

A UML Profile for Agent-Oriented Modeling

Gerd Wagner

Eindhoven Univ. of Technology, Faculty of Technology Management,
G.Wagner@tm.tue.nl,
<http://tmitwww.tm.tue.nl/staff/gwagner>

Abstract. We present a UML profile for an agent-oriented modeling approach called *Agent-Object-Relationship (AOR)* modeling, where an entity is either an agent, an event, an action, a claim, a commitment, or an ordinary object, and where special relationships between agents and events, actions, claims and commitments supplement the fundamental association, aggregation/composition and generalization relationships of UML class models.

1 Introduction

In [Wag02], we have proposed an agent-oriented modeling language for the analysis and design of organizational information systems, called *Agent-Object-Relationship* modeling language (AORML). In the AORML, an entity is either an agent, an event, an action, a claim, a commitment, or an ordinary object. Special relationships between agents and events, actions, claims and commitments supplement the fundamental association, generalization and aggregation relationships of UML class models. AORML can be viewed as an extension of the Unified Modeling Language (UML). We believe that AORML, by virtue of its agent-oriented categorization of different classes, allows more adequate models of organizations and organizational information systems than plain UML.

Casting the AOR metamodel as a UML profile allows AOR models to be notated using standard UML notation. This means that most UML tools (specifically the ones that support the extension mechanisms of UML, such as stereotypes and tagged values) can be used to define AOR models. Standard practice for defining UML profiles has been adopted. A mapping of AOR metamodel classes to their base UML classes, with accompanying stereotypes, tagged values and constraints is presented. An implementation of this mapping can be used, for example, to generate XMI metadata conforming to the AOR metamodel from models notated using the UML profile. Specialized AOR tools will more likely directly use the AOR metamodel rather than the UML profile as a basis for storing and manipulating models.

There are two basic types of AOR models: external and internal ones. An external AOR model adopts the perspective of an external observer who is observing the (prototypical) agents and their interactions in the problem domain under consideration. In an internal AOR model, we adopt the internal (first-person) view of a particular agent to be modeled. This distinction suggests the following

system development path: in the analysis phase, draw up an external AOR model of the domain under consideration including one or more focus agents (this yields a domain model); in the design phase, for each focus agent, transform the external AOR model into an internal one according to the agent's perspective (this is called "internalization", resulting in a functional design model); then, refine the internal AOR model of each focus agent into an implementation model for the target language (such as SQL or Java). A complete internal AOR model is a formal specification of a high-level state transition system, where perception, reaction and commitment/claim handling provide the basic transition types.

Current information system technologies are largely based on the Entity-Relationship (ER) metamodel of [Che76] and the Relational Database (RDB) model of [Cod70]. Driven both by the inherent shortcomings of relational databases and the success of the object-oriented (OO) programming paradigm, concepts and techniques from OO programming are now increasingly applied in the area of information systems. However, there is no single generally acknowledged model of object-orientation, and OO programming differs from object-oriented information systems engineering in various respects. In OO programming, all software artifacts – both static (passive objects) and dynamic (processes, called 'active objects') – are treated as objects, from GUI push buttons to entire server programs, while in object-oriented information systems, typical examples of objects are customers, bank accounts and other 'business objects' which are represented in (object-)relational database tables, but (business) processes are not considered to be objects.

The UML does not support the concept of an agent as a first class citizen. In the UML, there is a certain ambiguity with respect to the agent concept. Human and artificial agents, if they are 'users' of a system, are called *actors* being involved in *use cases* but remaining external to the system model, while software agents within the boundaries of the system are called 'active objects'. In the UML, the customers and the employees of a company would have to be modeled as 'objects' in the same way as rental cars and bank accounts, while in the AOR approach they would be modeled as institutional or human agents to be represented in the system of that company (which itself could be modeled as an artificial agent).

Since interaction between agents takes place in a social context, deontic concepts such as commitments and claims with respect to external agents, and rights and duties with respect to internal agents, are essential for understanding and controlling coherent interaction between agents and other systems. Neither ER modeling nor UML provide any means to account for the deontic aspects of an information system.

In AOR modeling, only agents can communicate, perceive, act, make commitments and satisfy claims. Objects do not communicate, cannot perceive anything, are unable to act, and do not have any commitments or claims.

1.1 Agents

The agent metaphor subsumes both natural and artificial systems. A formal agent concept for the purpose of representing agents in an information system, and for agentifying information systems, may abstract away from many of the higher-level cognitive aspects of human agents. It only needs to capture those aspects that are relevant for realizing the interactions of interest. In an enterprise information system, for instance, only perceptions (in the form of signals), beliefs, memories and commitments associated with business processes are of interest.

According to our ontological distinction between agents and objects, only agents can perceive events, perform actions, communicate, or make commitments. Ordinary objects are passive entities with no such capacities. This contrasts with the language used in the literature on object-oriented programming, where objects ‘communicate’ or ‘interact’ with each other by sending ‘messages’. Notice that the UML term ‘collaboration’ between objects corresponds only to a very low-level sense of communication and interaction. In fact, sending a ‘message’ in the sense of OO programming corresponds rather to a (possibly remote) procedure call, and not to a communication act (or *speech act*): while an OO message has no generic structure at all, a speech act message has the mandatory form $m(c)$ where m is the message type (expressing the ‘illocutionary force’), and c is the message content (composed of propositions and/or action terms).

Conceptually, it is therefore not justified to model customers and suppliers as ‘business objects’ in the same way as bank accounts and software artifacts (such as GUI push buttons). Object-orientation does not capture communication and interaction in the high-level sense of business processes. Our view is shared by Jacobson [Jac94] who remarks (p.36) that “it is bizarre to apply the way of thinking that governs computer systems to business processes”.

1.2 Information Systems as Agents

The following definition of [HR95] summarizes the most important features of agency: *Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions.*

In the case of an agentified information system,

1. ‘*perception of dynamic conditions in the environment*’ refers to incoming messages representing communication events (such as receiving a request for a sales quotation or an acknowledgment of a sales order) and to incoming signals representing environment events (such as receiving a payment);
2. ‘*action to affect conditions in the environment*’ refers to communication acts of the agentified IS (such as acknowledging a sales order) and to physical acts (such as delivering goods or making a payment);
3. finally, ‘*reasoning to interpret perceptions, solve problems, draw inferences, and determine actions*’ refers to things like the proper processing of incoming

messages, the computational inference of correct answers to queries, and the determination of proper actions (such as locking all sales orders of a customer whose credibility is in question or issuing an alert when the fulfillment of a commitment is overdue).

An information system may be explicitly designed as an agent by

1. treating its information items as its *beliefs* or *knowledge*;
2. adding further mental components such as *perceptions* (in the form of incoming messages and signals), *memory*, and *commitments*;
3. providing support for agent-to-agent communication on the basis of a standard agent communication language.

2 AOR Models

While plain UML supports the design of *object-oriented* information systems realized with the help of relational and object-relational database technology, AORML is to support the high-level design of *agent-oriented* information systems. An AOR model consists of an *External AOR Model*, corresponding to a domain analysis model, and an *Internal AOR Model*, corresponding to a design model, as shown in Figure 1.

In an External AOR model, we adopt the view of an external observer who is observing the (prototypical) agents and their interactions in the problem domain under consideration. Typically, an External AOR model has a *focus*, that is an agent, or a group of agents, for which we would like to develop a state and behavior model.

An Internal AOR model depicts *the world* as it may be represented in the mental state of the focus agent. If the focus agent is an organization, the Internal AOR model represents its view of *the world*, and may be used to design its information system. Thus, AOR modeling suggests the following development path for organizational information systems:

1. In the domain analysis, develop an external AOR model of an organization (or a group of organizations) and its (or their) environment from the perspective of an external observer of the scenario.
2. Transform the external AOR model into an internal AOR model for the focus agent for that an information system is to be developed (typically an organization or an organizational unit). If there are several focus agents, and for each of them an information system is to be developed, this step can be iterated.
3. Transform the internal AOR models obtained in the previous step into database design models (logical database schemas), e.g. for object-relational (SQL-99) database management systems, or into sets of corresponding logical data structure definitions in a target language such as Java.

4. Refine the design models into implementation models (physical database schemas) by taking performance and storage management issues, as well as the specific features of the target language (such as SQL-99 or Java), into consideration.
5. Generate the target language code.

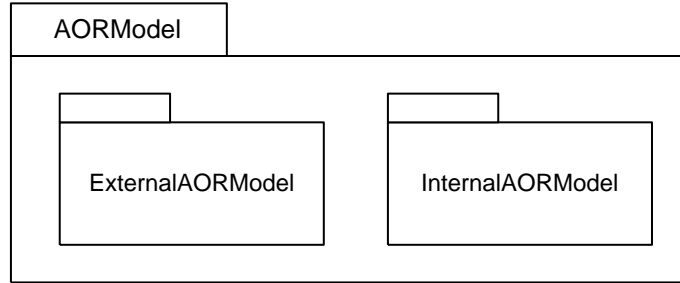


Fig. 1. An AORModel consists of an ExternalAORModel (corresponding to a domain analysis model) and an InternalAORModel (corresponding to a design model).

The meta-concepts of AOR modeling that are common to both Internal and External AOR modeling are listed as UML *stereotypes* in Table 1.

<i>Stereotype</i>	<i>Base Class</i>	<i>Parent</i>	<i>Constraints</i>
AORModel	Model	NA	
Agent	Class	NA	Restricted generalization.
BiologicalAgent	Class	Agent	Restricted generalization.
HumanAgent	Class	BiologicalAgent	Restricted generalization.
ArtificialAgent	Class	Agent	Restricted generalization.
SoftwareAgent	Class	ArtificialAgent	Restricted generalization.
Robot	Class	ArtificialAgent	Restricted generalization.
EmbeddedSystem	Class	ArtificialAgent	Restricted generalization.
InstitutionalAgent	Class	Agent	Restricted generalization.
Organization	Class	InstitutionalAgent	Restricted generalization.
OrganizationalUnit	Class	InstitutionalAgent	Restricted generalization.
Object	Class	NA	Restricted generalization.

Table 1. A summary of the stereotypes that are common to both Internal and External AOR modeling. *Restricted generalization* means that whenever a generalization relationship involves a class of that stereotype as either subclass or superclass, the other class involved must also be of that stereotype.

2.1 Object Classes

Object classes, such as sales orders or product items, are visualized as rectangles essentially in the same way like entity types in ER diagrams, or object classes in UML class diagrams. They may participate in *association*, *generalization* or *aggregation/composition* relationships with other object classes, and in *association* or *aggregation/composition* relationships with agent classes.

2.2 Agent Classes

We distinguish between *artificial* agents, *human* agents and *institutional* agents.¹ Examples of human agent classes are Person, Employee, Student, Nurse, or Patient. Examples of institutional agents are organizations, such as a bank or a hospital, or organizational units. An institutional agent consists of a number of internal agents that perceive events and perform actions on behalf of it, by playing certain *roles*.

In certain application domains, there may also be artificial agent classes, such as software agents (e.g., involved in electronic commerce transactions), embedded systems (such as automated teller machines), or robots. For instance, in an automated contract negotiation or in an automated purchase decision, a legal entity may be represented by an artificial agent. Typically, an artificial agent is owned, and is run, by a legal entity that is responsible for its actions.

In AOR diagrams, an agent class is visualized as a rectangle with rounded corners. Icons indicating a single human, a group, or a robot may be used for visualizing the distinction between human, institutional and artificial agent.

Agents may be related to other entities by means of ordinary domain relationships (associations). In addition to the designated relationships *generalization* and *composition* of ER/OO modeling, there are further designated relationships relating agents with events, actions and commitments. They are discussed below.

2.3 External and Internal Agents

With respect to an institutional agent, one has to distinguish between external and internal agents. Internal agents, by virtue of their contractual status (or ownership status, in the case of artificial internal agents), have certain rights and duties, and assume a certain position within the subordination hierarchy of the institution they belong to. In the case of a hospital, examples of human internal agents are doctors and nurses; examples of artificial internal agents are communication-enabled information systems and agentified embedded systems, such as patient monitoring systems.

¹ Notice that we do not distinguish between ‘agents’ and ‘actors’. Both terms denote the same concept. By default, we use the term ‘agent’.

2.4 Commitments and Claims

Representing and processing commitments and claims in information systems explicitly helps to achieve coherent behavior in (semi-)automated interaction processes. In [Sin99], the social dimension of coherent behavior is emphasized, and commitments are treated as ternary relationships between two agents and a ‘context group’ they both belong to. For simplicity, we treat commitments as binary relationships between two agents.

Commitments to perform certain actions, or to see to it that certain conditions hold, typically arise from certain communication acts. For instance, sending a sales quotation to a customer commits the vendor to reserve adequate stocks of the quoted item for some time. Likewise, acknowledging a sales order implies the creation of a commitment to deliver the ordered items on or before the specified delivery date.

Obviously, a commitment of a_1 (the debtor) towards a_2 (the creditor) to do the action α is mirrored as a claim of a_2 against a_1 to create the action event α .

3 External AOR Models

In the external-observer-view adopted in external AOR models, *the world* (i.e. the application domain) consists of various types of

1. *agents*,
2. communicative and non-communicative *action events*,
3. *non-action events*,
4. *commitments/claims* between two agent classes,
5. ordinary *objects*,
6. various *designated relationships*, such as *sends* and *does*,
7. ordinary *associations*.

In the view of an external observer, actions are also events, and commitments are also claims, exactly like two sides of the same coin. Therefore, an external AOR model contains, besides the agent and object classes of interest, the action event classes and commitment/claim classes that are needed to describe the interaction between the focus agent(s) and the other types of agents.

Object classes, in an external AOR model, belong to one or several agents (or agent classes). They define containers for beliefs. If an object class belongs exclusively to one agent or agent class (in the sense of a private belief type), the corresponding rectangle is drawn inside this agent (type) rectangle. If an object class represents beliefs that are shared among two or more agents (or agent classes), the object class rectangle is drawn outside the respective agent (type) rectangles.

An external AOR model does not include any software artifacts. It rather represents a conceptual analysis view of the problem domain and may also contain elements which are merely descriptive and not executable by a computer program (as required for enterprise modeling).

An external AOR model may comprise one or more of the following diagrams:

Agent Diagrams depicting the agent classes of the domain, certain relevant object classes, and the relationships among them (an example is shown in Figure 3).

Interaction Frame Diagrams depicting the action event classes and commitment/claim classes that determine the possible interactions between two agent classes (or instances).

Interaction Sequence Diagrams depicting proto-typical instances of interaction processes.

Interaction Pattern Diagrams focusing on general interaction patterns expressed by means of a set of reaction rules defining an interaction process type.

The agent diagrams, interaction frame diagrams and interaction pattern diagrams of a model may be merged into a single all-encompassing *External AOR Diagram (EAORD)*. Interaction sequence diagrams are normally not included in such an EAORD, since they depict instances only, and are not at the type level.

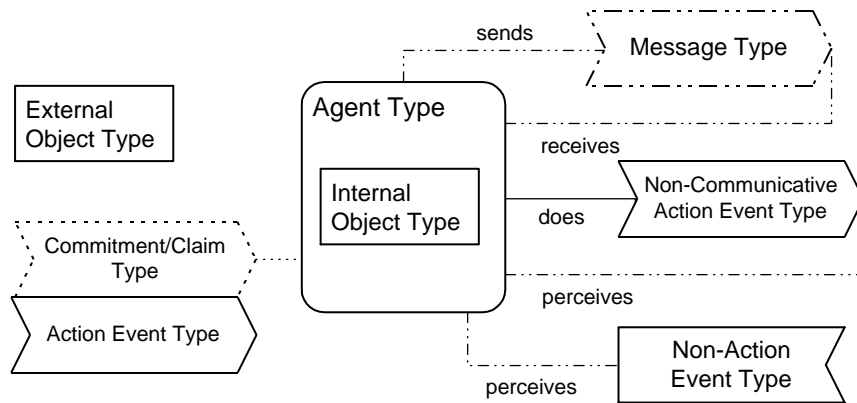


Fig. 2. The core elements of External AOR modeling.

Each agent has beliefs about its internal agents, about its objects, and about all external agents and shared objects that are related to it.

3.1 Actions Are Events but Not All Events are Actions

In the external observer perspective, all actions of agents are at the same time also events that may be perceived by other agents. The other way around, there are many events that are created by the corresponding actions of agents. However, there are also events which are not created by actions (e.g., temporal events, or events created by natural forces). Consequently, we make a distinction between *action events* and *non-action events*.

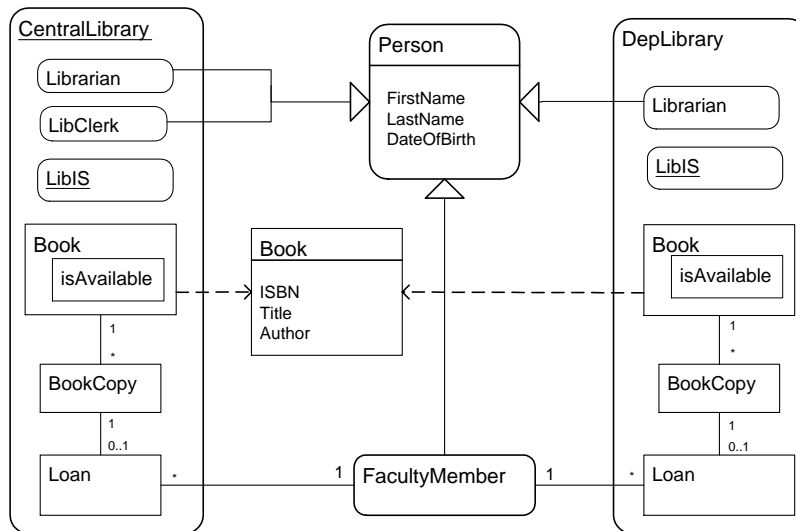


Fig. 3. An AOR agent diagram for the university libraries domain. The central library and the department libraries are institutional agents, having librarians as human internal agents and a library information system as an artificial internal agent. Also, **FacultyMember** is an agent class in this domain. Important object classes are **Book**, **BookCopy** and **Loan**, that is, libraries have beliefs about their books, their book copies, and their loans. The subclass **Book.isAvailable** is formed by all books that satisfy the status predicate **isAvailable**, that is, for which there is at least one book copy available.

In an External AOR Diagram, an action event type is graphically rendered by a special arrow rectangle where one side is an incoming arrow linked to the agent (or agent class) that performs this type of action, and the other side is an outgoing arrow linked to the agent (or agent class) that perceives this type of event. Communicative action event rectangles have a dot-dashed line. In the case of a non-action event, the corresponding event rectangle does not have an outgoing arrow (see Figure 4).



Fig. 4. A communicative action event, a non-communicative action event, and a non-action event.

<i>Stereotype</i>	<i>Base Class</i>	<i>Parent</i>	<i>Constraints</i>
Event	Class	NA	Restricted generalization. No aggregation.
Action Event	Class	Event	Restricted generalization. No aggregation.
Communicative Action Event	Class	Action Event	Restricted generalization. No aggregation.
Non-Communicative Action Event	Class	Action Event	Restricted generalization. No aggregation.
NonActionEvent	Class	Event	Restricted generalization. No aggregation.
CommitmentClaim	Class	NA	Restricted generalization. No aggregation.
does	Association	NA	The domain class must be an agent type and the range class must be a non-communicative action event type. Multiplicity is one-to-many.
perceives	Association	NA	The domain class must be an agent type and the range class must be a non-communicative action event type or a non-action event type. Multiplicity is one-to-many.
sends	Association	NA	The domain class must be an agent type and the range class must be a communicative action event type. Multiplicity is one-to-many.
receives	Association	NA	The domain class must be an agent type and the range class must be a communicative action event type. Multiplicity is one-to-many.
hasClaim	Association	NA	The domain class must be an agent type and the range class must be a commitment/claim type. Multiplicity is one-to-many.
hasCommitment	Association	NA	The domain class must be an agent type and the range class must be a commitment/claim type. Multiplicity is one-to-many.

Table 2. A tabular definition of the stereotypes of external AOR modeling. *No aggregation* means that classes of that stereotype must not participate in any aggregation.

3.2 Commitments/Claims

In external AOR modeling, a *commitment* of agent a_1 towards agent a_2 to perform an action of a certain type (such as a commitment to deliver an item) can also be viewed as a *claim* of a_2 against a_1 that an action event of that type will happen. Commitments/claims are conceptually coupled with the type of action event they refer to (such as *deliverItem* action events). This is graphically rendered by an arrow rectangle with a dotted line on top of the action event rectangle it refers to, as depicted in Figure 2.

3.3 Interaction Frame Diagrams

In an external AOR model, there are four types of designated relationships (association stereotypes) between agents and action events: **sends** and **receives** are associations that relate an agent with communicative action events, while **does**

and *perceives* are associations that relate an agent with non-communicative action events. In addition, there are two types of associations between agents and commitments/claims: *hasCommitment* and *hasClaim*. These association stereotypes are visualized with particular connector types as depicted in Figure 5.

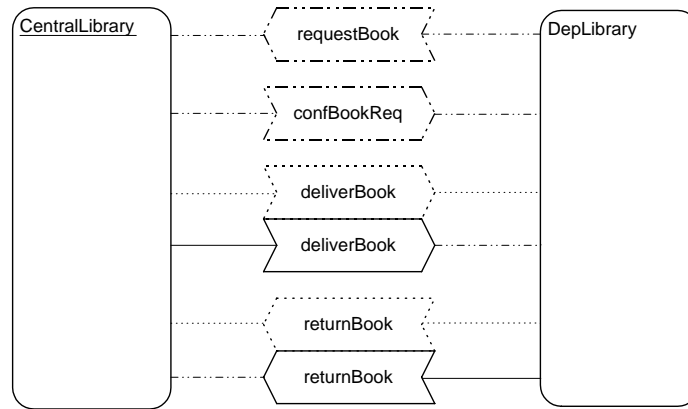


Fig. 5. The interaction frame between the central library and the department libraries: a department library may request a book from the central library; when such a book request has been confirmed by the central library, then there is a commitment to deliver the requested book (visualized by the dashed-line *deliverBook* arrow rectangle); normally, such a commitment leads to a corresponding action (visualized by the solid-line *deliverBook* arrow rectangle); after a book has been delivered, there is a commitment to return it in due time.

An *interaction frame diagram*, in an external AOR model, describes the possible interactions between two (types of) agents. It consists of various types of

1. communicative action events,
2. non-communicative action events,
3. commitments/claims (coupled with the corresponding types of action events),
and
4. non-action events.

An example of an interaction frame diagram is shown in Figure 5.

3.4 Interaction Sequence Diagrams

An interaction sequence diagram depicts (some part of) an instance of an interaction process. An *interaction process* is a sequence of action events and non-action events, performed and perceived by agents, and following a set of rules (or protocol) that specifies the type of the interaction process. Agents may interact with their inanimate environment, or they may interact with each other. A simple

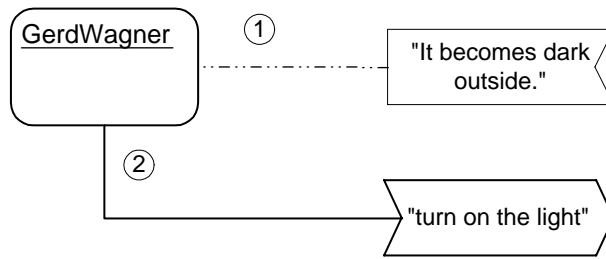


Fig. 6. A non-social interaction process involving an agent and his inanimate environment.

example of the former type of interaction process is my reaction to turn on the light in my office when it becomes dark outside, that is depicted in Figure 6.

A *social interaction process* is a temporally ordered, coherent set of action events and non-action events, involving at least one communicative action event, performed and perceived by agents, and following a set of rules, or protocol, that is governed by norms, and that specifies the type of the interaction process.² An example of a social interaction process is shown in Figure 7. Social norms imply, for instance, that after having confirmed a book request, the library is committed to deliver the requested book.

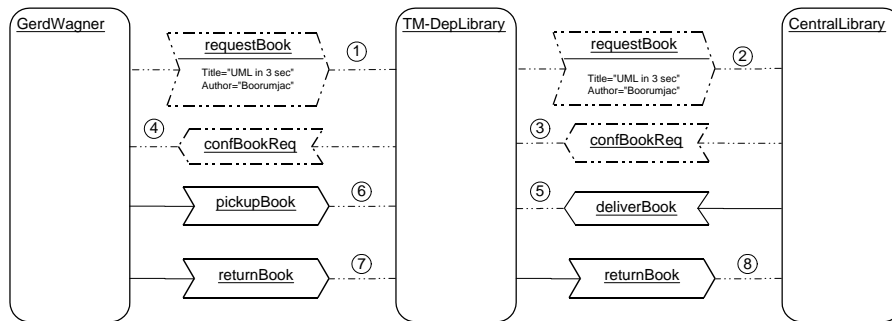


Fig. 7. A social interaction process involving three agents. It is an option to display the object classes referred to in the arguments of messages and action events within the agents that have to deal with them.

We consider a business process as a special kind of a *social interaction process*. Unlike physical or chemical processes, social interaction processes are based on communication acts that may create commitments and are governed by norms.

² Notice that we did not choose *activities* as the basic elements of a process. While an *action* happens at a time instant (i.e., it is immediate), an *activity* is being performed during a time interval (i.e., it has duration), and may consist of a sequence of actions.

We distinguish between an interaction process type and a concrete interaction process (instance), while in the literature the term ‘business process’ is ambiguously used both at the type and the instance level. Social interaction process types are modeled in *Interaction Pattern Diagrams* with the help of *reaction rules* (see [TW01,Wag02]).

4 Internal AOR Models

In an *internal* AOR model, we adopt the internal view of a particular agent to be modeled. In this first-person-view, *the world* (i.e. the application domain) consists of various types of

1. other *agents*,
2. *actions*,
3. *commitments* towards other agents to perform certain actions,
4. *events*,
5. *claims* against other agents that certain action events happen,
6. ordinary *objects*,
7. various *designated relationships*, such as *isSentTo* and *isPerceivedBy*,
8. ordinary *associations*.

These meta-classes of internal AOR modeling are listed as stereotypes in Table 3.

An internal AOR model may comprise one or more of the following diagrams:

Reaction Frame Diagrams depicting other agents (or agent classes) and the action and event classes, as well as the commitment and claim classes that determine the possible interactions with them.

Reaction Sequence Diagrams depicting proto-typical instances of interaction processes in the internal perspective.

Reaction Pattern Diagrams focusing on the reaction patterns of the agent under consideration expressed by means of reaction rules.

The reaction frame diagrams and reaction pattern diagrams of a model may be merged into a single all-encompassing *Internal AOR Diagram (IAORD)*. Reaction sequence diagrams are normally not included in such an IAORD, since they depict instances only, and are not at the type level.

In an IAORD, internal agents (or agent classes) of the agent to be modeled are distinguished from external agents by dashing their rectangle line.

4.1 Actions and Events

In the perspective of an institutional agent, only the actions of internal agents performed on behalf of the institution count as actions, while the actions of external agents count as events.

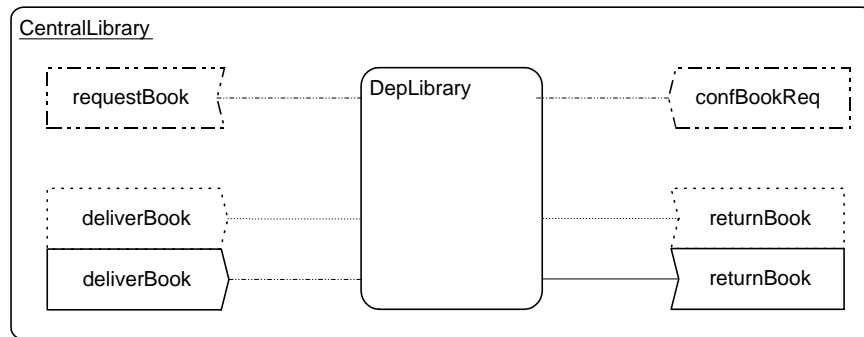


Fig. 8. In an internal AOR model of the central library, a `requestBook` message isReceivedFrom a department library, an `confBookReq` message isSentTo the department library, a `deliverBook` action isPerceivedBy the department library, and a `returnBook` event isCreatedBy a department library. A commitment towards the department library (to deliver a book) is modeled together with the associated action (of carrying out the delivery). A claim against the department library (to return a book) is modeled together with the associated event (of returning the book).

An event type is graphically rendered by a rectangle with an incoming arrow side, while an action type is graphically rendered by a rectangle with an outgoing arrow side. Communication event classes and communication act classes are visualized by a dashed-dotted line.

In an internal model of an agent, events and actions are related to other agents by means of four association stereotypes: an outgoing message isSentTo – and an incoming message isReceivedFrom – another agent, while a physical action isPerceivedBy – and an action event isCreatedBy – another agent. These association stereotypes are graphically rendered with particular connector types as illustrated in Figure 8.

4.2 Commitments and Claims

A commitment towards another agent (such as a commitment towards a customer to deliver an item) is coupled with the associated action (such as a *deliverItem* action) to be performed. It is visualized as a rectangle with a dotted line on top of the associated action rectangle as in Figure 8.

A claim against another agent (such as a claim against a customer to pay an invoice) is coupled with the associated event (such as a *payInvoice* event). It is visualized as a rectangle with a dotted line on top of the associated event rectangle as in Figure 8.

4.3 Interaction Frames

In an internal AOR model, the interaction frame with respect to an agent class *A* is described by means of six kinds of classes:

1. **communication events**, or incoming messages, created by the communication acts of instances of *A*;
2. **communication acts**, or outgoing messages, directed to instances of *A*;
3. **claims** against instances of *A*;
4. **non-communicative events** created by the actions of instances of *A*;
5. **commitments** towards instances of *A*;
6. **non-communicative actions** which may be performed in order to fulfill corresponding commitments towards instances of *A*, or in response to events within the interaction frame.

Figure 8 shows a Reaction Frame Diagram for the central library describing the interaction frame with respect to the agent class `DepLibrary`. There is the incoming message type `requestBook`, the outgoing message type `confBookReq`, the commitment type `deliverBook` coupled with the corresponding action type, and the claim type `returnBook` coupled with the corresponding event type.

5 Related Work

Some predefined UML ‘stereotypes’ come quite close to some of the AOR meta-concepts:

Signals are defined as a class stereotype. They correspond to some degree to a communicative action event (or message) type in external AOR models. For activity diagrams, there are two signal symbols: one for sent signals, and one for received signals, corresponding to the AOR distinction between communication acts (outgoing messages) and communication events (incoming messages). Strangely, the receipt of a signal is treated as an action that may follow any activity (which seems to denote the special action type *wait for signal*).

Active objects are another example of a class stereotype. An active object is an object that “owns a thread and can initiate control activity” (cited from the OMG Unified Modeling Language Specification). Thus, active objects are rather an implementation, and not a domain, modeling concept. In a certain sense, they form a superclass of software agents, but they do not reflect the AOR distinction between agent and object.

The UML Profile for Business Modeling The UML 1.4 standard contains a *UML Profile for Business Modeling* that defines the following class stereotypes: ‘worker’, ‘case worker’, ‘internal worker’, and ‘entity’. A `«Worker»` is “an abstraction of a human that acts within the system”. Although it is not clear what “system” means here, the concept of a worker seems to correspond to the AOR concept of an internal human agent. While an `«Internal Worker»` does not interact with actors outside the system, a `«Case Worker»` does. All other (passive) business objects are called `«Entity»`. In addition, the concepts *organization unit* and *work unit* are proposed as subsystem stereotypes. An

«Organization Unit» is “a subsystem corresponding to an organization unit of the actual business”; it “contains organization units, work units, classes (workers and entities), and relationships”; thus, it corresponds to the AOR concept of an internal institutional agent. A «Work Unit» is “a subsystem that contains one or more entities”; it is “a task-oriented set of objects that form a recognizable whole to the end user”.

The UML Profile for Business Modeling seems to be a rather ad-hoc proposal for making a distinction between active and passive ‘business objects’ and for resolving some of the conceptual difficulties arising from the UML definition of an «Actor». While it shares some of its motivations with the AOR metamodel, it is, in many respects, quite incomplete. For instance, the only specific semantics assigned to «Worker» (by means of well-formedness rules for associations) is that they may «communicate» with each other and may «subscribe» to an «Entity». However, it is not explained, what these special associations mean.

The Eriksson-Penker Business Extensions In [EP99], Eriksson and Penker propose an approach to business modeling with UML based on four primary concepts: resources, processes, goals, and rules. In this proposal, there is no specific treatment of agents. They are subsumed, together with “material, information, and products” under the concept of *resources*. This unfortunate subsumption of human agents under the traditional ‘resource’ metaphor prevents a proper treatment of many agent-related concepts such as commitments, authorization, and communication/interaction.

5.1 Agent UML

In [OvDPB00], an agent-oriented extension of UML, called *Agent UML (AUML)*, mainly concerning sequence diagrams and activity diagrams, has been proposed. However, UML class diagrams are not modified, and no distinction between agents and objects is made in AUML.

6 Conclusion

Similar to ‘object’, the term ‘agent’ denotes an abstraction that leads to more natural and more modular software concepts. It helps to capture more semantics about natural and artificial systems an information system has to represent and to interact with.

We have shown that the AOR modeling language can be viewed as a UML profile. In fact, we need to define two profiles, one for external and one for internal AOR models. It is, however, difficult to express the AOR behavior modeling construct of *reaction rules* since these rules are not expressible as UML ‘stereotypes’. So, we can only cast the AOR state modeling fragment as a UML profile. The inclusion of reaction rules for AOR behavior modeling is not supported by the UML extension mechanisms.

References

- [Che76] P. Chen. The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [Cod70] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 1970.
- [EP99] H.E. Eriksson and M. Penker. *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, 1999.
- [HR95] B. Hayes-Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence*, 72:329–365, 1995.
- [Jac94] I. Jacobson. *The Object Advantage*. Addison-Wesley, Workingham (England), 1994.
- [OvDPB00] J. Odell, H. van Dyke Parunak, and B. Bauer. Extending UML for agents. In G. Wagner, Y. Lesperance, and E. Yu, editors, *Proc. of the 2nd Int. Workshop on Agent-Oriented Information Systems*, Berlin, 2000. iCue Publishing.
- [Sin99] M.P. Singh. An ontology for commitments in multiagent systems. *Artificial Intelligence and Law*, 7:97–113, 1999.
- [TW01] K. Taveter and G. Wagner. Agent-oriented enterprise modeling based on business rules. In *Proc. of 20th Int. Conf. on Conceptual Modeling (ER2001)*, pages 527–540, Yokohama, Japan, November 2001. Springer-Verlag. LNCS 2224.
- [Wag02] G. Wagner. The Agent-Object-Relationship metamodel: Towards a unified conceptual view of state and behavior. Technical report, Eindhoven Univ. of Technology, Fac. of Technology Management, Available from <http://AOR.research.info>, May 2002. To appear in *Information Systems*.
- [YM95] E. Yu and J. Mylopoulos. From E-R to ‘A-R’ – modeling strategic actor relationships for business process reengineering. *Int. J. of Intelligent and Cooperative Information Systems*, 4(2-3):125–144, 1995.
- [Yu95] E.S.K. Yu. *Modeling Strategic Relationships for Process Reengineering*. PhD thesis, Computer Science Department, Univ. of Toronto, Toronto (Canada), 1995.

<i>Stereotype</i>	<i>Base Class</i>	<i>Parent</i>	<i>Constraints</i>
InternalAORModel	Model	NA	
Event	Class	NA	Restricted generalization. No aggregation.
ActionEvent	Class	Event	Restricted generalization. No aggregation.
Communicative ActionEvent	Class	Action	Restricted generalization. No aggregation.
NonCommunicative ActionEvent	Class	Event	Restricted generalization. No aggregation.
NonActionEvent	Class	Event	Restricted generalization. No aggregation.
Action	Class	NA	Restricted generalization. No aggregation.
Communicative Action	Class	Action	Restricted generalization. No aggregation.
NonCommunicative Action	Class	Action	Restricted generalization. No aggregation.
Commitment	Class	NA	Restricted generalization. No aggregation.
Claim	Class	NA	Restricted generalization. No aggregation.
isCreatedBy	Association	NA	The domain class must be a non-communicative action event type and the range class must be an agent type. Multiplicity is many-to-one.
isPerceivedBy	Association	NA	The domain class must be a non-communicative action type and the range class must be an agent type. Multiplicity is many-to-one.
isSentTo	Association	NA	The domain class must be a communicative action event (or message) type and the range class must be an agent type. Multiplicity is many-to-one.
isReceivedFrom	Association	NA	The domain class must be a communicative action event (or message) type and the range class must be an agent type. Multiplicity is many-to-one.
holdsAgainst	Association	NA	The domain class must be a claim type and the range class must be an agent type. Multiplicity is many-to-one.
holdsTowards	Association	NA	The domain class must be a commitment type and the range class must be an agent type. Multiplicity is many-to-one.

Table 3. A summary of the stereotypes of internal AOR modeling.