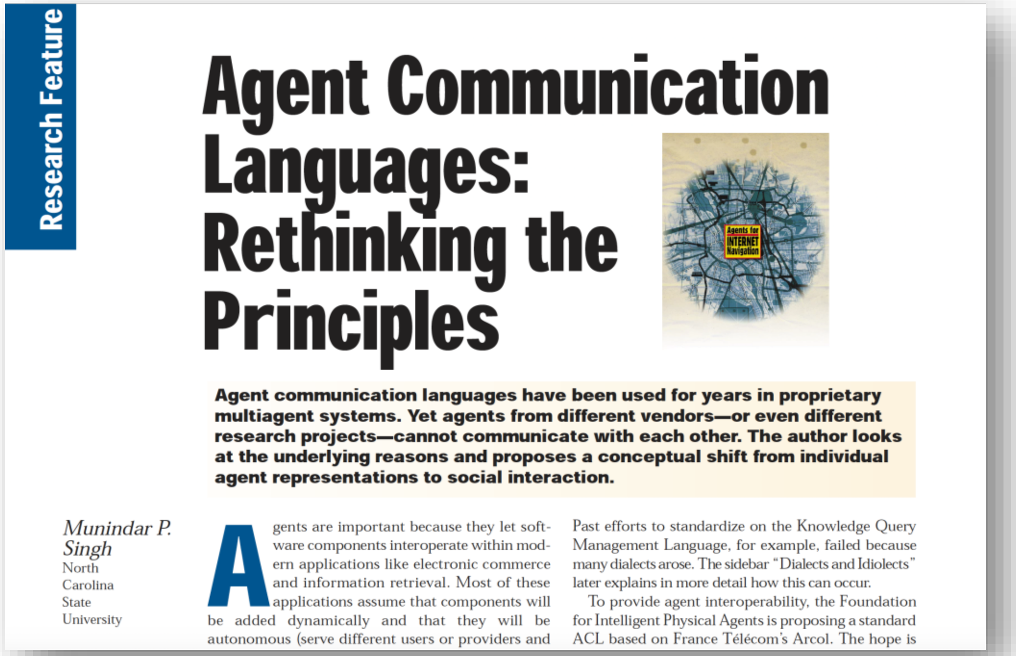
FIPA Contract Net Protocol

FIPA Abstract Architecture, 2002: http://fipa.org/specs/fipa00001/SC00001L.html

Speech Acts: Limitations?

Agent communication and interaction based on speech acts is intuitive for developers and simple to use in practice, but relies on two important assumptions:

- agents participating to interactions have mental states
- agents behave sincerely (works best for closed systems)



Research Feature

Agent Communication Languages: Rethinking the Principles

Agent communication languages have been used for years in proprietary multiagent systems. Yet agents from different vendors—or even different research projects—cannot communicate with each other. The author looks at the underlying reasons and proposes a conceptual shift from individual agent representations to social interaction.

Munindar P. Singh
North Carolina State University

Agents are important because they let software components interoperate within modern applications like electronic commerce and information retrieval. Most of these applications assume that components will be added dynamically and that they will be autonomous (serve different users or providers and

Past efforts to standardize on the Knowledge Query Management Language, for example, failed because many dialects arose. The sidebar "Dialects and Idiolects" later explains in more detail how this can occur.

To provide agent interoperability, the Foundation for Intelligent Physical Agents is proposing a standard ACL based on France Télécom's Arcol. The hope is

Later in the course we'll talk about **trust and reputation**, **social power**, and **organizations** — which provide means to filter messages!

Agent communication:
mental agency vs. **social agency**

More on **social agency** and **commitments** in **Lecture #8**!

Munindar P. Singh. *Agent Communication Languages: Rethinking the Principles*. Computer, vol. 31, no. 12. 1998.

Universität St.Gallen

Our Journey

Prerequisites:
- **ASSE**
- **(…)**

Week 1:
Introduction

Week 2:
A Web for Machines

Week 3:
Knowledge Representation and Reasoning for the Web

Week 4:
Linked Data and Distributed Knowledge Graphs

Week 5:
Hypermedia Agents (Arch. and Programming)

Week 6 (Coordination I):
Agent Communication and Interaction

Week 7 (Coordination II):
Self-Organization and Stigmergy

Week 8 (Coordination III):
Normative MAS and Organizations

Week 9 (Coordination IV):
Trust & Reputation

Week 10:
Game Theory and Social Choice

Week 11:
Reinforcement Learning and Multi-Agent Learning

Week 12:
An Industry Perspective

Exercises

Ex1: Writing Your First Agent(s)!	Ex7: Ant Colony Optimization
Ex2: Automated Planning	Ex8: Organized Agent
Ex3: Web Ontologies	Ex9: Trustworthy Agents
Ex4: Operating on Linked Data	Ex10: Axelrod’s Agents
Ex5: BDI Agent	Ex11: Reinforcement Learning Agents
Ex6: Interacting Agents on the Web	Course Review and Q&A

Any Questions / Comments / Doubts / Concerns?

Images

https://www.istockphoto.com/
https://freepik.com
https://www.bostondynamics.com/products/spot
https://billiards.colostate.edu/faq/cut/estimating-angle/
https://www.linearmotiontips.com/designing-linear-motion-tracks-robotic-positioning/
https://www.utas.edu.au/news/2017/6/7/301-a-day-in-the-life-of-a-typical-phd-student/