Specifying Communication Aspects in Multi-Agent Systems using OPM/MAS

Arnon Sturm[1], Dov Dori[2], Onn Shehory[3]

[1]Department of Information Systems Engineering, Ben Gurion University of the Negev, Beer Sheva, 84105, Israel
sturm@bgumail.bgu.ac.il
[2] Faculty of Industrial Engineering and Management, Technion – Israel Institute of Technology, Haifa, 32000, Israel
dori@ie.technion.ac.il
[3] IBM Haifa Research Lab, Haifa University, Haifa, 31905, Israel
onn@il.ibm.com

Abstract. Communication has an essential role in agent-based systems and thus should be specified as part of the system. Agent-oriented modeling methods support the specification of communication aspects of agent-based systems to a limited extent. They lack in addressing some communication specification needs such as protocol reusability, specification synthesis and validability, and accessibility. In this paper, we propose a solution to this problem. We utilize the Object-Process Methodology for Multi-Agent Systems (OPM/MAS) for specifying the communication aspects. We also provide an algorithm for protocol validation. Via a feature-based comparison to other well-known methodologies, we show that the suggested new approach improves upon previous ones in addressing the communication specification needs.

1. Introdcution

Interactions are an essential aspect of Multi-Agent Systems (MAS). An interaction is a form of communication that consists of the technical means to communicate: a protocol, a communication language, and content. Communication can be referred to as messages and protocols, where the messages are the core building blocks and the protocol is an ordered set of messages that together define the admissible patterns of a particular type of interaction between entities. Clearly, communication has an essential role in agent-based systems and thus should be specified as part of the system. Indeed, many agent-oriented modeling methods[1] address communication aspects of agent-based systems. Yet, as we show, existing methods address the communication specification needs of agent-based systems only in part. In particular, those methods lack in specifying aspects such as protocol reusability, specification synthesis and validability, and in weaving the communication specification into the system specification. The latter results in a decreased accessibility of a method.

[1] For brevity, we shall henceforth use the term *method* in reference to a modeling method.

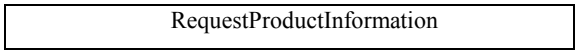
In this paper, we propose a solution to these shortcomings. In particular, our solution addresses protocol reusability, protocol specification synthesis and validability, and the weaving of communication specification into system specification. Protocol reusability is achieved by separating its specification from the system functional specifications. Protocol specification synthesis and validability is increased by providing formal definitions for all communication aspects. The weaving of the communication specification into the system specification is achieved by providing specification core elements that are common for both communication aspects and system specification. To arrive at these solutions we utilize OPM/MAS [13] in the following manner. Communication-related building blocks and their relationships were defined such that a protocol and a message are independently defined, thus increasing reusability. The formal definition of these building blocks and their relationships allows for protocol specification synthesis and validability. In addition, utilizing the OPM/MAS single model we eliminate the weaving problem. We found that OPM/MAS is adequate for specifying the communication aspects of a MAS, since it utilizes the integrated approach (of structure and behavior) of the Object-Process Methodology (OPM) [6] and is built on top of its core elements (i.e., objects and processes). This utilization enables the integration of the communication aspects into the method because it encompasses both structural and dynamic properties.

In the following, we survey well-known agent-oriented methods that have communication specification capabilities. We examine these methods in order to analyze the gap between the existing capabilities of specification methods and the needed capabilities, and to demonstrate various approaches towards communication specification. Based on this survey, we can infer what additional capabilities are required, and then utilize this understanding in suggesting a method that overcomes limitations of previous ones.

1.1. Related Work

The methods we examine are GAIA [14], MaSE [4, 5], and AUML [1, 2]. Other methods usually adopt modeling concepts (e.g., state charts or agent interaction protocol diagram) from these three methods. We demonstrate the communication specification capabilities of these modeling methods using a simple communication protocol for Requesting and Getting Product Information (in short, RGPI).

GAIA has two communication building blocks: a message and a protocol. In GAIA, the specification of the messages and the protocols is done using an interaction diagram as shown in Figure 1. The diagram shows two messages that follow the RGPI protocol mentioned before: **`RequestProductInformation`** and **`ProductInformation`**. The arrow indicates the message order. Each message is defined by its name (e.g., **`RequestProductInformation`**), its initiating role (the left side, e.g., **`AuctionParticipant`**), its responding role (the right side, e.g., **`AuctionManager`**), its description (the lower box), and parameters (on the horizontal lines on the right side of each message) that are required by the sender or the responder.



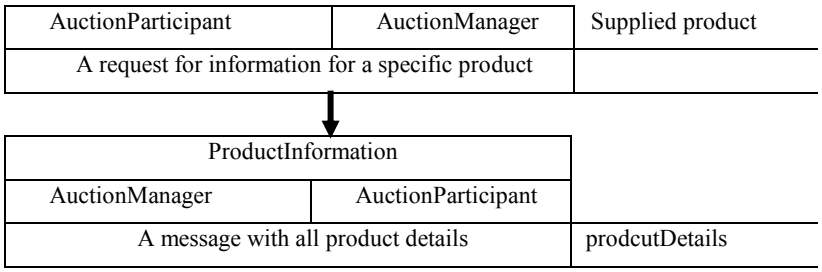


Figure 1. GAIA interaction diagram

MaSE uses different notions to specify the interactions among agents. The first footprint of interaction specification is done during the (MaSE) refinement roles stage within the analysis phase, when communication relationships between tasks are determined. The interaction specifications are done during the design phase within the conversation construction stage. The conversations detail the interactions between agent types where each conversation (interaction) in modeled by two communication class diagrams: one for the initiating agent and the other for the responding agent. A communication class diagram is a finite state automaton that defines the conversation states of the two participating agent classes. The diagram consists of states (in which an agent may perform some activities) and transitions in the form of "**`rec-mess(args1)[cond]/action^trans-mess(args2)`**", where **`rec-mess(arg1)`** is the message received with its argument, the **`[cond]`** indicates the required condition to move to the next state, the action is an activity performed during the transition, and the **`^trans-mess(args2)`** is the sent message with its argument.

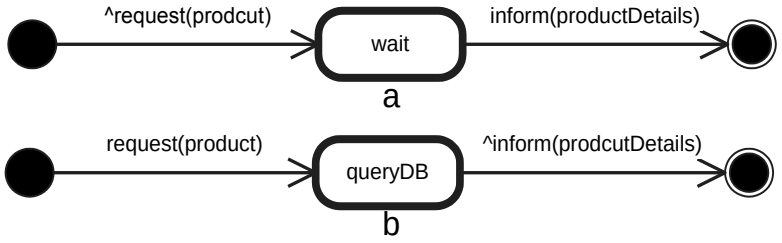


Figure 2. MaSE conversation specification: (a) the initiating agent communication class diagram; (b) the responding agent communication class diagram.

Figure 2 depicts the conversation between two agents. The initiating agent (Figure 2a) sends a request (indicated by the "^" symbol) and enters a "wait" state. It wakes up upon receiving the required information. The responding agent (Figure 2b) gets the request message and enters into a "queryDB" state in which it searches for the required information. Upon completing the search, the agent replies with the required information.

The most widely used technique for specifying interaction in an agent-based system is an agent interaction protocol diagram, which is part of Agent Unified Modeling Language (AUML). AUML is being developed under by one of the technical commit-

tees within FIPA [7]. That technical committee is trying to standardize agent-based systems modeling based on AUML. One of the aspects that this committee handles is the interaction aspect. The AUML modeling capabilities for agent interactions are specified in [9]. The agent interaction protocol diagram is based on a sequence diagram notation but with different semantics: (1) the lifelines are of roles and not of objects; and (2) the arrows are messages rather than method invocations. Figure 3 specifies the RGPI protocol, which consists of two participant roles: the **`AuctionParticipant`** and the **`AuctionManager`**.

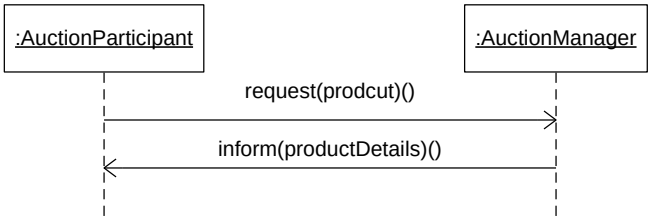


Figure 3. AUML agent interaction protocol diagram

Analyzing the specifications and the capabilities of the methods mentioned above, we found the following communication-related specification problems: in AUML it is not clear how the interaction diagram weaves into the system model; in GAIA there is no support for ACL; in MaSE each protocol is defined between two specific agent types, exclusively for these types. As a result, other agent types among which the same protocol is required cannot reuse the protocol, and hence it should be redefined for these agents types. Moreover, the three methods use a specific view for specifying some of the communication aspects, this view comes in addition to other system views. Views' multiplicity complicates models; the addition of another view increases complexity, in particular due to the need for further consistency checking.

In this paper, we endow a modeling method with the capabilities of specifying the communication aspects of multi-agent systems. This modeling method overcomes the aforementioned problems. Thus, the contribution of this paper is in proposing a specification method for agent communication that weaves into the entire model specification in a formal, reusable, yet, intuitive way.

The paper is organized as follows. Section 2 describes the Object-Process Methodology for Multi-Agent Systems, emphasizing the way according to which the communication aspects are being specified, and presents a validation algorithm. Section 3 evaluates the new approach by comparing its capabilities in modeling communication aspects of agent-based systems to the capabilities of the well-known agent-oriented methods and Section 4 concludes.

2. Object-Process Methodology for Multi-Agent Systems

The Object-Process Methodology for Multi-Agent Systems (OPM/MAS) was first presented in [13]. OPM/MAS is based on the Object-Process Methodology [6], which

inherits its capabilities from both object- and process-oriented paradigms. OPM is an integrated approach to the study and development of software systems. The basic premise of OPM is that objects and processes are two types of equally important classes of things, which together describe the function, structure and behavior of systems in a single framework in virtually any domain. OPM unifies the system lifecycle stages—specification, design and implementation—within one frame of reference, using a single diagramming tool – a set of Object-Process Diagrams (OPDs) and a corresponding subset of English, called Object-Process Language (OPL). Processes, objects and states are connected via structural or procedural links.

OPM/MAS is a specific extension for multi-agent systems using the domain analysis approach. It follows the OMG-MOF [12] approach that defines four abstraction layers for specifying information, systems, and domains. The first layer is the information layer, which is comprised of the desired data. The model layer, which is the second layer, is comprised of the metadata that describes data in the information layer. The third metamodel layer is comprised of the descriptions that define the structure and semantics of metadata. Finally, the meta-metamodel layer is comprised of the description of the structure and semantics of meta-metadata. Thus, for specifying multi-agent systems we use the OPM at the meta-metamodel, we also use it in the meta-model layer in which we specify the domain model of multi-agent systems and then we use that meta-model to specify a specific system that follows the rules defined within the meta-model.

Figure 4 and Figure 5 depict the upper levels of the OPM-based domain model for multi-agent systems. The elements within that model were already presented in [13]. Figure 4 presents the system diagram of the MAS metamodel, which explains our approach towards MAS application development and defines the relationships between the different building blocks. For example, a **Society** exhibits **Organizations** and **Agents**. Figure 5 presents the in-zoomed agent OPD. For example, an **Agent** consists of **Task processes**, **Mobilizing processes** and **Messaging processes**, and exhibits **Objects**, **Facts**, and **Services**, as well as **Locations**, **Priorities**, **Agent Names**, **Communication Acts**, **Types**, and **Results**.

Following the domain layer, the next layer is a MAS model. The model follows the rules specified within the MAS metamodel. Thus, the metamodel serves two related purposes. One is to define domain-specific building blocks or components that are comprised of lower-level OPM entities (objects, processes, and states), which enhance the reuse of the MAS building blocks and prevent the need to reinvent them from scratch. The second purpose of the metamodel is to verify the correct use of the building blocks and the correctness of the relationships between them within the application model. The linkage between the metamodel and the application model is done via the domain labels. These labels are recorded in the upper left side of an entity of each thing (i.e., an object or a process) within the application model.

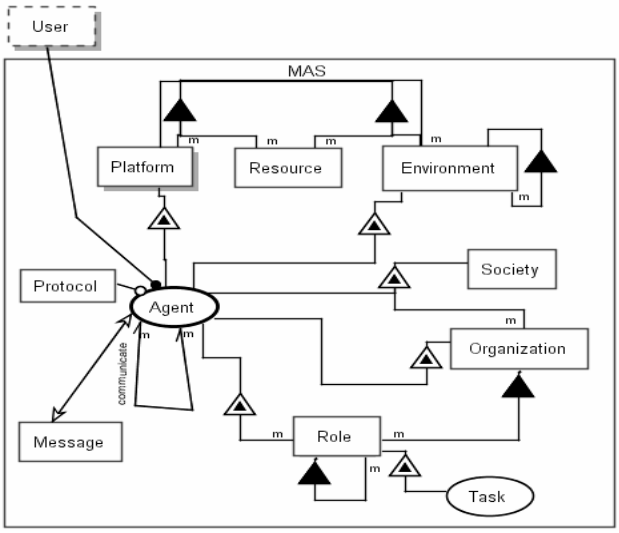


Figure 4. The multi-agent system domain – system diagram

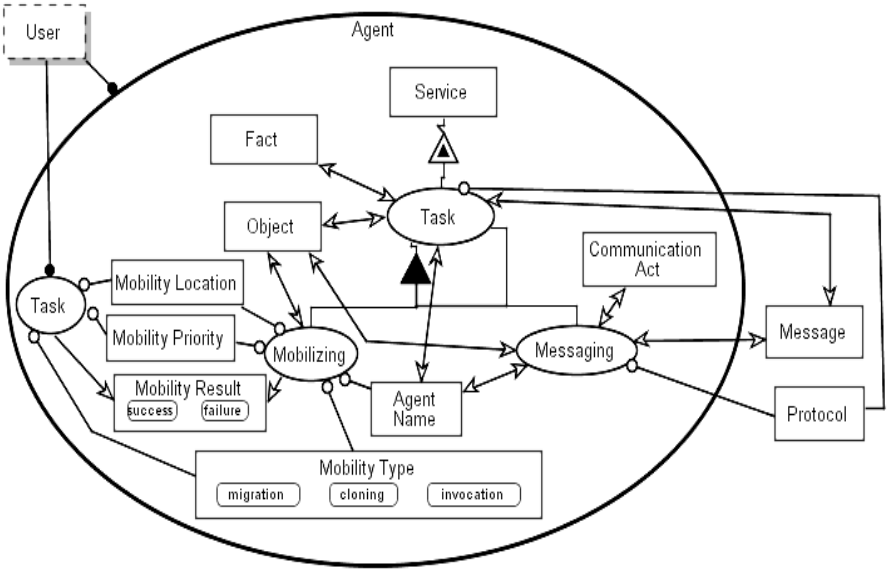


Figure 5. The multi-agent system domain - Agent in-zoomed

2.1. Modeling Protocols in OPM/MAS

In this paper, we focus on the design of the communication aspects within a multi-agent system. We introduce the building blocks that present the basic communication elements and show the relationships among these that were not shown in Figure 4 and

Figure 5. The communication building blocks described below were selected by analyzing the state of the art research on modeling communication aspects, such as [9]. We find these building blocks to adequately address the communication aspects' modeling needs. The communication building blocks are the following:

- **Message:** A means of exchanging facts or objects between agents. It is equivalent to an ACL message
- **Messaging:** A process (set of procedures) of building and transferring messages. On the sending end it consists of building the message, associating objects with it, and sending it out. On the receiving end, it consists of receiving the message, translating it, and associating it with the appropriate objects.
- **Protocol:** An ordered set of messages that together define the admissible patterns of a particular type of interaction between entities.
- **Communication Act:** A building block that specifies the performative, which is a straightforward utterance, related to a specific messaging process.
- **Message Group:** A building block that represents a group of messages. The messages within that group can be connected by and/or/xor relationships.

The first step in modeling the communication aspects of agent-based systems is to specify the communication patterns, that is, protocols. We first define the protocol metamodel. The protocol metamodel in Figure 6 includes a protocol (which is the main building block in this context), which may consist of other protocols (to enable nested protocol definition), message groups (to enable gathering of several message options), and communication acts that can be related to each other and to message groups. In OPM/MAS a protocol is an independent element which is not restricted to an agent or to a role, but is part of the multi-agent system (as shown in Figure 4).

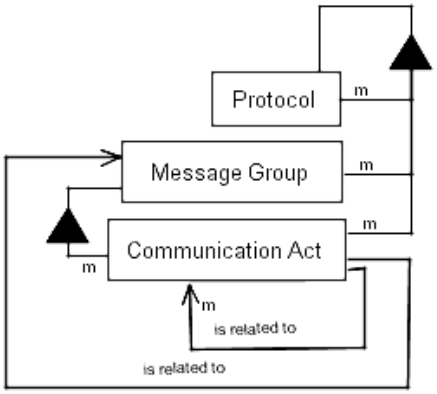


Figure 6. The multi-agent system domain - Protocol unfolded

The OPM/MAS model of the RGPI protocol is shown in Figure 7. The order of the messages is determined by their vertical position from top to bottom, i.e., **Request** and then **Inform**. This is based on the OPM convention that vertically arranged things are by default ordered with the first being placed at the top.

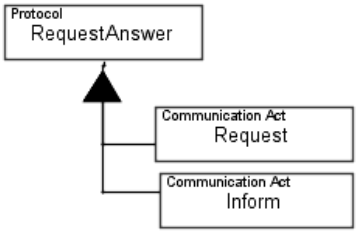


Figure 7. OPM/MAS simple protocol specification

To demonstrate additional OPM/MAS capabilities for protocol specification, we model the FIPA-ContractNet protocol [8], as shown in Figure 8.

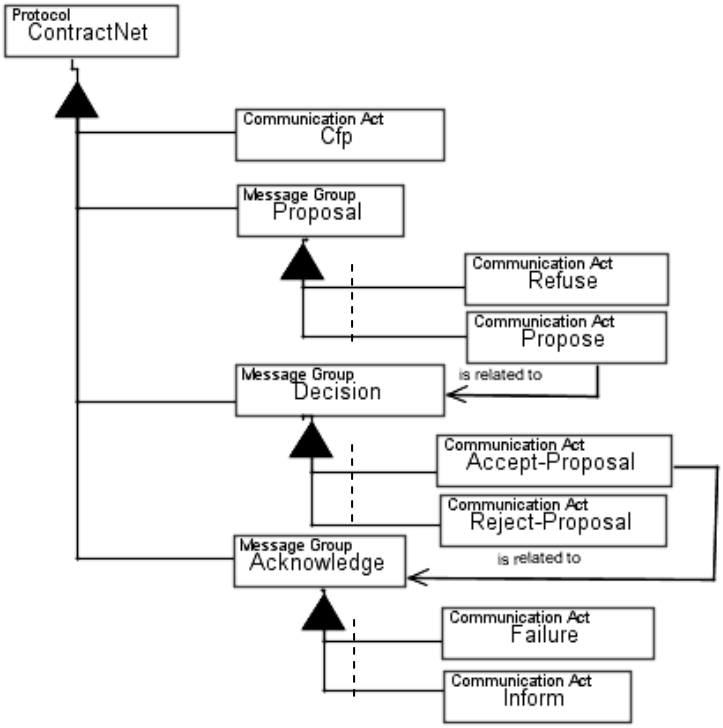


Figure 8. ContractNet protocol specification in OPM/MAS

As before, the order of the messages is from top to bottom, i.e., **Cfp Communication Act**, **Proposal Message Group**, **Decision Message Group**, and **Acknowledge Message Group**. The dashed line within each one of the message groups indicates a XOR relationship between the communication acts in that message group, which means that exactly one type of communication act can participate in an instance of the protocol. The default unidirectional structural relations between the **Propose Communication Act** and the **Decision Message Group**, and between the **Accept-Proposal Communication Act** and the **Acknowledge Message Group** indicate the flow of the protocol and the order of the messages. For example,

the **Propose Communication Act** is followed by a message from the **Decision Message Group**. In the case of **Refuse Communication Act** and **Reject-Proposal Communication Act**, the protocol ends, because the pre-conditions do not hold.

2.2. Modeling Messages in OPM/MAS

As stated before, modeling the protocol is the first stage of the system communication specification. We should further weave the protocol into the agent behavioral specification. In OPM/MAS, as the meta-model suggests, the messages are part of the system flow. In the following, we provide an example that demonstrates the specification of the communication aspects and its integration into the system specification (i.e., model). In Figure 9 a system level OPD of an electronic market multi-agent system is depicted. The system is aimed at managing the interaction among contractors. Upon receiving a task specification a contractor look for the appropriate agents and send a CFP to them following the FIPA contract net protocol rules. The system consists of three agents: **Contractor**, **Provider**, and **Market Manager**, which reside on different platforms. The **Contractor Agent** and the **Provider Agent** should follow the **ContractNet Protocol** as indicated by the links associating the protocol and the agents. The specification of that protocol is depicted in Figure 8.

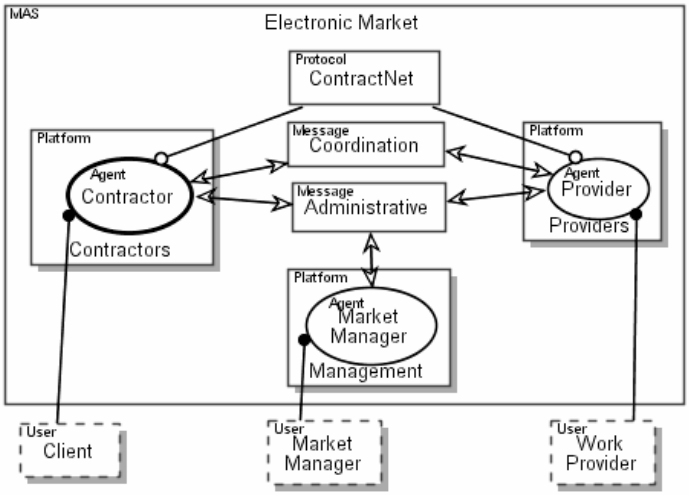


Figure 9. Electronic Market Multi-Agent System – System Diagram

In Figure 10, the Contractor Agent is in-zoomed, showing its activities. It starts with the Specification Task, followed by messaging processes aimed at looking for possible sub-contractors. Then, the Analysis Task, which is elaborated in Figure 11, takes place. In that OPD, a specific agent is selected and the contract-net protocol takes place as depicted in Figure 8. Specifying the communication aspects within the agent functionality, each messaging process may be associated with the following building block: an incoming/outgoing ACL message, a communication act, and a

protocol. A protocol associated with a messaging process indicates that the message should follow the protocol rules, i.e., the messages' order.

In the example we present, we use the same model for specifying the protocols and other system aspects. In addition, we demonstrate the way according to which the protocol is integrated into the system specification via the messaging process. This model enables the designer to formally specify the communication aspects. This formality allows the designer to perform verification over the protocol usage within the system model. In the next section we provide an algorithm for verifying the proper use of a specified protocol with a system specification, i.e., the messages' order.

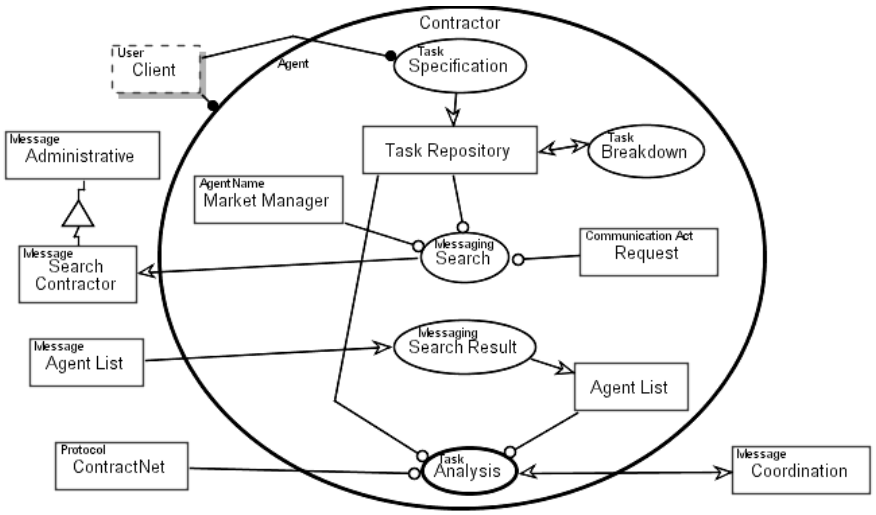


Figure 10. The Contractor Agent in-zoomed

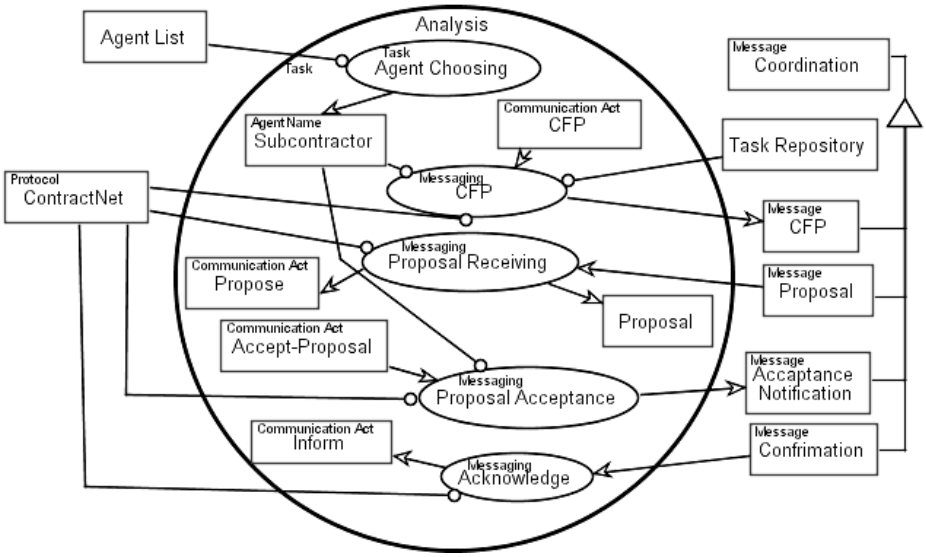


Figure 11. The Analysis Task in-zoomed

2.3. Verifying Protocol Usage in OPM/MAS

The verification of the protocol usage within a system may occur twice: once during the design phase to check whether the designer follows the protocol rules within the model, and once during run time as part of the implementation. In this section we focus on the former.

In the OPM/MAS model the communication aspects are specified using two main elements: a protocol and a message. These can be easily tracked, since they are explicitly marked as domain labels within the system model. For example, the act of sending or receiving a message is determined by the messaging domain label. The verification algorithm uses these domain labels to gather the information regarding the communication aspects. The algorithm has two stages: the communication information gathering and the protocol rules checking.

1. Communication Information Gathering

At this stage the algorithm gathers all communication information within the agent flow. That is, all the messages according to their order are collected. In OPM/MAS the order of the processes is determined according to the vertical position of the process from top to bottom in each OPD. A process may terminate when all of its subprocesses have terminated. The collection process should gather all information related to the messaging process, that is, its name, the communication act associated with it, the agent name associated with it, and the protocol associated with it. Note that the algorithm filters out messages that are not associated with a protocol. In our example, the Search and the Search Result messages should be omitted, and the results of this stage will be the following:

Name	Level	Communication Act	Agent Name	Protocol
CFP	1	CFP	Subcontractor	ContractNet
Proposal Receiving	1	Propose		ContractNet
Proposal Acceptance	1	Accept-Proposal	Subcontractor	ContractNet
Acknowledge	1	Inform		ContractNet

In case of concurrent messaging processes specified within the model, a hierarchy of these messages will be added and the message level will be determined.

2. Protocol Rules Checking

At this stage the algorithm checks the messaging order according to their associated protocol.

For each message a usage check is performed. This check verifies that the communication act associated with the messaging process exists within the associated protocol. If it does not exist, the algorithm reports an error.

When the algorithm finds a messaging process associated with the first communication act defined within a protocol, an instance of that protocol is created. Note, that this instance is not part of the design, but rather an instance within the verification mechanism. That instance will accumulate all messages until the last message of a protocol is checked. In our example, when traversing the messages table and reaching the CFP messaging process a ContractNet protocol instance is created. In case that another CFP messaging process appears in the table, a new instance of the ContractNet protocol will be created.

Then, the algorithm continues to the next message, checking whether it is associated with the next valid communication act as specified within the protocol. In case the message follows the protocol rules the algorithm continues to the next message and in case of a message that does not follow the protocol rules, the algorithm notifies an error and continues to the next message. In our example, the next message is proposal receiving, which is associated with the propose communication act. That message follows the protocol rules, thus the algorithm continues to the next message until it checks all messages.
In case of ambiguity, due to a lack of information (e.g., a missing communication act or a missing agent name) within system specification, the algorithm notifies of an error and continues to the next message.

3. Comparison of Communication Aspects Specification within Agent-Oriented Methods

Thus far, we have exemplified the modeling capabilities of OPM/MAS with respect to the specification of protocols and messages. In this section we compare the MAS communication specification capabilities of GAIA, MaSE, AUML, and OPM/MAS. Our comparison is summarized below and is based on a set of externally defined criteria, partially taken from [11]:

1. **Reusability:** is the capability to utilize the same interaction protocol when a new agent or role is introduced.
   GAIA: In GAIA, the protocol participants are roles. Thus, when introducing a new agent there is no need to change a protocol. Yet, when introducing new roles, new interactions, that is, protocols, should be defined. This reduces the reusability of GAIA with respect to the communication aspects.
   MaSE: In MaSE, the protocol participants are agents. Thus, for every combination of two coordinating agents there is a need to define the protocol. Thus, MaSE is lacking in reusability of protocols.
   AUML: In AUML, the protocol participants are roles. Thus, when introducing a new agent there is no need to change a protocol. Yet, when introducing new roles, new interactions should be defined. This reduces the reusability of AUML with respect to the communication aspects.
   OPM/MAS: In OPM/MAS, the protocol participants are not declared. Thus, the protocol can be utilized for any combinations of agents/roles, without the need to redefine them.
2. **Expressiveness:** is a capability of presenting system concepts. In this paper we refer to the following aspects:
   a. **Synchronization:** is the ability to define whether an agent has to wait for an answer or it can continue without waiting.
      GAIA: This aspect is not dealt with within GAIA.
      MaSE: Synchronization is specified within the communication class diagram.
      AUML: Synchronization is specified using a special notation.

OPM/MAS: Synchronization is specified within the regular agent control flow. In OPM/MAS, the default mode of a message is asynchronous. In case of specifying a synchronous message there is a need to define an additional message process that accepts the response from the sending agent.

b. **Concurrency:** is the ability to define the sending and receiving of multiple messages at the same time.
   GAIA: This aspect is not dealt with within GAIA.
   MaSE: In MaSE, the basic assumption is that all of the communication classes and tasks are concurrent.
   AUML: In AUML, concurrency is achieved by using multiple lifelines within the agent interaction protocol diagram.
   OPM/MAS: In OPM/MAS, concurrency is specified within the regular agent control flow. That is, a messaging process is a behavioral element within OPM/MAS, thus its graphical position within the diagram determines its execution order, whether sequential or concurrent.

c. **Looping:** the ability to define the sending of a set of messages many times, depending on a specific condition.
   GAIA: This aspect is not dealt with within GAIA.
   MaSE: Loops are specified using the behavior specification.
   AUML: Loops are specified using a special notation.
   OPM/MAS: Loops are specified within the regular agent control flow.

d. **Temporal constraints:** the ability to specify time constraints over messages.
   GAIA: This aspect is not dealt with within GAIA.
   MaSE: Temporal constraints can be specified using the behavior specification.
   AUML: Temporal constraints are specified using a special notation. For example, it can be specified using conditions over the messages.
   OPM/MAS: Temporal constraints can be specified within the regular agent control flow. For example, it can be specified using the event mechanism and conditions.

e. **Message dependencies:** the ability to define relations among messages.
   GAIA: Message dependencies are specified using arrows, which determined the messages' order.
   MaSE: Message dependencies are specified using the behavior specification via the finite state automaton.
   AUML: Message dependencies are specified using the blocks on the lifelines.
   OPM/MAS: Message dependencies are specified using the structural relationships between messages and message groups within a protocol specification. For example, in Figure 8, the Decision Message Group is dependent on a propose message.

f. **System specification integration:** is the ability to integrate the protocol specification into the system specification.
   GAIA: In GAIA, protocols are integrated within the responsibilities of a role.
   MaSE: In MaSE a protocol is a part of each agent specification.
   AUML: In AUML, the integration of the protocol within the entire system model is unclear. There are no guidelines for integrating communication aspects into the system models.

OPM/MAS: In OPM/MAS all communication aspects are part of the single unified model.

3. **Accessibility:** is the ease of specifying a protocol and understanding it.
   GAIA: GAIA specifications are easy to use and understand.
   MaSE: MaSE communication specification is a bit confusing because of the similarity to the task diagram and their appearance in several models.
   AUML: In AUML, protocols are easy to model and understand. Yet, understanding the overall system and the use of the protocol specification is difficult.
   OPM/MAS: In OPM/MAS, protocols are easy to define and understand using the static elements of OPM/MAS. In addition, it is clear how to integrate the communication aspects into the entire system model.
4. **Validability:** is the ability to employ algorithms or tools for validating the correctness and proper use of protocols and messages.
   GAIA: In GAIA this issue is not dealt with.
   MaSE: In MaSE, there is a validation algorithm.
   AUML: No validation algorithm is provided.
   OPM/MAS: The validation algorithm was discussed in Section 3.3.
5. **Synthesis:** is the ability to transform the interaction specification into a code skeleton.
   GAIA: This aspect is not dealt with within GAIA.
   MaSE: Using agentMOM [3], a skeleton code can be produced out of MaSE specifications.
   AUML: There are several studies and tools showing the capability of transforming UML models into code skeleton. Yet, there are no guidelines for transforming the protocol diagrams into code skeleton.
   OPM/MAS: OPM/MAS is accompanied by a special tool called Generic Code Generation (GCG) that enables the designer to set up rules for transformation and enables to get OPM/MAS specification and transforms it to code skeleton following the pre-defined rules. For example, when considering JADE[10], which is a commonly used platform for developing multi-agent systems, as the target platform, one can transform the protocol specification into a special behavior supporting the interaction protocols or transform each message to a special ACLMessage behavior such as SenderBehaviour or RecieverBehaviour.

Summarizing our evaluation we found that GAIA does not address many of the above criteria. MaSE addresses many of them but lacks adequate reusability and accessibility. The use of AUML agent interaction protocol diagram is easy, yet it lacks in integrating the protocols into the overall system specification. OPM/MAS proposes a new method for handling protocols, which is both accessible and expressive. It also enhances reusability and has validation rules and synthesis capabilities.

4. Summary

In this paper, we present a new approach for specifying communication aspects of multi-agent systems. This new approach overcomes limitations of previous methods

for specifying communication aspects of MAS. In particular, protocol reusability, protocol specification synthesis and validability, and weaving the communication specification into the system specification, are addressed. We utilize OPM/MAS to resolve the shortcomings of existing methods with respect to these aspects. We do so by formally defining the communication-related building blocks and their relationships, in a way that a protocol and a message are independently defined, thus increasing reusability. The formal definition of these building blocks and their relationships allows for protocol specification, synthesis and validability. In addition, utilizing the OPM/MAS single model, we avoid the integration problem. We demonstrate the use of OPM/MAS for specifying MAS communication aspects, provide a verification algorithm and evaluate the new approach via a featured-based comparison to three other methods: GAIA, MaSE, and AUML. Our evaluation shows that OPM/MAS addresses well the evaluation criteria. This result shows the advantages of using OPM/MAS as a modeling method for MAS. It may also enhance extensions and modifications of other methods to address shortcomings found in this study.

References

1. AUML, http://www.auml.org, 2003.
2. Bauer B., Muller J. P., and Odell J., "Agent UML: A Formalism for Specifying Multiagent Software Systems", The International Journal of Software Engineering and Knowledge Engineering, 11 (3), pp. 207-230, 2001.
3. DeLoach S. A, agentMom User Manual, 2001.
4. DeLoach S. A., Wood M. F., and Sparkman Cl. H., "Multiagent Systems Engineering", The International Journal of Software Engineering and Knowledge Engineering, 11 (3), pp. 231-258, 2001.
5. DeLoach S. A. and Wood M., "Developing Multiagent Systems with agentTool", Proceedings of The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL), LNCS 1986, Springer Verlag, pp. 46-60, 2001.
6. Dori D., Object-Process Methodology - A Holistic Systems Paradigm, Springer Verlag, 2002.
7. FIPA, www.fipa.org, 2003.
8. FIPA, "FIPA Contract Net Interaction Protocol Specification",http://www.fipa.org/specs/fipa00029/, 2003.
9. Huget M.-P., and James Odell, "Representing Agent Interaction Protocol with Agent UML", Proceeding of the Fifth International Workshop on Agent-Oriented Software Engineering, 2004.
10. JADE, http://sharon.cselt.it/projects/jade/, 2003.
11. Koning J-L., Huget M-P., Wei J., and Wang X., "Extended Modeling Languages for Interaction Protocol Design", Proceeding of the Second International Workshop on Agent-Oriented Software Engineering,. LNCS 2222, Springer-Verlag, pp. 68-83, 2002.
12. OMG -MOF, "Meta-Object Facility (MOF™)", version 1.4, 2002.
13. Sturm A., Dori D., and Shehory O., "Single-Model Method for Specifying Multi-Agent Systems", Proceeding of Second International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 121-128, 2003.

14. Zambonelli F., Jennings N. R. and Wooldridge M. , "Developing multiagent systems: the Gaia Methodology", ACM Trans on Software Engineering and Methodology 12 (3), pp. 317-370, 2003.