# Engineering data-aware commitment-based multiagent systems

Alina Aleksandrova

Vienna University of Technology

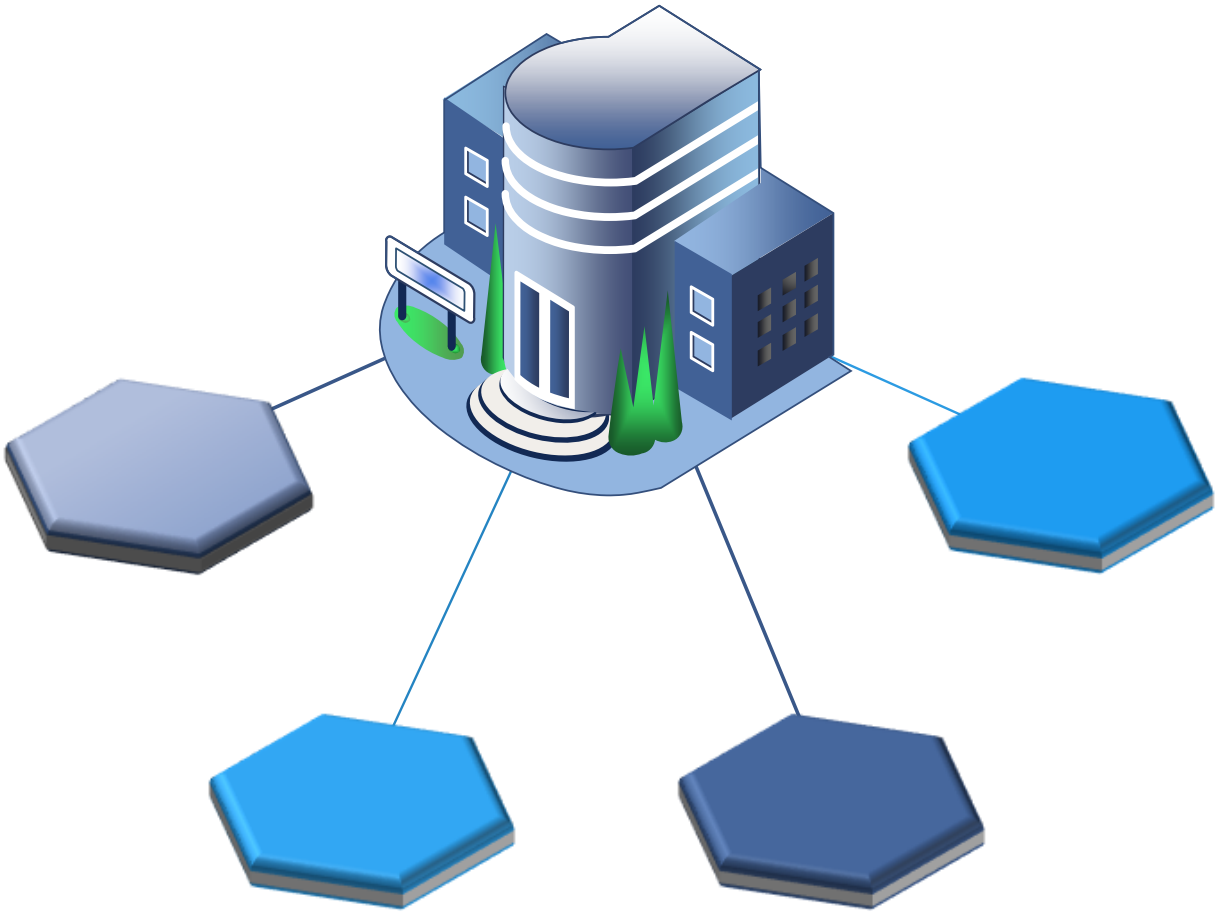
Supervisor:
Dr. Marco Montali

Co-supervisor:
Prof. Stefan Woltran

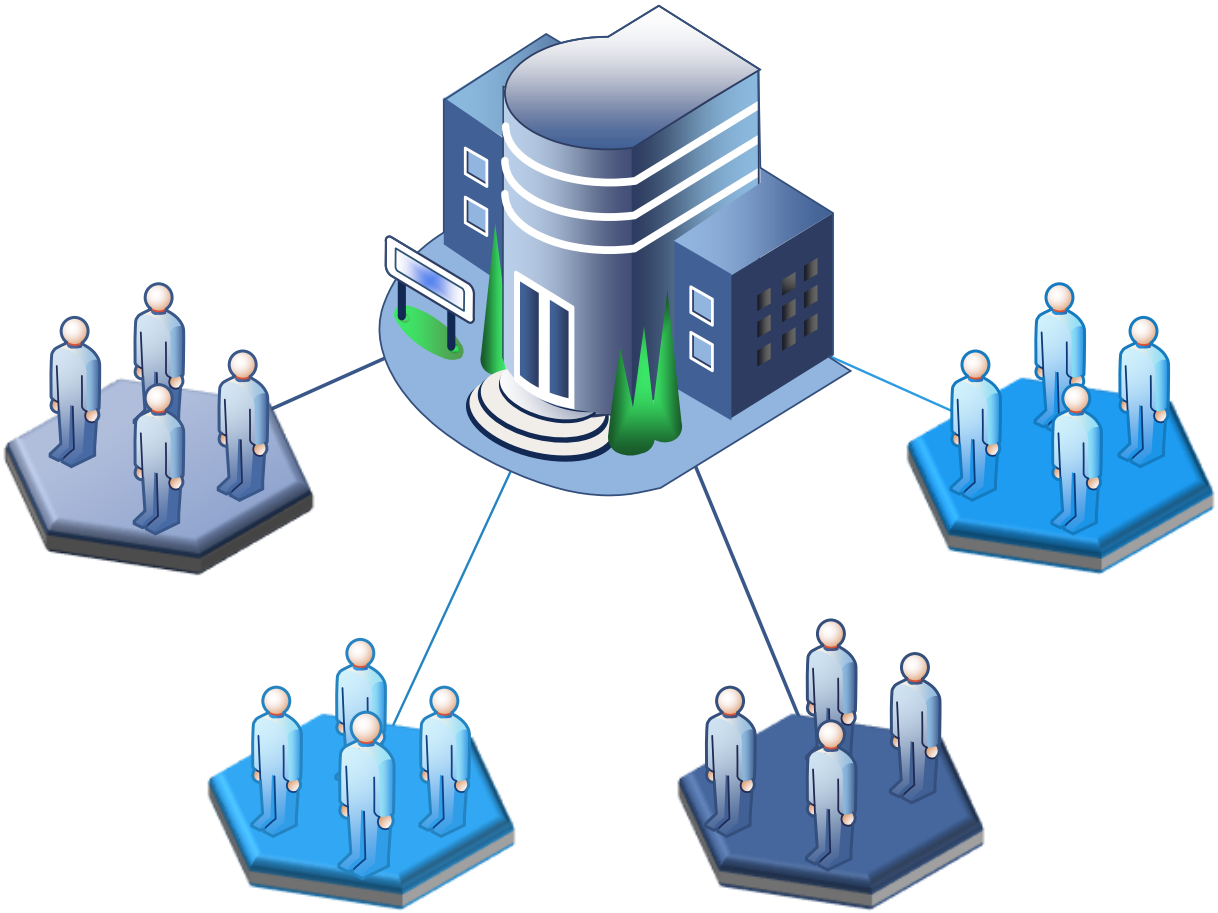
11th February, 2016

# Motivation

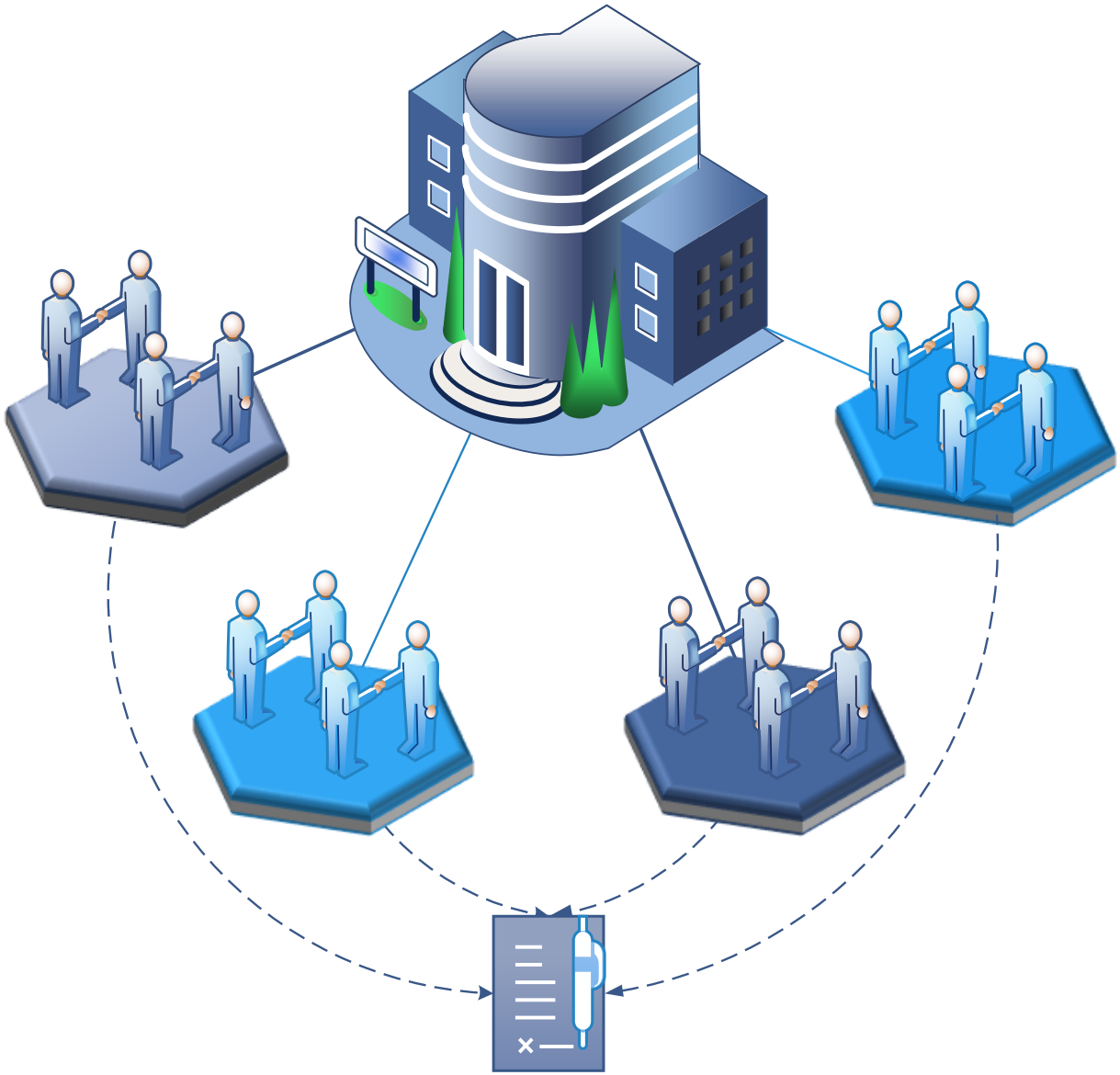
# Motivation



# Motivation



# Motivation



# Multiagent systems

According to M. Wooldridge:

## What is MAS?

"A multiagent system consists of a number of agents interacting with each other."

To successfully interact, agents will require the ability to *cooperate*, *coordinate* and *negotiate* with each other.

# Multiagent systems

According to M. Wooldridge:

## What is MAS?

"A multiagent system consists of a number of agents interacting with each other."

To successfully interact, agents will require the ability to *cooperate*, *coordinate* and *negotiate* with each other.

## Agent properties

- autonomy
  *Its behaviour is determined by its* **own experience**.

# Multiagent systems

According to M. Wooldridge:

## What is MAS?

"A multiagent system consists of a number of agents interacting with each other."

To successfully interact, agents will require the ability to *cooperate*, *coordinate* and *negotiate* with each other.

## Agent properties

- autonomy
- reactivity
  *It responds to changes in the* **environment**.

# Multiagent systems

According to M. Wooldridge:

## What is MAS?

"A multiagent system consists of a number of agents interacting with each other."

To successfully interact, agents will require the ability to *cooperate*, *coordinate* and *negotiate* with each other.

## Agent properties

- autonomy
- reactivity
- pro-activeness
  *It* **takes an initiative**; *recognizes opportunities.*

# Multiagent systems

According to M. Wooldridge:

## What is MAS?

"A multiagent system consists of a number of agents interacting with each other."

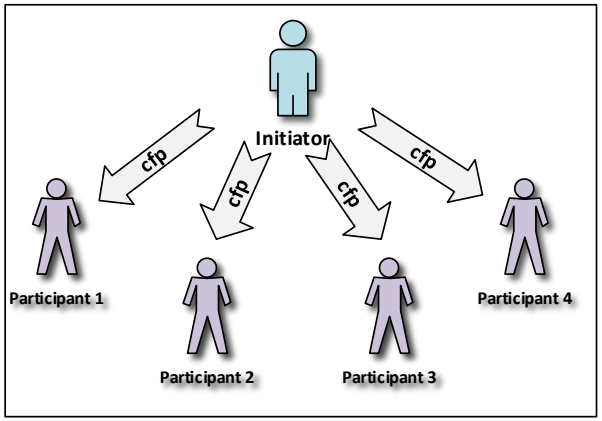
To successfully interact, agents will require the ability to *cooperate*, *coordinate* and *negotiate* with each other.

## Agent properties

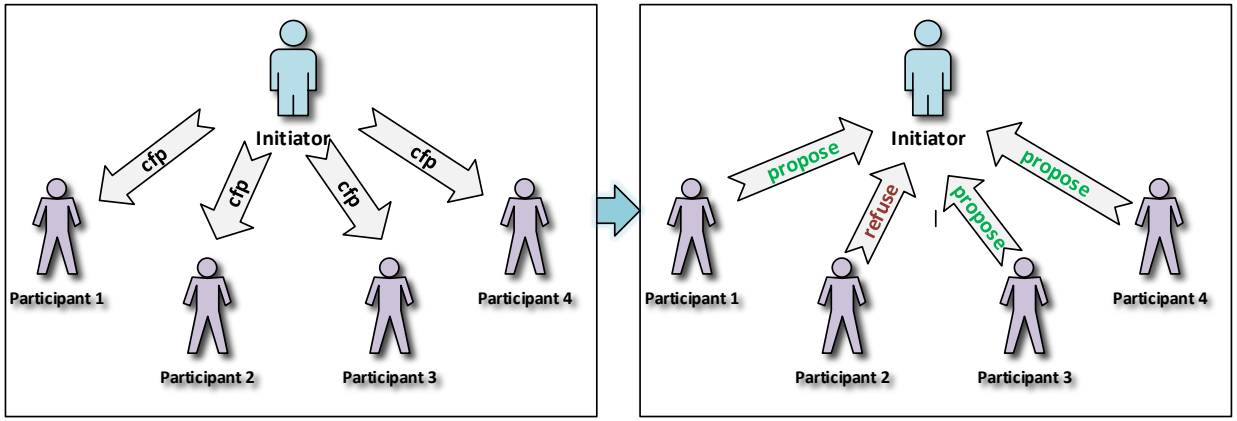
- autonomy
- reactivity
- pro-activeness
- social ability

  *It can* **interact** *with other agents.*

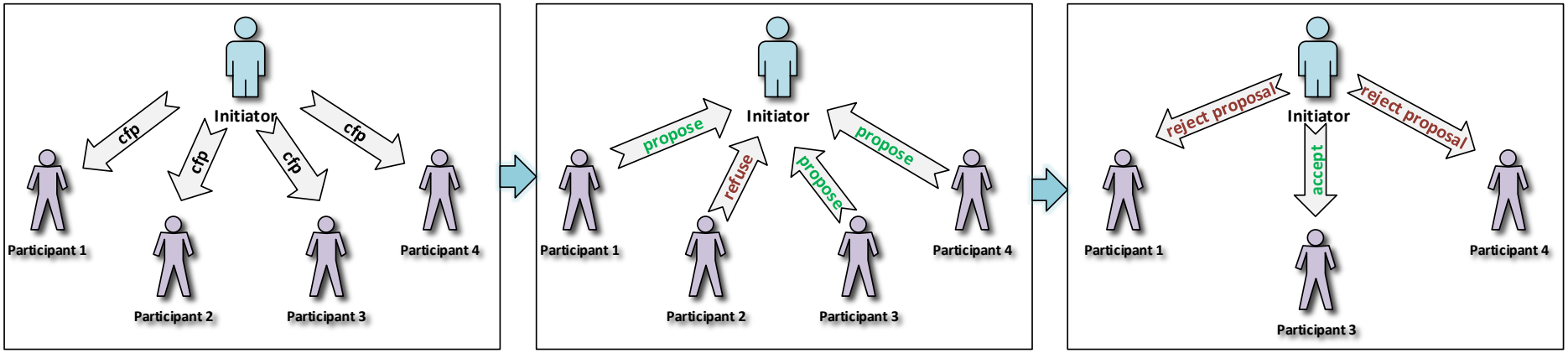
# Contract Net Protocol



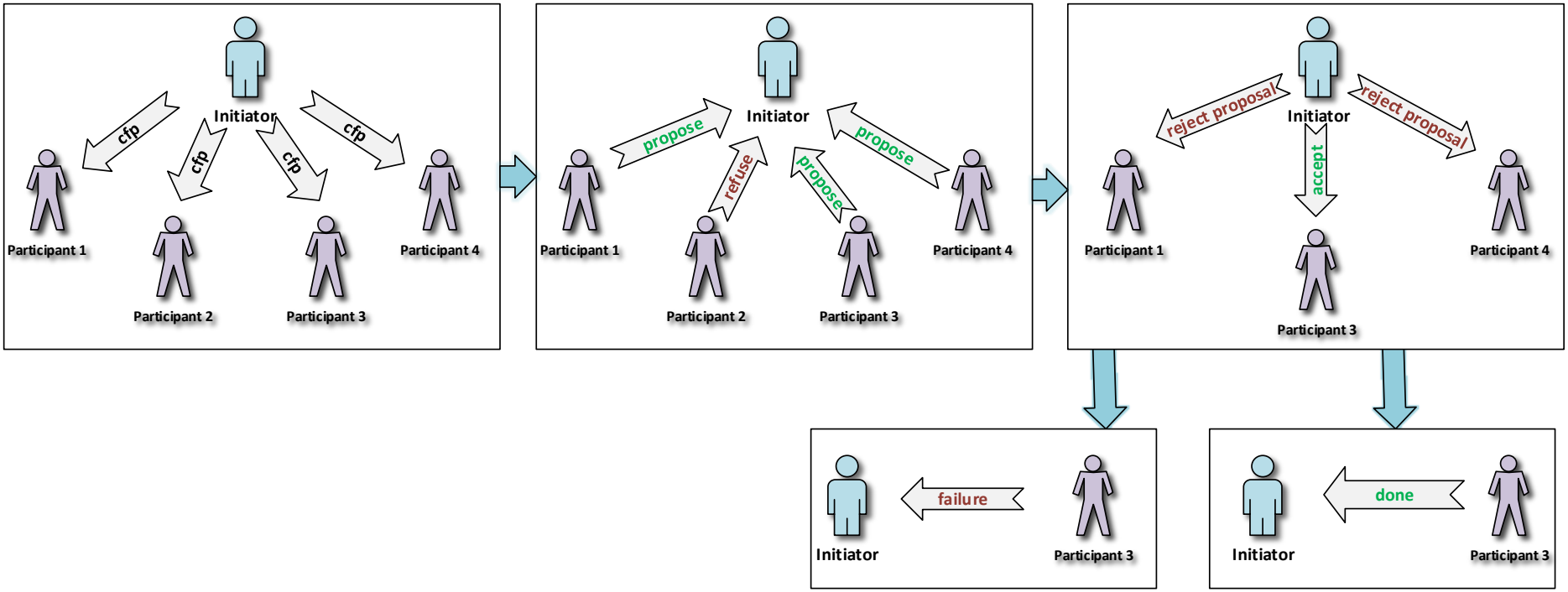
# Contract Net Protocol



# Contract Net Protocol



# Contract Net Protocol



# Data-Awareness in MAS

- All formal approaches to MAS miss *data perspective*.

# Data-Awareness in MAS

- All formal approaches to MAS miss *data perspective*.
- Many logics to reason about agent behaviours and interactions
  - **Exchanged information is neglected!**

# Data-Awareness in MAS

- All formal approaches to MAS miss *data perspective*.
- Many logics to reason about agent behaviours and interactions
  - **Exchanged information is neglected!**
- $\rightsquigarrow$ The model of the system is not in line with the implementation used at execution time.

# Relational MAS

- Each agent maintains its own relational database.

# Relational MAS

- Each agent maintains its own relational database.
- Agents behaviour is regulated by communicative and reactive rules.

# Relational MAS

- Each agent maintains its own relational database.
- Agents behaviour is regulated by communicative and reactive rules.

## Communicative rules

$$Q(t, \vec{x}) \textbf{ enables } \mathtt{EV}(\vec{x}) \textbf{ to } t$$

1. Determine which messages with payload can be sent.
2. Agent autonomously chooses which message to send.

# Relational MAS

- Each agent maintains its own relational database.
- Agents behaviour is regulated by communicative and reactive rules.

## Reactive rules

- **on** $\texttt{EV}(\vec{x})$ **to** $t$ **if** $Q(\vec{y_1})$ **then** $\alpha(\vec{y_2})$ *(on-send)*
- **on** $\texttt{EV}(\vec{x})$ **from** $s$ **if** $Q(\vec{y_1})$ **then** $\alpha(\vec{y_2})$ *(on-receive)*

Agent conditionally reacts to an incoming/outgoing message by invoking an update action

# Relational MAS

- Each agent maintains its own relational database.
- Agents behaviour is regulated by communicative and reactive rules.

## Update actions

$\alpha(\vec{x}) : \{e_1, ..., e_n\}$. Each update effect $e$ has the form:

$$Q(\vec{p}, \vec{x}) \rightsquigarrow \textbf{add } A, \textbf{ del } D$$

- $A$ set of "add" facts, possibly including *service calls*;
- $D$ set of "delete" facts;

# Example - Contract Net Protocol

## Initiator

- $Task(t, \text{“}todo\text{”}) \wedge Agent(a) \wedge \Phi_{sui}(a, t) \wedge \neg Contacted(a, t)$ **enables** $cfp(t)$ **to** $a$

## Participant

# Example - Contract Net Protocol

## Initiator

- $Task(t, \text{“}todo\text{”}) \wedge Agent(a) \wedge \Phi_{sui}(a, t) \wedge \neg Contacted(a, t)$ **enables** $cfp(t)$ **to** $a$
- **on** $cfp(t)$ **to** $a$ **if** *true* **then** MARKCONTACTED$(a, t)$
  - MARKCONTACTED$(a, t)$ : $\{[\text{true}] \rightsquigarrow \textbf{add}\{\text{Contacted}(a, t)\}\}$

## Participant

# Example - Contract Net Protocol

## Initiator

- $Task(t, \text{“}todo\text{”}) \wedge Agent(a) \wedge \Phi_{sui}(a, t) \wedge \neg Contacted(a, t)$ **enables** $cfp(t)$ **to** $a$
- **on** $cfp(t)$ **to** $a$ **if** $true$ **then** MARKCONTACTED$(a, t)$
  - MARKCONTACTED$(a, t)$ : $\{[\text{true}] \rightsquigarrow \textbf{add}\{\text{Contacted}(a, t)\}\}$

## Participant

- **on** $cfp(t)$ **from** $a$ **if** $true$ **then** ASSIGNPRICE$(t, a)$
  - ASSIGNPRICE$(t, a)$ : $\{[true] \rightsquigarrow$ **add**$\{\text{SuggestedPrice}(t, getPrice(t), a)\}\}$

# Implementation of RMAS

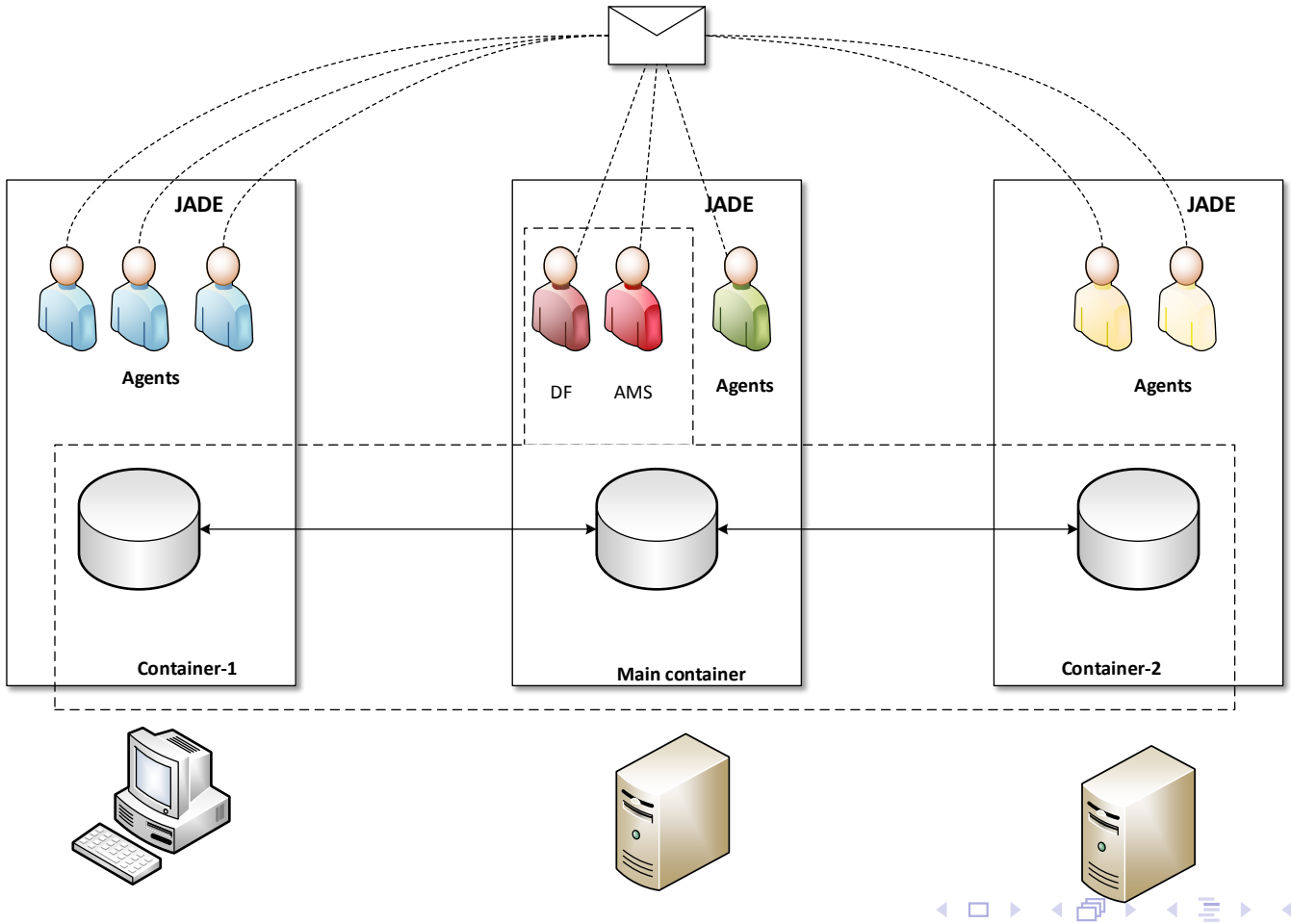
1. There are many different platforms for multiagent systems

# Implementation of RMAS

1. There are many different platforms for multiagent systems
   - JADE is a *FIPA compliant* agent platform and a Java framework for the development of MAS.

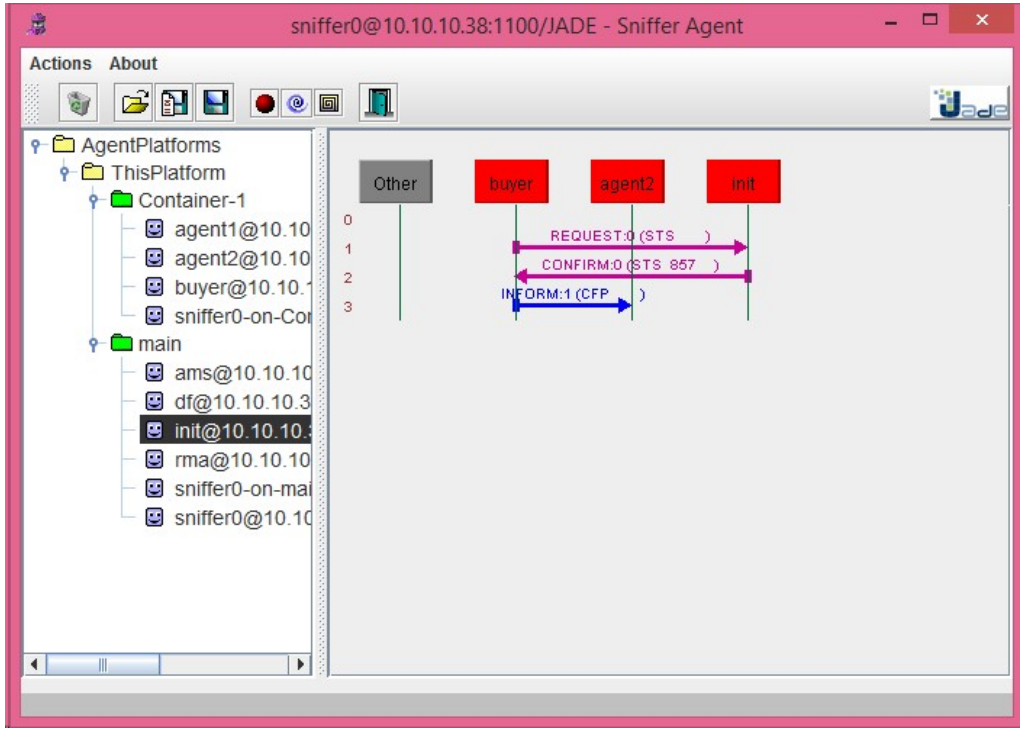
# Implementation of RMAS

1. There are many different platforms for multiagent systems
   - JADE is a *FIPA compliant* agent platform and a Java framework for the development of MAS.



# Implementation of RMAS

1. There are many different platforms for multiagent systems
   - JADE is a *FIPA compliant* agent platform and a Java framework for the development of MAS.



# Implementation of RMAS

2. We provide implementation of RMAS as a combination of JADE and an engine for data-aware processes.

# Implementation of RMAS

2. We provide implementation of RMAS as a combination of JADE and an engine for data-aware processes.
    - Engine: KRDB research group at UniBZ.

# Implementation of RMAS

2. We provide implementation of RMAS as a combination of JADE and an engine for data-aware processes.
   - Engine: KRDB research group at UniBZ.
   - We have improved the engine by adding syntax for multiagent aspects of the system.

# Implementation of RMAS

2. We provide implementation of RMAS as a combination of JADE and an engine for data-aware processes.
   - Engine: KRDB research group at UniBZ.
   - We have improved the engine by adding syntax for multiagent aspects of the system.
   - The whole system can be specified using *textual* or *XML* notation.

# Implementation of RMAS

2. We provide implementation of RMAS as a combination of JADE and an engine for data-aware processes.
   - Engine: KRDB research group at UniBZ.
   - We have improved the engine by adding syntax for multiagent aspects of the system.
   - The whole system can be specified using *textual* or *XML* notation.
   - All queries in agent specification are domain-independent.
     - $\rightsquigarrow$ They can be expressed using SQL

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;
2. for interacting with external environment and injecting new data.

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;
2. for interacting with external environment and injecting new data.

Nondeterminism can be implemented in the following ways:

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;
2. for interacting with external environment and injecting new data.

Nondeterminism can be implemented in the following ways:

- by interacting with humans;

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;
2. for interacting with external environment and injecting new data.

Nondeterminism can be implemented in the following ways:

- by interacting with humans;
- by interacting with external services;

# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;
2. for interacting with external environment and injecting new data.

Nondeterminism can be implemented in the following ways:

- by interacting with humans;
- by interacting with external services;
- by randomly chosen agent decision.

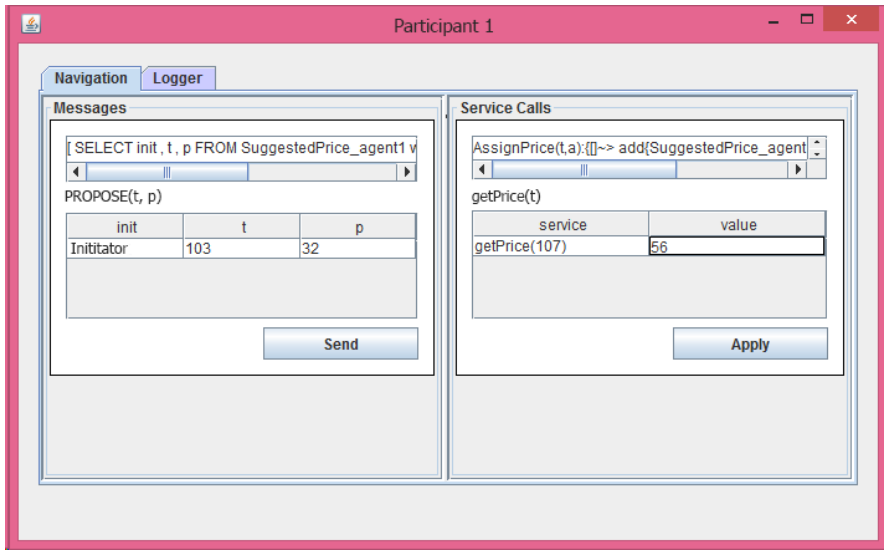
# Nondeterministic Aspects of Agents

In RMAS there are several situations when non-determinism occurs in the system:

1. during the execution of communicative rules;
2. for interacting with external environment and injecting new data.

Nondeterminism can be implemented in the following ways:

- by interacting with humans;
- by interacting with external services;
- by randomly chosen agent decision.



# Commitments

Commitment is a guarantee that binds the debtor towards the creditor about the achievement of a specific condition.

# Commitments

Commitment is a guarantee that binds the debtor towards the creditor about the achievement of a specific condition.

## Social commitment

$$CC(debtor, creditor, q_{antecedent}, q_{consequent})$$

An agent *debtor* commits to an agent *creditor* to bring about the consequent $q_{consequent}$, whenever $q_{antecedent}$ holds.

# Commitments

Commitment is a guarantee that binds the debtor towards the creditor about the achievement of a specific condition.

## Social commitment

$$CC(debtor, creditor, q_{antecedent}, q_{consequent})$$

An agent *debtor* commits to an agent *creditor* to bring about the consequent $q_{consequent}$, whenever $q_{antecedent}$ holds.

## Example

$$CC(university, student, accepted, admission\_letter\_delivered)$$

# Commitments

Commitment is a guarantee that binds the debtor towards the creditor about the achievement of a specific condition.

## Social commitment

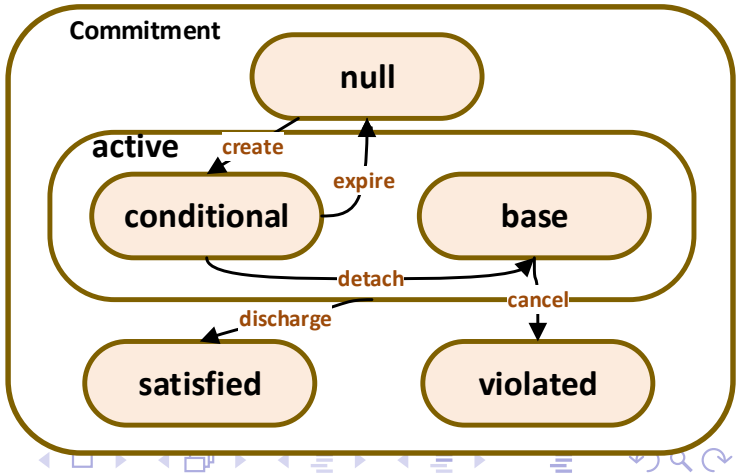
$$CC(debtor, creditor, q_{antecedent}, q_{consequent})$$

An agent *debtor* commits to an agent *creditor* to bring about the consequent $q_{consequent}$, whenever $q_{antecedent}$ holds.

## Example

$$CC(university, student, accepted, admission\_letter\_delivered)$$

- Commitments evolve through time and have life cycles.
- The evolution of commitments is regulated by the commitment machine



# First-order commitments

For supporting first-order commitments it requires to introduce the following components:

## Contractual specification

Set of commitment rules of the form:

$$\textbf{on } \mathtt{EV}(\vec{x}) \textbf{ from } s \textbf{ to } r \textbf{ if } Q_c(s, r, \vec{x})$$
$$\textbf{then } CC_{\mathtt{n}}(s, r, Q_p(s, r, \vec{x}, \vec{y}), Q_d(s, r, \vec{x}, \vec{y})),$$

# First-order commitments

For supporting first-order commitments it requires to introduce the following components:

## Contractual specification

Set of commitment rules of the form:

$$\textbf{on } \mathtt{EV}(\vec{x}) \textbf{ from } s \textbf{ to } r \textbf{ if } Q_c(s, r, \vec{x})$$
$$\textbf{then } CC_{\mathtt{n}}(s, r, Q_p(s, r, \vec{x}, \vec{y}), Q_d(s, r, \vec{x}, \vec{y})),$$

## Example – Contract Net Protocol

$$\textbf{on } \mathit{propose}(t, \mathit{price}) \textbf{ from } p \textbf{ to } i \textbf{ if } [\mathit{true}] \textbf{ then}$$
$$\mathtt{CC}_{\mathit{acceptance}}(p, i, [\exists \mathit{price}.\mathit{AssignedTo@}i(p, t, \mathit{price})],$$
$$[\mathit{ExecTask@}p(i, t, \text{“}\mathit{success}\text{”}) \vee \mathit{ExecTask@}p(i, t, \text{“}\mathit{fail}\text{”})])$$

## Commitment Box

Special relations obtained from the contractual specification. For each conditional commitment:

- One relation for its instances, keeping track of *debtor*, *creditor*, *payload*.
- One relation for the instances of its base-level version, keeping track also of the extended payload and the commitment state (active, satisfied, violated, ...)

## Commitment Box

Special relations obtained from the contractual specification. For each conditional commitment:

- One relation for its instances, keeping track of *debtor*, *creditor*, *payload*.
- One relation for the instances of its base-level version, keeping track also of the extended payload and the commitment state (active, satisfied, violated, ...)

## Example

The *proposal* commitment leads to

- $AcceptanceCC(debtor, creditor, task, price)$
- $AcceptanceC(debtor, creditor, state, task, price)$

# First-order commitments

## First-order Commitment Machine

1. Commitments can be manipulated:

# First-order commitments

## First-order Commitment Machine

1. Commitments can be manipulated:
   - Explicit operations: delegation, cancellation, release ...

# First-order commitments

## First-order Commitment Machine

1. Commitments can be manipulated:
   - Explicit operations: delegation, cancellation, release ...
   - Implicit operations: handled by the institutional agent whenever antecedent/consequent becomes true.

# First-order commitments

## First-order Commitment Machine

1. Commitments can be manipulated:
   - Explicit operations: delegation, cancellation, release ...
   - Implicit operations: handled by the institutional agent whenever antecedent/consequent becomes true.
2. On-exchange rules automatically generated from the contractual specification and executed by the institutional agent.

# First-order commitments

## First-order Commitment Machine

1. Commitments can be manipulated:
   - Explicit operations: delegation, cancellation, release ...
   - Implicit operations: handled by the institutional agent whenever antecedent/consequent becomes true.
2. On-exchange rules automatically generated from the contractual specification and executed by the institutional agent.

### Example

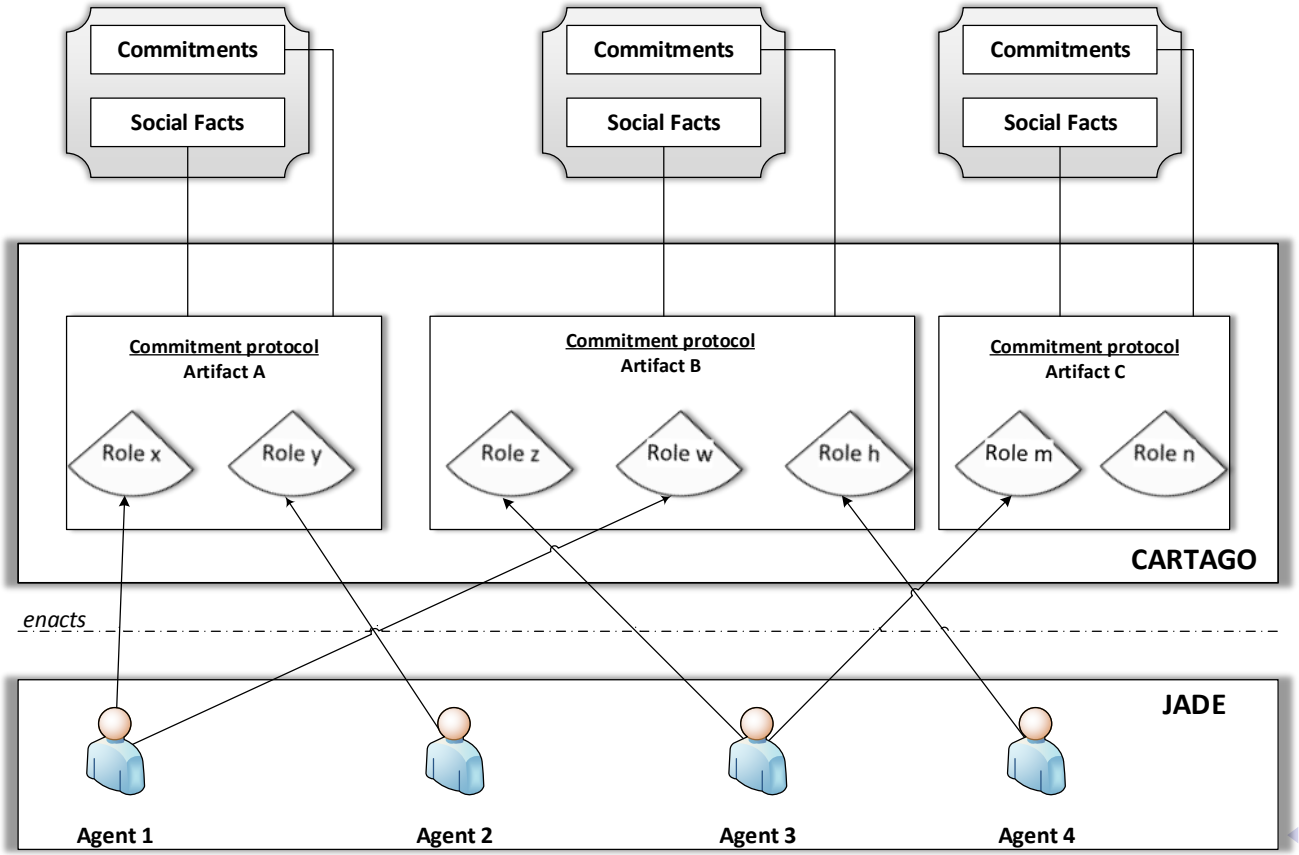
**CC creation**:
**on** $propose(task, price)$ **from** $p$ **to** $i$ **if** $[true]$ **then** $create\_AcceptanceCC(p, i, task, price)$
where $create\_AcceptanceCC(p, i, task, price) : \{[true] \rightsquigarrow$
$\mathbf{add}\{AcceptanceCC(p, i, task, price)\}\}$
**C creation** (triggered for any exchanged event):

$createC(d, c) :\{[AcceptanceCC(d, c, task, price) \wedge (\exists price.AssignedTo@i(d, task, price))]$
$\rightsquigarrow$ **add** $\{AcceptanceC(d, c, active, task, price)\}\}$

# Implementation

## 2COMM

- *2COMM("Communication & Commitment")* is the framework for building commitment-based protocols fully implemented in Java.
- Provides two connectors for existing agent platforms: JADE and Jason.



# Implementation

## Different ways of extending 2COMM

We propose different architectures extending 2COMM framework with the possibility of supporting first-order commitments. These architectures depend on:

# Implementation

## Different ways of extending 2COMM

We propose different architectures extending 2COMM framework with the possibility of supporting first-order commitments. These architectures depend on:

1. message passing services established between agents;

# Implementation

## Different ways of extending 2COMM

We propose different architectures extending 2COMM framework with the possibility of supporting first-order commitments. These architectures depend on:

1. message passing services established between agents;
2. the database usage credentials assigned to the entities;

# Implementation

## Different ways of extending 2COMM

We propose different architectures extending 2COMM framework with the possibility of supporting first-order commitments. These architectures depend on:

1. message passing services established between agents;
2. the database usage credentials assigned to the entities;
3. the agents allowed to manipulate commitments;

# Implementation

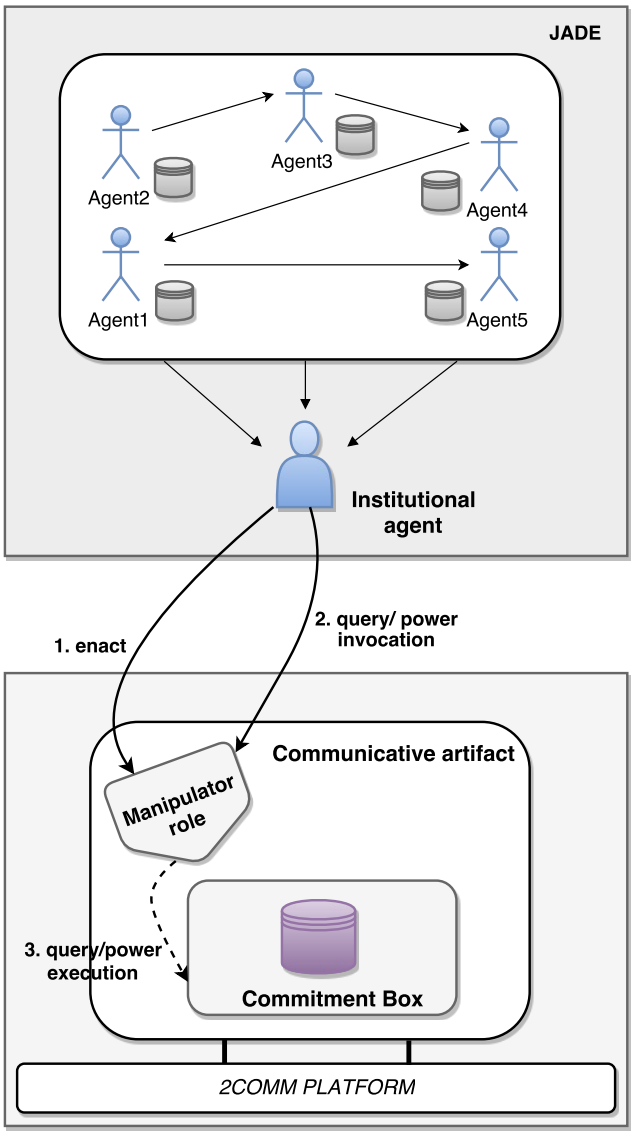
## Different ways of extending 2COMM

We propose different architectures extending 2COMM framework with the possibility of supporting first-order commitments. These architectures depend on:

1. message passing services established between agents;
2. the database usage credentials assigned to the entities;
3. the agents allowed to manipulate commitments;
4. the ways of representing artifacts.

# Implementation

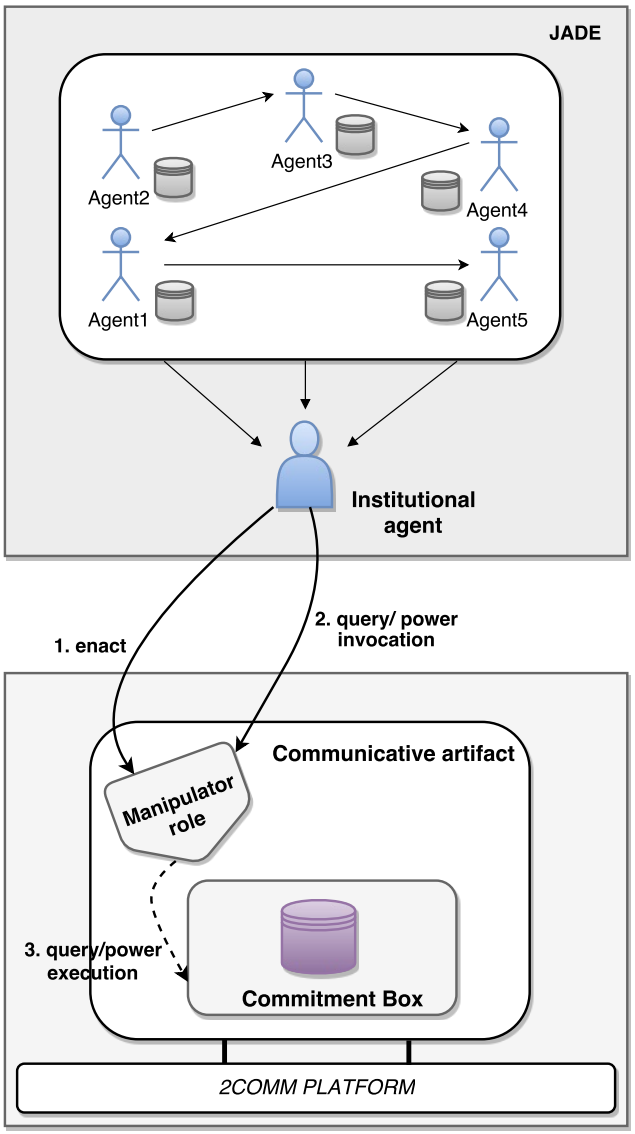
## Combining 2COMM and RMAS




- Agents communicate via JADE message services.

# Implementation

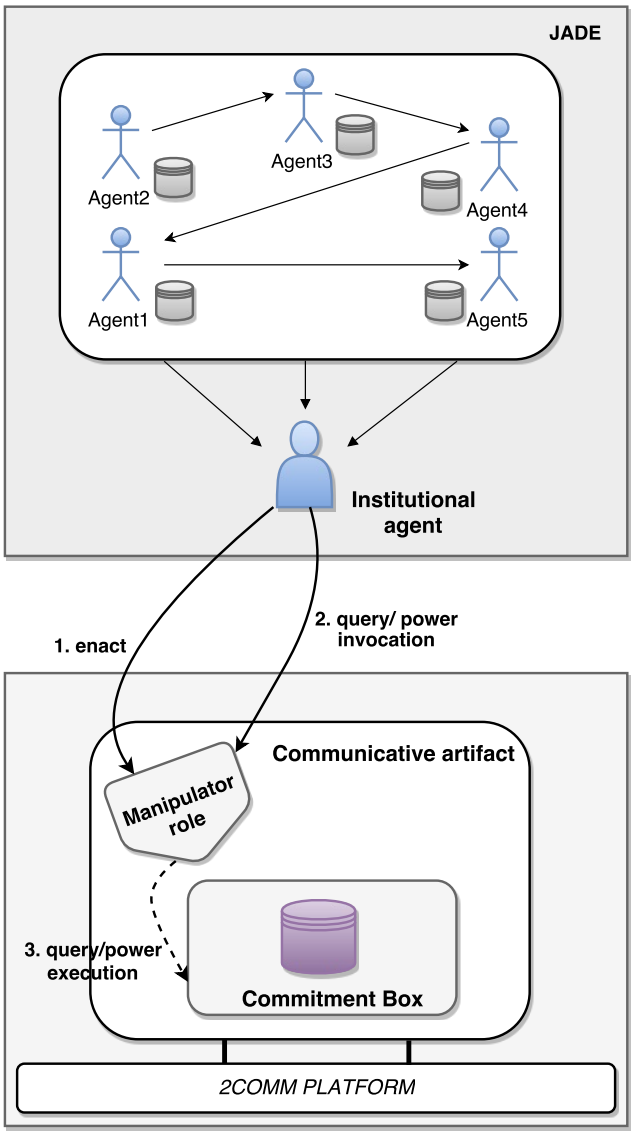
## Combining 2COMM and RMAS




- Agents communicate via JADE message services.
- The institutional agent can observe all messages in the system.

# Implementation

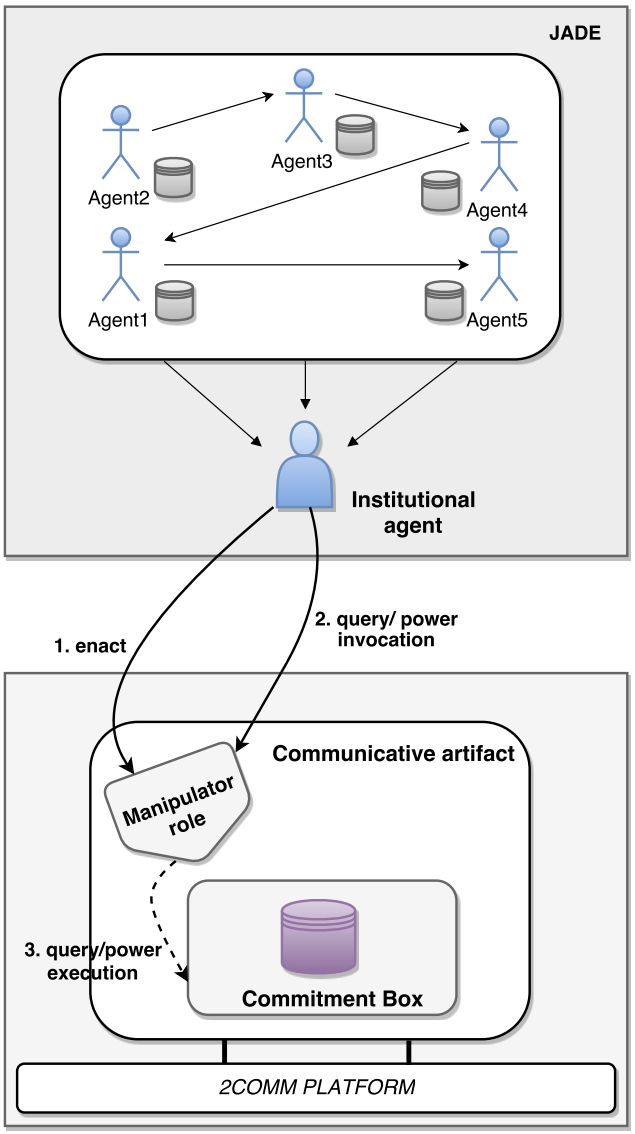
## Combining 2COMM and RMAS




- Agents communicate via JADE message services.
- The institutional agent can observe all messages in the system.
- Commitment protocols can be specified using declarative language.

# Implementation

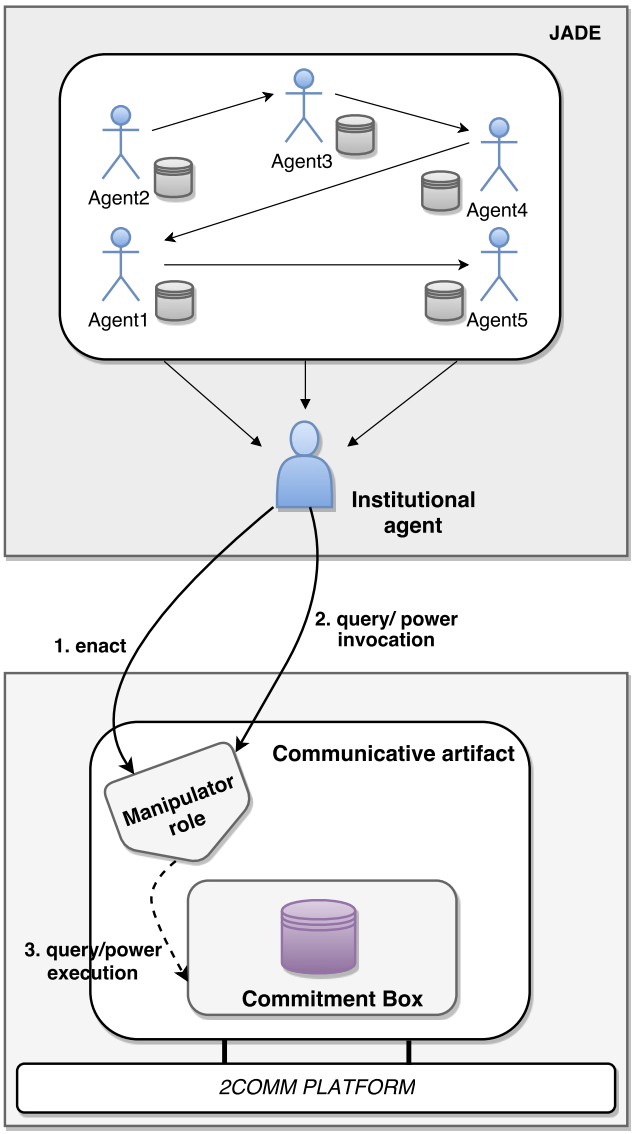
## Combining 2COMM and RMAS




- Agents communicate via JADE message services.
- The institutional agent can observe all messages in the system.
- Commitment protocols can be specified using declarative language.
- Commitments are manipulated either explicitly or implicitly by means of data-aware artifacts..

# Implementation

## Combining 2COMM and RMAS




- Agents communicate via JADE message services.
- The institutional agent can observe all messages in the system.
- Commitment protocols can be specified using declarative language.
- Commitments are manipulated either explicitly or implicitly by means of data-aware artifacts..
- The institutional agent is supplied with the GUI.

# Conclusions

- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.

# Conclusions

- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.

# Conclusions

- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.
- We have made this studies and proposals operational on top of JADE+2COMM.

# Conclusions

- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.
- We have made this studies and proposals operational on top of JADE+2COMM.
- Future work:

# Conclusions

- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.
- We have made this studies and proposals operational on top of JADE+2COMM.
- Future work:
  1. Temporal aspects of commitments.

# Conclusions

- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.
- We have made this studies and proposals operational on top of JADE+2COMM.
- Future work:
  1. Temporal aspects of commitments.
  2. Efficient access of agent databases.

# Conclusions

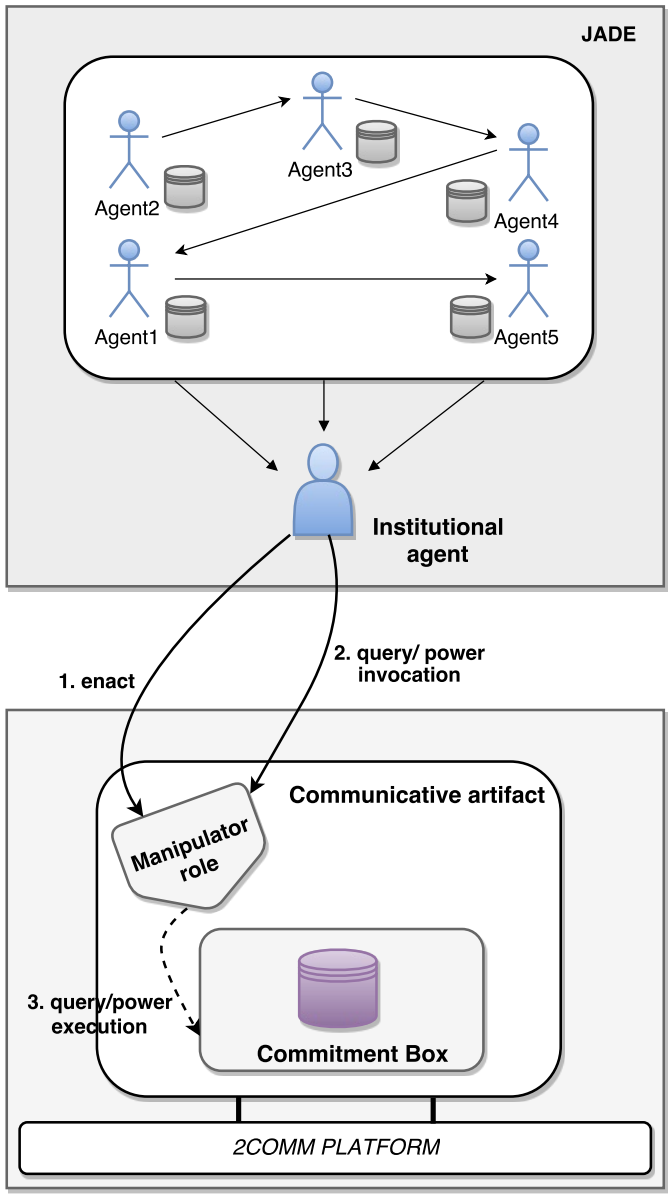
- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.
- We have made this studies and proposals operational on top of JADE+2COMM.
- Future work:
  1. Temporal aspects of commitments.
  2. Efficient access of agent databases.
  3. Nested commitments.

# Conclusions

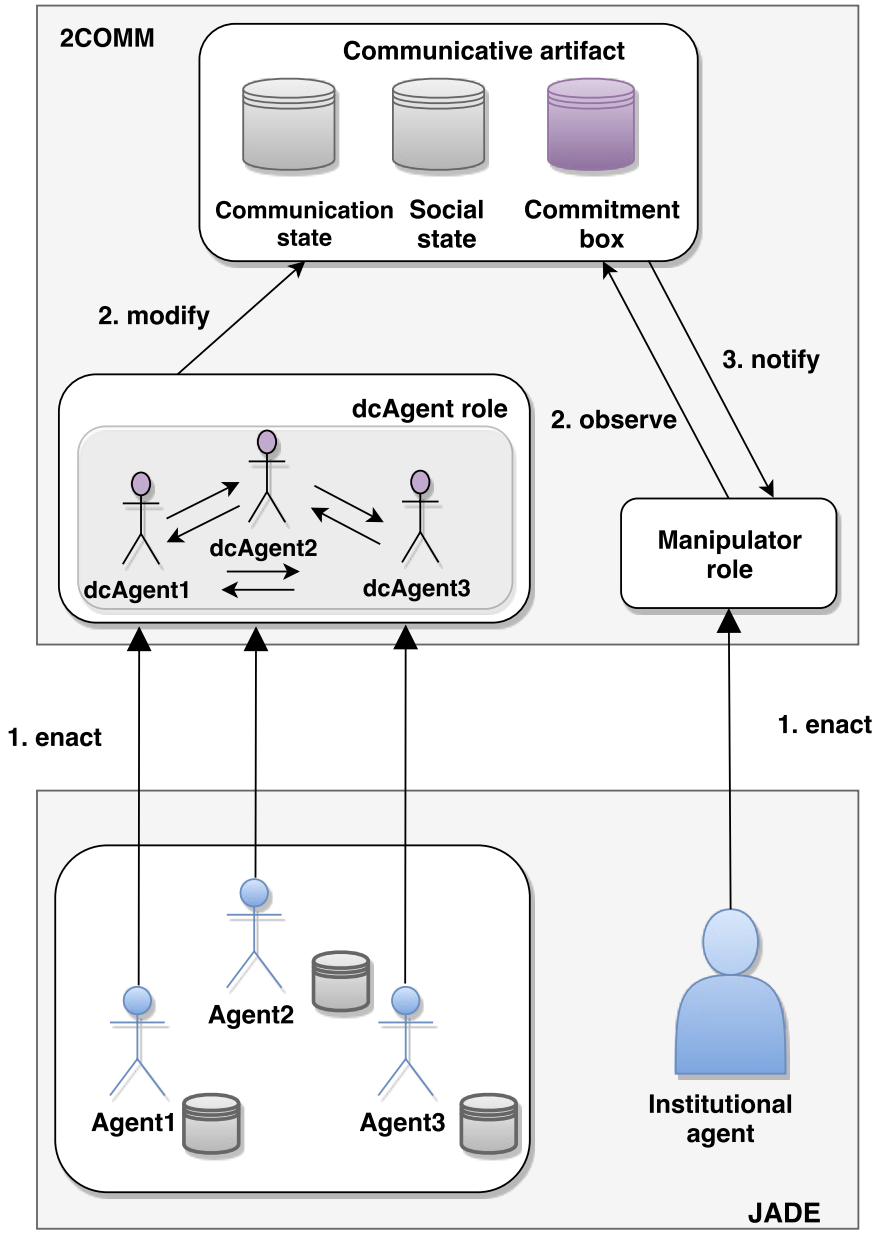
- Data-awareness is fundamental in MASs, but neglected in virtually any formal approach in the literature.
- We showed how to conceptualize the relational MAS, where data and processes are equally central, and how it can be extended for supporting commitment-based protocols.
- We have made this studies and proposals operational on top of JADE+2COMM.
- Future work:
  1. Temporal aspects of commitments.
  2. Efficient access of agent databases.
  3. Nested commitments.
  4. Typed relational multiagent systems.

# Thank you for attention!

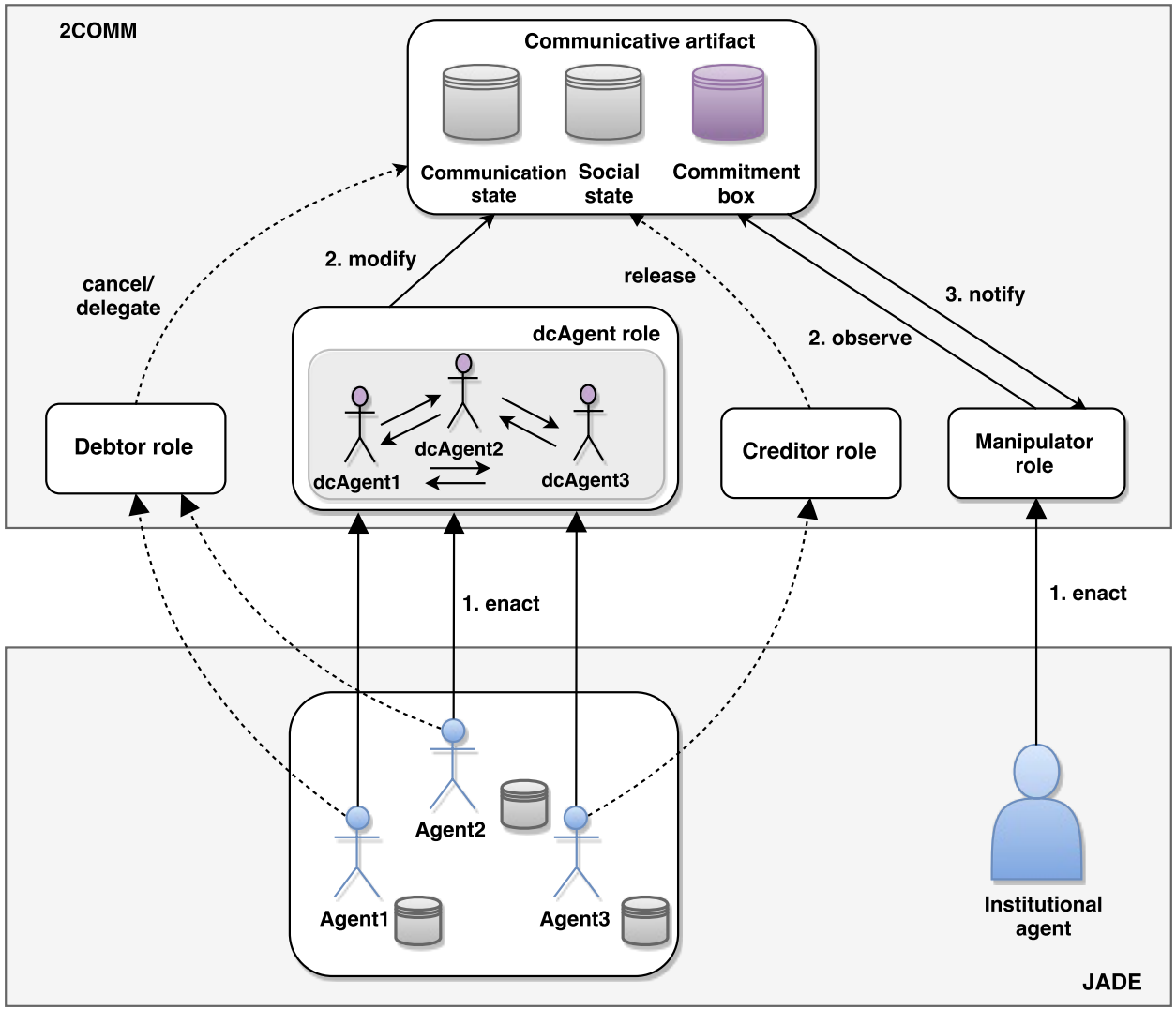
# Architectures



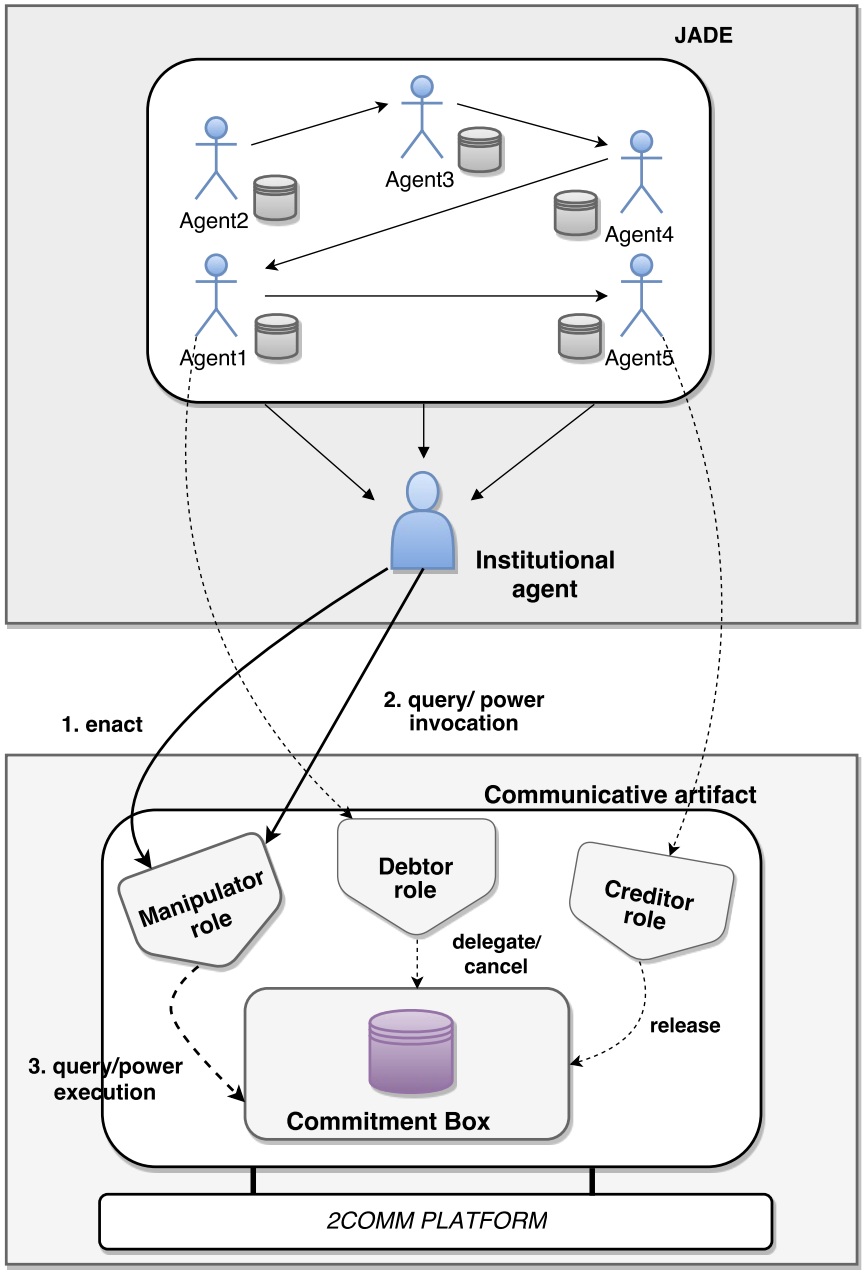
# Architectures



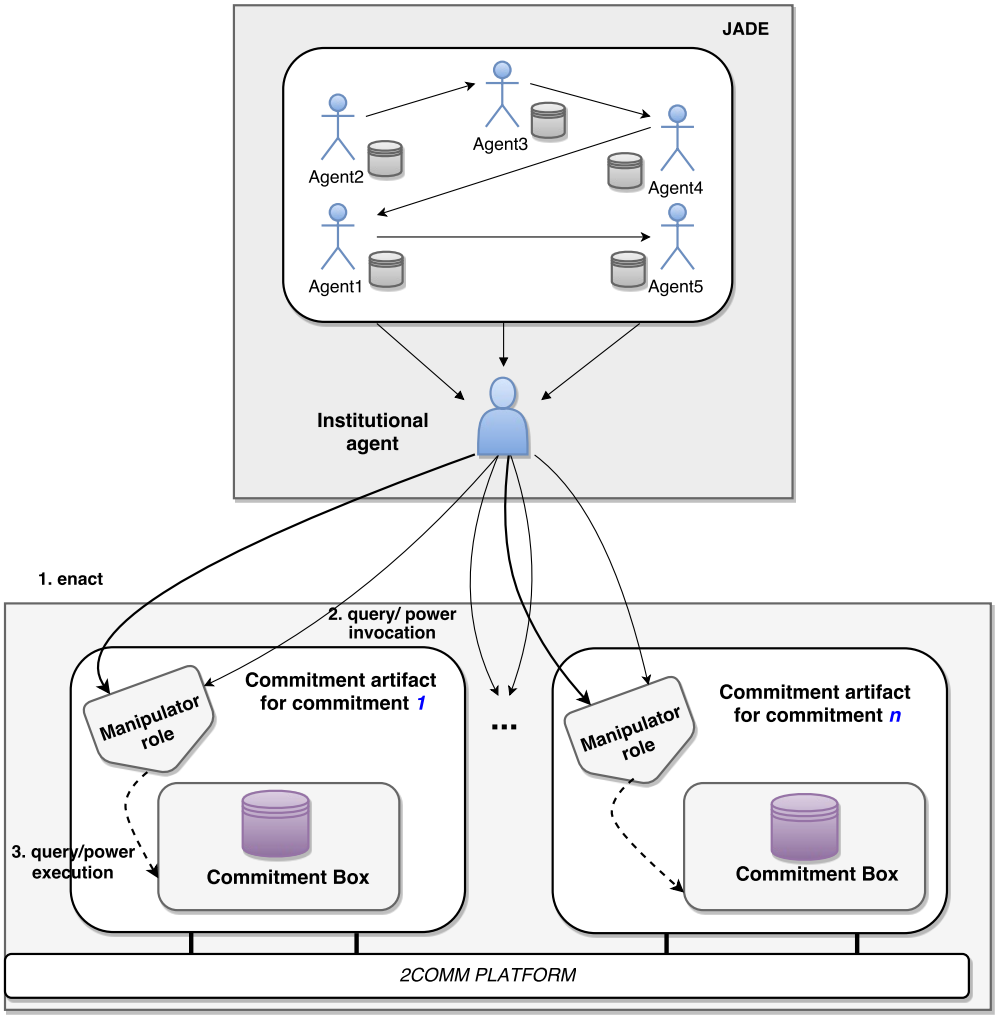
# Architectures



# Architectures



# Architectures



# Architectures