Compositional Design and Reuse of a Generic Agent Model

Article *in* Applied Artificial Intelligence · October 1999

DOI: 10.1080/088395100403397 · Source: CiteSeer

3 authors:

Frances M.T Brazier
Delft University of Technology
343 PUBLICATIONS **3,400** CITATIONS
SEE PROFILE

Catholijn M. Jonker
Delft University of Technology
541 PUBLICATIONS **6,283** CITATIONS
SEE PROFILE

Jan Treur
Vrije Universiteit Amsterdam
774 PUBLICATIONS **7,756** CITATIONS
SEE PROFILE

Compositional Design and Reuse of a Generic Agent Model

Frances M.T. Brazier, Catholijn M. Jonker, Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{frances,jonker,treur}, Email: {frances,jonker,treur}@cs.vu.nl

Abstract

This paper introduces a formally specified design of a compositional generic agent model (GAM). This agent model abstracts from specific application domains; it provides a unified formal definition of a model for weak agenthood. It can be (re)used as a template or pattern for a large variety of agent types and application domain types. The model was designed on the basis of experiences in a number of application domains. The compositional development method DESIRE was used to design the agent model GAM at a conceptual and logical level. It serves as a unified, precisely defined coneptual structure which can be refined by specialisation and instantiation to a large variety of other, more specific agents. To illustrate reuse of this agent model, specialisation and instantiation to model co-operative information gathering agents is described in depth. Moreover, it is shown how GAM can be used to describe in a unified and hence more comparable manner a large number of agent architectures from the literature.

Keywords: agent architecture, reuse, unified, generic model, template, pattern, compositional

1 Introduction

The term agent has become popular, and has been used for a wide variety of applications, ranging from simple batch jobs and simple email filters, to mobile applications, to intelligent assistants, and to large, open, complex, mission critical systems (such as systems for air traffic control). Some of the key concepts in agent technology lack universally accepted definitions. In particular, there is only partial agreement on what an agent is. For example, simple batch jobs are termed agent because they can be scheduled in advance to perform tasks on a remote machine, mobile applications are termed agent because they can move themselves from computer to computer, and intelligent assistants are termed agents because they present themselves to human users as believable characters that manifest intentionality and other aspects of a mental state normally attributed only to humans. Besides this variety in different appearances of agents, applications of agents often are concentrated on specific implementations of agents (often in Java). Often the only precise description of an agent is its implementation code, which is dependent on the chosen implementation platform. Therefore, existing agent architectures are often only comparable in an

informal manner. A principled design description of an agent at a conceptual and logical level lacks, which makes it difficult to compare agents from different applications.

As agents show a variety of appearances, perform a multitude of tasks, and their abilities vary significantly (Nwana, 1996; Nwana and Ndumu, 1998), attempts have been made to define what they have in common. In (Wooldridge and Jennings, 1995b,c) the *weak notion of agent* is introduced; this is often used as a reference. This notion will be explained in more detail in Section 2; a number of primitive concepts relevant for this type of agent are identified in Section 3. During the design of such agents, these concepts have to be incorporated, and a number of generic agent processes can be identified; for example relating to interaction with the world or to social behaviour with respect to other agents.

To obtain a unified, formally defined conceptual but implementation-independent description, Section 5 describes the compositional design of a generic agent model (GAM) at a conceptual and logical level, in which generic agent concepts and processes related to the weak agent notion are predefined. This generic agent model abstracts from specific application domains; by refinement (specialisation and instantiation) it can be (re)used as a core design for a large variety of agent types and application domains. The model was designed on the basis of experiences in applications to, among others, monitoring, diagnosis and restoration of an electricity network (Brazier, Dunin-Keplicz, Jennings and Treur, 1995) and negotiation for load balancing of electricity use (Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak and Treur, 1998). The compositional development method used to design this agent model, DESIRE, is briefly introduced in Section 4. To illustrate reuse of this agent model, an application to co-operative information gathering agents is described in more depth in Section 6. In Section 7 it is discussed how the model GAM can be used to obtain a unified, and thus comparable, description at the level of design of a large variety of agent architectures occurring in the literature. In Section 8 the paper concludes with a discussion on design and reuse of this generic agent model.

2 Agent Notions

The *weak notion of agent* was introduced In (Wooldridge and Jennings, 1995b,c) and is often used as a reference in the literature (see also (Jennings and Wooldridge, 1998a)).

2.1 Weak Notion of Agent

The weak notion of agent is a notion that requires the behaviour of agents to exhibit at least the following four types of behaviour:

- Autonomous behaviour
- Responsive behaviour (also called reactive behaviour)
- Pro-active behaviour
- Social behaviour

Autonomy relates to control: although an agent may interact with its environment, the processes performed by an agent are in full control of the agent itself. Jennings and Wooldridge (1998a) define *autonomous* behaviour as:

> … the system should be able to act without the direct intervention of humans (or other agents) and should have control over its own actions and internal state.

This means that an agent can only be requested to perform some action, and, as Jennings and Wooldridge (1998a) state:

> The decision about whether to act upon the request lies with the recipient.

Examples of autonomous processes are: process control systems (e.g., thermostats, missile guiding systems, and nuclear reactor control systems), software deamons (e.g., one that monitors a user's incoming email and obtains their attention by displaying an icon when new, incoming email is detected), operating systems.

Many processes that exhibit autonomous behaviour are being termed agents. However, if such agents do not exhibit flexible behaviour, they are not, in general, considered to be intelligent agents. An intelligent agent is defined in (Jennings and Wooldridge, 1998a) to be a computer system that is capable of flexible autonomous actions in order to meet its design objectives. Intelligence requires flexibility with respect to autonomous actions, meaning that intelligent agents also exhibit responsive, social, and pro-active behaviour.

An agent exhibits *responsive* (or *reactive*) behaviour if it reacts or responds to new information from its environment. Jennings and Wooldridge define responsive behaviour as follows:

> Agents should perceive their environment (which may be the physical world, a user, a collection of agents, the Internet, etc.) and respond in a timely fashion to changes that occur in it.

A barometer is a simple example of a system that exhibits responsive behaviour: It continually receives new information about the current air pressure and responds to this new information by adjusting its dial.

Pro-active behaviour is defined by (Jennings and Wooldridge, 1998a) as follows:

> Agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate.

Pro-active behaviour is the most difficult of the required types of behaviour for an agent defined according to the weak agent notion. For example, pro-active behaviour can occur simultaneously with responsive behaviour. It is possible to respond to incoming new information in an opportunistic manner according to some goals. Also initiatives can be taken in response to incoming new information from the environment, and thus this behaviour resembles responsive behaviour. However, it is also possible to behave pro-actively when no new information is received from the

environment. This last behaviour can by no means be called responsive behaviour. A more elaborate comparison between responsive behaviour and pro-active behaviour can found in (Jonker and Treur, 1998).

An agent exhibits *social* behaviour if it communicates and co-operates with other agents. Jennings and Wooldridge define social behaviour as follows:

> Agents should be able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities.

An example of an agent that exhibits social behaviour is a car: it communicates with its human user by way of its dials (outgoing communication dials: speed, amount of fuel, temperature) and its control mechanisms (incoming communication control mechanisms: pedals, the steering wheel, and the gears). It co-operates with its human user, e.g., by going in the direction indicated by the user, with the speed set by that user.

2.2 Other Notions of Agent

Agents can also be required to have additional characteristics. In this section three of these characteristics are discussed: adaptivity, pro-creativity, and intentionality.

Adaptivity is a characteristic that is vital in some systems. An adaptive agent learns and improves with experience. This behaviour is vital in environments that change over time in ways that would make a non-adaptive agent obsolete or give it no chance of survival. This characteristic is modelled often in simulations of societies of small agents, but also, for example, in adaptive user interface agents.

Pro-creativity is of similar importance to find agents that satisfy certain conditions. The chance of survival is often measured in terms of a fitness function. This characteristic is modelled often in simulations of societies of small agents (see the literature in the area of Artificial Life). A computer virus is a very infamous form of a pro-creative agent.

According to Dennett (1987) an *intentional system* is an entity

> … whose behaviour can be predicted by the method of attributing beliefs, designs and rational acumen.

Mentalistic and intentional notions such as *beliefs*, *desires*, *intentions*, *commitments*, *goals*, *plans*, *preference*, *choice*, *awareness*, may be assigned to agents. The *stronger notion of agenthood* in which agents are described in terms of this type of notions provides additional metaphorical support for the design of agents.

3 Primitive Agent Concepts

The notions of agenthood discussed in Section 2 are highly abstract notions. In order to design agents, it is necessary to be familiar with a number of primitive agent concepts. These primitive concepts serve as an ontology or vocabulary used to express analyses and designs of applications of agents and multi-agent systems. Two classes of primitive notions are distinguished: those used to describe the behaviour of agents in terms of their external (or public) states and interactions (Section 3.1), and those used to describe the behaviour of agents in terms of their internal (or private) states, and processes (Section 3.2). In Section 3.3, to illustrate the concepts, an example agent is discussed in terms of these concepts: an elevator.

3.1 External primitive concepts

Two types of interaction of an agent with its environment are distinguished, depending on whether the interaction takes place with an agent or with something else (called an *external world*), for example a database, or the material world. For each of these two types of interaction specific terminology is used.

Interaction with the external world

Two primitive types of interaction with the external world are distinguished. The first type of interaction, *observation*, changes the information the agent has about the world, but does not change the world state itself, whereas the second type, *performing an action*, does change the world state, but does not change the information the agent has about the world. Combinations of these primitive types of interaction are possible; for example, performing an action, and observing its results.

Observation
In which ways is the agent capable of observing or sensing its environment? Two types of observation can be distinguished: the agent passively receives the results of observations without taking any initiative or control to observe (*passive observation*), or the agent actively initiates and controls which observations it wants to perform; this enables the agent to focus its observations and limit the amount of information acquired (*active observation*).

Execution of actions in the external world
An agent may be capable of making changes to the state of its environment by initiating and executing specific types of actions.

Communication with other agents

Two directions of communication are distinguished, which can occur together: *outgoing communication* (is the agent capable of communicating to another agent; to which ones ?), and i*ncoming communication* (is the agent capable of receiving communication from another agent; from which ones ?).

3.2 Internal primitive concepts

A description in terms of the external primitive concepts abstracts from what is inside the agent. In addition to descriptions of agents in terms of the external concepts, often descriptions in terms of internal concepts are useful. The following internal primitive agent concepts are distinguished.

World and Agent Models

An agent may create and maintain information on (a model of) external *world* based on its observations of that world, on information about that world communicated by other agents, and its own knowledge about the world. The agent may also create and maintain information on (models of) *other agents* in its environment based on its observations of these agents as they behave in the external world, on information about these agents communicated by other agents, and knowledge about the world.

Self Model and History

Some agents create and maintain information on (a model of) their own characteristics, internal state, and behaviour. Or the agent creates and maintains a history of the world model, or agent models, or self model, or own and group processes.

Goals and Plans

To obtain pro-active, goal-directed behaviour, an agent often represents, generates, and uses explicit goals and its own plans of action in its processing.

Group Concepts

Besides individual concepts, often agents use group concepts that allow it to co-operate with other agents. For example, *joint goals*: is the agent capable of formulating or accepting and using goals for a group of agents, i.e., goals that can only be achieved by working together? Or *joint plans*: is the agent capable of representing, generating, and using plans of action for joint goals, i.e., involving which actions are to be performed by which agents in order to achieve a certain joint goal? Also *commitments* to joint goals and plan, *negotiation protocols* and *strategies* can be useful group concepts for agents, depending on their role and function.

3.3 An Example Analysis

These agent concepts introduced in Section 3.1 and 3.2 are illustrated by an example: an elevator is analysed from the agent perspective using these basic concepts; see Tables 1, through 3 below and their motivation.

I. External primitive concepts	elevator
Interaction with the world	
obserations	
passive observations	presence of objects between doors (optically) total weight its position
active observations	presence of objects between the doors (mechanically)
performing actions	moving opening and closing doors
Communication with other agents	
incoming communication	from users in the elevator: where they want to go (pushing button in elevator) from users outside: where they want to be picked up (pushing button outside elevator)
outgoing communication	to users in the elevator: where we are (display) there is overweight (beep) to users outside: where is the elevator (display) in which direction it moves (display)

Table 1 Elevator: External primitive concepts

3.3.1 The elevator in terms of external primitive concepts

Observation

No reasoning is performed as to when observations are to be performed. However, it is capable of receiving *passive* observation results on the presence of objects between the doors (an optical sensor), the total weight of its contents, andpossibly, its position in the building (at which floor). Besides it is able to perform *active* observation: the presence of objects between the doors (a mechanical sensor which is moved in the door opening just ahead of the doors themselves).

Performing actions

It performs actions in the world like moving itself (and people) vertically from one position to another and opening and closing doors.

Incoming communication

The elevator receives communication from users by buttons that have been pressed (providing information about the floor to which they wish to be transported).

Outgoing communication

The elevator communicates to a user by lighting buttons (information on the floor) and sounding beeps (information about overload).

II. Internal primitive concepts	elevator
A. World Model	the current floor, max load, current load
B. Agent Models	a user wants to be picked up from floor X a user wants to go to floor Y
C. Self Model	when maintenance is next due
D. History	when maintenance was last performed
E. Goals	to go to the floor X to pick up somebody to go to the floor X to deliver somebody
F. Plans	the order in which the required floors are visited sometimes: the speed that is taken
G. Group Concepts	
Joint goals	With other elevators to transport people and goods as efficiently as possible
Joint plans	Some elevators are capable of distributing the work
Commitments	The elevators then commit to their part of the work
Negotiation protocol	To reach a good distribution, they may have to negotiate
Negotiation strategies	To reach a good distribution, they may have to negotiate

Table 2 Elevator: Internal primitive concepts

3.3.2 The elevator in terms of internal primitive concepts

World and Agent Models

Elevators need to know *world information* on at which floor they are. They may maintain this knowledge themselves based on the actions (going two floors up, going one floor down) they perform. Another possibility is that the elevator immediately observes where it is, then it would not need to maintain a world state. Furthermore, the elevator needs to know how much weight its physical self is capable of transporting. The *agent information* of the user goals (where they want to go) may be maintained as well.

Self Model and History

The agent might have an explicit representation of when its physical form needs maintenance. It does not need to know what actions it previously performed to perform its current task. It might have an explicit representation of when it has last received maintenance.

Goals and Plans

Most modern elevators make use of the explicit goals (adopted from the goals communicated by the

users). The goals are used to determine which actions to perform. They may even make plans for reaching these goals: determine the order of actions, for example when one of the users has the goal to be at a highter floor and another on a lower floor.

Group Concepts

The elevator co-operates with its users. Sometimes the elevator can also co-operate with other elevators so that they could strategically distribute themselves over the floors. *Joint goals*: The goals adopted from the goals communicated by the users are joint goals (joint with the users), and sometimes even joint with the other elevators. *Joint plans*: Modern elevators are capable of distributing the work load, and thus of making joint plans. *Commitments*: To achieve the joint goals an elevator must commit to its part of the work as specified in the joint plans. *Negotiation protocols*: To make a joint plan, the elevators may negotiate as to which elevator goes where. Negotiation is only possible if a negotiation protocol is followed. *Negotiation strategies*: To make a joint plan, the elevators may negotiate as to which elevator goes where. Negotiation is only possible if each elevator has at least one strategy for negotiation.

III. Types of behaviour	elevator
Autonomy	yes
Responsiveness	in reaction to user requests in immediate reaction to observed objects between the doors
Pro-activeness	taking the initiative to go to a normally busy floor, if empty and not being called by a user
Social behaviour	co-operation with users, and, sometimes, with other elevators
Own adaptation and learning	often not possible

Table 3 Elevator: Types of behaviour

3.3.3 Types of behaviour of the elevator

Autonomy

As soon as it is activated, no system or human is controlling its machinery, and (normally) it is not switched off and on by the user. The fact that it responds to the immediate stimuli of buttons being pressed is not the same as being controlled. The elevator has full control of its motor, doors, and lights.

Pro-activeness

The most simple elevators stay where they are (some take the initiative to close their doors) when no longer in use, but more intelligent elevators go to a strategic floor (e.g., the ground floor).

Reactiveness

The elevator reacts to the immediate stimuli of buttons pressed, therefore, it is shows reactive behaviour. Furthermore, elevators often show delayed-response behaviour in picking up people.

People often have to wait for the elevator as the elevator picks up people on other floors, however, the elevator does not forget a signal and will, eventually, come to the requested floor.

Social behaviour

The elevator co-operates with users and, sometimes, with other elevators.

Own adaptation and learning

Simple elevators are not capable of adjusting their own behaviour to new situations, nor are they capable of learning. However, it is possible to conceive of more intelligent elevators that can learn the rush hours for the different floors.

4 Compositional Development of Multi-Agent Systems

The example multi-agent system described in this paper has been developed using the compositional development method DESIRE for multi-agent systems (DEsign and Specification of Interacting REasoning components); for the underlying principles, see (Brazier, Jonker and Treur, 1998), for a real-world case study, see (Brazier, Dunin-Keplicz, Jennings and Treur, 1995). The development of a multi-agent system is supported by graphical design tools within the DESIRE software environment. Translation to an operational system is straightforward; the software environment includes implementation generators with which formal specifications can be translated into executable code of a prototype system. In DESIRE, a design consists of knowledge of the following three types: process composition, knowledge composition, the relation between process composition and knowledge composition. These three types of knowledge are discussed in more detail below.

4.1. Process Composition

Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of (is composed of) lower level processes. Processes can be described at different levels of abstraction; for example, the process of the multi-agent system as a whole, processes defined by individual agents and the external world, and processes defined by task-related components of individual agents. The identified processes are modelled as *components*. For each process the *input and output information types* are modelled. The identified levels of process abstraction are modelled as *abstraction/specialisation relations* between components: components may be *composed* of other components or they may be *primitive*. Primitive components may be either reasoning components (i.e., based on a knowledge base), or, components capable of performing tasks such as calculation, information retrieval, optimisation. These levels of process abstraction provide process hiding at each level. The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *process composition*. This composition of processes is described by a specification of the possibilities for *information exchange* between processes (*static view* on the composition), and a specification of *task control knowledge* used to control processes and information exchange (*dynamic view* on the composition).

4.2. Knowledge Composition

Knowledge composition identifies the knowledge structures at different levels of (knowledge) abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction levels may correspond to the process abstraction levels, but this is often not the case. The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. Knowledge structures can be identified and described at different levels of abstraction. At higher levels details can be hidden. An *information type* defines an ontology (lexicon, vocabulary) to describe objects or terms, their sorts, and the relations or functions that can be defined on these objects. Information types can logically be represented in order-sorted predicate logic. A *knowledge base* defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented by formulae in order-sorted predicate logic, which can be normalised by a standard transformation into rules. Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of more specific knowledge bases. The compositional structure is based on the different levels of knowledge abstraction distinguished, and results in information and knowledge hiding.

4.3. Relation between Process and Knowledge Composition

Each process in a process composition uses knowledge structures. Which knowledge structures are used for which processes is defined by the relation between process composition and knowledge composition.

4.4 Generic Models and Reuse

Instead of designing each and every new agent application from scratch, an existing generic model can be used. Generic models can be distinguished for specific types of agents, of specific agent tasks and of specific types of multi-agent organisation. The use of a generic model in an application structures the design process: the acquisition of a conceptual model for the application is based on the generic structures in the model. A model can be generic in two senses:

- generic with respect to the *processes or tasks*
- generic with respect to the *knowledge structures*

Genericity with respect to processes or tasks refers to the level of process abstraction: a generic model abstracts from processes at lower levels. A more specific model with respect to processes is a model within which a number of more specific processes are distinguished, at a lower level of process abstraction. This type of refinement is called *specialisation*. Genericity with respect to knowledge refers to levels of knowledge abstraction: a generic model abstracts from more specific knowledge structures. Refinement of a model with respect to the knowledge in specific domains of application, is refinement in which knowledge at a lower level of knowledge abstraction is explicitly included. This type of refinement is called *instantiation*.

In Section 5 a generic agent model for weak agency is presented. The application for co-operative information gathering agents presented in Section 6 of this paper is an instantiation of this generic agent model. *Reuse* as such, reduces the time, expertise and effort needed to design and maintain

system designs. Which components, links and knowledge structures from the generic model are applicable in a given situation depends on the application. Whether a component can be used immediately, or whether instantiation, modification and/or specialisation is required, depends on the desired functionality. Other existing (generic) models can be used for specialisation of a model; existing knowledge structures (e.g., ontologies, thesauri) can be used for instantiation. Which models and structures are used depends on the problem description: existing models and structures are examined, rejected, modified, specialised and/or instantiated in the context of the problem at hand.

5 The Generic Agent Model: GAM

The characteristics of weak agency and the primitive agent concepts, introduced in Sections 2 and 3, provide a means to reflect on the tasks an agent needs to be able to perform. Pro-activeness and autonomy are related to the primitive concepts self model, goals, and plans. Reactivity and social ability are related to the primitive concepts world model, agent models, history, communication with other agents, and interaction with the external world. The ability to communicate with other agents and to interact with the external world often relies on the knowledge an agent has of the world and other agents.

The design of the generic agent model (GAM) in a compositional approach entails consideration of the processes and knowledge an agent needs to perform and the composition of related components and knowledge structures.

5.1 Process composition

Process composition within the generic agent model identifies the processes within an agent at the highest level of abstraction, and the manner in which they are composed to obtain the agent process (composition relation). Section 5.1.1 identifies the processes and their levels of abstraction. In Section 5.1.2 their interface information types are identified. The way in which these processes are composed is defined by information links and task control knowledge. Sections 5.1.3 (information links) and 5.1.4 (task control) address this composition relation.

5.1.1 Processes at different levels of abstraction

Identification of a process includes its abstraction level and its interface information types. The processes modelled within the generic agent model are depicted in Figure 1. The processes involved in controlling an agent (e.g., determining, monitoring and evaluating its own goals and plans) but also the processes of maintaining a self model are the task of the component own process control. The processes involved in managing communication with other agents are the task of the component agent interaction management. Maintaining knowledge of other agents' abilities and knowledge is the task of the component maintenance of agent information. Comparably, the processes involved in

managing interaction with the external (material) world are the task of the component world interaction management. Maintaining knowledge of the external (material) world is the task of the component maintenance of world information. The specific task for which an agent is designed (for example: design, diagnosis, information retrieval), is modelled in the component agent specific task. Existing (generic) task models may be used to further structure this component. In addition, a component co-operation management may be distinguished for all tasks related to social processes such as co-operation in a project, or negotiation. This component is not discussed in this chapter, but is addressed elsewhere in more detail (Brazier, Jonker and Treur, 1997).

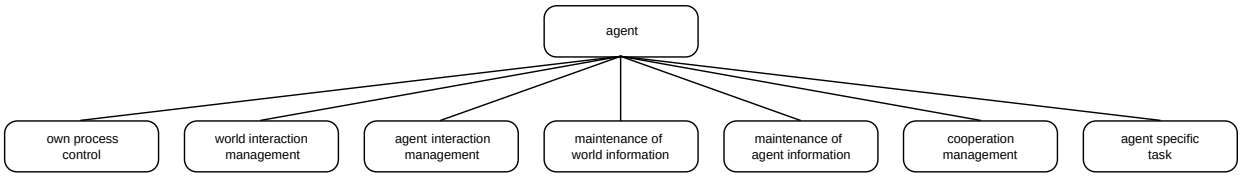

Figure 1 Processes at the two highest process abstraction levels within the agent

The four characteristics of weak agency discussed in Section 2 are related to these components in the following sense. Perception of the environment is performed by world interaction management (managing the perception process), maintenance of world information and maintenance of agent information (representation of perception information obtained from the environment). Actions in the world are managed by world interaction management. Social actions are managed by the tasks agent interaction management and cooperation management. The task cooperation management is not explained further in this chapter. Performing the agent's processes is initiated and co-ordinated by the task own process control; thus the agent's autonomous and pro-active behaviour is modelled.

5.1.2 Interface information types

A number of generic information types can be distinguished for the input and output of the generic agent model (based on external concepts) and for the generic processes within the agent (based on internal concepts).

Interface information types of the agent

An agent capable of communication with other agents may receive incoming communication info and may send outgoing communication info. Moreover, the agent may observe and perform actions in the external (material) world. The information type observation info models the observations that are to be performed in the component external world. The information type observation result info models the incoming results of observations. The information type action info models the actions the agent performs. In Table 4 an overview of the agent's interface information types is specified, based on the external primitive agent concepts.

The information types that express communication information are composed of information types on the subject of communication, and an information type to specify the agent from, or to whom, the communication is directed.

process	*input information types*	*output information types*
agent	incoming communication info observation result info	outgoing communication info observation info action info

Table 4 Specification of interface information types of the agent

Interface information types of components within the agent

The interface information types of the components within the agent are based on the internal primitive agent concepts; these interface information types are listed in Table 5. Within the agent component, the component own process control uses belief information on other agents and the external (material) world, as input. This information is modelled in the information type belief info which is composed of belief info on world and belief info on agents. The output of the component own process control includes the agent's characteristics (modelled in the information type own characteristics), used by the components agent interaction management and world interaction management. In addition to this information type, the component agent interaction management also receives the incoming communication received by the agent (and forwarded directly to the component agent interaction management), modelled in the input interface in the information type incoming communication info, and world and agent information,

process	*input information types*	*output information types*
own process control	belief info	own characteristics
agent interaction management	incoming communication info own characteristics belief info	outgoing communication info maintenance info
world interaction management	observation result info own characteristics belief info	observation info action info maintenance info
maintenance of agent information	agent info	agent info
maintenance of world information	world info	world info

Table 5 Specification of interface information types within the generic agent model

modelled in the input information type belief info. The output generated by the component agent interaction management includes the output for the agent as a whole (outgoing communication info), extended with maintenance info which is composed of maintenance info on agents and maintenance info on world (communicated information on the world and other agents that needs to be maintained).

	from		*to*	
information link	*process*	*information type*	*process*	*information type*
communicated info	agent	incoming communication info	agent interaction management	incoming communication info
info to be communicated	agent interaction management	outgoing communication info	agent	outgoing communication info
observation results to wim	agent	observation result info	world interaction management	observation result info
observations and actions	world interaction management	observation info action info	agent	observation info action info
communicated world info	agent interaction management	maintenance info on world	maintenance of world info	assumption world info
communicated agent info	agent interaction management	maintenance info on agents	maintenance of agent info	assumption agent info
observed world info	world interaction management	maintenance info on world	maintenance of world info	assumption world info
observed agent info	world interaction management	maintenance info on agents	maintenance of agent info	assumption agent info
world info to aim	maintenance of world information	epistemic world info	agent interaction management	belief info on world
agent info to aim	maintenance of agent information	epistemic agent info	agent interaction management	belief info on agents
world info to wim	maintenance of world information	epistemic world info	world interaction management	belief info on world
agent info to wim	maintenance of agent information	epistemic agent info	world interaction management	belief info on agents
own process info to aim	own process control	own characteristics	agent interaction management	own characteristics
own process info to wim	own process control	own characteristics	world interaction management	own characteristics
own process info to mwi	own process control	own characteristics	maintenance of world information	target world info
own process info to mai	own process control	own characteristics	maintenance of agent information	target agent info
world info to opc	maintenance of world info	epistemic world info	own process control	belief info on world
agent info to opc	maintenance of agent info	epistemic agent info	own process control	belief info on agents

Table 6 Specification of information exchange in table format

The component maintenance of agent information receives new information on other agents (the agent's beliefs on other agents) in its input interface. These beliefs on other agents are made available to other components in the output interface of the component maintenance of agent information. Likewise the component world interaction management receives the agent's characteristics in the input information type own characteristics, observation results received by the agent (and forwarded directly to the component world interaction management) in the input interface type observation result info, and information the agent has about the world and agents in the information type belief info. The output generated by the component world interaction management includes the output for the agent as a whole (action info, observation info), extended with maintenance info (information obtained from observation of the world and other agents that needs to be maintained).

The component maintenance of world information receives new information on the world (the agent's beliefs on the world) in its input interface. Beliefs on the world are available in the output interface of the component maintenance of world information.

5.1.3 Composition relation: information exchange

Information exchange within the agent is specified by the information links listed in Table 6, and depicted in Figure 2.

Observation results are transferred through the information link observation result info to wim from the agent's input interface to the component world interaction management. In addition, this component receives belief information from the component maintenance of world information through the information link world info to wim, and the agent's characteristics from the component own process control through the link own process info to wim. The selected actions and observations (if any) are transferred to the output interface of the agent through the information link observations and actions.

The component maintenance of world information receives meta-information on observed world information from the component world interaction management, through the information link observed world info and meta-information on communicated world information (through the link communicated world info) from the component agent interaction management. Epistemic information from maintenance of world information, epistemic world info, is transferred to input belief info on world of the components world interaction management, agent interaction management and own process control, through the information links world info to wim, world info to aim and world info to opc.

Comparably the component maintenance of agent information receives meta-information on communicated information from the component agent interaction management, through the information link communicated agent info and meta-information on observed agent information (through the link observed agent info) from the component world interaction management. Epistemic information, epistemic agent info, is output of the component maintenance of agent information, becomes input belief info on agents of the components world interaction management, agent interaction management and own process control, through the information links agent info to wim, agent info to aim and agent info to opc.

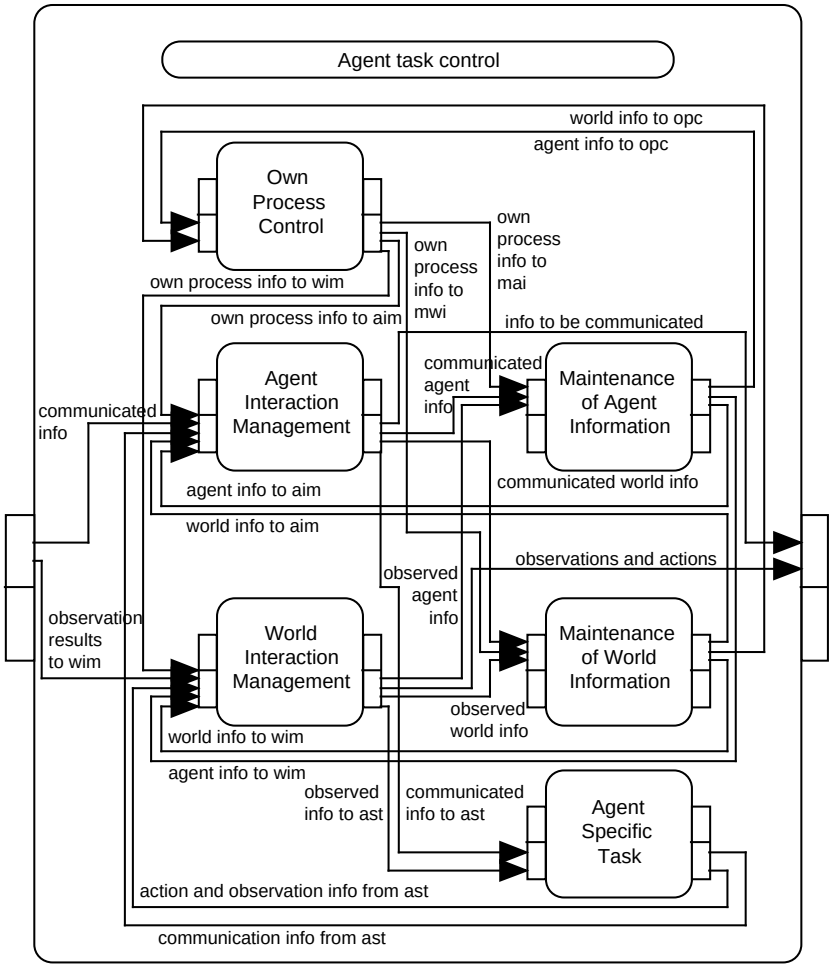

Figure 2 Information exchange at the highest process abstraction level within the agent

5.1.4 Composition relation: task control

Task control at the highest process abstraction level within the agent is simple: all components and links are made awake when the agent is awakened, which means that they all process (in an asynchronous manner) information as soon as it arrives.

5.2 Knowledge composition

A number of generic knowledge structures, in particular information types, can be distinguished: application domain independent knowledge structures which can be instantiated for a particular domain of application.

Information types provide the ontology with which knowledge used in the processes can be expressed. Information types provide the *ontology* (or lexicon, or vocabulary) for the languages

used in one (or more) components, knowledge bases and information links. In information type specifications the following concepts are used: sorts, objects, relations, functions, and meta-descriptions. Furthermore, information types can be composed from other information types. Each concept is

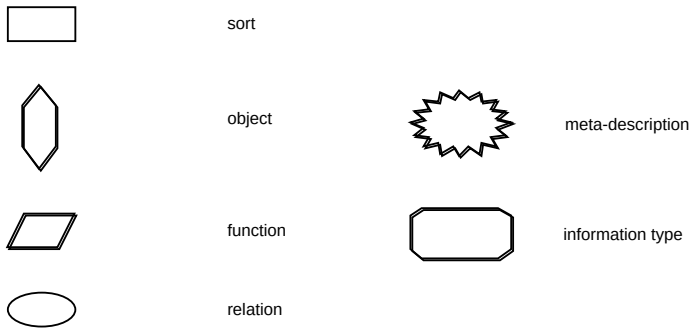

Figure 3 Information types: Legenda

represented graphically, see Figure 3. The icon for information types is used as depicted in this Figure 3 (containing only the name of an information type), but also as depicted in Figure 4 containing the sorts, object, functions, relations, and meta-descriptions used in the design of that information type.

information type	*short explanation*
belief info	information on the beliefs of the agent (information the agent has on the world and on other agents)
incoming communication info	information on communication the agent has received from another agent
outgoing communication info	information on communication the agent has decided to perform
observation info	information on observations the agent has decided to perform
observation result info	information on the observation results the agent has obtained
action info	information on the actions the agent has decided to perform
own characteristics	information on the agent's characteristics
maintenance info	information to be remembered by the agent

Table 7 Information types at the highest level of knowledge abstraction

5.2.1 Information types at different levels of knowledge abstraction

In this section the information types at the highest level of knowledge abstraction are presented, and the way in which they are composed of other information types.

Information types at the highest level of knowledge abstraction

At the highest level of knowledge abstraction, information types are distinguished to represent generic agent concepts such as: belief information (on the world and on other agents), (incoming and outgoing) communication information, information on observation (information on observations to be performed and obtained observation results), action information, information on the agent's characteristics, information to be remembered. These notions (abstracting from lower levels of knowledge abstraction), are modelled by the information types listed in Table 7.

Composition relations between information types

Each of the information types in Table 7 is composed of information types at a lower level of knowledge abstraction. Two of the information types (belief info and maintenance info) are composed of two more specific information types: one for information on the world and one for information on other agents. All information types are (either directly or indirectly) composed of (1) generic information types and (2) domain specific information. Generic information types are fully specified within the generic model. Domain specific information types are defined by references; they are instantiated for a specific domain of application. For example, the information type action info is composed of the generic information type actions to be performed and the domain specific information type domain actions (see Figure 4). The specific actions for a given domain of application are not specified within the generic model.

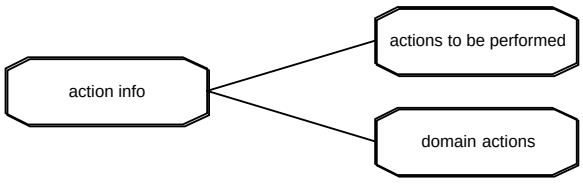

Figure 4 action info as a composition of a generic and domain specific information type

In a similar manner:

- the information type observation info is composed of the generic information type obs to be performed and the domain specific information type domain meta-info.

- the information type observation result info is composed of the generic information types observation results and truth indication, and the domain specific information type domain meta-info

- the information type incoming communication info is composed of the generic information types incoming communication and truth indication, and the domain specific information type domain meta-info.

- the information type outgoing communication info is composed of the generic information types outgoing communication and truth indication, and the domain specific information type domain meta-info

The information type domain meta-info is composed of world meta-info, agent meta-info and meta-info hierarchy. The information type world meta-info is a meta-description of the information type world info, using the sort WORLD INFO ELEMENT, as will be shown in Section 5.2.2. Similarly, the information type agent meta-info is a meta-description of the information type agent info using the sort AGENT INFO ELEMENT. The information type meta-info hierarchy defines the sorts WORLD INFO ELEMENT and AGENT INFO ELEMENT to be sub-sorts of the sort INFO ELEMENT (see also Figure 7 in Section 5.2.2).

The information types maintenance info on world and maintenance info on agents are composed of two generic information type (maintenance on world, resp. maintenance on agents and truth indication) and a domain specific information type (world meta-info, resp. agent meta-info). Comparable information type compositions have been defined for belief information. The information type own characteristics is composed of the generic information type agent characteristics and the domain specific information type domain agent characteristics. Finally, the standard meta-information types assumption info, epistemic info, required info and target info are used to define (by composition) specific variants information types for the given world information and agent information separately. The information type meta-input agent info is a meta-description of the information type agent info using the sort for input atoms IA; the other variants of meta-information types are defined similarly.

5.2.2 Generic information types

The information types world meta-info and agent meta-info include meta-descriptions of the information types world info and agent info using the sort WORLD INFO ELEMENT and AGENT INFO ELEMENT, respectively. Note that within the generic model the information types world info and agent info are only references. They can be instantiated for a specific domain of application.

Generic information types for observations and actions

The generic information type observation results enables the agent to express statements on observation results. In applications the observations can be *passive*: without taking any initiative, the agent automatically receives the observation results from the external world, or, *active*: observations initiated by the agent; the agent decides to do a specific observation and transfers this

decision to perform an observation to the external world. After receipt of this selected observation the world executes this observation and transfers observation results back to the agent. The decision of an agent to perform an active observation, for example depends on its own goals (*pro-active observation behaviour*) or on requests of other agents (*reactive observation behaviour*). Using the generic information type obs to be performed, the observations selected by the agent are expressed by the relation to be observed (see Figure 5). The generic information type truth indication defining the sort SIGN and the objects pos and neg in this sort, is also used in the information type observation results.

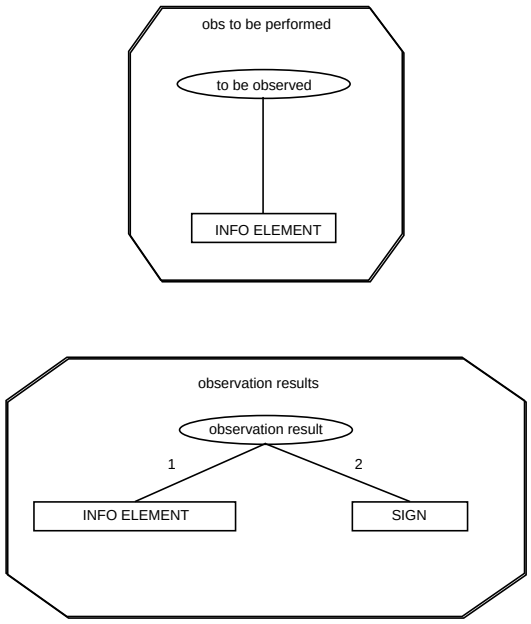

Figure 5 Generic information types on observation

Using these information types it is possible to make statements about the process of observation of the state of the world in contrast to statements about the world. It is possible for the statement 'my observation result is that the pressure is high' to be true, while in the world state 'the pressure is high' is false. For example, a sensor could give the wrong information. Similarly, it could also be the other way around: the statement 'the pressure is high' could be true in the world state, while the statement 'my observation result is that the pressure is high' is false, simply because it was not observed. Note also that 'I did not observe that the pressure is high' means something different from 'I observed that the pressure is not high'. A statement of the form 'my observation result is that the pressure is high' cannot be expressed using the information type that describes the world. For example, the statement 'the pressure is high' is not adequate. Therefore, another structure is necessary to express statements about statements. Statements about statements are called *meta-level* statements. The statements that form the subjects of such meta-level statements are called *object level* statements. The generic information type actions to be performed enables the agent to reason about actions; see Figure 6.

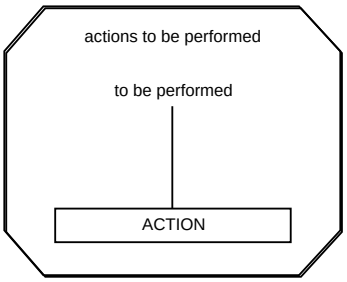

Figure 6 Generic information type: actions to be performed

Generic information types for communication

A social agent is able to receive incoming communication and to generate outgoing communication. The generic information types for communication are depicted in Figure 7.

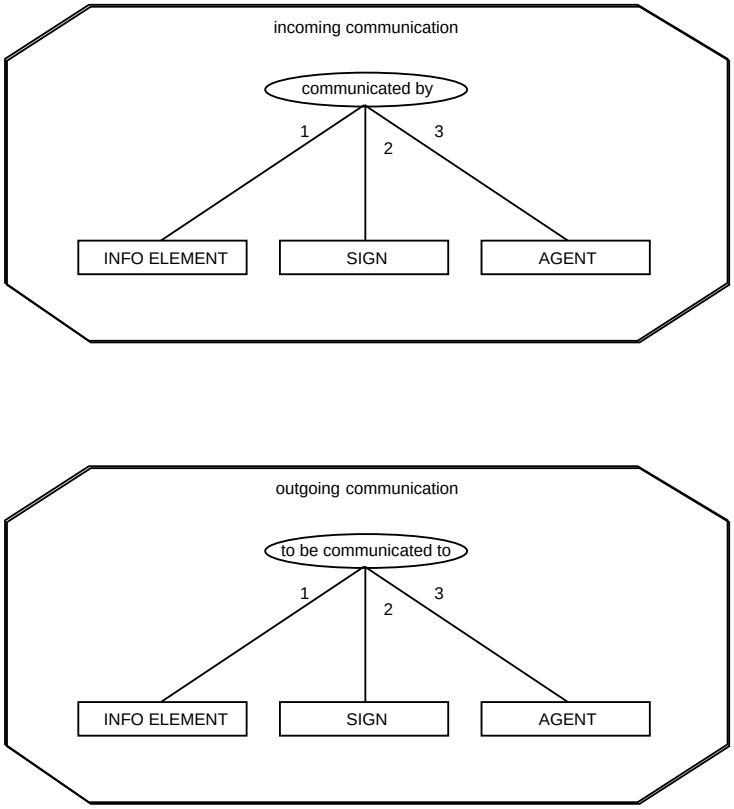

Figure 7 Generic information types on communication

By these information types it is possible to make statements about the process of communication (in contrast to, for example, statements about the world). It is possible for the statement 'I was told that the

pressure is high' to be true, while in the world state 'the pressure is high' is false: the other agent may simply not tell the truth. It could also be the other way around: the statement 'the pressure is high' could be true in the world state, while the statement 'sombody told me that the pressure is high' is false, simply because nobody told me. Note also that 'he did not tell me that the pressure is high' does not mean the same as 'he told me that the pressure is not high'. Similar to statements about observation, statements about communication are meta-level statements.

Generic information types for internal information

The information communicated to the agent may be used to extend or update an agent's beliefs on the world or on other agents. The information received is analysed, selected and prepared to be stored as information either on the world or on other agents; the related information types are depicted in Figure 8.

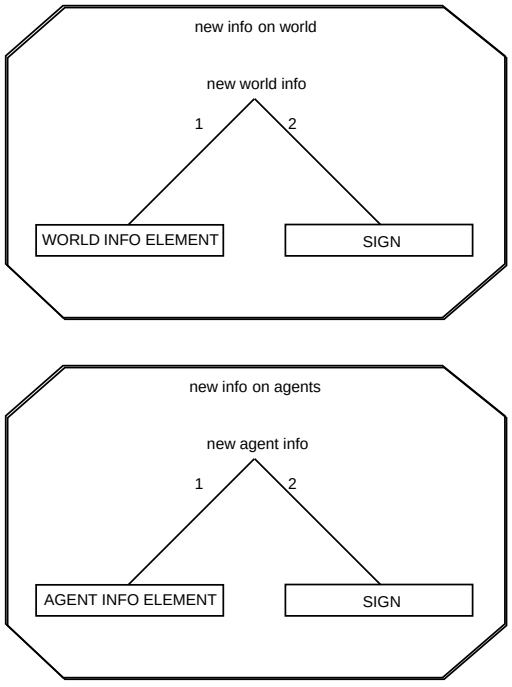

Figure 8 Generic information types: maintenance on agents, maintenance on world

The generic information type beliefs can be used to maintain information on the world and other agents; see Figure 9.

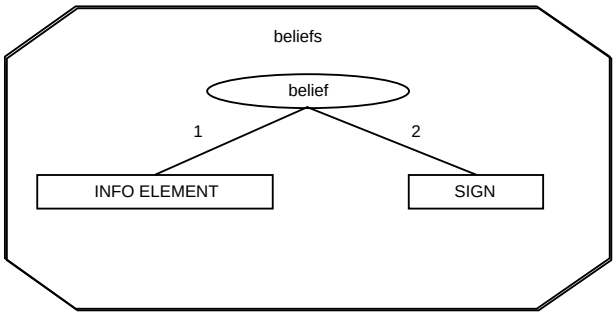

Figure 9 Generic information type: beliefs

The generic information type agent characteristics can be used to express meta-information about the agent's characteristics in an explicit, declarative manner.

Standard meta-information types

Generic standard meta-information types on assumption information and epistemic information are included. The sort IA models the input atoms of the component in which this information type is used. Similarly the sort IOA models the input and output atoms. Target information expresses on which output atoms (modelled by the sort OA) a component can focus. A target type expresses whether the focus is on confirmation (truth value true) or rejection (truth value false) of an information element, or just on determination of its truth value. The information type required info specifies the input atoms needed to derive target atoms. This meta-information makes it possible to focus the reasoning process: to provide input needed to derive the targets.

A generic standard meta-information type is of a form named by meta-input <*information-type-name*>, meta-output <*information-type-name*> and meta-interface <*information-type-name*>. These information types are meta-descriptions of the information type named, using sort IA, OA, or IOA, respectively. Note that all standard information types as described are pre-defined and as such known in any component. They do not need to be explictly specified, but can be used in information links.

5.2.3 Domain specific information types

Within the knowledge composition specified in Sections 5.2.1 and 5.2.2 references occur to domain specific information types. Application of the generic model concentrates on instantiation of these information types for the specific application domain at hand, and on domain specific knowledge bases. Generic information types can simply be reused. For completeness the domain specific information types (which need to be instantiated) are summarised in Table 8, with a short explanation.

specific information type	*short explanation*
world info	expresses specific information on the world
agent info	expresses specific information on other agents
agent identification	identifies the names of the other agents
domain actions	describes the names of the actions the agent can perform
domain agent characteristics	expresses characteristics of agents, such as varaiants of pro-activeness and reactiveness

Table 8 Domain specific information types

5.2.4 Generic knowledge bases

Often the knowledge to be used for a specific application strongly depends on the application domain. However, sometimes parts of the knowledge can be formulated in a more generic, domain independent manner, which makes reuse possible in domains with similar characteristics. These generic knowledge bases are available to be used in the agent model. They may be (re)used in a specific application depending on their relevance. If during an application of the generic model to a specific domain, the knowledge is applicable and relevant, it can be reused. If they are not relevant, they simply can be left out.

An example of a generic knowledge base in the generic model is the following knowledge base observation result extraction kb which can be used within the component world interaction management to identify the observed world information and observed agent information that is to be maintained:

if observation_result(I:WORLD_INFO_ELEMENT,S: SIGN)
then new_world_info(I: WORLD_INFO_ELEMENT,S: SIGN);

if observation_result(I:AGENT_INFO_ELEMENT,S: SIGN)
then new_agent_info(I:AGENT_INFO_ELEMENT,S: SIGN);

This generic knowledge expresses that the agent blindly trusts its own observations. In applications the knowledge can be refined, for example by adding conditions. Similarly the generic knowledge base communicated info extraction kb is part of the generic model. This knowledge base may be used within the component agent interaction management to identify the communicated world information and communicated agent information that needs to be maintained:

if communicated_by(I:WORLD_INFO_ELEMENT,S: SIGN, A:AGENT)
then new_world_info(I: WORLD_INFO_ELEMENT,S: SIGN);

if communicated_by(I:AGENT_INFO_ELEMENT,S: SIGN, A:AGENT)
then new_agent_info(I:AGENT_INFO_ELEMENT,S: SIGN);

This generic knowledge expresses that the agent blindly trusts what other agents communicate. In applications also this knowledge can be refined, for example by adding conditions.

5.2.5 Relations between knowledge bases and information types

The knowledge bases defined in Section 5.2.4 are related to information types depicted in Figure 9.

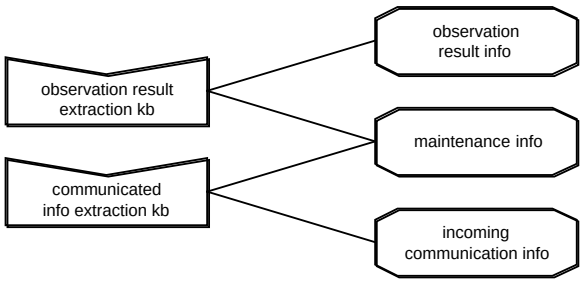

Figure 9 Relation between generic knowledge bases and information types

5.3 Relations between process and knowledge composition

The generic information types described in this section are all used in interfaces of components. The relations between the two generic knowledge bases introduced in Section 5.2.5 and processes in which they occur is straightforward: observation result extraction kb is used within component world interaction management, and communicated info extraction kb within agent interaction management.

6 Refinement of the Generic Agent Model for an Application Domain

In this section an example application of the generic agent model is presented: co-operative information gathering agents.

6.1 Problem description : co-operative information gathering

This example multi-agent system consists of two agents that can each gather partial information on the world, but can only draw further conclusions by combining their individual information.

6.1.1 The domain

The application is as follows. Assume two agents A and B start a small project: they have to do some investigation and make up a report on some topic. Each of the agents has access to useful sources of information, but which information differs for the two agents. By co-operation they can benefit from the exchange of information that is only accessible to the other agent. If both types of information are combined, conclusions can be drawn that would not have been achievable for each of the agents separately. Co-operation may fail for a number of reasons. For example one of the agents, say A, may not be pro-active in its individual search for information. This may be compensated if the agent B is pro-active in asking the other agent for information, but then at least A has to be reactive (and not entirely inactive in information search). Another reason for failure is that one of the agents may not be willing to share its acquired information with the other agent. Yet another reason for failure may be that although both agents are active in searching and exchanging

information, none of them is able to combine different types of information and deduce new conclusions.

To make the example more precise: the example multi-agent model is composed of three components: two information gathering agents A and B and a component W representing the external world. Each of the agents is able to acquire partial information about the external world (by observation). Each agent's own observations are insufficient to draw conclusions of a desired type, but the combined information of both agents is sufficient: they have to co-operate to be able to draw conclusions. Therefore communication is required; the agents can communicate their own observation results and requests for observation information of the other agent. For reasons of presentation, this, by itself quite common situation for co-operative information agents, is materialised in the following more concrete form. The world situation consists of an object that has to be classified. One agent can only observe the bottom view of the object, the other agent the side view. By exchanging and combining observation information they are able to classify the object. In the example interview protocol presented below, two experts in the field of classification of three-dimensional objects are studied; agent A has done this job for almost twenty-five years, and agent B only started a year ago. Their daily work consists of observing three-dimensional objects, and trying to identify the nature of these objects. They need to co-operate to be successful, because they each can only see one side of the object. Agent B can only see the bottom, and A can only see one of the sides. They need to combine their two two-dimensional views to come to a correct conclusion about the object, using the knowledge depicted in Figure 10.

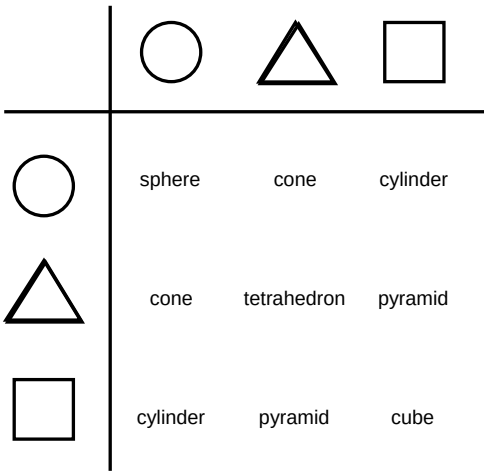

Figure 10 Object classification knowledge

6.1.2 The requirements

Based on the generic agent model described in Section 5, some variants of agents that can play the role of Arnie and Bernie are designed. The variants of agents can differ in some of their characteristics; an agent may or may not be *pro-active*, in the sense that it takes the initiative to:

- perform observations
- communicate its own observation results to the other agent
- ask the other agent for its observation results
- determine the classification of the object (by reasoning)

Moreover, it may be *reactive* to the other agent in the sense that it responds to a request for observation information:

- by communicating its observation result as soon as they become available
- by starting to observe for the other agent upon request

These agent characteristics can be represented explicitly as facts in the agent's component own process control. By varying these facts, different variants of this agent can be defined. Of course, the impact of these explicitly specified characteristics needs to be specified in the model. For example, if an agent has the characteristic that it always takes the initiative to communicate its observation results as soon as they are acquired, then the agent needs to behave accordingly, but if the agent does not have this characteristic, then the agent need not behave this way. This requires an adequate interplay between the component own process control and the component agent interaction management within the agent, and adequate knowledge within the component agent interaction management.

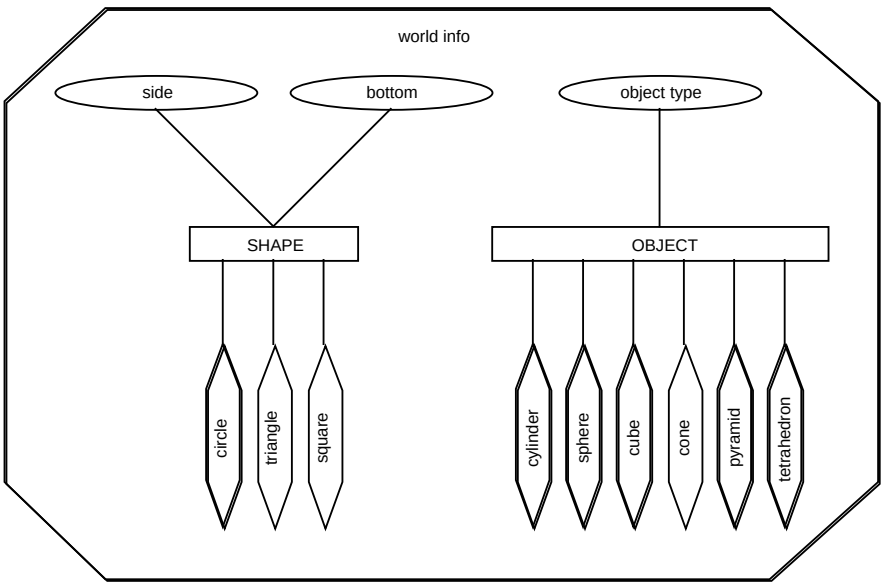

Figure 11 Instantiation of the information type world info

6.2 An agent model for co-operative information gathering

In this section the generic agent model introduced in Section 5 is applied to the application domain described in Section 6.1. Reusing a generic model entails that instantiations are made for a number of domain specific information types of the model. However, also some (preferably minor) extensions or modifications of the model are often made. For example, in this domain of application a component for the agent-specific task (named object classification) and some information links are added. In this section first the information types are discussed (Section 6.2.1), and next the knowledge bases (Section 6.2.2). Finally, the model is slightly extended by adding an information link from agent interaction management to world interaction management and information links from own process control and maintenance of world information to object classification (Section 6.2.3).

6.2.1 Domain specific information types

The information types needed to model the example of co-operative information gathering agents are the instantiations of the domain specific information types of the generic model and a few additional domain specific information types.

Instantiations of domain specific information type of the generic model

In Section 5.2.3 the domain specific information types are listed: world info, agent info, agent identification, domain actions, domain agent characteristics. For some of these information types domain specific instantiations are needed. The information types agent info and domain actions can be left empty in this domain, as the agents do not perform actions. In Figure 11 the instantiation of the information type world info is modelled. Six different types of objects form the sort OBJECT. The two-dimensional shapes that can be observed form the sort SHAPE. The two perspectives are modelled by the relations side and bottom. Finally, the classification of the type of object is expressed by the relation object type.

The agent characteristics are taken from Section 3.2. An agent can be pro-active with respect to taking the initiative to observe, to inform the other agent if information is available, to request information from the other agent, and to reason in order to draw a conclusion on the object classification. It can be reactive with respect to providing the other agent with available information upon request and observation for the other agent, if the requested information is not yet available. The instantiation of the information type domain agent characteristics is depicted in Figure 12. The information type world meta-info is used in domain agent characteristics.

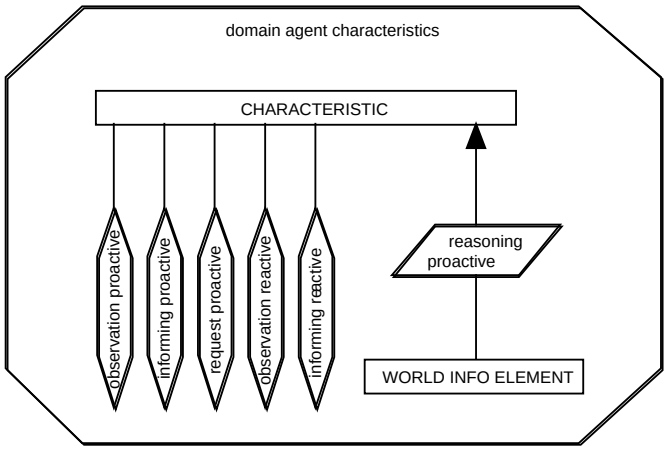

Figure 12 Instantiation of the information type domain agent characteristics

To distinguish communicated information in requests and information provision, functions and relations requested and info are defined in additional information types.

6.2.2 Domain knowledge

In this section the domain specific knowledge bases are discussed in the context of the component in which they are used.

Object classification knowledge

The knowledge used to classify the object based on available observation information can easily be taken from the table depicted in Figure 15:

if	bottom(circle)	**and**	side(circle)	**then**	object_type(sphere);
if	bottom(circle)	**and**	side(square)	**then**	object_type(cylinder);
if	bottom(square)	**and**	side(circle)	**then**	object_type(cylinder);
if	bottom(circle)	**and**	side(triangle)	**then**	object_type(cone);
if	bottom(triangle)	**and**	side(circle)	**then**	object_type(cone);
if	bottom(square)	**and**	side(square)	**then**	object_type(cube);
if	bottom(square)	**and**	side(triangle)	**then**	object_type(pyramid);
if	bottom(triangle)	**and**	side(square)	**then**	object_type(pyramid);
if	bottom(triangle)	**and**	side(triangle)	**then**	object_type(tetrahedron);

It is assumed that objects are placed in the correct orientation. For example, viewed from the bottom a cylinder is always a circle. Note that there is no situation in which the conclusion can be drawn on the basis of one observation only.

World interaction knowledge

As the agents in the domain do not perform actions, world interaction focusses entirely on observation. No passive observations exist in the domain. There are two reasons to actively perform

an observation: the agent may be pro-active (expressed by the first knowledge element below) or reactive (expressed by the second knowledge element) with respect to observation. Note that an observation is only selected if no information is available.

```
if   own_characteristic(observation_proactive)
   and not belief(side(X:SHAPE), pos)
   and not belief(side(X:SHAPE), neg)
then   to_be_observed(side(X:SHAPE));

if   own_characteristic(observation_reactive)
   and  requested(side(X:SHAPE))
   and not belief(side(X:SHAPE), pos)
   and not belief(side(X:SHAPE), neg)
then   to_be_observed(side(X:SHAPE));
```

Actually, this knowledge base is meant for one of the agents. For the other agent side must be replaced by bottom.

Communication knowledge

The component agent interaction management makes use of knowledge to analyse incoming communication, and to generate outgoing communication.

Knowledge to analyse incoming communication

Generic knowledge needed to analyse incoming information is defined in the generic model; see Section 6.2.2.4 above. This knowledge identifies the information on the world that is to be maintained. However, in line with the communication differentiation added in this example model, a more sensitive treatment is preferred. The first knowledge element below expresses that the information provided by the other agent is identified as world information that is to be maintained. The second knowledge element identifies the information requested. The choice is made to only use this information in the component world interaction management, and to not maintain this information separately.

```
if communicated_by(info(I:WORLD_INFO_ELEMENT),S: SIGN, A:AGENT)
then new_world_info(I: WORLD_INFO_ELEMENT,S: SIGN);

if communicated_by(request(I:WORLD_INFO_ELEMENT),S: SIGN, A:AGENT)
then requested(I:WORLD_INFO_ELEMENT);
```

Knowledge to generate outgoing communication

Whether or not the agent actively communicates information to other agents depends on its own characteristics. If an agent is pro-active with respect to information provision, the first knowledge element below is applicable.:

if own_characteristic(informing_proactive)
  and belief(I: WORLD_INFO_ELEMENT, S: SIGN)
then to_be_communicated_to(info(I: WORLD_INFO_ELEMENT),S: SIGN,bernie);

If an agent is reactive in informing the other agent upon request then the second and third knowledge element are relevant:

if own_characteristic(informing_reactive)
  and communicated_by(requested(I: WORLD_INFO_ELEMENT), pos, A:AGENT)
  and belief(I: WORLD_INFO_ELEMENT, S: SIGN)
then to_be_communicated_to(info(I: WORLD_INFO_ELEMENT),S: SIGN,A:AGENT);

if own_characteristic(observation_reactive)
  and communicated_by(requested(I: WORLD_INFO_ELEMENT), pos, A:AGENT)
  and belief(I: WORLD_INFO_ELEMENT, S: SIGN)
then to_be_communicated_to(info(I: WORLD_INFO_ELEMENT),S: SIGN,A:AGENT);

The fourth knowledge element is applicable for pro-active behaviour with respect to requesting:

if own_characteristic(request_proactive)
  and not belief(bottom(S:SHAPE), pos)
  and not belief(bottom(S:SHAPE), neg)
then to_be_communicated_to(requested(bottom(S:SHAPE)), pos, bernie);

Own process control knowledge

The knowledge base for the component own process control contains meta-information that defines the agent character. For each agent the own process control knowledge is defined by a sub-set of the following set of meta-facts

own_characteristic(observation_proactive)
own_characteristic(observation_reactive)
own_characteristic(informing_proactive)
own_characteristic(informing_reactive)
own_characteristic(request_proactive)
own_characteristic(reasoning_proactive(object_type(O:OBJECT)))

Each sub-set defines a specific type of agent (the possibility of having represented the negation of an own characteristic is not considered). For example, the empty sub-set defines a totally apathic agent: it does nothing except maintain the information it receives. The complete set defines a fully pro-active and reactive agent.

6.2.3 Addition of information links

Three information links are added to the model. One of these links takes care of requests. The management of requests from the other agent and information provision to the other agent could be

modelled as an additional agent specific task. However, because the management is rather simple, the choice has been made to have the two components agent interaction management and world interaction management take care of all request management. To this end, the information link requests is added to transfer requests from agent interaction management to world interaction management. The information types requests and world meta-info are used in this link, both at the source and destination. Furthermore, two information links are added to connect the agent specific task object classification. One information link is used to transfer the information from maintenance of world information to object classification. The other information link is used to transfer information of the form

own_characteristic(reasoning_proactive(object_type(O:OBJECT)))

from own process control to the information that the output atom object_type(O:OBJECT) is a target of the component object classification.

agent B agent A	obs proactive reas proactive	obs proactive reas proactive inf proactive	obs proactive reas proactive inf proactive req proactive	obs reactive	obs reactive reas proactive	inf reactive obs proactive
obs proactive reas proactive	-	A	A	-	-	-
obs proactive reas proactive inf proactive	B	A, B	A, B	-	-	-
obs proactive reas proactive inf proactive req proactive	B	A, B	A, B	A	A, B	A
obs reactive	-	-	B	-	-	-
obs reactive reas proactive	-	-	A, B	-	-	-
inf reactive obs proactive	-	-	B	-	-	-

Table 9 Some of the outcomes of two co-operative information gathering agents

6.3 The behaviour of co-operative information gathering agents

The behaviour of the co-operative information gathering agents strongly depends on their characteristics. The number of sub-sets of the set of six agent characteristic facts in Section 6.2.2.4

is 64. Whether or not an agent succeeds in classification of the object also depends on the behaviour of the other agent. In principle it is possible to create a 64 by 64 matrix to identify the behaviour of all 4096 combinations of two agents. For practical reasons, only a small subset of such combinations is discussed in this section. Table 9 indicates which of the agents will be able to classify the object for 36 combinations of two agents.

The table shows that two pro-active but purely individualistic agents (both observation pro-active and reasoning pro-active) will never find a solution. Nevertheless, if one of these agents is also social in communicating its observation results (informing pro-active), the other agent (but not the agent itself) will find a solution. A fully pro-active agent will find a solution as soon as its partner is observation reactive, or informing reactive and observation pro-active, or informing pro-active and observation pro-active. An observation reactive and reasoning pro-active agent will find a solution if the other agent is request pro-active, observation pro-active and informing pro-active. Agents that both are only reactive in communication will not succeed. These are only some of the possibilities. A more complete analysis of the conditions under which one of the two or both agents will find a solution can be found in (Jonker and Treur, 1997).

7 Comparison with existing Agent Architectures and Applications

In the agent literature, various agent architectures can be found, often specialised to a particular type of application. The design of most of these agent architectures is not formally specified in detail; usually they are only available in the form of an implementation, and at the conceptual level some informal pictures and natural language explanations. In general, the aim for the development of these agent architectures in the first place is to have a working piece of software for a specific type of application. The design of the generic agent model GAM introduced in this paper has a different aim. The generic agent model GAM was meant as a unified design model for weak agency, formally specified in an implementation- and domain-independent manner at a high level of abstraction. A success criterion for this aim is the possibility to specialise and instantiate the agent model GAM to obtain conceptual, formal specifications of design models for a variety of (implemented, but not formally specified) agent types and agent behaviours. Thus a unified design description is obtained which enables comparison of these agent architectures at a conceptual but yet formally defined level. Evaluation of this aim has taken place for two different groups of agent architectures:

- *agent architectures for new applications* designed, after an informal analysis, as a formally specified refinement of GAM

- *existing agent architectures,* developed for specific applications without formal specification of a design model; in the context of the research reported here they have been reverse-engineered at a conceptual design level using the structure of GAM

Evaluation for the first of these two groups of agent architectures has shown that GAM is an adequate means to design specific types of agents, given a variety of requirements imposed by specific application domains. Evaluation for the second group of requirements shows that GAM is an adequate means for reverse engineering, to obtain unified, comparable formal descriptions of different types of existing agent architectures. For a summarizing overview, see Table 10.

7.1 Applications designed on the basis of GAM

The following types of agents tuned to specific application domains have been developed using (refinements of) the structure of GAM to obtain a formally specified design model.

Simulated animal behaviour

Instantiations of the generic agent model GAM have been designed to fulfill the requirements imposed by purely reactive, delayed response, pro-active goal-directed, and social animal behaviour, as identified in the literature on animal behaviour; e.g., see (Vauclair, 1996). Within the model for purely reactive behaviour, only one component is instantiated to model the associations between observations and actions used in the direct interaction with the world. For the model with delayed response behaviour, a separate component for memory (maintenance of world information) was instantiated, in addition to world interaction management. For pro-active behaviour, also the component own process control was instantiated, to represent specific agent characteristics and to generate goals. To obtain a model for a specific type of social behaviour, in addition, the components maintenance of agent information (where the pick order between the animals is represented) and agent interaction management (to generate and interpret growling) have been instantiated. For more details, see (Jonker and Treur, 1998b).

Negotiating agents to achieve load balancing of electricity use

The application to load balancing of electricity use by means of a flexible form of one-to-many negotiation was made in co-operation with Swedish electricity industry. A precursor of the generic agent model GAM was used to develop this application. Within this application, the component cooperation management has a more complex refinement to address the evaluation and generation of bids. Also the components own process control (representing agent characteristics that have impact on the negotiation, and decisions to start or stop a negotiation process) and agent interaction management (to transfer the bids to the other agents) are present in an instantiated form. The component AST was instantiated to the task ‘determine balance of predicted use’. For more details, see (Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak and Treur, 1998).

Personal information agents and information brokering agents at the World Wide Web

For different applications of information agents in a World Wide Web context, agent models have been developed on the basis of GAM. First, an instantiation of GAM has been designed to serve as an information broker agent. This broker agent model has instantiations of all components of GAM. For example, within maintenance of world information information on the objects of the brokering is

maintained (i.e., meta-information of the brokered information objects), and within maintenance of agent information, (interest) profiles of users and other agents are created and maintained. Within the agent specific task different matching forms have been specified. Within the instantiated component world interaction management it is specified how the agent can observe tags with meta-information in a HTML page at a given Website. In (Jonker and Treur, 1998c), the broker agent model, and an application to a Personal Assistant to support researchers in the exchange of scientific papers is described. Moreover, it is described how the information broker agent model can support its own maintenance by installing at run-time new ontologies and knowledge bases communicated to the agent by maintenance agents (instantiation of own process control). In (Jonker and Treur, 1999), a multi-agent architecture of an intelligent Website is introduced, based on (a number of instantiations of) the information broker agent model, and illustrated for the domain of a department store. Here the information agents play the role of servants at the Website, who are able to have an informed dialogue with visitors of the Website, tailored to the background and needs of the visitor. In (Jonker, Lam and Treur, 1999) an application of this architecture to a Website for employees of an insurance company is described.

Agents in social simulation applications based on deliberate normative behaviour

To simulate societies in which agents can behave in a deliberate normative manner, a model has been developed for a deliberate normative agent (Castelfranchi, Dignum, Jonker, and Treur, 1999). This type of agent has explicit mental representations of norms, which are interpreted operationally as (meta-)goals for its own behaviour. The deliberation also incorporates deciding about when to follow a norm and when to violate it. The model has been designed as a refinement of GAM in the following manner. Besides components for maintenance of world information and maintenance of agent information, also a component maintenance of society information is added. In this component the norms distinguished in the society are maintained. Society information could have been represented within maintenance of agent information as a specific, global form of agent information; however it was decided that it is more natural to include a separate component for this 'Society Model' to make society norms more explicitly visible as distinct from personal norms of specific agents. Other components reused are agent interaction management, world interaction management and own process control. The latter component is refined into four sub-components: norm management, goal management, plan management, and strategy management. In the first of these components decisions on (personal) norm adoption are made. The adopted norms are operationalised within strategy management in terms of control of the goal management and plan management processes.

7.2 Reverse engineering of existing agent architectures and applications

A number of existing applications have been reverse-engineered at a conceptual design level using the structure of GAM as a starting point for refinement. The generic model GAM has been refined to obtain a formally specified design description of the following types of agents.

- *monitoring*, *diagnostic* and *restoration* agents in electricity transportation management

The multi-agent system for electricity transportation management developed in the ARCHON project was one of the first operational real-world applications of agent technology (Cockburn, and Jennings, 1995; Jennings, Corera, Laresgoiti, Mamdani, Perriolat, Skarek, and Varga, 1996). It is currently running on-line in a control room in the North of Spain. An electricity transportation network carries electricity from generation sites to the local networks where it is distributed to customers. Managing this network is a complex activity which involves a number of different subprocesses: monitoring the network, diagnosing faults, and planning and carrying out maintenance when such faults occur. The application involves two co-operating diagnostic agents, a monitoring agent, and a restoration agent. The reverse engineering application of GAM to ARCHON can be found in (Brazier, Dunin-Keplicz, Jennings and Treur, 1995). All of the agents maintain a World Model, which clearly can been obtained as an instantiation of the component maintenance of world information in GAM. Moreover, they maintain information about the other agents in the system in socalled Acquaintance Models, obtained as an instantiation of GAM's component maintenance of agent information. Furthermore, Monitor Incoming Data and Monitor Process State were obtained as an instantiation of own process control in GAM. The agent-specific task component AST was instantiated to obtain the different specialisations of the agents: it is refined to a complex diagnostic model for the diagnosis agents, to a model for monitoring disturbances and the progress of restoration processes for the monitor agent, and to a model for restoration planning for the restoration agent. More details can be found in (Brazier, Dunin-Keplicz, Jennings and Treur, 1995).

- *co-operative agents* based on joint intentions

In (Jennings, 1995) an informally described multi-agent model for cooperative problem solving is proposed. Essential elements of this model are the dynamic organisation and management of joint activities, susceptive to change due to unexpected events. As described, the model only provides a restricted amount of detail to support analysis, modelling and implementation of co-operative agents in specific domains. In (Brazier, Jonker and Treur, 1997) it is described how a formal design model of this cooperative agent architecture has been made as a refinement of GAM. Within this model monitoring, planning, control of own activities, and monitoring, planning, allocation, and communication of activities with other agents are explicitly distinguished. To obtain this cooperative agent model both the components own process control (for the monitoring, planning and control of own activities) and the component cooperation management (for the monitoring, planning, allocation and communication about activities involving others) have been refined to more complex, composed components; see (Brazier, Jonker and Treur, 1997) for more details.

- *BDI-agents*

The wellknown BDI architecture (Rao and Georgeff, 1991), and its predecessor PRS (Georgeff and Lansky, 1987), is organised around the notions beliefs, desires, and intentions. How the generic agent model GAM can be refined to obtain a formally specified design model of the BDI-

architecture, can be found in (Brazier, Dunin-Keplicz, Treur and Verbrugge, 1999). The beliefs on the environment (the world and the other agents) are maintained within the components maintenance of world information and maintenance of agent information. The desires and intentions are represented within a refinement of component own process control, which in this case has a more complex, compositional structure, based on components belief determination, desire determination and intention and commitment determination. The latter component is composed of components goal determination and plan determination, which, in turn are composed of intended goal determination and committed goal determination, resp. intended plan determination and committed plan determination. For more details, see (Brazier, Dunin-Keplicz, Treur and Verbrugge, 1999).

- *agents in social simulation experiments*

In (Cesta, Miceli and Rizzo, 1996) experiments are reported with which social theories are tested by simulating interaction between different types of simple agents (i.e., agents with limited knowledge and capabilities). Four types of agents are distinguished on the basis of their social characteristics: social agents, parasite agents, solitary agents and selfish agents. The effect of an agent's social characteristic on interaction with other agents is measured by simulating agent behaviour in a situation in which 30 agents try to survive on a 15 * 15 grid in which 60 pieces of food are continually available in random positions. An agent's welfare is measured on the basis of its energy level. The end result of a simulation is the number of agents that survive in a given society of agents, given the energetic value of the food available. Agents do not communicate explicitly but implicitly: a hungry agent changes colour, and this can be seen by other agents. Agents' social characteristics are assumed to be static. An agent does not change from being, for example, selfish to social. The implications of agents' social characteristics for its behaviour is as follows. A solitary agent will always search for food, regardless of its internal energy level. Likewise, a parasite agent will always look for help. A selfish agent will look for help only if it is in danger, otherwise it searches for food. A social agent will also look for help if it is in danger. If it is in a hungry state, it will search for food. If it is in a normal state, then it will search for food if no help-seeking agents are seen. Otherwise, the social agent will give food to one of the help-seeking agents nearby.

The experiments reported in (Cesta, Miceli and Rizzo, 1996) have been replicated and extended by reverse engineering based on GAM. The refinement of the generic model GAM to obtain the four types of agents was performed on the basis of the informal, textual descriptions provided by (Cesta, Miceli and Rizzo, 1996). The only components within the generic agent model, applicable to these small agents, are the components own process control and world interaction management. The component own process control is composed of four components: own resource management, own characteristics, goal determination and plan determination. The component own resource management receives information about its current energy level and the resources it has consumed, with which it determines its new energy level. On the basis of information the component goal determination receives about its own social characteristics and its own energy level, it determines the goals the agent is to pursue: for example to find food, or to look for help. The component own characteristics receives information on the agent's energy level from the component own resource management. This information is used to determine the agent's next state (e.g., hungry, normal or in danger). The component plan determination

receives information (1) from the component own characteristics, namely the agent's current state, (2) from the component goal determination, namely which goals are to be pursued and (3) from outside the component, namely the current state of the world. With this information the component plan determination determines which actions to take in the external world.

The component world interaction management interprets information it receives from the external world, and transforms information about actions to be taken in the external world into specifications for actions which the external world can execute. Two components are defined to perform these tasks: the component observation information interpretation and the component action execution preparation. For more details, see (Brazier, Eck, and Treur, 1997).

- *Touring Machines, INTERRAP, ZEUS, and ADEPT*

In the remainder of this section it is discussed how the generic agent model GAM can be refined to obtain a formally specified design model for four other existing agent architectures: Touring Machines (Ferguson, 1992), INTERRAP (Müller, Pischel, and Thiel, 1995; Müller, 1996), ZEUS (Nwana, Ndumu and Lee, 1998), and ADEPT (Jennings, Faratin, Norman, O'Brien, Wiegand, Voudouris, Alty, Miah, and Mamdani, 1996).

The Touring Machines architecture described in (Ferguson, 1992) distinguishes three layers: a reactive layer, a planning layer, and a modelling layer; all layers process concurrently. The reactive layer can be formally specified as an instantiation of the the components world interaction management and agent interaction management in the generic agent model GAM. If reactions on combined input from observation and communication have to be modelled, two information links between world interaction management and agent interaction management are added for direct information exchange, avoiding modelling this information as beliefs. The planning layer can be specified as a refinement of component own process control; also the Control Rules are part of this refinement of own process control. The modelling layer can be obtained by instantiation of the components maintenace of world information and maintenance of agent information, where models of the agent's environment are maintained. The specific approach to control by Control Rules (in the form of Censors and Suppressors) entails that all incoming and outgoing information has to be filtered by the Control Rules within own process control. This means that, although in principle all layers are meant to be connected independently to the outside world, in order to do the filtering, in practice these connections come together in the Control Rules component within own process control. This confirms analyses of this agent architecture available in the literature; e.g., see (Müller, 1996).

Within the INTERRAP architecture (Müller, Pischel, and Thiel, 1995; Müller, 1996), the following components play a role: *World Interface* (Sensors, Communication, and Actors), *Agent KB* (Social Model (SM), Mental Model (MM), World Model (WM)), *Agent Control Unit* (Cooperative Planning Layer (CPL), Local Planning Layer (LPL), Behaviour-Based Layer (BBL)). A formal design specification of the World Interface can be obtained as an instantiation of the components agent interaction management (communication) and world interaction management (sensors, actors) within GAM. A design specification of Agent KB's Social Model can be obtained as an instantiation of the

component maintenance of agent information and the World model of maintenance of world information. The Mental Model can be obtained as a refinement within own proces control, as far as mental concepts referring to the agent itself are concerned. If also mental concepts such as joint intentions are involved, these can be included within cooperation management. The Local Planning Layer can be obtained as a refinement of own process control, the Cooperative Planning Layer of cooperation management, and the Behaviour-Based Layer of the components agent interaction management and world interaction management. The INTERRAP model has a much richer structure than the generic agent model GAM, especially in control aspects. Control differs from the Touring Architecture in that only the Behaviour-Based Layer is connected to the outside world, and the Local Planning Layer (within own process control) becomes involved as soon as the Behaviour-Based Layer indicates that the situation is assessed as beyond its competence. Similarly, own process control can indicate that the situation is beyond its (individual) competence and involve the Cooperative Planning Layer (in cooperation management). For the refinement of GAM this means that it is specified that the appropriate control information is exchanged between world interaction management and agent interaction management, own process control and cooperation management.

The ZEUS architecture distinguishes: Mailbox, Message Handler, Co-ordination Engine, Execution Monitor, Acquaintance Model, Planner and Scheduler, Task/Plan Database, Resource Database. The Mailbox and the Message Handler together can be formally specified as a specialisation and instantiation of the component agent interaction management within GAM. The Co-ordination Engine can be obtained as a refinement of the component cooperation management. The Execution Monitor with the Planner and Scheduler, and the Task/Plan Database together can be specified as a specialisation and instantiation of the component own process control. The Acquaintance Model can be obtained as an instantiation of component maintenance of agent information. Although interaction with the External World is not explicitly modelled within a ZEUS agent, the Resource Database may include some of this information.

	WIM	AIM	MWI	MAI		OPC	CM	AST
animal behaviour	observation action generation	growling	memory	pick order		goal generation	-	-
negotiating agents	-	bids transfer	-	-		decision to negotiate or stop	composed negotiation model	determine balance
information agents	inspecting the WWW	interests and provided information	information on (info) objects	profile information		pro-activeness and reactiveness characteristics	-	models for strict and soft matching
deliberate normative agents		communication about norms	information on world	MAI info about norms of other agents	MSI info about norms in society	composed of: norm management, stategy management, goal management, plan management	-	-
electricity transportation management	-	refined to Generate Outgoing, Receive Incoming	World Model	Acquaintance Models		refined for Monitor Incoming Data, Evaluate Process State	-	Diagnostic Process Model, Analysis Model of Incoming Data, Planning and Monitoring Model
cooperative agent architecture	-	-	-	-		refined for Monitoring, Planning and Control of own activities	refined for Project Generation, Project Monitoring	-
BDI	Beliefs on other agents	Commitment transfer	Beliefs on world	Commitments of others		refined for Handling Beliefs, Desires and Intentions	-	-
society simulation	refined for Observation Information Interpretation, Action Ex. Preparation	-	-	-		refined for Own Resource Management, Own Char., Goal Det., Plan Det.	-	-
Touring Machines	Reactive Layer	Reactive Layer	Environment Model	Agent Models		refinement for Planning Layer, Control Rules	-	-
INTERRAP	Sensors, Actors, Behaviour-Based Layer	Communicat-ion , Behaviour-Based Layer	World Model	Social Model		refinement for Mental Model, Local Planning Layer	Social Mental Model, Cooperative Planning Layer	-
ZEUS	-	Mailbox, Message Handler	Resource Database	Acquaintance Models		refined for Planner and Scheduler, Task/Plan Database, Execution Monitor	Coordination Engine	
ADEPT	-	CM	-	AM		SAM SM	IMM	SEM

Table 10 Overview of refinements of GAM to designs for various agent architectures

The architecture ADEPT (Advanced Decision Environment for Process Tasks; see (Jennings, Faratin, Norman, O'Brien, Wiegand, Voudouris, Alty, Miah, and Mamdani, 1996)) represents business processes by a hierarchy of cooperative agents. The hierarchy ensures that communication overhead between agents and the autonomy of the agents are balanced. Within this model, agents have the following modules: a communication module, an interaction management module (IMM), a situation assessment module (SAM), a service execution module (SEM), a self model (SM), acquaintance models (AM). These modules have been specified as a refinement of GAM as follows: the module IMM as a refinement of the component cooperation management, the modules SAM and SM as components within a specialisation of the component own process control, the module SEM can clearly be described as a specialisation of the component maintenance of agent information.

8 Discussion

This section, first summarizes the process of designing and reusing a generic model, on the basis of the generic agent model GAM. Next current and future research issues are discussed.

8.1 Designing a generic model

The generic agent model GAM was not designed from scratch. Conceptual analysis of agent capabilities and characteristics is the main motivation for the components distinguished in the generic agent model. These components have been distinguished in agent models in different domains of application. Example agent models for the applications described in (Brazier et al., 1995; Brazier et al., 1998), based on a precursor of GAM were an important input for the process of designing the generic agent model in more detail. Further generic structures were extracted from these example models and combined, leaving out domain specific elements.

In a number of cases a choice had to be made. Some other information types could have been included as well. The more structures are included, the more support is given when reusing the generic model. However, this only holds for applications for which the generic structures are relevant: the richer a generic model is, the more restrictive is its scope of application. Since the generic model GAM has been designed to be a very widely applicable model, the choice has been made to limit the number of structures included. As discussed in Section 7, more specialised agent models have been developed as well: for example, a generic model for BDI-agents, in which the component own process control is refined (Brazier, Dunin-Keplicz, Treur and Verbrugge, 1999), and a generic model for co-operation, in which both the components own process control and co-operation management are refined (Brazier, Jonker and Treur, 1997).

8.2 Reusing a generic model

The scope of applicability of the generic agent model GAM covers a variety of application domains, as discussed in Section 7. As the generic model was constructed to subsume a large number of applications, it should not be difficult to reuse the generic model in similar application domains. This paper shows in more depth how the generic agent model can be applied to another application domain: co-operative information gathering agents. As a first step in the reuse of GAM the domain specific knowledge structures were instantiated: domain specific information types and knowledge bases. The information types domain actions and agent information were not considered to be relevant for this application, so these information types remained empty. In fact, the component maintenance of agent information was not used at all and could have been removed. One of the generic knowledge bases in the generic agent model could be reused (observation result extraction kb). Another generic knowledge base (communicated info extraction kb) was replaced by a more specific knowledge base. Moreover, knowledge bases were added to generate communication and observation.

A second step was the addition of two new information types to handle requests for information or observation. Finally, a third step was to add an information link to transfer requests from agent interaction management to world interaction management. The process of reusing a generic model as summarised above has realistic characteristics. In general, if a suitable generic model is available, during the design process:

- most but not all parts of the generic model can be reused as is
- parts that are not used are modified, remain empty or can be removed
- some additional knowledge structures may be needed and added
- some additional information links may be needed and added
- maybe some additional components are needed and added or modified

The example process of reusing the generic agent model GAM discussed in this paper shows almost all of these characteristics.

8.3 Current and Future Research

Current research focuses on requirements engineering and verification for agent systems, and on applications to information brokering agents and Electronic Commerce. Within requirements engineering the aim is to obtain appropriate informal, semi-formal and formal representations of functional or behavioural properties of a multi-agent system, of the agents within a multi-agent system and of components within an agent. A first proposal can be found in (Herlea, Jonker, Treur and Wijngaards, 1999). Requirements specifications can be expressed in generic forms and reused in conjunction with generic models such as GAM. Compositional verification is an approach to establish that behavioural properties of a multi-agent system hold, given properties of agents and of their components; e.g., see (Jonker and Treur, 1998a).

References

Brazier, F.M.T., Cornelissen, F., Gustavsson, R., Jonker, C.M., Lindeberg, O., Polak, B., and Treur, J. (1998). Agents Negotiating for Load Balancing of Electricity Use. In: M.P. Papazoglou, M. Takizawa, B. Krämer, S. Chanson (eds.), Proceedings of the 18th International Conference on Distributed Computing Systems, ICDCS'98, IEEE Computer Society Press, 1998, pp. 622-629.

Brazier, F.M.T., Dunin-Keplicz, B., Jennings, N.R. and Treur, J. (1995). Formal specification of Multi-Agent Systems: a Real-World Case. In: V. Lesser (Ed.), *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95*, MIT Press, Cambridge, MA, pp. 25-32. Extended version in: International Journal of Cooperative Information Systems, M. Huhns, M. Singh, (Eds.), special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, vol. 6, 1997, pp. 67-94.

Brazier, F.M.T., Dunin-Keplicz, B., Treur, J., and Verbrugge, L.C. (1999). Modelling Internal Dynamic Behaviour of BDI agents. In: A. Cesto and P.Y. Schobbes (eds.), Proceedings of the Third International Workshop on Formal Models of Agents, MODELAGE'97. Lecture Notes in AI, Springer Verlag. In press, 1999, pp. 21.

Brazier, F.M.T., Jonker, C.M., Jüngen, F.J., and Treur, J. (1999). Distributed Scheduling to Support a Call Centre: a Co-operative Multi-Agent Approach. Applied Artificial Intelligence Journal, vol. 13, 1999, pp. 65-90. H.S. Nwana and D.T. Ndumu (eds.), Special Issue on Multi-Agent Systems. Earlier shorter version in: H.S. Nwana and D.T. Ndumu (eds.), Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'98. The Practical Application Company Ltd, 1998, pp. 555-576.

Brazier, F.M.T., Jonker, C.M., Treur, J. (1997). Formalisation of a cooperation model based on joint intentions. In: J.P. Müller, M.J. Wooldridge, N.R. Jennings (eds.), *Intelligent Agents III (Proc. of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96),* Lecture Notes in AI, volume 1193, Springer Verlag, 1997, pp. 141-155.

Brazier, F.M.T., Jonker, C.M., and Treur, J. (1998). Principles of Compositional Multi-agent System Development. In: J. Cuena (ed.), Proceedings of the 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98, 1998, pp. 347-360.

Castelfranchi, C., Dignum, F., Jonker, C.M. and Treur, J. (1999). Deliberate Normative Agents: Principles and Architecture. In: N.R. Jennings, Y. Lesperance (eds.), Proc. of the Sixth International Workshop on Agent Theories, Architectures and Languages, ATAL'99. To be published as Intelligent Agents VI. Lecture Notes in AI, Springer Verlag.

Cesta, A., M. Micelli and P. Rizzo (1996). Effects of different interaction attitudes on a multi-agent system performance. In: W. van de Velde and J.W. Perram (Eds.) *Agents Breaking Away*.

Proc. 7th Eur. Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'96. Lecture Notes in Artificial Intelligence, vol. 1038 Springer-Verlag, pp. 128-138.

Chandrasekaran, B. (1986) Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. *IEEE Expert*, Vol. 1, pp. 23–30.

Cockburn, D. and N. R. Jennings (1995). ARCHON: A Distributed Artificial Intelligence System for Industrial Applications. In: G. M. P. O'Hare and N. R. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, Wiley & Sons, pp. 319-344.

Dennett, D. (1987). The Intentional Stance, MIT Press, Cambridge, MA.

Ferguson, I.A. (1992). Touring Machines: An Architecture for Dynamic, Rational, Mobile Agents. Ph.D. Thesis. Computer Laboratory, University of Cambridge, UK.

Ferguson, I.A. (1995). Integrated control and coordinated behaviour. In: (Wooldridge and Jennings, 1995a), pp. 203-218.

Georgeff, M.P. and A.L. Lansky (1987). Reactive Reasoning and Planning. Proc. of the National Conference of the American Association for AI, AAAI'87. Morgan Kaufman.

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E., Specification of Behavioural Requirements within Compositional Multi-Agent System Design. In: F.J. Garijo, M. Boman (eds.), Multi-Agent System Engineering, Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99. Lecture Notes in AI, vol. 1647, Springer Verlag, Berlin, 1999, pp. 8-27.

Jennings, N.R. (1995). Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, *Artificial Intelligence Journal* 74 (2)

Jennings, N. R. , J. Corera, I. Laresgoiti, E. H. Mamdani, F. Perriolat, P. Skarek and L. Z. Varga (1996). Using ARCHON to develop real-word DAI applications for electricity transportation management and particle accelerator control, *IEEE Expert - Special issue on Real World Applications of DAI* .

Jennings, N.R., Faratin, P., T. Norman, T.J. O'Brien, P. Wiegand, M. E. Voudouris, C., Alty, J. L., Miah, T. and Mamdani, E. H. (1996). ADEPT: Managing Business Processes using Intelligent Agents. In: Proc. BCS Expert Systems 96, Conference (ISIP Track), Cambridge, UK 5-23.

Jennings, N.R., and M. Wooldridge (1998a), Applications of Intelligent Agents. In: (Jennings and Wooldridge, 1998b), pp. 3-28.

Jennings, N.R., and M. Wooldridge (eds.) (1998b), Agent Technology: Foundations, Applications, and Markets. Springer Verlag.

Jonker, C.M., Lam, R.A., and Treur, J. (1999). A Multi-Agent Architecture for an Intelligent Website in Insurance. In: Proceedings of the Third International Workshop on Cooperative Information Agents, CIA'99. Lecture Notes in AI, Springer Verlag, 1999. In press.

Jonker, C.M. and Treur, J. (1998a). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380.

Jonker, C.M., and J. Treur, (1998b), Agent-based Simulation of Reactive, Pro-active and Social Animal Behaviour. In: J. Mira, A.P. del Pobil, and M. Ali (eds.), *Methodology and Tools in Knowledge-Based Systems (Proceedings of the 11th International Conference on Industrial and Engineering Applications of AI and Expert Systems, IEA/AIE'98*, vol. I), Lecture Notes in AI, vol. 1415, Springer Verlag, 1998, pp. 584-595.

Jonker, C.M., and Treur, J. (1999). Information Broker Agents in Intelligent Websites. In: I. Imam, Y. Kodratoff, A. El-Dessouki, and M. Ali (eds.), Multiple Approaches to Intelligent Systems (Proc. of the 12th International Conference on Industrial and Engineering Applications of AI and Expert Systems, IEA/AIE'99). Lecture Notes in AI, vol. 1611, Springer Verlag, 1999, pp. 430-439.

Müller, J.P. (1996). The Design of Intelligent Agents: a Layered Approach. Lecture Notes in AI, vol. 1177, Springer Verlag, 1996

Müller, J.P., M. Pischel, and M. Thiel (1995). Modelling reactive behaviour in vetically layered agent architectures. In: (Wooldridge and Jennings, 1995a), pp. 261-276

Nwana, H.S., (1996), Software Agents: an Overview, *Knowledge Engineering Review*, vol. 11(3), pp. 205 - 244.

Nwana, H.S., and D.T. Ndumu, (1998), A Brief Introduction to Software Agent Technology. In: (Jennings and Wooldridge, 1998b), pp. 29 – 47.

Nwana, H.S., Ndumu, D.T. and Lee, L.C. (1998). ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems. In: Proceedings of the Third International Conference on the Application of Intelligent Agents and Multi-Agent Technology (eds. Nwana, H.S. and Ndumu, D.T.), The Practical Application Company, Blackpool, 377-391. Also in Applied AI, vol. 13, pp. 129

Rao, A.S. and Georgeff, M.P. (1991). Modeling rational agents within a BDI architecture. In: R. Fikes and E. Sandewall (eds.), Proceedings of the Second Conference on Knowledge Representation and Reasoning, Morgan Kaufman, pp. 473-484.

Vauclair, J. (1996). Animal Cognition. Harvard Univerity Press, Cambridge, Massachusetts.Wooldridge, M., Jennings, N.R. (eds.) (1995a) *Intelligent Agents*, Lecture Notes in Artificial Intelligence, Vol. 890, Springer Verlag, Berlin,

Wooldridge, M.J., Jennings, N.R. (1995b) Agent theories, architectures, and languages: a survey. In: (Wooldridge and Jennings, 1995a), pp. 1-39.

Wooldridge, M.J., and N.R. Jennings (1995c). Intelligent Agents: Theory and practice. In: *Knowledge Engineering Review*, 10(2), pp. 115-152.