

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

## From Objects To Agents

### **This is the author's manuscript**

*Original Citation:*

*Availability:*

This version is available <http://hdl.handle.net/2318/1845182> since 2022-03-02T17:03:27Z

*Terms of use:*

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)



# From Objects To Agents

WOA 2021 – Miniscuola

---

Matteo Baldoni

1 settembre 2021

Università degli Studi di Torino - Dipartimento di Informatica

## Please, note that

some slides are partially based on the book “M. Wooldridge, *An Introduction to Multi-Agent Systems*, Wiley, 2009”. A special thanks to Alberto Martelli, Cristina Baroglio, Roberto Micalizio, and Stefano Tedeschi, all from Università degli Studi di Torino.

# Who am I?

## Matteo Baldoni

Associate Professor (INF/01)

Dipartimento di Informatica, Università degli Studi di Torino

In the current a.y. I am teaching: *Sviluppo di Applicazioni Software Modellazione di Dati e Processi Aziendali, Agenti Intelligenti, Informatica per le scienze naturali, and Modellazione di Processi Amministrativi e Compliance Normativa*

## Email

baldoni AT di DOT unito DOT it



# Who am I?

## Matteo Baldoni

Associate Professor (INF/01)

Dipartimento di Informatica, Università degli Studi di Torino

Recently, I co-organized the XIII and the XIX International Conference of the Italian Association for Artificial Intelligence (2013 and 2020), and the XXII International Conference on Principle and Practice of Multi-Agent Systems (2019).

## Email

baldoni AT di DOT unito DOT it



# Table of Contents

1. Why intelligent agents?
2. What is meant by intelligent agents?
3. Let me open a parenthesis...
4. Parenthesis closed
5. The Triangle of Computation
6. The Awakening of the Action Force: Norms
7. JaCaMo
8. Robustness for Multiagent Systems



## Why intelligent agents?

---



Five ongoing **trends** have marked the history of computing:

- **ubiquity**
- **interconnection**
- **intelligence**
- **delegation**
- **human-orientation**





- The continual reduction in cost of computing capability has made it possible to introduce processing power into places and devices that would have once been uneconomic
- As processing capability spreads, sophistication (and intelligence of a sort) becomes ubiquitous
- What could benefit from having a processor embedded in it?



- Computer systems today no longer stand alone, but are networked into large distributed systems
- The internet is an obvious example, but networking is spreading its ever-growing tentacles
- Since distributed and concurrent systems have become the norm, some researchers are putting forward theoretical models that portray computing as primarily a process of interaction

# Intelligence and Delegation



- The complexity of tasks that we are capable of automating and delegating to computers has grown steadily
- Computers are doing more for us, without our intervention
- We are giving control to computers, even in safety critical tasks. For example:
  - *fly-by-wire aircraft*, where the machine's judgment may be trusted more than an experienced pilot
  - *fly-by-wire cars*, intelligent braking systems, cruise control that maintains distance from car in front



- The movement away from **machine-oriented** views of programming toward concepts and metaphors that more closely reflect the way we ourselves understand the world
- Programmers (and users) relate to the machine differently
- Programmers conceptualize and implement software in terms of higher-level — more human-oriented – **abstractions**

# Where does it bring us?

Delegation and Intelligence imply the need to build computer systems that can act effectively on our behalf.

This implies:

- The ability of computer systems to act **independently**
- The ability of computer systems to act in a way that **represents our best interests** while interacting with other humans or systems

# Cooperation and agreements



- Interconnection and distribution have become core motifs in computer science
- **But** interconnection and distribution, coupled with the need for systems to represent our best interests, implies systems that can **cooperate** and **reach agreements** (or even **compete**) with other systems that have different interests (much as we do with other people)

# We need to study autonomous agents and multiagent systems

All of these trends have led to the emergence of a new field in Computer Science: **autonomous agents (AA)** and **multiagent systems (MAS)**

In AAMAS, we address questions such as:

- How can cooperation emerge in societies of **self-interested** agents?
- What kinds of languages can agents use to communicate?
- How can self-interested agents **recognize conflict**, and how can they (nevertheless) **reach agreement**?
- How can autonomous agents coordinate their activities so as to cooperatively achieve goals?

## An agent

is a computer system that is capable of independent action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)

## Agent Design

How do we build agents capable of independent, autonomous action, so that they can successfully carry out tasks we delegate to them?



# Multiagent Systems, a definition

## A multiagent system (MAS)

is one that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to **cooperate**, **coordinate**, and **negotiate** with each other, much as people do.

## Society Design

How do we build agents that are capable of interacting (cooperating, coordinating, negotiating) with other agents in order to successfully carry out those delegated tasks, especially when the other agents cannot be assumed to share the same interests/ goals?

# Agents and Multiagent Systems: an interdisciplinary field

- The field of Multiagent Systems is influenced and inspired by many other fields:
  - Economics
  - Philosophy
  - Game Theory
  - Logic
  - Ecology
  - Social Sciences
- This can be both a strength (infusing well-founded methodologies into the field) and a weakness (there are many different views as to what the field is about)
- This has analogies with artificial intelligence itself



## Isn't it all just AI?

- We don't need to solve all the problems of artificial intelligence (i.e., all the components of intelligence) in order to build really useful agents
- **Classical AI ignored social aspects of agency.** These are important parts of intelligent activity in real-world settings



## Isn't it all just Economics/Game Theory?

- These fields also have a lot to teach us in multiagent systems, but:
  - Insofar as game theory provides descriptive concepts, it doesn't always tell us how to compute solutions; we're concerned with computational, resource-bounded agents
  - Some assumptions in economics/game theory (such as a rational agent) may not be valid or useful in building artificial agents



## Isn't it all just Social Science?

- We can draw insights from the study of human societies, but there is no particular reason to believe that artificial societies will be constructed in the same way
- Again, we have inspiration and cross-fertilization, but hardly subsumption



**What is meant by intelligent agents?**

---

# What is an agent?

- The main point about agents is they are **autonomous**: capable of acting independently, exhibiting control over their internal state

## Thus

an agent is a computer system capable of **autonomous** action in some **environment** in order to meet its design **objectives**

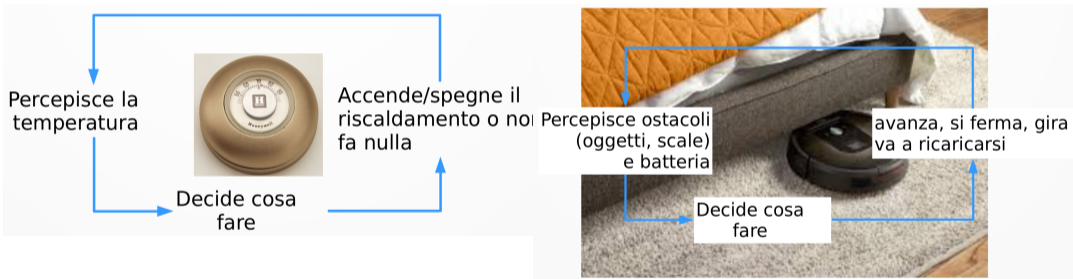


# What is an agent?





# What is an agent?



# What is an intelligent agent?

Autonomy and controllability:

- Outside the AI/AAMAS community, intelligent autonomous agents are perceived as: **self-aware**, **uncontrollable** entities whose autonomy emerges as an “*extra-program*” property
- An agent can take steps that are not included since from the beginning in its program



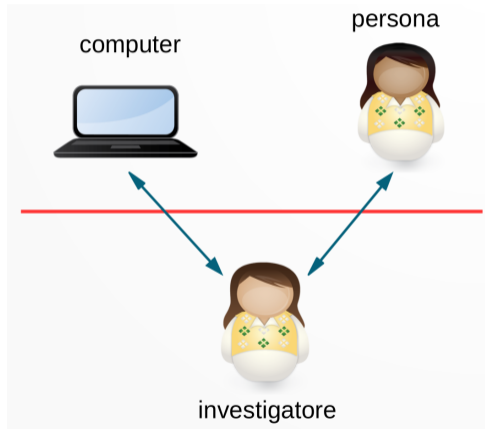


**Let me open a parenthesis...**

---

# Turing's Test (1950)

- A person or a computer are hidden from an investigator
- Interaction:
  - The investigator asks (written) questions
  - The hidden entity provides (written) responses
  - Will the investigator understand if on the other part there is a person or a computer?



# Intelligence is understanding

Is it enough to produce the expected outputs to say that there is understanding?

XYZ ?



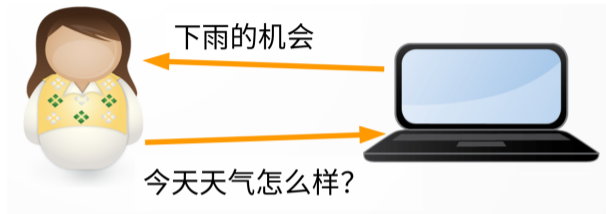
ABC !

XYZ ?



ABC !

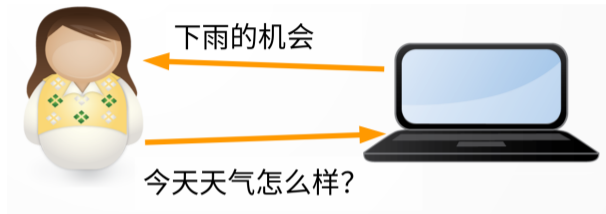
## The Chinese room (J. Searle)



- A computer, it is programmed to respond with certain Chinese characters to other Chinese characters received as input. The human interlocutor speaking in Chinese does not see the computer that is locked in a room. What will he think of who is in the room?<sup>1</sup>

<sup>1</sup>Searle, John. R. (1980) Minds, brains, and programs. Behavioral and Brain Sciences 3 (3): 417-457

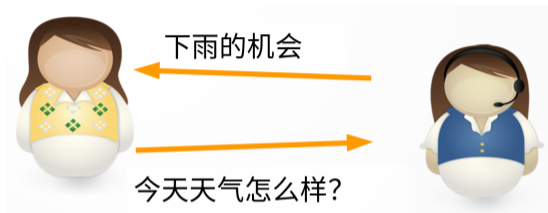
# The Chinese room (J. Searle)



- A computer, it is programmed to respond with certain Chinese characters to other Chinese characters received as input. The human interlocutor speaking in Chinese does not see the computer that is locked in a room. What will he think of who is in the room?<sup>1</sup>
- **Does he/she/it speak Chinese? Does he/she/it understand?**

<sup>1</sup>Searle, John. R. (1980) Minds, brains, and programs. Behavioral and Brain Sciences 3 (3): 417-457

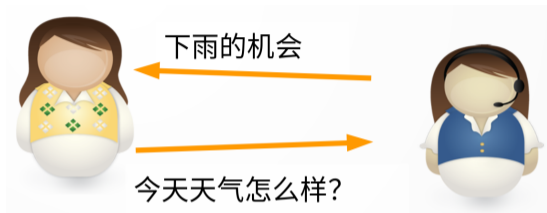
# The Chinese room (J. Searle)



- A person locked in the room has instructions for respond with certain Chinese characters in response to other Chinese ideograms



# The Chinese room (J. Searle)

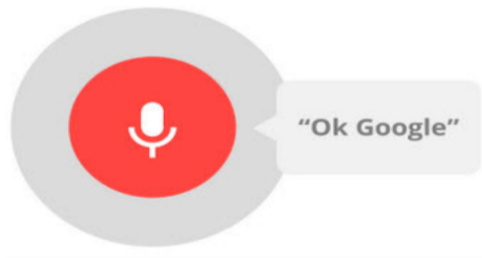


- A person locked in the room has instructions for respond with certain Chinese characters in response to other Chinese ideograms
- **Does he/she/it speak Chinese? Does he/she/it understand?** *"Instantiating a computer program is never by itself a sufficient condition of intentionality"*

An impressive application

Does he/she/it speak Italian/English/Chinese?

Does he/she/it understand?

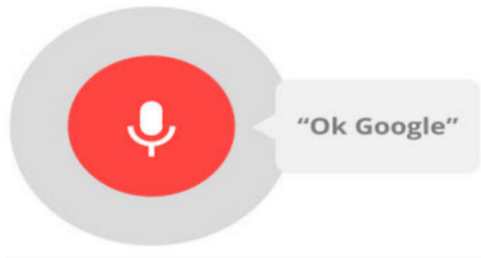


An impressive application

Does he/she/it speak Italian/English/Chinese?

Does he/she/it understand?

- It is a user interface
- It associates sounds with written words and uses these words as search keywords in the wide repertoire of documents accessible via the web
- The association is built by using statistical natural language processing techniques



# Strong and weak AI

- **Strong AI:** is it possible to reproduce human intelligence?  
Including self-awareness, sentience, . . .
- **Weak AI:** are there automatic ways to solve problems that require intelligence for a human being?  
Task-oriented, study of human thought and behavior

# What can a calculator do? Functions of the kind $f : \mathbb{N} \rightarrow \mathbb{N}$

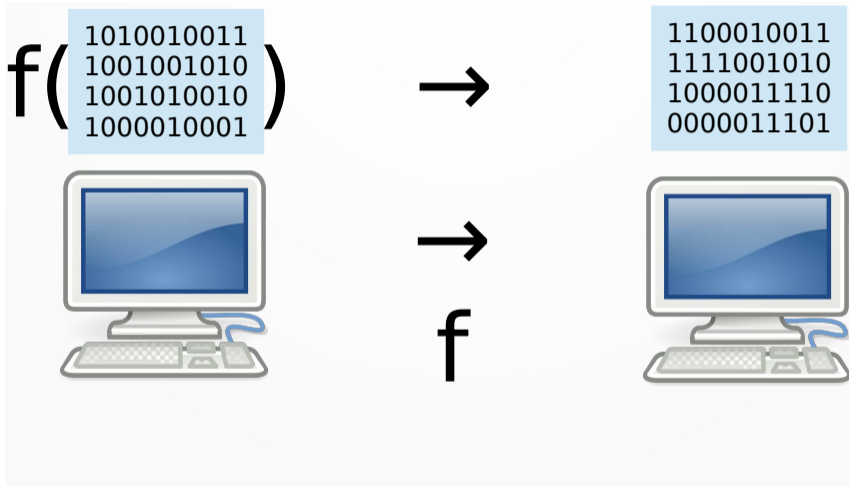
```
1010010011  
1001001010  
1001010010  
1000010001
```



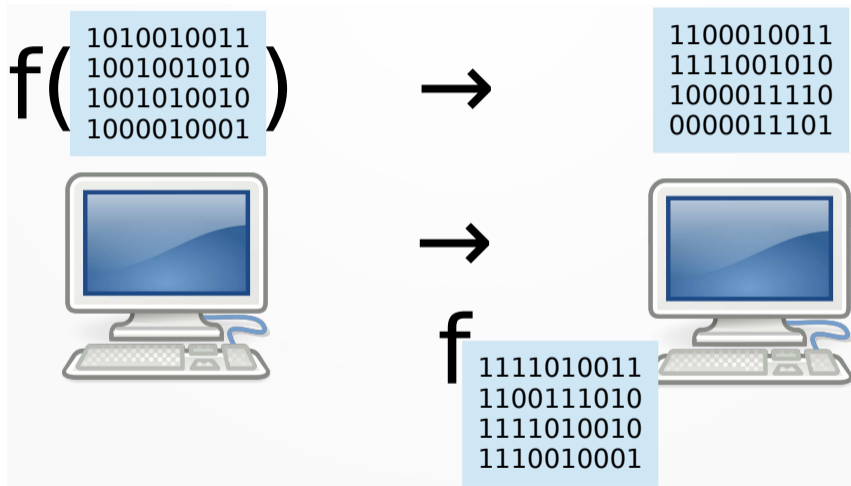
```
1100010011  
1111001010  
1000011110  
0000011101
```



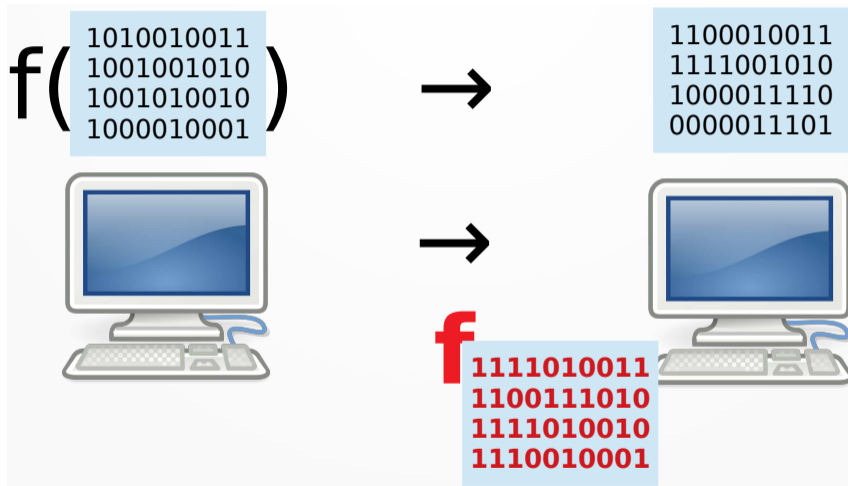
# What can a calculator do? Functions of the kind $f : \mathbb{N} \rightarrow \mathbb{N}$



# What can a calculator do? Functions of the kind $f : \mathbb{N} \rightarrow \mathbb{N}$



# What can a calculator do? Functions of the kind $f : \mathbb{N} \rightarrow \mathbb{N}$







**The software is neither “good” nor “bad” but it can be poorly programmed**



**Parenthesis closed**

---

# What is an intelligent agent?

Trivial (non-interesting) agents::

- thermostat
- UNIX daemon

## An intelligent agent

is a computer system capable of **flexible autonomous** action in some **environment**.

By **flexible**, we mean:

- **reactive**
- **pro-active**
- **social**



- If a program's environment is guaranteed to be fixed, the program need never worry about its own success or failure – program just executes blindly (eg. a compiler)
- The real world is not like that: things change, information is incomplete. Many (most?) interesting environments are **dynamic**
- Software is hard to build for dynamic domains: program must take into account possibility of failure – ask itself whether it is worth executing!

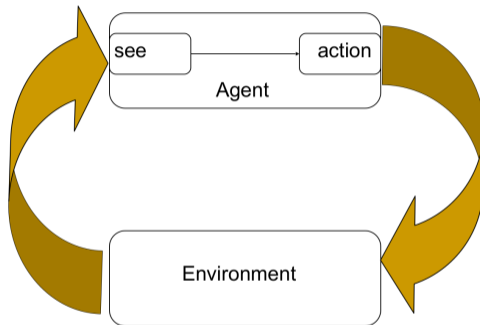
## A reactive agent

is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful).

A reactive agent can be implemented by simple conditional rules such as:

**“if car-in-front-is-braking then  
initiate-braking”**

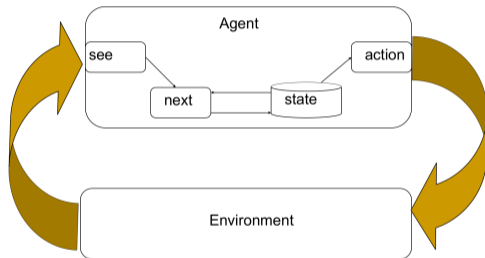
If the agent perceives that the car in front is braking, it initiates immediately to brake to avoid collision, without wasting time to reason.



- Reacting to an environment is easy (eg.: event/stimulus → response rules)
- But, we generally want agents to do “*things for us*”
- Hence **goal directed behavior**
- **proactiveness**: generating and attempting to achieve goals; not driven solely by events; **taking the initiative**
- **Recognizing opportunities**

## Proactive agents maintain a state.

- The state of the agent can contain two kinds of knowledge:
  - The agent needs some information about how the environment evolves
  - The agent needs some information about how the agent's own actions affect the world
- The agent needs some sort of **goal** information, so that it can act accordingly to fulfil the goal. To do this, the agent must be able to reason about **plans** to achieve its goals.



## Reactivity VS Proactiveness

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
  - We want our agents to systematically work towards long-term goals
- 
- These two considerations can be at odds with one another
  - Designing an agent that can balance the two remains an open research problem





- The real world is a **multi-agent environment**: we cannot go around attempting to achieve goals without taking others into account
- Some goals can only be achieved with the cooperation of others
- Similarly for many computer environments

- The real world is a **multi-agent environment**: we cannot go around attempting to achieve goals without taking others into account
- Some goals can only be achieved with the cooperation of others
- Similarly for many computer environments

## Social ability

in agents is the **ability to interact** with other agents (and possibly humans) via some kind of **agent-communication language (ACL)**, and perhaps cooperate with others.

# Is it an Agent, or just a Program?

Franklin e Graesser, “Is it an Agent or just a Program?”, 1996.

An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

As an extreme case, humans are autonomous agents according to this definition.

At the other extreme we have very simple structures such as a thermostat.

A payroll program senses the world through input and output. It is **not an agent** because his output has no effect on what it later perceives. Furthermore, there is no “over time” continuity.



# The Triangle of Computation

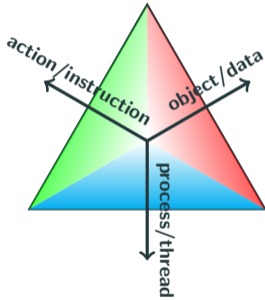
---



- Software Engineering aspires to *quality software*
  - *correctness*
  - *robustness*
  - *extensibility*
  - *reusability*
- A crucial ingredient to achieve these goals is a proper modularization of software architecture

# The Meyer's Triangle of Computation [Meyer, 1997]

- *“To execute a software system is to use certain processors to apply certain actions to certain objects.”*



- **process(or)**: a physical CPU, a process or a thread
- **action**: operations making up the computation (machine language operations up to subroutines)
- **object**: data structures to which the actions apply (computation-dependent or files, databases, etc.)

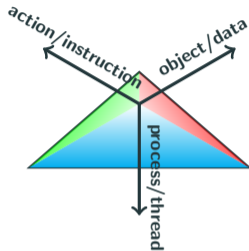
# The Meyer's Triangle of Computation [Meyer, 1997]

- *"To execute a software system is to use certain processors to apply certain actions to certain objects."*



- **process(or)**: a physical CPU, a process or a thread
  - **action**: operations making up the computation (machine language operations up to subroutines)
  - **object**: data structures to which the actions apply (computation-dependent or files, databases, etc.)
- Meyer's Triangle provides a common ground for comparing different approaches to modularization

# Functional Decomposition



*Puts at the center the process force*

## Pros

- simple and intuitive: builds a system by stepwise decomposition
- algorithmic-oriented: appropriate when a single top goal is specified

## Cons

- hardly maintainable: difficult to incorporate new “top goals”
- hardly scalable in presence of shared data and concurrent processes



# Object-Oriented Decomposition



*Puts at the center the object force*

## Pros

- objects have a life on their own, independent of the processes the use them
- *data operations*: provide the actions by which it is possible to operate on them
- object are the fundamental notion of the model

## Cons

- objects are passive: external processes take the decisions of what actions to invoke on objects
- no decoupling between the *use of an object and the management of that object*
- no conceptual support to the specification of tasks (processes)



- Actors have their own thread of computation
- The actor model does not properly address the **coordination** problem (although extensions have been proposed)
- In terms of Meyer's forces the actor model:
  - Supports **object/data management processes** (i.e., internal behaviors of actors)
  - *Does not* support the design and modularization of **processes using these objects** (i.e., processes external to actors)

- Create an **explicit representation of the activities of an organization** (e.g., an enterprise)
  - Describe how a set of interrelated activities lead to a **precise and measurable result** (a product or a service) **in response to an external, triggering event**
  - Emphasis on the **process** force: **activity-centric** view
    - *Prescriptive workflows*
    - *No proper abstractions for capturing data* manipulated along the flows
- ⇒ same limitations of the functional decomposition!

- Shifting from activity-centric to **data-centric** of business processes
- **Business Artifact (BA)**
  - is a **conceptual business entity** used in guiding the operations of a business (e.g., package delivery, patient visit, ...)
  - includes:
    - **information model**: hold relevant data about the artifact as it moves through the workflow
    - **lifecycle model**: states through which data can evolve + transitions between states (triggered by task execution)
  - emphasis on **object-data** force:
    - design and modularization of processes operating on BA is not explicitly considered



The agent paradigm relies on two **first-class abstractions**:

- *agents* (process force)
- *environment* (data/object force)

*“The environment is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both interaction among agents and the access to resources”*

[Weyns et al., 2007]

**Agents** are:

- **autonomous**: deliberative cycle (sense-plan-act) aimed at reaching a goal (*proactiveness*)
- **situated**: sense and manipulate their **environment**

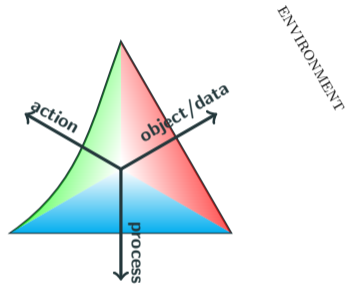
## Agents vs Objects

- objects do not have control over their own behavior (passive/reactive)
- objects do not exhibit a flexible behavior (no deliberative cycle)
- objects are single-threaded

## Agents vs Actors

- actors are not deliberative/proactive

# Agent-Oriented Paradigm



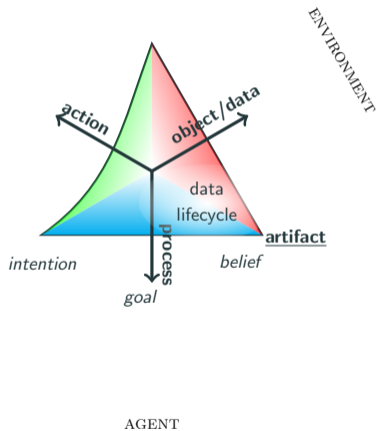
AGENT

# Agent-Oriented Paradigm

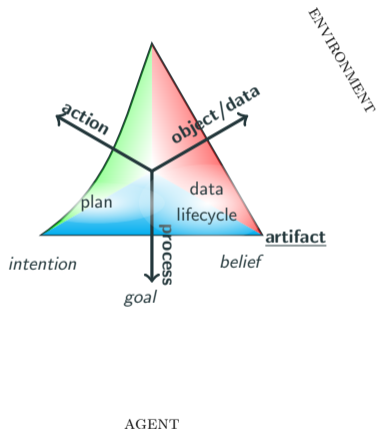




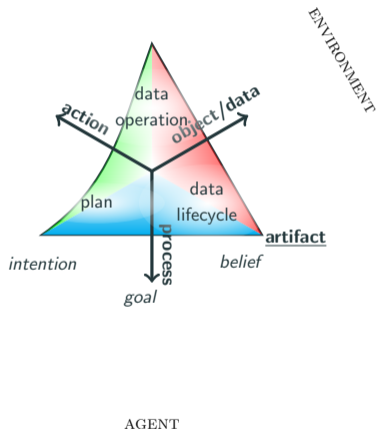
# Agent-Oriented Paradigm




# Agent-Oriented Paradigm



# Agent-Oriented Paradigm





## **The Awakening of the Action Force: Norms**

---



- **Norms condition the way in which processes operate on objects**
- Norms are about *“doing the right thing”*; processes are about *“doing what leads to a goal”* [Therborn, 2002]
- Relying on norms agents can reason about the social consequences of their actions

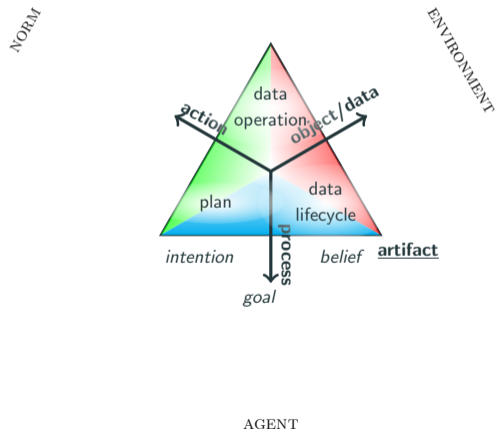
- **Norms condition the way in which processes operate on objects**
- Norms are about “*doing the right thing*”; processes are about “*doing what leads to a goal*” [Therborn, 2002]
- Relying on norms agents can reason about the social consequences of their actions

**Norms abstract the action force**

See [Baldoni et al., 2016a].

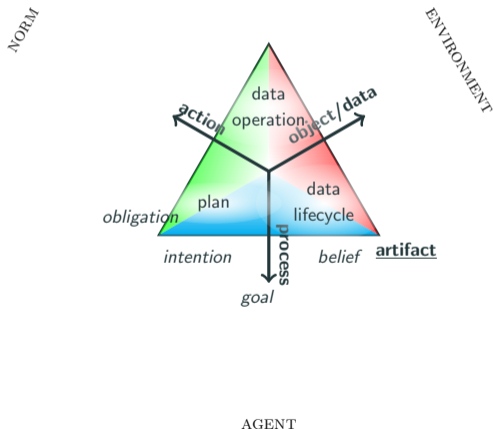
- **Regulative Norms** [Jones and Carmo, 2001]
  - specify patterns of **acceptable behavior** (i.e., actions and interactions) via obligations, permissions, prohibitions
- **Constitutive Norms** [Boella and van der Torre, 2004, Criado et al., 2014]
  - defines **institutional actions** that make sense only within the institution they belong to
  - specify the **applicability conditions** under which these actions can be used

# Revisiting the Meyer's forces [Baldoni et al., 2016a]

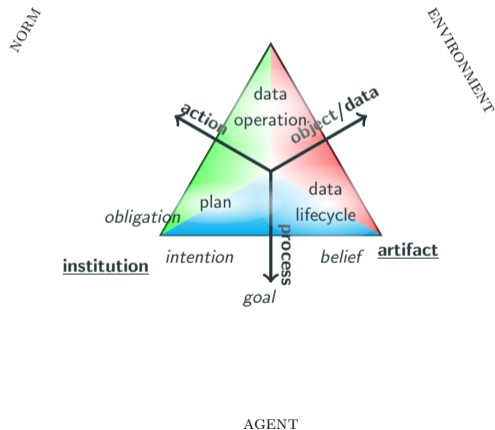




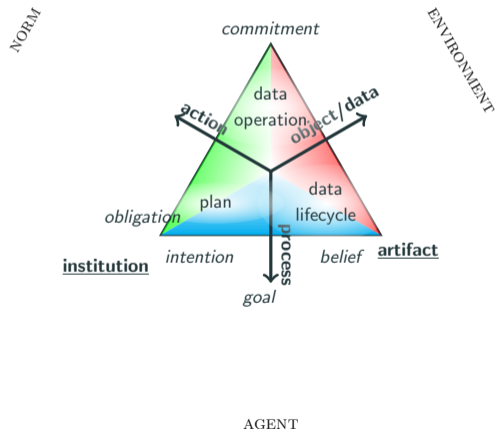
# Revisiting the Meyer's forces [Baldoni et al., 2016a]



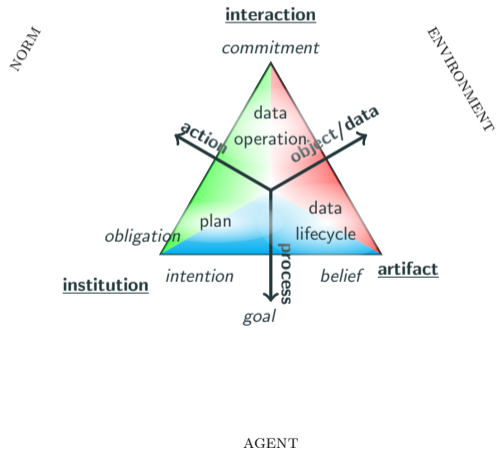
# Revisiting the Meyer's forces [Baldoni et al., 2016a]



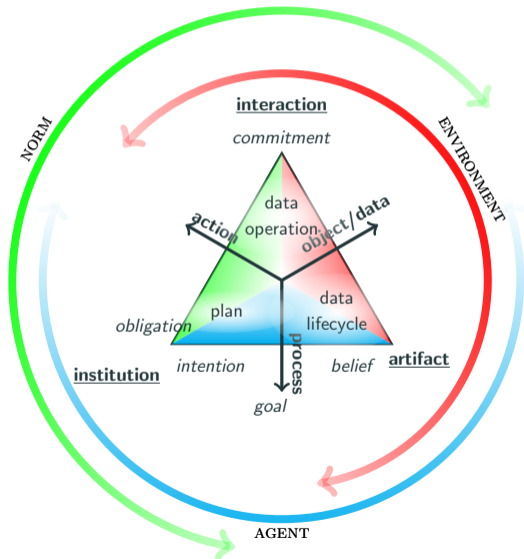
# Revisiting the Meyer's forces [Baldoni et al., 2016a]



# Revisiting the Meyer's forces [Baldoni et al., 2016a]



# Revisiting the Meyer's forces [Baldoni et al., 2016a]





**JaCaMo**

---

# JaCaMo: a Multi-Agent Oriented Programming platform

- JaCaMo is a Multi-Agent Oriented Programming (MAOP) platform
- it aims at programming systems by providing a seamless integration of three dimensions:

- **Agent** via **Jason**  
[Bordini et al., 2007],
- **Environment** via **CARTaGO**  
[Ricci et al., 2009],
- **Organization** via **Moise**  
[Hübner et al., 2010]

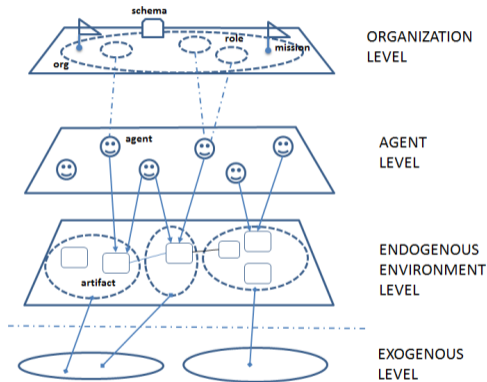
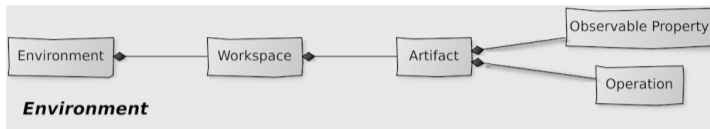


Figure 1: MAOP levels [Boissier et al., 2019]

# Environment dimension (from [Boissier et al., 2019])



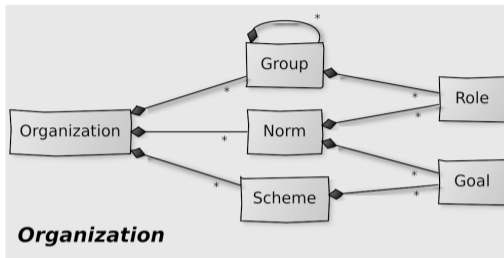
Simplified conceptual view (A&A meta-model [Omicini et al., 2008])

Simple artifact  
program:

```
public class Counter extends Artifact {  
    void init(int initialValue) {  
        defineObsProperty("count", initialValue);  
    }  
  
    @OPERATION void inc() {  
        ObsProperty prop = getObsProperty("count");  
        prop.updateValue(prop.intValue()+1);  
    }  
}
```



# Organization dimension (from [Boissier et al., 2019])



Simplified Conceptual View (Moise meta-model [Hübner et al., 2009])

Excerpts from organisation program:

```
<structural-specification>
<role-definitions>
<role id="auctioneer" />
<role id="participant" />
</role-definitions>
<group-specification id="auctionGroup">
<roles>
<role id="auctioneer" min="1" max="1"/>
<role id="participant" min="0" max="300"/>
</roles>
</group-specification>
</structural-specification>
```

Structural spec.

```
<functional-specification>
<scheme id="doAuction">
<goal id="auction">
<argument id="Id" />
<argument id="Service" />
<plan operator="sequence">
<goal id="start" />
<goal id="bid" ttf="10 seconds" />
<goal id="decide" ttf="1 hour" />
</plan>
</goal>
<mission id="mAuctioneer" min="1" max="1">
<goal id="start" />
<goal id="decide" />
</mission>
```

Functional spec.

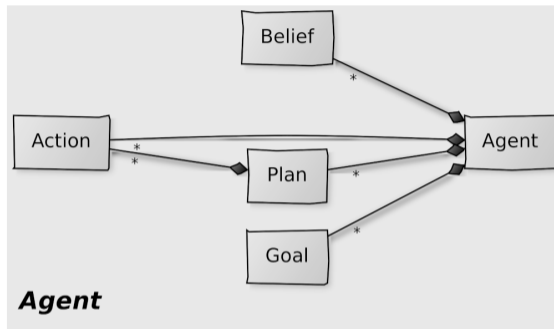
```
<normative-specification>
<norm id="n1" type="permission"
role="auctioneer"
mission="mAuctioneer" />
<norm id="n2" type="obligation"
role="participant"
mission="mParticipant" />
</normative-specification>
```

Normative spec.

```
norm n1 : plays(A, auctioneer, G) ->
forbidden(A, n1, plays(A, participant, G),
'forever').
```

program in NPL

# Agent dimension (from [Boissier et al., 2019])



Simplified Conceptual View (Jason meta-model [Bordini et al., 2007]):

Simple Agent Program:

```
happy(bob). // initial belief
!say(hello). // initial goal
/* Plans */
+!say(X) : happy(bob) <- .print(X).
// ...
```

example bob.asl

```
+happy(A) <- !say(hello(A)).
+!say(A) : not today(friday) <- .print(X); !say(X).
+!say(X) : today(friday) <- .print("stop").
-happy(A) : .my_name(A) <- .drop_intention(say(_)).
```

example carl.asl

# Agent's plans for obligations

## Obligation to commit to a mission

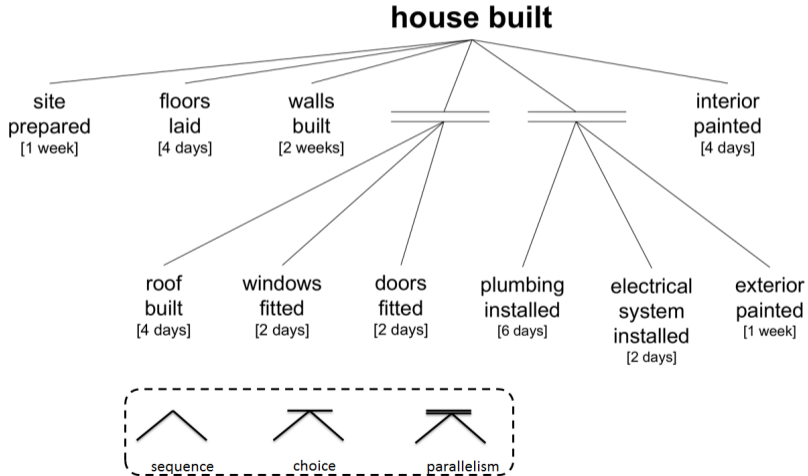
```
+obligation(Ag, Norm, committed(Ag, Mission, Scheme), Deadline)
  : .my_name(Ag)
  <- .print("I am obliged to commit to ", Mission);
     commit_mission(Mission, Scheme).
```

## Obligation to a goal

```
+obligation(Ag, Norm, achieved(Sch, Goal, Ag), Deadline)
  : .my_name(Ag)
  <- .print("I am obliged to achieve goal ", Goal);
     !Goal[scheme(Sch)];
     goal_achieved(Goal, Sch).
```

# The building house example

## Functional decomposition





# Robustness for Multiagent Systems

---

# Robustness: an important property of software systems

## Sys. and Soft. Eng. Vocabulary ISO/IEC/IEEE 24765

Robustness as the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

## Robustness refers to a system property

A property of a system is robust if it is **invariant with respect to a set of perturbations** [Alderson and Doyle, 2010].

- **reliability** as robustness to component failure
- **efficiency** as robustness to lack of resources
- **scalability** as robustness to change to the size and complexity of the system as a whole
- **modularity** as robustness to structured component rearrangements
- **evolvability** as robustness of lineages to changes on long time scales

The availability of feedback is seen as crucial in gaining robustness [Alderson and Doyle, 2010].

## Feedback

A piece of **information**, some facts that are obtained **retroactively**, that **objectively concern** an execution of interest, and **that are passed** from one component to another.

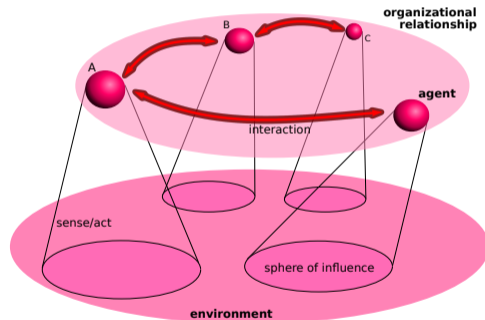
## Significance and quality of feedback

are crucial in making a system robust: [Alderson and Doyle, 2010].

- only information that is **functional** to the desired kind of robustness
- only information that comes from **reliable source**

# Multiagent Organizations

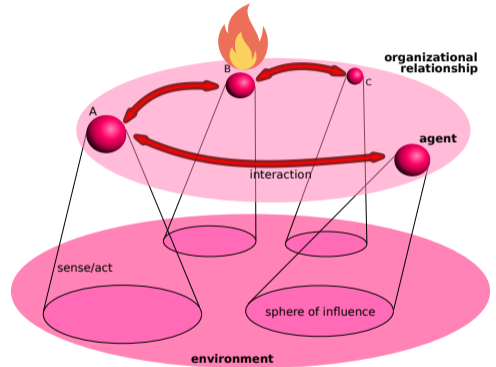
- “An organization provides a structure of constraints that allow a system consisting of many parts to act as a whole, with the aim of achieving goals that otherwise would not be achievable (or not as easily)” [Elder-Vass, 2011]
- **Norms** (rules, protocols, etc.) to define what is expected of each agent
- **Sanctions** as deterrents **to prevent** norm violation





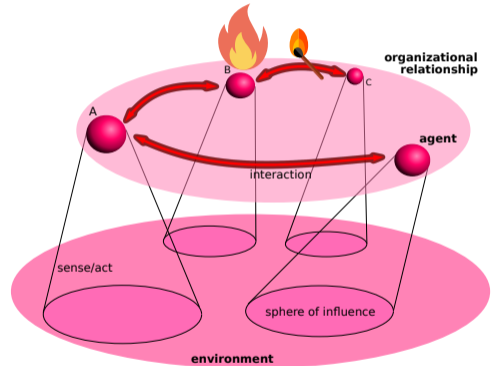
# Perturbations

- Unfortunately, when the system faces an abnormal situation (**perturbation**) and some agent *fails* to achieve a goal, **sanctions are of little utility**, if any



# Perturbations

- Unfortunately, when the system faces an abnormal situation (**perturbation**) and some agent *fails* to achieve a goal, **sanctions are of little utility**, if any
- The agent may have tried its best to do what expected, but something which is not under its control might hinder the achievement

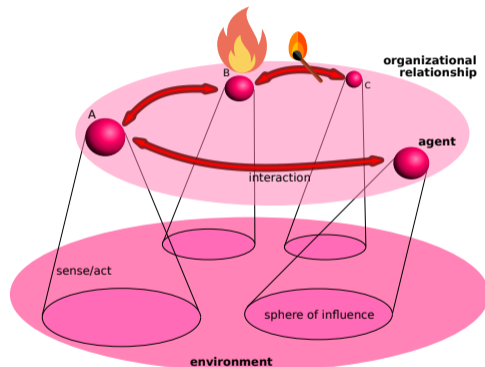


# Perturbations

- Unfortunately, when the system faces an abnormal situation (**perturbation**) and some agent *fails* to achieve a goal, **sanctions are of little utility**, if any
- The agent may have tried its best to do what expected, but something which is not under its control might hinder the achievement

## Broader problem

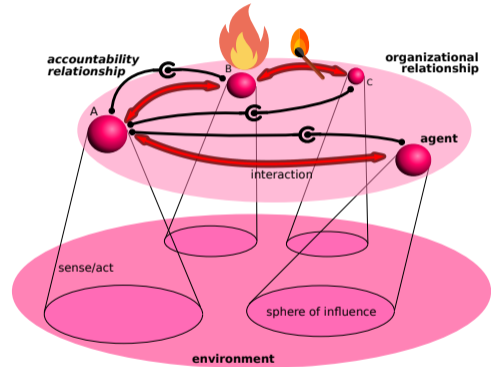
No structured way for collecting and propagating information about encountered situations



# Robustness

When a system meets a perturbation it needs **to reconfigure**. To this aim:

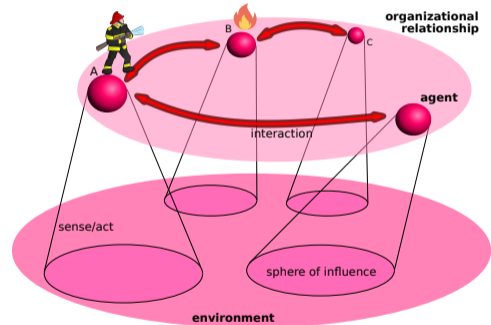
1. the agents need the means for:  
understanding who is entitled to ask what  
to whom
2. the information of interest must be asked  
to an informed source and must be  
delivered in the right format
3. the information will be delivered to whom  
is equipped with the right abilities and will  
be entitled to perform certain tasks,  
needed to cope with the situation



# Robustness

When a system meets a perturbation it needs **to reconfigure**. To this aim:

1. the agents need the means for:  
understanding who is entitled to ask what  
to whom
2. the information of interest must be asked  
to an informed source and must be  
delivered in the right format
3. the information will be delivered to whom  
is equipped with the right abilities and will  
be entitled to perform certain tasks,  
needed to cope with the situation



- The **agents' autonomy** is an enabler of the system's adaptability, which is crucial to achieve robustness
  - However, adaptability requires the system to be equipped with the ability to produce proper feedback, propagate it, and process it, so to enable the selection and enactment of behavior that is appropriate to cope with the situation
- The **normative system** enables the exploitation of the agents' autonomy, creating expectations on their activities, which is crucial to achieve system robustness
  - However, agents may fail the expectations (the obligations). Whenever sanctions are not accompanied by feedback and feedback handling mechanisms, they do not provide a means that support robustness

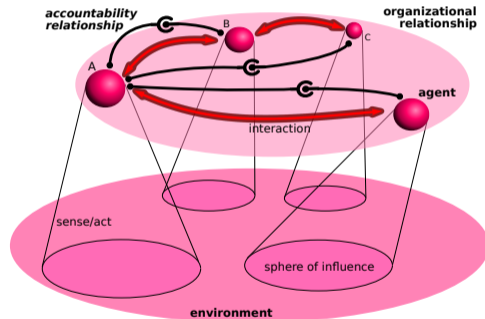
- The current design methodologies for MAS fall short in addressing robustness in a systematic way at design time.

## Accountability

We exploit the notion of **accountability** [Garfinkel, 1967, Grant and Keohane, 2005, Dubnick and Justice, 2004, Baldoni et al., 2016b, Baldoni et al., 2019] as a mechanism for building feedback/reporting frameworks, similarly to what is often done in human organizations [Sustainable Energy for All Initiative, , Zahran, 2011].

# Accountability as a means for robustness in MAS

- We claim that **account** and **accountability** are the crucial tools for making organizations more **robust**
- In the **human world/organizations** accountability it provides the means to address recurring and systemic issues, and to incorporate lessons learned into future activities





A relationship between two parties:

- One of the parties (the “**account taker**” or *a-taker*) can legitimately ask, under some agreed conditions, to the other party an **account** about a process of interest
- the other party (the “**account giver**” or *a-giver*) is legitimately required to provide the account to the a-taker

## The two dimensions of accountability

1. **Normative dimension** → Legitimacy of asking and availability to provide accounts
2. **Structural dimension** → For being accountable about a process, an agent must have control over that process and have awareness of the situation it will account for

# Extending JaCaMo for Accountability

- Changes mostly concern the **organization specification**; i.e., the Moise component
- Changes are **as conservative as** possible (when no accountability relationship is specified, we fall back to standard JaCaMo)
- Changes satisfy three needs:
  - **specify accountability agreements** within an organization
  - translate accountability agreements into a **corresponding body of norms**
  - give agents the capability of **producing accounts** and **marking goals** not only as *achieved*, but also as *failed* or *released*



## How does accountability come into play while programming agents?

Three basic programming patterns can be exploited for capturing a variety of situations

- Information gathering
- Context-aware adaptation
- Exception handling

## Exception

Event that causes suspension of normal program execution. [ISO/IEC/IEEE, 2010]

- The purpose of an exception handling mechanism is to:
  1. Identify when an **exception** (i.e., a *perturbation*) occurs
  2. Apply suitable **handlers**, capable of treating the exception and recover

Exception handling is a matter of **responsibility distribution**:

1. Always involves two parties: a party that is **responsible** for raising an exception, and another party that is **responsible** for handling it
2. Captures the need for some **information/account** from the former to the latter that allows coping with the exception

**Exception handling can be built upon accountability relationships**  
[Baldoni et al., 2021a, Baldoni et al., 2021b]

- Meyer's forces of computation have been revised under Agent-Oriented perspective
- Agent-Paradigm is still a promising approach to design and develop of complex software systems
- Integration of data and norm awareness might be the key to increase the appealing of industries
- JaCaMo+ [Baldoni et al., 2018a, Baldoni et al., 2018b] and JaCaMo for accountability and exception handling [Baldoni et al., 2021a, Baldoni et al., 2021b] is a first step along this path



Alderson, D. L. and Doyle, J. C. (2010).

**Contrasting views of complexity and their implications for network-centric infrastructures.**

*IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(4).



Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., and Tedeschi, S. (2021a).

**Distributing Responsibilities for Exception Handling in JaCaMo.**






In Emdriss, U., Nowé, A., Dignum, F., and Lomuscio, A., editors, *Proc. of 20th International Conference on Autonomous Agents and Multiagent Systems, Demonstration Track, AAMAS 2021*, Online. IFAAMAS, Richland, SC.




Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., and Tedeschi, S. (2021b).

**Exception Handling in Multiagent Organizations: Playing with JaCaMo.**

In Alechina, N., Baldoni, M., and Logan, B., editors, *Proc. of the 9th International Workshop on Engineering Multi-Agent Systems, EMAS 2021, held in conjunction with AAMAS 2021*, Online.

- 
-  Baldoni, M., Baroglio, C., Calvanese, D., Micalizio, R., and Montali, M. (2016a).  
**Towards Data- and Norm-aware Multiagent Systems.**  
In Baldoni, M., Müller, J. P., Nunes, I., and Zalila-Wenkstern, R., editors, *Post-Proc. of the 4th International Workshop on Engineering Multi-Agent Systems, EMAS 2016, Revised Selected and Invited Papers*, number 10093 in LNAI, pages 22–38. Springer.
-  Baldoni, M., Baroglio, C., Capuzzimati, F., and Micalizio, R. (2018a).  
**Commitment-based Agent Interaction in JaCaMo+.**  
*Fundamenta Informaticae*, 159(1-2):1–33.
-  Baldoni, M., Baroglio, C., Capuzzimati, F., and Micalizio, R. (2018b).  
**Type Checking for Protocol Role Enactments via Commitments.**  
*Journal of Autonomous Agents and Multi-Agent Systems*, 32(3):349–386.
-  Baldoni, M., Baroglio, C., May, K. M., Micalizio, R., and Tedeschi, S. (2016b).  
**Computational Accountability.**






In Chesani, F., Mello, P., and Milano, M., editors, *Deep Understanding and Reasoning: A challenge for Next-generation Intelligent Agents, URANIA 2016*, volume 1802, pages 56–62, Genoa, Italy. CEUR, Workshop Proceedings.

 Baldoni, M., Baroglio, C., May, K. M., Micalizio, R., and Tedeschi, S. (2019).  
**MOCA: An ORM MOdel for Computational Accountability.**







*Journal of Intelligenza Artificiale*, 13(1):5–20.

 Baldoni, M., Baroglio, C., and Micalizio, R. (2020).  
**Fragility and Robustness in Multiagent Systems.**

In Baroglio, C., Hubner, J. F., and Winikoff, M., editors, *Post-Proc. of the 8th International Workshop on Engineering Multi-Agent Systems, EMAS 2020, Revised Selected Papers*, number 12589 in LNAI, pages 61–77, Auckland, New Zealand. Springer.






 Baldoni, M., Baroglio, C., Micalizio, R., and Tedeschi, S. (2021c).  
**Robustness based on Accountability in Multiagent Organizations.**

In Emriss, U., Nowé, A., Dignum, F., and Lomuscio, A., editors, *Proc. of 20th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2021*, pages 142–150, Online. IFAAMAS, Richland, SC.

- 
-  Boella, G. and van der Torre, L. W. N. (2004).  
**Regulative and constitutive norms in normative multiagent systems.**  
In Dubois, D., Welty, C. A., and Williams, M., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, Whistler, Canada, June 2-5, 2004, pages 255–266. AAAI Press.
-  Boissier, O., Bordini, R., J.F., H., and Ricci, A. (2019).  
**Multi-Agent Oriented Programming - A short and practical introduction to the JaCaMo platform.**
-  Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007).  
**Programming multi-agent systems in AgentSpeak using Jason.**  
John Wiley & Sons.
-  Bovens, M. (2010).  
**Two concepts of accountability: Accountability as a virtue and as a mechanism.**  
*West European Politics*, 33(5):946–967.
-  Criado, N., Argente, E., Noriega, P., and Botti, V. (2014).





**Reasoning about constitutive norms in BDI agents.**

*Logic Journal of IGPL*, 22(1):66–93.

-  Dubnick, M. J. (2014).  
**Accountability as a Cultural Keyword, pages 23–38.**  
Oxford University Press.
-  Dubnick, M. J. and Justice, J. B. (2004).  
**Accounting for accountability.**  
Annual Meeting of the American Political Science Association.
-  Elder-Vass, D. (2011).  
**The Causal Power of Social Structures: Emergence, Structure and Agency.**  
Cambridge University Press.
-  Garfinkel, H. (1967).  
**Studies in ethnomethodology.**  
Prentice-Hall Inc., Englewood Cliffs, New Jersey.
-  Grant, R. W. and Keohane, R. O. (2005).

## **Accountability and Abuses of Power in World Politics.**

*The American Political Science Review*, 99(1).

-  Hübner, J. F., Boissier, O., Kitio, R., and Ricci, A. (2010).  
**Instrumenting multi-agent organisations with organisational artifacts and agents.**  
*Auton. Agents Multi Agent Syst.*, 20(3):369–400.
-  ISO/IEC/IEEE (2010).  
**ISO/IEC/IEEE International Standard - Systems and software engineering – Vocabulary.**  
*ISO/IEC/IEEE 24765:2010(E)*, pages 1–418.
-  Jones, A. J. and Carmo, J. (2001).  
**Deontic logic and contrary-to-duties.**  
In Gabbay, D., editor, *Handbook of Philosophical Logic*, page 203–279. Kluwer.
-  Meyer, B. (1997).  
**Object-oriented Software Construction (2Nd Ed.).**  
Prentice-Hall, Inc., Upper Saddle River, NJ, USA.



 Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009).

**Environment programming in cartago.**

In Bordini, R. H., Dastani, M., Dix, J., and Seghrouchni, A. E. F., editors, *Multi-Agent Programming, Languages, Tools and Applications*, pages 259–288. Springer.

 Sustainable Energy for All Initiative.

**Accountability framework.**

 Therborn, G. (2002).

**Back to norms! on the scope and dynamics of norms and normative action.**

*Current Sociology*, 50:863–880.


 Weyns, D., Omicini, A., and Odell, J. (2007).

**Environment as a first class abstraction in multiagent systems.**

*JAAMAS*, 14(1):5–30.

 Zahran, M. (2011).

**Accountability Frameworks in the United Nations System.**



[https://www.unjiu.org/sites/www.unjiu.org/files/jiu\\_document\\_files/products/en/reports-notes/JIU%20Products/JIU\\_REP\\_2011\\_5\\_English.pdf](https://www.unjiu.org/sites/www.unjiu.org/files/jiu_document_files/products/en/reports-notes/JIU%20Products/JIU_REP_2011_5_English.pdf)

UN Report.