Protocol-Based Communication for Situated Multi-Agent Systems

Danny Weyns, Elke Steegmans and Tom Holvoet
AgentWise, DistriNet, K.U.Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{danny.weyns, elke.steegmans, tom.holvoet@cs.kuleuven.ac.be}

Abstract

In this paper we introduce a model for direct communication in situated multi-agent systems. Direct communication is typically associated with cognitive agents, where the information encoded in the messages is related to a mental state. This generally assumed view on communication however, does not fit the approach of situated, behavior-based agents. We propose a protocol-based communication model for situated agents. Communication specified in terms of protocols, i.e. well-defined sequences of messages, shifts the focus of communication from the reasoning upon messages towards the relationship between the exchanged messages. The model decomposes communication into three functional modules: message decoding, communicating and message encoding. The core of the model, the communicating module (1) interprets decoded messages and reacts to them in accordance with the applicable protocol, and (2) initiates or continues conversations when the conditions imposed by the applicable protocol are satisfied.

1. Introduction

In the approach of situated multi-agent systems[1] (situated MASs), agents and the environment constitute complementary parts of a multi-agent world that can mutually affect each other. Situatedness places an agent in a context in which it is able to perceive its environment and in which it can (inter)act. Intelligence in a situated MAS originates from the interactions of the agents in their environment rather than from the capabilities of individual agents. Where the approach of situated MASs started from the rejection of classical agency based on symbolic AI, nowadays the original opposition tends to evolve towards convergence with different schools emphasizing different aspects. In recent years, several researchers have shown that a mutual integration of visions can yield a synergetic surplus value, see e.g. [6][11][20]. This paper contributes in this evolution by integrating direct communication in situated MASs.

Situated, behavior-based agents usually communicate indirectly, e.g. by depositing pheromone trails in the environment. To set up explicit collaborations, reflected in mutual commitments, the agents need to communicate directly with one another. Direct communication however, is typically associated with cognitive agents, where the information encoded in the messages is related to a mental state. This generally assumed view on communication does not fit the approach of situated, behavior-based agents. Situated agents do not reason upon mental state but select actions based on internal stimuli and stimuli perceived in their environment. In this paper we study the research problem of how to enable direct communication in situated MASs. Enabling direct communication sets a different perspective on the social interaction between situated agents. We propose communication specified in terms of protocols instead of mental states as an approach to shift the focus of communication from the reasoning upon individual messages towards the relationship between the exchanged messages.

Throughout the paper we use the Packet-World [16] as an illustrative case. Fig. 1 depicts an example of the Packet-World. The goal of the agents in the Packet-World is to bring colored packets (squares) to correspondingly colored destinations (circles). Agents can observe the environment, however only to a limited extent. Agents are allowed to make one step at a time to a free neighboring field or pick up a packet from a neighboring field. An agent is able to carry one packet at a time, which it can put down at any free neighboring field or at the destination. Agents can also send messages to each other either for exchanging information or for setting up collaborations. Performing actions requires energy. Therefore agents are equipped with a battery. Energy is of vital importance for the agents. The battery can be charged at one of the available charge stations. To find the way to a charge station, each charge station transmits a gradient field. Agents can follow such a field and "climb up" the gradient towards the station. In the example of Fig. 1 there is one charge station which is indicated by an electric

1 Alternative designations are behavior-based agents [2], adaptive autonomous agents [10] or hysteretic agents [6].

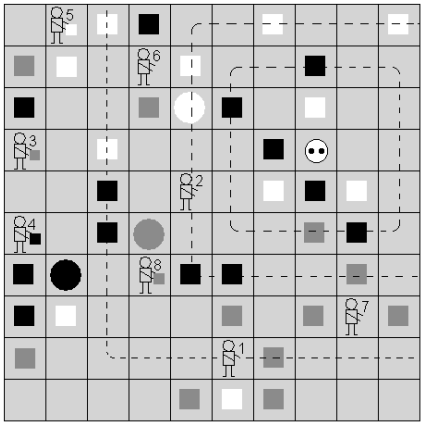

Figure 1. The Packet-World

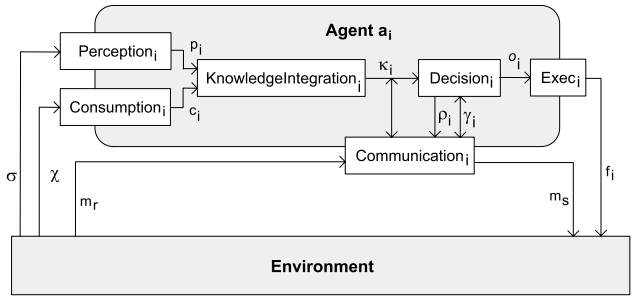

Figure 2. Generic model for a situated agent

socket symbol. The gradient field is marked by concentric rounded squares around the charge station. The intensity decreases proportional to the distance to the charge station.

This paper is structured as follows. Section 2 explains how the model for communication fits in a generic architecture for situated agents and introduces the basic concepts. Section 3 introduces the model for protocol-based communication. First we give a high level overview of the model. Then, in section 4, we zoom in on the model and formally describe the constituent modules and their interplay. Finally we conclude in section 5.

2. Generic agent architecture and concepts

In this section we first situate the model for communication in a general model for situated agents. Then we introduce the basic concepts of roles and situated commitments.

2.1. Agent architecture

The model for protocol-based communication we present in this paper fits in a generic model for situated MASs we have described in previous work [18]. This generic model formally describes an abstract architecture for situated MASs. The model builds upon Ferber's theory for action, described in [6]. According to this theory, agents produce influences into the environment and subsequently the environment reacts by combining the influences to deduce a new state of the world.

An overview of the generic architecture with a focus on the functional decomposition of an agent's behavior is depicted in Fig. 2. We touch briefly on the different modules. The $Perception_i$ module takes care of the perception of the environment of agent a_i, i.e. it maps the local state of the environment σ onto a percept, denoted as p_i. The $Consumption_i$ module selects from the set of consumptions χ a consumption c_i for agent a_i. A consumption is an effect of the reaction of the environment to the most recently produced influences for a particular agent. When an agent "consumes" a consumption, the consumed effect can be absorbed (e.g. food that is turned into energy), the agent may simply hold an element (e.g. an object it has picked up) or the consumption may affect the agent's state (e.g. the arm of a robot is wrenched through an external force). Perception and consumption depend on an agent's ontology [19].

The $KnowledgeIntegration_i$ module uses the most recent percept and consumption to update the internally registered state and to produce the *current knowledge* κ_i. The current knowledge is determined by the actual context the agent is situated in. Contrary to knowledge-based agents, situated agents do not build up an internal model of the environment. Instead, they favor to employ the environment itself as a source of information. Internal representations are described in terms of the immediate environment, known as indexical-functional representation [9][6]. Situated agents use representations to direct their decision making process, however this is done "here and now". Such representations do not oblige the agent to keep track of a hypothetical future state or investigate the implications of it on a plan.

The $Decision_i$ module is responsible for action selection. To decide about its next action, the decision module takes the agent's current knowledge κ_i, the set of available roles ρ_i and the activated situated commitments γ_i and selects an operator o_i for execution. We explain the concepts of a role and a situated commitment below. $Exec_i$ executes the selected operator producing an influence f_i into the environment. The environment collects the influences of simultaneously acting agents [17] and calculates the reaction, i.e. state changes in the environment and consumptions for the acting agents, according to a set of domain specific laws.

The final building block in the model of a situated agent is the $Communication_i$ module that is the subject of this paper. In short, the communication module processes incoming messages and produces outgoing messages according to well-defined communication protocols. Agents typically modify their state (current knowledge and/or situ-

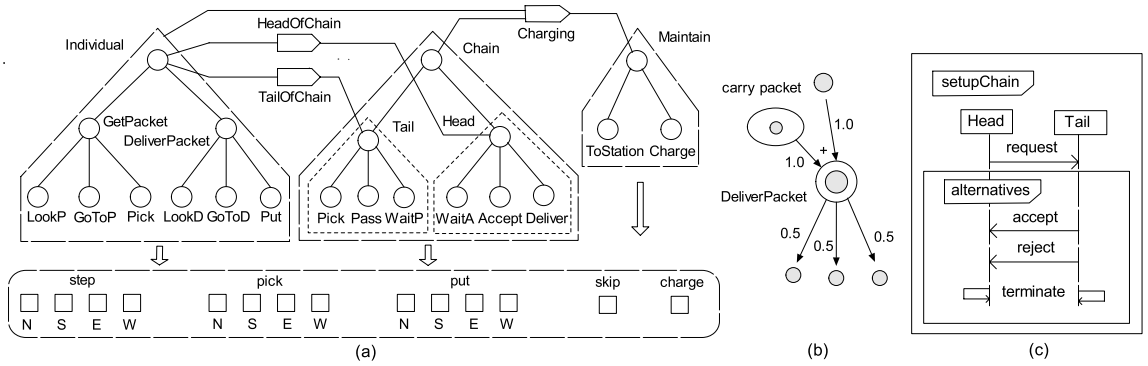

Figure 3. Model for action selection (a); detail of the activity flow (b); a communication protocol (c)

ated commitments) implied by the communicative interactions. Messages are exchanged through the communication medium that is part of the environment.

2.2. Roles and situated commitments

Direct communication is the basis for explicit collaborations between agents, reflected in mutual commitments. The attitude of a commitment has been studied extensively, however always from the perspective of cognitive agents, see e.g. [4][7][5]. These traditional approaches take a *psychological* viewpoint on commitment, i.e. a commitment is essentially based on the mutual beliefs of the involved agents. We introduce the notion of a *situated commitment* as a *social* attitude of situated agents. Contrary to the traditional approaches on commitment which are based on the mutually dependent mental states of the involved agents and a goal-oriented plan, a situated commitment is based on the *roles* of the involved agents and the *local context* they are placed in. We share the sociological viewpoint on commitment proposed in [14], however Singh's work focuses on cognitive agents in information rich environments while our focus is on situated, behavior-based agents.

As indicated in Fig. 2, roles (ρ_i) and situated commitments (γ_i) are related to action selection as well as to communication. To clarify the concepts of a role and a situated commitment we take a closer look at action selection, which is the responsibility of the $Decision_i$ module. For a detailed study we refer to our previous work, mentioned above.

The model for action selection is based on a hierarchical, free-flow network [12]. In [15] T. Tyrrell demonstrated that free-flow decision structures for behavior-based action selection are preferable over hierarchical or flat decision structures. For a recent discussion see [3]. However, existing free-flow architectures are designed from the viewpoint of individual agents. They lack *explicit* support for social behavior. The concepts of a role and a situated commitment enable explicit social behavior of situated, behavior-based agents. Fig. 3(a) depicts a simplified partial action selection model for an agent in the Packet-World.

The hierarchy is composed of *nodes* which receive information from internal and external stimuli in the form of *activity*. The nodes feed their activity down through the hierarchy until the activity arrives at the *action nodes* where a winner-takes-it-all process decides which action is selected. Fig. 3(b) shows in detail how the $DeliverPacket$ node collects its activity and feeds it downwards in the hierarchy.

We define a role as a subtree in the hierarchy that covers a logical functionality of the agent. The root node of such a subtree is denoted as the *top node* of the role. A role is named as its top node. A role may consist of a set of sub-roles, and sub-roles of sub-sub-roles etc. All roles of the agent are constantly active and contribute to the final decision making by feeding subsets of actions with activity. However the contribution of each role depends on the activity it has accumulated from the affecting stimuli of its nodes. In the example, there are three roles demarcated by dotted triangles. In the role $Individual$, the agent searches for packets and brings them to the destination. The role of $Chain$ is composed of two sub-roles: $Head$ and $Tail$ denoting the two roles of agents in a collaboration to form a chain to pass packets to each other. Finally in the role of $Maintain$ the agent recharges its battery.

A situated commitment defines a relationship between one role, called the *goal* role, and a non-empty set of other roles, i.e. the *source* roles, of an agent. Situated commitments have a well-known name. Explicitly naming the commitments enables agents to set up mutual commitments in a collaboration [14]. However, a single agent can also commit to itself. The connector $Charging$ in Fig. 3(a) denotes the situated commitment of an agent to itself to recharge its battery. The connectors $HeadOfChain$ and $TailOfChain$ denote the mutual situated commitments of two agents to collaborate in a chain. With each link between (the top node of) a source role and the commitment, a *weight factor* is associated that determines the amount of influence of the

source role on the goal role via the situated commitment.

Besides a name, each situated commitment is characterized by a *relations* set, a *context*, an *activation condition*, a *deactivation condition*, a *status* (activated or deactivated) and an *addition function*. The relations set contains the identity of the related agent(s) in the situated commitment. The context describes contextual properties of the situated commitment such as descriptions of objects in the local environment. Activation and deactivation conditions can be boolean expressions based on internal state or perceived information, or event occurrences such as sending or receiving a message. When the activation condition becomes true, the situated commitment is activated. The situated commitment then injects an additional amount of activity in the goal role defined by the addition function. The weight factors of the links from the source roles determine the fraction of activity of each source role that is taken into account by the addition function. The top node of the goal role combines the additional activity of the situated commitment with the regular activity accumulated from its stimuli. As soon as the deactivation condition becomes true, the situated commitment is deactivated. Then the situated commitment no longer influences the activity level of its goal role.

In general one agent can be involved in different situated commitments at the same time. Activity received through different situated commitments is then combined with the regular activity received from stimuli into one result.

As an example, consider agent 8 (see Fig. 1) that commits to be $HeadOfChain$ in a collaboration to pass packets to agent 1. The activation condition for the situated commitment is the receipt of the confirmation from agent 1 to collaborate. The relations set for the commitment is the singleton agent 1. The context of the commitment denotes that gray packets are passed along the chain. The deactivation condition is a change in the environment that indicates that the collaboration has finished, e.g. agent 1 has left its post.

Summarizing, agents agree on mutual situated commitments in a collaboration via direct communication. Activating situated commitments and initiating their properties (relations set and context) are typical responsibilities of the $Communcation_i$ module. Once activated, the situated commitment will affect the selection of actions in the $Decision_i$ module. The situated commitment induces extra activity in the hierarchy, favoring the goal role of the commitment relatively to the source roles. Contrary to traditional approaches of commitment (e.g. a joint commitment [4]) where the agents have the obligation to mutually communicate with each other when the conditions for a committed cooperation no longer hold, for a situated commitment it is typically the local context in which the involved agents are placed that regulates the duration of the commitment. This approach fits the general principles of situatedness and robustness of situated MASs.

3. Model for protocol-based communication

In this section we give a high level description of the model for protocol-based communication. First we examine communication protocols, then we introduce the model for protocol-based communication.

3.1. Communication protocols

Communication in MASs is traditionally based on speech act theory [1][13]. Speech act theory treats communication as actions, however the communicative acts are considered in isolation. For example, the original KQLM specification only suggests an implicit sequencing of messages in agent interactions [8]. In practice speech acts are mostly part of logically related series of communicative acts. In addition, communicative acts are typically specified in terms of mental states which imposes consequences on the nature of the agents. Communication specified in terms of protocols shifts the focus of communication from reasoning upon individual messages towards the relationship between the exchanged messages.

A *communication protocol* specifies a well-defined sequence of messages, each message referring to a speech act. We consider both binary and n-ary communication protocols. A binary protocol involves two communication peers (one as the initiator), whereas an n-ary protocol involves multiple communication peers (also with one initiator). *Protocol-based communication* is the interaction between agents based on the exchange of messages according to a specific communication protocol. We use the notion of a *conversation* to refer to such an ongoing interaction. A conversation is initiated by the *initial* speech act of a communication protocol. At each stage in the conversation there is a limited set of possible speech acts. Terminal states determine when the conversation comes to an end. During the progress of a conversation agents typically modify their state implied by the communicative interaction.

As an example, we look at the communication protocol to set up a chain for passing packets in the Packet-World as shown in the UML interaction diagram of Fig. 3(c). If the conditions for an agent hold to enter the role of $Head$ of a chain, it *requests* the candidate tail to cooperate. Request is the initial speech act. The requested agent then investigates the proposal. Depending on the context it answers with *accept* or *reject*. In case the agent accepts the request it activates the situated commitment $TailOfChain$, see Fig. 3(a). After receiving the acceptance, the initiating agent activates the situated commitment $HeadOfChain$. The cooperation is then settled and continues until the situated commitments are deactivated. Deactivation may be communicated explicitly by communicating "end of cooperation" messages, but may also be induced by changes in the envi-

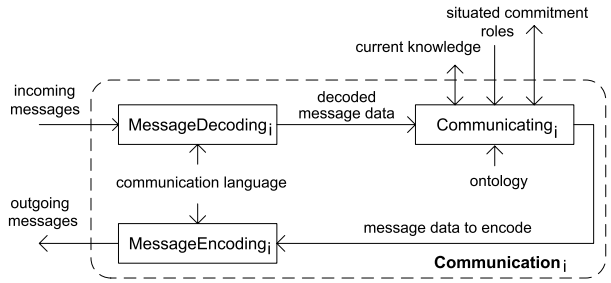

Figure 4. High level overview of the model

ronment, e.g. when all packets are passed or when an agent for some reason leaves its post. In case the requested agent rejects the proposal the conversation terminates without an agreement. Finally, when the requested agent neglects the request, e.g. when it urgently left to recharge its battery, the initiator detects this and *terminates* the conversation.

3.2. High level model

Fig. 4 depicts an overview of the model. The model decomposes communication into three functional modules: message decoding, communicating and message encoding.

The message decoding module stores incoming messages in a buffer and decodes the buffered messages one by one. Decoding extracts the information from a message according to a well defined *communication language*. A communication language defines the format of the messages, i.e. the subsequent fields the message is composed of. We denote the extracted information as *decoded message data*. Decoded message data describes the information of a message in a form that can be understood by the agent's communicating module. As an example, consider an agent a_i with identity y_i that requests an agent a_j with identity y_j to form a chain for passing gray packets. The decoded message data of the received message may contain the following data:

$$< 2475, y_i, request, \{Tail, chain(packet(gray))\} >$$

This decoded message data consists of four fields. 2475 is a unique identifier for the conversation. This identifier is assigned by the initiator of the conversation and is used by the participants to refer unambiguously to the conversation. y_i is the identity of the sender, $request$ is the performative of the message and $\{Tail, chain(packet(gray))\}$ is the obvious content of the message. In the model, we assume that agents speak (one and) the same language. This limitation avoids complexity, however the model can be extended to multi-language communication.

The communicating module is the heart of the communication system and has a twofold task. First, this module is responsible for interpreting the decoded message data derived from incoming messages and for reacting appropriately. Second the module is responsible to initiate or continue a conversation when the conditions imposed by an applicable protocol step become true. To deal with these tasks, the communicating module uses the agent's repository of protocols, its current knowledge, the set of available roles, the status of its situated commitments and a common *ontology*. The ontology defines the terminology of the modelled domain, i.e. (1) a vocabulary of words that are used to represent concepts in the domain and (2) a set of relationships that express interconnections between these concepts. For example, the content of the decoded message data $\{Tail, chain(packet(gray))\}$ connects four concepts of the agent's vocabulary $Tail$, $chain$, $packet$ and $gray$ into a relationship with an obvious semantic. In the example, the conditions for agent y_j to accept the proposal to collaborate in the chain are: (1) it knows the protocol to set up a chain, (2) the agent has the role $Tail$, (3) there are gray packets in the agent's neighborhood to pass on and (4) the agent is not yet engaged in a commitment that conflicts with this new request. If all these conditions hold, the agent accepts the proposal, otherwise it rejects it. When accepting it, the agent activates the situated commitment $TailOfChain$ and composes *message data to encode* like:

$$< 2475, y_i, accept, \{TailOfChain, packet(gray)\} >$$

This message data contains the necessary information to encode the confirmation message of the collaboration.

The message encoding module encodes the newly composed message data into messages and passes them to the message transport system of the environment. To deal with possible delays, this module also provides a buffer.

4. Formal description of the model

In this section we formalize the model for protocol-based communication. The formal notation is based on set theory and in accordance with [18][19]. First we zoom in on $MessageDecoding_i$ and $MessageEncoding_i$, then we look at the $Communicating_i$ module.

4.1. Message decoding and encoding

Fig. 5 depicts an overview of the decoding and encoding modules. First we introduce a number of definitions:

$Ag = \{a_1, \ldots, a_n\}$: the set of agents in the MAS

$y_i \in Y$: the identity of a_i with $Y = \{y_1, ..., y_n\}$ the set of unique identities, one for each agent in the MAS

$m_k \in M$: a formatted structure of characters representing a message with M the set of all possible messages in the system

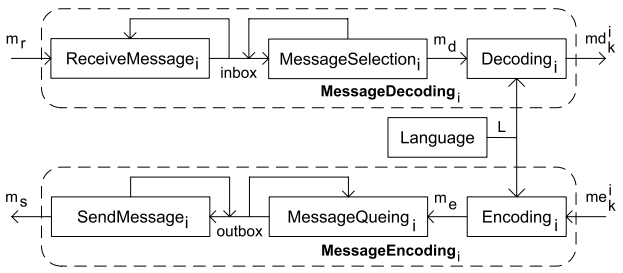

Figure 5. Message decoding and encoding

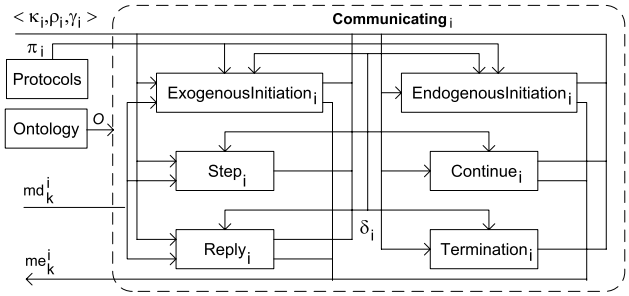

Figure 6. Detailed model for communicating

$L \in \Lambda$: the communication language[2] that defines the message format; a message has the following fields:

1. $cid \in Cid$: a unique id of the conversation, with Cid the set of conversations ids. The function $getCid()$ returns a new id to the initiator of a conversation
2. $sender \in Y$: the id of the sender of the message
3. $addressees \in 2^Y$: the ids of the addressees
4. $perform \in P$: the performative of the message, with P the set of all performatives of L
5. $content \in C$: the content of the message, with C the set of all contents of L

$inbox \in M^i_{in}$: a set of messages in the input buffer of agent a_i with $M^i_{in} \subseteq 2^M$ all message sets with agent a_i as addressee

$outbox \in M^i_{out}$: a set of messages in the output buffer of a_i, with $M^i_{out} \subseteq 2^M$ all message sets with agent a_i as sender

$md^i_k = <cid, sender, perform, content>$: decoded message data, i.e. the data of a message received by a_i, with $md^i_k \in Dd$ and Dd the set of all decoded message data in the system

$me^i_k = <cid, addressees, perform, content>$: the message data to encode a message to be sent by a_i, with $me^i_k \in De$ and De the set of all message data to encode in the system

Based on these definitions we formalize the functionality of the decoding and encoding modules. We limit the description to the decoding part. $ReceiveMessage_i$ is a function that adds a message received by agent a_i to its $inbox$. The function is typed as follows:

$$ReceiveMessage_i : M \times M^i_{in} \rightarrow M^i_{in}$$
$$ReceiveMessage_i(m_r, inbox) = inbox^+$$

with[3] $inbox^+ = inbox + m_r$

$MessageSelection_i$ picks up a message from the inbox and sends it to the decoding module:

$$MessageSelection_i : M^i_{in} \rightarrow M \times M^i_{in}$$
$$MessageSelection_i(inbox) = < m_d, inbox^- >$$

with $inbox^- = inbox - m_d$

$Decoding_i$ decodes the selected message resulting in decoded message data. To decode the received messages, the decoding module uses a communication language L. This function is typed as follows:

$$Decoding_i : M \times \Lambda \rightarrow Cid \times Y \times P \times C$$
$$Decoding_i(m_d, L) = < cid, sender, perform, content >$$

The integral $MessageDecoding_i$ function is typed as follows:

$$MessageDecoding_i : M \times M^i_{in} \times \Lambda \rightarrow Cid \times Y \times P \times C$$
$$MessageDecoding_i(m_r, inbox, L) = < cid, sender, perform, content >$$

Summarizing, the message decoding module collects received messages in a buffer from where it selects the messages one by one for decoding. To decode a message the agent uses the communication language. Decoding results in decoded message data that can be interpreted by the communicating module of the agent.

4.2. Communicating

Now we direct our attention to the core of protocol-based communication, the $Communicating_i$ module, see Fig. 6. First we introduce a number of extra definitions. Then we discuss the decomposition of the communicating module.

4.2.1. Definitions

$O \in \Theta$: the ontology[4] that defines the terminology of the modelled world, i.e. a tuple $< V, F >$ defined as:

1. $v \in V$: the representation of a concept that the agent understands with V the vocabulary of the domain concepts
2. $f \in F$: a relationship between concepts of V, with V the set of all relationships of the ontology

$p(c_q, \ldots, c_r) \in R^O$: the representation of a part of knowledge based on an ontology O, with p a predicate, c_i a value and R^O the set of all knowledge representations in the system

2 Since we assume one language, the set Λ only contains the singleton L.

3 The binary operator '$+$' denotes the addition of an element to a set, while the '$-$' operator analogously denotes the substraction.

4 We suppose that agents use one and the same ontology, thus $\Theta = \{O\}$.

$\kappa_i \in K_i^O$: the current knowledge of agent a_i, with $K_i^O \subseteq K^O$ all possible knowledge sets of a_i and $K^O \subseteq 2^{R^O}$ all sets of knowledge in the agent system

$r \in \rho_i$: the name of a role, in short a role, with $\rho_i \subseteq \Re$ the role set of a_i and $\Re$ the set of all roles in the system

$g \in \gamma_i$: a situated commitment, i.e. a 3-tuple $< name, status, initiate >$ with $name \in V$ defined in the ontology O and $status \in \{activated, deactivated\}$ the commitment status; $initiate$ is a function that enables an agent to set contextual properties of the commitment ($relations$ and $context$, see section 2.2). $\gamma_i \in \Gamma$ is a_i's set of situated commitments, $\Gamma \subseteq 2^G$ the set of all commitment sets and G all situated commitments in the agent system

$s \in S$: a protocol step, i.e. a tuple $< conditions, effects >$ with $conditions$ a set of boolean expressions that determine whether the protocol step is applicable (based on received messages, the roles and the state[5] of the agent); $effects$ describe the effects of the protocol step (composing a new message data and/or updating the state of the agent)

$p = \{s_u, \ldots, s_w\}$: a communication protocol, i.e. a well-defined set of protocol steps with $p \in \pi_i$ and $\pi_i \in \Pi$ the set of protocols available to agent a_i; $\Pi = 2^P$ is the set of all protocol sets and P the protocol set in the agent system

$d \in \delta_i$: a conversation, i.e. a 3-tuple $< protocol, cid, history >$ with $protocol$ the used protocol, cid a unique identifier, $history$ the sequence of message data of received/sent messages of the conversation. $\delta_i \in \Delta_i$ is the set of conversations agent a_i currently is involved in and $\Delta_i \subseteq \Delta$ the set of all conversation sets agent a_i can be involved in; $\Delta \subseteq 2^D$ is the set of all conversation sets and D the set of all conversations in the agent system

The communication function abstracts from the concrete implementation of a role and a situated commitment in the $Decision_i$ module. By explicitly naming roles and situated commitments agents can set up mutual commitments in a collaboration. We illustrate the use of a communication protocol for a collaboration to set up a chain in the Packet-World, see also Fig. 3. The protocol consists of four protocol steps:

$$setupChain = < request, accept, reject, terminate >$$

We take a closer look at the $request$ step that initiates the collaboration:

$$\begin{aligned}
&request = < \\
&\quad \{\kappa_i[distance(self, destination(c)) = |1|, \\
&\qquad distance(self, agent(y_j) = |2|, \\
&\qquad distance(destination(c), agent(y_i)) \leq 2\,], \\
&\quad \rho_i[r = Head], \\
&\quad \gamma_i[\neg status.any = activated\,], \\
&\quad \delta i[\neg\, protocol.any = setupChain\,]\,\}, \\
&\quad \{cid = getCid(), \\
&\qquad me_k^i = < cid, \{y_j\}, request, \{Tail, chain(packet(c))\} >, \\
&\qquad conversation = < setupChain, cid, \{me_k^i\} > \\
&\qquad \delta_i.add(conversation)\ \} \\
&>
\end{aligned}$$

An agent a_i with id y_i is able to *request* an agent a_j with id y_j to collaborate as $Tail$ in a chain for passing packets with color c when, according to its current knowledge, the requesting agent a_i is near the destination for packets of color c and the requested agent a_j is positioned at the right distance to cooperate. Besides agent a_i must be able to perform the role of $Head$, it may not have activated any other commitment ($\gamma_i[\neg status.any = activated]$) and it may not be involved in any other conversation to set up a chain ($\delta i[\neg\, protocol.any = setupChain]$). When all these conditions hold agent a_i asks for a new conversation id, composes a new message data and a new conversation with the appropriate data and adds the new conversation to its set of ongoing conversations ($\delta_i.add(conversation)$). When agent a_j receives the encoded message it verifies the request and informs agent a_i whether it *accepts* or *rejects* the collaboration. However, when for some reason agent a_j neglects the request, e.g. when it leaves the context to recharge its battery, the $terminate$ condition for agent a_i becomes true and that finishes the conversation.

4.2.2. Communicating functionality Fig. 6 depicts the decomposition of the communicating module. Each sub-module takes care of a particular kind of conversation step. We identify four classes of functions in a conversation: initiating a conversation, reacting to an incoming message, continuing a conversation and terminating a conversation. $ExogenousInitiation_i$ and $EndogenousInitiation_i$ represent functions to initiate a new conversation, by another agent or the agent itself respectively. $Step_i$ processes received message data without directly reacting to it, while $Reply_i$ immediately reacts to received message data. $Continue_i$ picks up the conversation after a break and finally $Termination_i$ "silently" terminates a conversation. An example of $Step_i$ is a manager that receives a bid of one of the participants in a Contract Net. An example of $Continue_i$ is a contractor who informs the manager about the result of the expedited task. Finally, $Termination_i$ is illustrated in the Packet-World example above where agent a_j neglects a request of a_i to collaborate in a chain. We limit the formal description to two representative modules, $EndogenousInitiation_i$ and $Reply_i$.

$EndogenousInitiation_i$ starts new conversations. This function is typed as follows:

$$\begin{aligned}
&EndogenousInitiation_i : \\
&\qquad \Pi \times \Delta_i \times K_i^O \times \Gamma_i \times \Re \times \Theta \rightarrow K_i^O \times \Gamma_i \times De \times \Delta_i \\
&EndogenousInitiation_i(\pi_i, \delta_i, \kappa_i, \gamma_i, \rho_i, O) = \\
&\qquad < \kappa_i', \gamma_i', me_j^i, \delta_i' >
\end{aligned}$$

5 Here state refers to the agent's current knowledge, the status of its situated commitments and the conversations it is involved in.

An agent is able to initiate a new conversation when the conditions of the initial step of an applicable protocol hold according to the set of conversations the agent is currently involved in, its current knowledge, the status of its situated commitments, the available roles and the ontology of the domain. As a result the agent updates its current knowledge, it possibly activates a new situated commitment, it composes the message data to encode the initial message of the newly initiated conversation and finally it adds the new conversation to the set of its conversations.

$Reply_i$ reacts to incoming messages of an ongoing conversation and is typed as follows:

$$Reply_i : Dd \times \Delta_i \times K_i^O \times \Gamma_i \times \Re \times \Theta \rightarrow K_i^O \times \Gamma_i \times De \times \Delta_i$$
$$Reply_i(md_k^i, \delta_i, \kappa_i, \gamma_i, \rho_i, O) = < \kappa_i', \gamma_i', me_k^i, \delta_i' >$$

To react to an incoming message, the agent selects from the set of conversations it is currently involved in the conversation that matches the conversation id of the decoded message data. The agent then verifies the conditions of the next protocol steps for the conversation based on its current knowledge, the status of its situated commitments, the available roles and the ontology of the domain. According to the effects of the protocol step for which the conditions hold, the agent updates its current knowledge and the set of situated commitments, it composes the message data for the reply and adds the new message data to the history of the conversation. To conclude, the encoding module encodes the new message data and passes the reply at the message transport system of the environment.

5. Conclusions and future work

In this paper we proposed a model for protocol-based communication in situated MASs. Direct communication allows situated agents to exchange information or to set up explicit collaborations. We use the notion of a situated commitment as a basis for explicit collaboration between situated agents. During collaboration setup, agents exchange messages. If the agents agree, i.e. when the conditions prescribed by the protocol hold, this results in mutual situated commitments. A situated commitment affects the agent's decision making in favor of the role it plays in the collaboration. The collaboration typically ends when the context of the involved agents changes such that the conditions to continue the collaboration expire.

To validate the model we implemented it in the Packet-World. In the paper, we showed how agents in the Packet-World use a communication protocol to coordinate in a collaboration to pass packets along a chain. The next step is to verify the model in a more complex case. In a current industrial research project we investigate how the multi-agent paradigm can be applied to Automated Guided Vehicle (AGV) warehouse systems. Traditional systems use one central controller that instructs the AGVs to perform jobs based on a calculated plan. By looking at AGVs as agents of a situated MAS the system can control itself in a distributed manner, improving adaptability and scalability.

Acknowledgements. This research is supported by the K.U.Leuven research council (AgCo2) and the Flemish Institute for Advancement of Research in Industry (EMC2).

References

[1] J. L. Austin. *How To Do Things With Words*. Oxford University Press, Oxford, UK, 1962.

[2] R. A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47, 1991.

[3] J. Bryson. Intelligence by design. *PhD thesis, MIT*, 2001.

[4] P. Cohen and H. Levesque. Teamwork. *Special Issue on Cognitive Science and Artifical Intelligence*, 1991.

[5] B. Dunin-Keplicz and R. Verbrugge. Calibrating collective commitments. *LNCS, Springer Verlag*, 2691, 2003.

[6] J. Ferber. *An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Great Britain, 1999.

[7] N. Jennings. Commitments and conventions. *The Knowledge Engineering Review*, 2(3), 1993.

[8] Y. Labrou. Standardizing agent communication. *LNCS, Springer Verlag*, 2086, 2001.

[9] P. Maes. *Designing Autonomous Agents, MIT Press*, 1990.

[10] P. Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1-2), 1994.

[11] V. Parunak and F. Zambonelli. From design to intention: Signs of a revolution. *1th International Conference on Autonomous Agents and Multi-Agent Systems, Bologna*, 2002.

[12] K. Rosenblatt and D. Payton. A fine grained alternative to the subsumbtion architecture for mobile robot control. *International Joint Conference on Neural Networks, IEEE*, 1989.

[13] J. R. Searle. *Speech Acts: An Essay in Philosophy of Language*. Cambridge University Press, 1969.

[14] M. Singh. Commitments among autonomous agents in information-rich environments. *8th Workshop on Modelling Autonomous Agents in a Multi-Agent World, Sweden*, 1997.

[15] T. Tyrrell. *Computational Mechanisms for Action Selection*. PhD thesis, University of Edinburgh, 1993.

[16] D. Weyns and T. Holvoet. The packet–world. *Demo presented at the 1th International Conference on Autonomous Agents and Multi-Agent Systems, Bologna*, 2002.

[17] D. Weyns and T. Holvoet. Regional synchronization for simultaneous actions in situated multi-agent systems. *LNCS, Springer Verlag*, 2691, 2003.

[18] D. Weyns and T. Holvoet. A formal model for situated multi-agent systems. *Formal Approaches for Multi-Agent Systems, Special Issue of Fundamenta Informaticae, to appear*, 2004.

[19] D. Weyns, E. Steegmans, and T. Holvoet. Model for active perception in situated multi-agent systems. *Special Issue of Journal on Applied Artificial Intelligence*, 18(8-9), 2004.

[20] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *Transactions on Software Engineering and Methodology, ACM*, 12(3), 2003.