Compositional Design and Maintenance of Broker Agents

Catholijn M. Jonker and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence

De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands.

URL: http://www.cs.vu.nl/~{jonker,treur}. Email: {jonker,treur}@cs.vu.nl

Abstract

A generic broker agent architecture is introduced, designed in a principled manner using the compositional development method for multi-agent systems DESIRE. A flexible, easily adaptable agent architecture results in which, in addition, facilities have been integrated that provide automated support of the agents own maintenance. Therefore, the agent is not only easily adaptable, but it shows adaptive behaviour to meet new requirements, supported by communication with a maintenance agent.

1. Introduction

To support users on the World Wide Web, various types of agents can be, and actually have been, developed. For example, to support brokering processes in electronic commerce, personal assistant agents can be developed that support a user offering products (or services) at the Web, or agents that support search for information on products within a user's scope of interest, or agents that combine both functionalities. Moreover, mediating agents can be developed that communicate both with personal assistant agents that provide information on products and with personal assistants that ask for information on products. Recently a few applications of broker agents have been addressed for this area; for example, see [5], [6], [9], [10], [11], [12]. In general, applications like these are implemented in an ad hoc fashion without an explicit design at a conceptual level, and without taking into account the dynamic requirements imposed by the domain of application and the maintenance problem (including desired extension or modification of functionality) implied by this dynamic character.

In this paper a generic agent architecture for agents acting in brokering processes is introduced which has been designed in a principled manner, using the compositional development method for multi-agent systems DESIRE. The agent architecture can be instantiated by adding specific types of knowledge to support functionalities and behaviour required. Depending on the choice of these requirements, an agent is created for a specific context by including the appropriate types of knowledge. For example, a search agent with functionality restricted to (incidental) search for information upon a user's request can be built by adding only knowledge needed for this task. Such an agent, for example, is not able to store and maintain the user’s query or information that has been found, nor is it able to provide information to other agents. If these functionalities are required as well, the necessary types of knowledge have to be added.

Requirements imposed at the time of creation determine the functionalities and behaviour of the agent. If later on additional or changed requirements come up, the agent has to be modified or redesigned to meet the new requirements. The agent architecture introduced here supports its own modification due to the fact that basic functionalities are specified in an explicit declarative manner, in the form of knowledge. It is possible to dynamically modify the agent by adding or deleting some of its knowledge. Since this declaratively represented knowledge can be the subject of communication between agents, modification can be performed at a distance: another agent (e.g., a dedicated maintenance agent) communicates the knowledge needed for the modification to the agent that is to be modified. Thus a flexible agent implementation is obtained that can be maintained and evolve over time on the basis of communication with other agents only.

In Section 2 an example problem domain for brokering processes is sketched. Section 3 introduces the design of the generic agent architecture. The different types of knowledge are presented in Section 4. In Section 5 the behaviour of the system is analysed by giving an overview of which types of knowledge are needed for which types of basic functionalities. Finally, in Section 6 it is shown how the agent can communicate that a specific functionality is required, and it is explained how, after appropriate maintenance knowledge has been communicated to the agent, the agent performs its own dynamic modification in order to obtain the required functionalities.

2. Electronic Commerce and Brokering

The process of brokering as often occurs in electronic commerce involves a number of agents. A user offering products may be supported by a provider agent that provides information about the products to other (human or computer) agents. A user looking for products may be supported by a personal assistant agent that takes its user’s queries and contacts other agents or looks at the Web directly to find information on products within the user’s scope of interest. Such a personal assistant agent may contact either provider agents immediately, or mediating agents, which in turn have contact with provider agents, or other mediating agents. Depending on the application, the chain of agents involved may include zero or more mediating agents.

The domain analysed for the agent architecture presented here is the domain of brokering (scientific) papers. This domain has a number of aspects in common with other domains within the area of electronic commerce. The task of a provider agent is to inform other researchers on papers available on Internet (a marketing aspect), for example an agent related to a Web site of a research group, announces new papers included in their Web site. If a researcher is looking for a paper with certain characteristics (scope), a personal assistant agent can ask other agents for information on papers with these characteristics. To be able to tune the information provided to users, a number of scopes of interest can be maintained for each of the users. For example, one of the users may be interested in papers on certain topics, such as work flow management systems, but also in papers on agents and the World Wide Web.

Topics can be basic (e.g., ‘work flow management systems’, or ‘agents’, or ‘World Wide Web’), or a combination of a number of topics (e.g., ‘agents and World Wide Web’). In the latter case the user interest is limited to papers which address both topics. Moreover, if it is added that the user is only interested in papers from the years 1995 to 1997, then either year in the range 1995-1997 is meant. Topics can be matched with, for instance, the set of keywords of a paper, or with the abstract, or the paper as a whole. In some disciplines, such as Medicine, an ontology of topics has been developed that serves more or less as a standard. Besides topics also other attributes of papers can be used to define a scope of interest, for example an author, a year, a research group, et cetera. These attributes can also be used in combination with each other. For this example, a shared ontology of topics is assumed. All agents in the brokering process express their information and interests using this shared ontology. It is assumed that the following attributes of a paper are available and can be used: title, authors, affiliation(s) of the authors, location on the World Wide Web where it can be found, topics covered by the paper, abstract, year, and reference. This information can be used to identify papers that are of interest to a user, but also forms the source for the information that can be provided to a user when a paper is proposed to him or her.

3. Compositional Design of the Generic Broker Agent

3.1 Compositional Design of Multi-Agent Systems

The emphasis in the compositional design method for multi-agent systems DESIRE (cf., [1]) is on the conceptual and detailed design. The design of a multi-agent system in DESIRE is supported by graphical design tools within the DESIRE software environment. The software environment includes implementation generators with which (formal) design 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*, and the *relation* between process composition and knowledge composition. These three types of knowledge are discussed in more detail below.

3.1.1 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.

Identification of Processes at Different Levels of Abstraction

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.

Composition of Processes

The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *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).

3.1.2. 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.

Identification of knowledge structures at different abstraction levels

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.

Composition of Knowledge Structures

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.

3.1.3 Relation between Process Composition 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.

3.2 Design of the Generic Broker Agent

For the design of the generic broker agent the main aspects introduced above are considered: process composition, knowledge composition, and relations between knowledge and process composition. A compositional generic agent model GAM (introduced in [3]), supporting the weak agency notion (cf. [13]) is used. At the highest abstraction level within an agent, a number of processes can be distinguished that support interaction with the other agents. First, a process that manages communication with other agents, modelled by the component agent interaction management in *Figure* 1. This component analyses incoming information and determines which other processes within the agent need the communicated information. Moreover, outgoing communication is prepared. Next, the agent needs to maintain information on the other agents with which it co-operates: maintenance of agent information. The component maintenance of world information is included to store the information on world information (e.g., information on attributes of products). The process own process control defines different characteristics of the agent and determines foci for behaviour. The component world interaction management is included to model interaction with the world (with the World Wide Web world, in the example application domain): initiating observations and receiving observation results.

The agent processes discussed above are generic agent processes. Many agents perform these processes. In addition, often agent-specific processes are needed: to perform tasks specific to one agent, for example directly related to a specific domain of application. In the current example the agent has to determine proposals for other agents. In this process information on available products (communicated by information providing agents and kept in the component maintenance of world information), and about the scopes of interests of agents (kept in the component maintenance of agent information), is combined to determine which agents might be interested in which products. For the broker agent this agent-specific task is called determine proposals. *Figure* 1 depicts how the broker agent is composed of its components (process composition); this composition is simply reused from the generic agent model GAM.

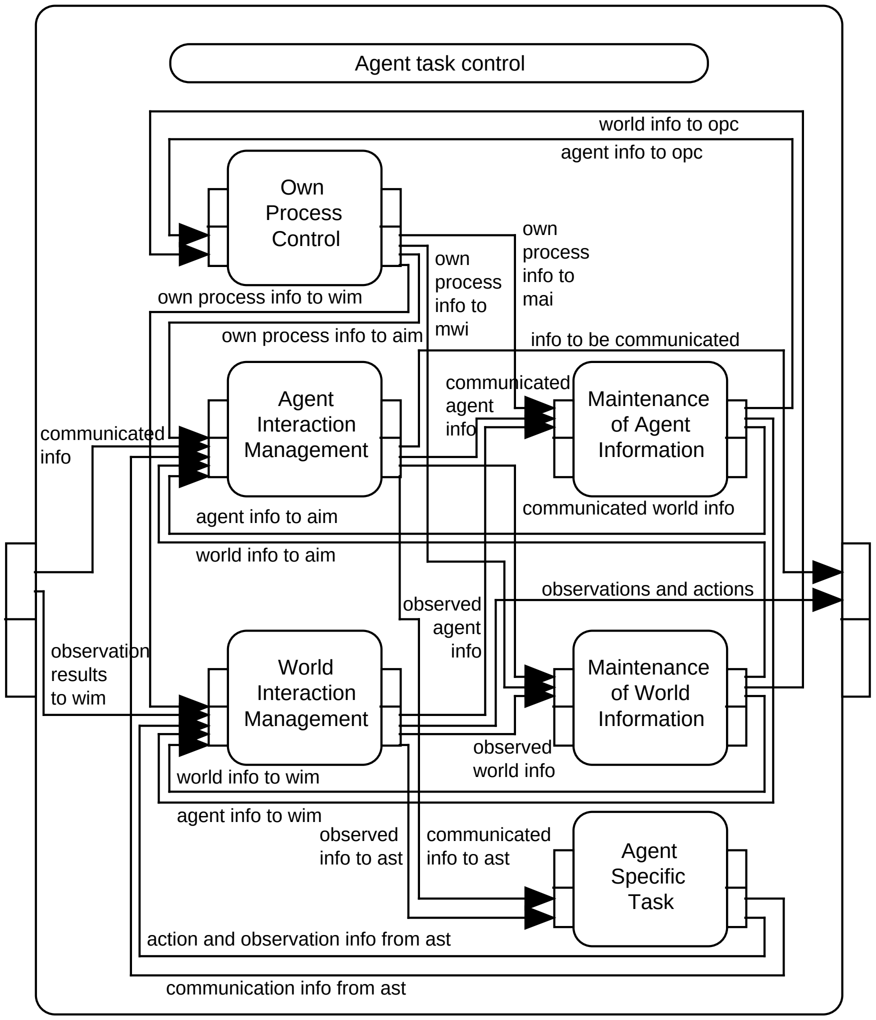

Figure 1 Composition at the highest level within the generic agent model GAM

Part of the exchange of information within the generic agent model GAM can be described as follows. The broker agent needs input about scopes of interests put forward by agents and information about attributes of available products that are communicated by information providing agents. It produces output for other agents about proposed products and the attributes of these products. Moreover, it produces output for information providers about interests. In the information types that express communication information, the subject information of the communication and the agent from or to whom the communication is directed are expressed. This means that communication information consists of statements about the subject statements that are communicated.

Within the broker agent, the component own process control uses belief input information and generates focus information: to focus on a scope of interest to be given a preferential treatment, i.e., pro-active behaviour will be shown with respect to this focus. The component agent interaction management has the same input information as the agent (incoming communication), extended with belief info and focus info. The output generated includes part of the output for the agent as a whole (outgoing communication), extended with maintenance info (information on the world and other agents that is to be stored within the agent), which is used to prepare the storage of communicated world and agent information.

Information on attributes of products is stored in the component maintenance of world information. In the same manner, the beliefs of the agent with respect to other agents' profiles (provider attribute info and interests) are stored in maintenance of agent information. The agent specific task determine proposal uses information on product attributes and agent interests as input to generate proposals as output.

For reasons of space limitation the generic and domain-specific information types within the agent model are not presented.

4. Generic and Domain Specific Knowledge

The different knowledge abstraction levels introduced for information types can also be exploited to structure the knowledge. Parts of the knowledge can be formulated in terms of scopes, abstracting from attributes and values. Other knowledge is used to perform the abstraction step: it can be used to derive conclusions in terms of scopes from input in terms of attributes and values. The knowledge bases are discussed below in the context of the component in which they are used.

4.1. Agent Specific Task: Determine Proposals

To determine proposals fitting a given scope of interest, information on products has to be compared to this scope of interest. To this end, the information on products, expressed in terms of their attributes has to be aggregated to information in terms of scopes. This can be derived using two knowledge bases, attribute and scope kb, which defines the relations between attributes and scopes in general, and product scope abstraction kb, which identifies for which scope(s) a product is relevant.

The composition of the knowledge in these two knowledge bases supports reuse. For example, if in one of the two knowledge bases, modifications are made, the other knowledge base still can be used. Moreover, the first knowledge base is specified independent of knowledge about products. It can be (re)used within the component maintenance of agent information as well, as will be shown below.

Given information on the scopes of products, by the knowledge base strict match kb it is defined how proposals to agents can be generated by matching the scopes of products and the scopes in which an agent is interested. For strict matching it consists of only the following knowledge.

```
if      A  is interested in scope S
  and   product P is in scope S
then    product P is interesting for A because of scope S
```

In formal specification form this is expressed as:

Knowledge base **strict match kb**

```
if      interested_in(A:AGENT, S:SCOPE)
  and   in_scope(P:PRODUCT, S:SCOPE)
then    is_interesting_for(P:PRODUCT, A:AGENT, S:SCOPE)
```

This knowledge simply states that if a product is in a scope an agent is interested in, then this product is interesting for this agent. Alternative knowledge bases can be used for non-strict matching.

4.2. Agent Interaction Management

The component agent interaction management makes use of four knowledge bases: (1) for incoming communication from agents asking for information on products, (2) incoming communication from agents providing information, (3) outgoing communication to agents interested in information on products, and (4) outgoing communication to agents providing information.

4.2.1. Incoming Communication

If an agent communicates her or his interests, then this information is identified as new agent interest information that is currently believed (which can be forgotten after the agent has reacted on it: knowledge base agent interest identification kb) or that has to be stored (in which case it can be remembered later: knowledge base agent interest maintenance identification kb). A condition for storage of interests information is that the type of contract is persistent. For agents with a weaker type of contract no requests are stored. The knowledge used in structured natural language is:

if	communicated by agent A:	interest in scope S
and	belief:	A has a contract of type persistent
then	new agent info:	A is interested in scope S

In concise formal form this is specified as:

Knowledge base **agent interest maintenance identification kb**

```
if      communicated_by(interest(S:SCOPE), V:SIGN, A:AGENT)
  and   belief(has_contract(A:AGENT, contract_type(persistent, Y)), pos)
then    new_agent_info(interested_in(A:AGENT, S:SCOPE), V:SIGN)
```

If an agent communicates that he or she wants to subscribe for a contract of a certain type, then this information is identified as new contract information that has to be stored. This identification makes use of the following knowledge.

if	communicated by agent A:	subscription for contract type C
then	new agent info:	A has contract of type C

In formal form:

Knowledge base **subscription identification kb**

```
if     communicated_by(subscription_for(C:CONTRACT_TYPE), V:SIGN, A:AGENT)
then   new_agent_info(has_contract(A:AGENT, C:CONTRACT_TYPE), V:SIGN)
```

If an agent communicates information about products it provides, this incoming information is analysed, new world information is identified as new information that can be used immediately and forgotten afterwards (knowledge base provider info identification kb), or has to be stored (knowledge base provider info maintenance identification kb). If an agent communicates information about products it provides, this incoming information can also be analysed, in order to obtain new agent information on the scopes of the information the agent (apparantly) can provide. This is expressed by:

```
if      communicated by agent A:    product P's attribute A has value V
then    new agent info:             agent A can provide attribute A with value V
```

In concise formal form:

Knowledge base provider scope maintenance identification kb:

```
if     communicated_by(attribute_has_value(P:PRODUCT, A:ATTRIBUTE, V:VALUE), pos, A:AGENT)
then   new_agent_info(can_provide(A:INFO_PROVIDER, A:ATTRIBUTE, V:VALUE), pos)
```

4.2.2. Outgoing Communication

New information (product identification, scope, or attribute information) on a product that may be interesting for an agent is communicated to this agent. This is expressed in the following knowledge.

```
if      belief:                        product P is interesting for agent A because of scope S
 and    belief:                        I  is info on product P (of the form P's attribute A has value V)
then    is to be communicated to A:    P is interesting because of scope S
  and   is to be communicated to A:    info I
```

In concise formal form:

Knowledge base proposal communication kb:

```
if     belief (is_interesting_for(P:PRODUCT, A:AGENT, S:SCOPE), pos)
  and  belief(attribute_has_value(P:PRODUCT, A:ATTRIBUTE, V:VALUE), pos)
then   to_be_communicated _to(is_interesting(P:PRODUCT, S:SCOPE), pos, A:AGENT)
  and  to_be_communicated _to(attribute_has_value(P:PRODUCT, A:ATTRIBUTE, V:VALUE), pos, A:AGENT)
```

The agent only communicates to an information provider if a scope has been taken as a focus, and if the information provider can provide products on this scope. This is expressed by:

```
if      in search focus:                       scope S
 and    belief:                                agent A can provide scope S
then    is to be communicated  to agent A:     interest in scope S
```

In concise formal form:

Knowledge base info provider request kb

```
if      in_search_focus(S:SCOPE)
  and   belief(can_provide_scope(A:AGENT, S:SCOPE), pos)
then    to_be_communicated _to(interest(S:SCOPE), pos, A:AGENT)
```

4.3. Own Process Control

By means of the knowledge base focus kb, used within own process control, the types of proposals to be determined and the scopes on which to focus search are determined, as indicated by the following knowledge base. For example, in the first knowledge element it is expressed that for an agent with a contract of any type, proposals will be determined that fit the agent's interests. This is in contrast with, for example, the second knowledge element which expresses that only for agents with a persistent contract type, their scopes of interests will be chosen as persisting search foci (otherwise these scopes of interest will be forgotten after being handled).

Knowledge base **focus kb**

```
if      belief(has_contract(A:AGENT, C:CONTRACT:TYPE))
  and   belief(interested_in (A:AGENT, S:SCOPE), pos)
then    product_to_be determined(in_scope (P:PRODUCT, S:SCOPE))
  and   proposal_to_be determined(is_interesting_for(P:PRODUCT, A:AGENT, S:SCOPE))

if      belief(has_contract(A:AGENT, contract_type(persisting, search_for_info)))
  and   belief(interested_in (A:AGENT, S:SCOPE), pos)
then    in_persisting_search_focus(S:SCOPE)

if      belief(has_contract(A:AGENT, contract_type(incidental, search_for_info)))
  and   belief(interested_in (A:AGENT, S:SCOPE), pos)
  and   not  search_focus_processed_for(S:SCOPE, A:AGENT)
then    in_incidental_search_focus(S:SCOPE)
  and   search_focus_chosen_for(S:SCOPE, A:AGENT)

if      in_persistent_search_focus(S:SCOPE)
then    in_ search_focus(S:SCOPE)

if      in_incidental_search_focus(S:SCOPE)
then    in_ search_focus(S:SCOPE)

if      in_ search_focus(S:SCOPE)
then    provider_to_be_determined_for(S:SCOPE)
```

4.4. World Interaction Management

The component world interaction management alllows the agent to look for information by observation. This entails generation of observations to be performed, and obtaining the observation results. The obtained observation results can be used incidentally after which the information is forgotten (using knowledge base observation info identification kb) or maintained to be used later as well (using knowledge base observation maintenance identification kb, similar to agent interaction management. The agent only observes if a scope has been taken as a focus. This is expressed using knowledge base observation initiative kb.

Knowledge base **observation initiative kb**

```
if      in_search_focus(S:SCOPE)
then    to_be_observed(S:SCOPE)
```

The actual execution of the observation does not take place within the agent, but in the external world. As part of the external world an engine can be used to search for products matching the pattern defined by the specified scope. The result of such an observation will be all information of any product that matches the scope. The knowledge base specified above is kept rather simple. To avoid too frequent repetition of observation, more sophisticated knowledge can be specified.

4.5. Maintenance of World and Maintenance of Agent Information

In principle, the components maintenance of world information and maintenance of agent information store information. The knowledge base attribute and scope kb defined above is also used in the component maintenance of agent information . In addition, the knowledge base provider scope abstraction kb is defined; it is similar to the knowledge base product scope abstraction kb mentioned above.

5. The Behaviour

The behaviour of the broker agent can be analysed in different ways. One way is to consider its basic functionalities with respect to its brokering task, and use these as building blocks to obtain behaviour. For example, its behaviour in terms of the weak notions of agency (autonomy, social ability, reactivity, and pro-activity) can be determined in terms of basic functionalities. Moreover, basic functionalities can be related to knowledge bases that are available within the agent. Using these two relationships, a relation can be identified between behaviour and available knowledge within the agent.

5.1. Basic Functionalities Depending on the Agent's Knowledge

The broker agent shows behaviour depending on certain basic functionalities. For the agent model presented, these basic functionalities have been specified in a declarative manner by the agent's knowledge. For each of the basic properties of the agent it has been established which knowledge bases are required. By varying the choice of knowledge for the agent, different types of agents can be designed.

1. *Observation of information available within a certain part of the world*

Observation requires the ability to iniate observations, specified in the knowledge base observation initiative kb, and the ability to identify the information resulting from an observation, specified in the knowledge base observation info identification kb. Both knowledge bases can be used within the component world interaction management.

2. *Communication with agents asking for information on products*

The basic functionality to communicate with agents asking for information on products requires the processing of incoming communication of asking agents and preparation of outgoing information. The incoming information may refer to scopes of interests of the asking agent, or to subscription. The communicated scopes of interest are identified using the knowledge base agent interest identification kb. Incoming communication on subscription is identified using knowledge base subscription identification kb. Outgoing communication containing product information to agents asking for information is prepared using knowledge base proposal communication info. All these knowledge bases are used within the component agent interaction management.

3. *Communication with agents providing information on products*

Communicated information on products can be processed in two different ways. First, the product information can be identified, using knowledge base provider info identification kb. Second, from the fact that information is provided on a product with certain characteristics, it can be abstracted (from the given product) that this provider is able to offer (at least some) products with these characteristics in general. This is done using knowledge base provider scope identification kb. Communication to an agent that may be able to provide information is prepared using knowledge base provider request kb. All these knowledge bases are used within component agent interaction management.

4. *Maintenance of acquired information on products*

The agent can identify that all communicated information on products has to be stored, using knowledge base provider info maintenance identification kb, within component agent interaction management. Moreover, by knowledge base observation info maintenance identification kb, within component world interaction management, new observation results on products to be stored can be identified.

5. *Maintenance of scopes of interest of agents*

The agent can identify that the incoming requests of agents are to be maintained. This functionality is specified by the knowledge base agent interest maintenance identification, used within component agent interaction management.

6. *Maintenance of scopes of products agents can provide*

Scopes of information agents can provide are stored, if the incoming communication is handled in an appropriate way using knowledge base provider scope maintenance identification kb, used within component agent interaction management.

7. *Own control*

Control of the agent's own processes is defined by the knowledge base focus kb, used within component own process control.

8. *Determining matches between products and scopes of interests*

To determine matches between products and scopes of interest the agent can use the knowledge bases attribute and scope kb, product scope abstraction kb, and strict match kb within component determine proposals.

basic functionality	*knowledge specifying functionality*	*in component*
1. observation	observation initiative kb observation info identification kb	WIM WIM
2. communication with agents asking for information	agent interest identification kb subscription identification kb proposal communication kb	AIM AIM AIM
3. communication with agents providing information	provider info identification kb provider scope identification kb provider request kb	AIM AIM AIM
4. maintenance of product information	observation info maintenance identification kb provider info maintenance identification kb	WIM AIM
5. maintenance of scopes of interest	agent interest maintenance identification kb	AIM
6. maintenance of scopes of products agents can provide	provider scope maintenance identification kb provider scope abstraction kb attribute and scope kb	AIM MAI MAI
7. own control	focus kb	OPC
8. match between products and scopes of interests	attribute and scope kb product scope abstraction kb strict match kb	DP DP DP

Figure 2 Relation between basic functionalities and knowledge required

Combinations of these functionalities define specific types of agents. For example, if a provider agent is designed, functionalities 2., 4., 5., 8. may be desired, whereas functionalities 1., 3., 6., 7. could be left out of consideration. If an agent is designed to support a user in finding information on products within a certain scope, functionalities 1., 3., 6., 8. (and perhaps 4.) may be desired, whereas 2. and 5. may be less relevant. For a mediating agent, or for an agent that has to play different roles, almost all functionalities (i.e., 2. to 8.) may be desired. The generic agent architecture introduced in Sections 3 and 4 can be instantiated in different manners to obtain, among others, the types of agents mentioned. The relation between the agent's basic functionalities, its knowledge, and where the knowledge is used is summarized in the table in *Figure* 2.

5.2. Reactive, Pro-active, and Other Forms of Behaviour

Depending on the choices made, the broker can show reactive behaviour towards agents asking for information on products and provider agents.

- In reaction to an agent that asks for products within a certain scope, the broker determines which of the products it knows, fit to this scope. The available information on the resulting products is communicated to the agent (e.g., author, title, year, topics, abstract, location, reference).
- Once an agent interest is known to the broker agent, it is reactive to any information providing agent that announces a product that fits the agent's scope. In such a case the information on this product is communicated to this agent (i.e., to all relevant agents).

Pro-active behaviour occurs when the broker agent has as a characteristic that it is pro-active with respect to certain agents. A pro-active broker, from time to time takes the initiative to ask provider agents for information on products which match some of its subscribed request profiles. It may focus on an agent's scopes of interest and actively select information providing agents and ask them whether they have products that fit in one of these scopes.

The behaviour of the broker agent may depend on other characteristics of the broker agent as well. In the example, the knowledge used within own process control was kept rather simple. It is not difficult to extend this knowledge in such a way that more complex forms of pro-active social behaviour are initiated and controlled. For example, it is also possible that the broker pro-actively determines an expected scope of interest of an agent and proposes products that fit this expected scope of interest.

6. Maintenance by Communication

The requirements on functionality of a broker agent may evolve over time; a user may first only ask for an agent that is able to communicate information on its own products, and later ask for more functionality. For example, functionality that allows the agent to look for information on products itself, and store this information. In the compositional agent model presented here, a substantial part of the agent's functionality is represented in a declarative manner by explicit knowledge. An important advantage of this form of representation is that this functionality specification can be adapted easily by replacing (parts of) the knowledge bases by others, and, even more important, the knowledge can easily be made a subject of communication. This can be exploited to design an agent that supports its own maintenance on the basis of (new or updated) knowledge that is communicated to it.

6.1. Communication of Maintenance Knowledge

The agent model has been extended with possibilities to support its own maintenance in the following manner. First, communication on knowledge has been added to the model by extending the information types dealing with communication by explicit representation of knowledge elements as terms:

```
knowledge_element(<kb-name>, <antecedent>, <consequent>)
```

Both antecedents and consequents are taken from the sort CONJ, which stands for conjunctions of literals. Besides knowledge it may be important to communicate which functionality is supported by a knowledge base. This is expressed by a relation

```
supports_functionality: KB *COMPONENT* FUNCTIONALITY
```

that can be used to represent the knowledge depicted in the table of *Figure* 2 in Section 5. Within the component agent interaction management a knowledge base is added to extract the maintenance information from the communication information:

Knowledge base maintenance knowledge identification kb

```
if    communicated_by(knowledge_element(K:KB, A:CONJ, C:CONJ), pos, maintenance_agent)
then  new_maintenance_info(knowledge_element(K:KB, A:CONJ, C:CONJ), pos)

if    communicated_by(supports_functionality(K:KB , C:COMPONENT, F:FUNCTIONALITY), pos,
      maintenance_agent)
then  new_maintenance_info(supports_functionality(K:KB , C:COMPONENT, F:FUNCTIONALITY), pos)
```

The extracted maintenance information is transferred to the agent component own process control and stored in this component using the relation maintenance info. It is also possible to model outgoing communication on maintenance. For example, if the broker agent decides that is needs a new functionality, this can be communicated to the maintenance agent:

```
to_be_communicated_to(required_functionality(F:FUNCTIONALITY), pos, maintenance_agent)
```

The maintenance agent can react on this communication by communicating maintenance information as described above.

6.2. Controlling Maintenance in own process control

Within the agent component own process control the (sub-)component own maintenance management is distinguished. Within this component the maintenance information received is analysed, and if found adequate, the decision is made to execute maintenance. The following knowledge is used:

```
if    maintenance_info(supports_functionality(K:KB , C:COMPONENT, F:FUNCTIONALITY), pos)
  and required_functionality(F:FUNCTIONALITY)
then  kb_to_be_added_in(K:KB , C:COMPONENT)

if    kb_to_be_added_in(K:KB , C:COMPONENT)
  and maintenance_info(knowledge_element(K:KB, A:CONJ, C:CONJ), pos)
then  kb_elt_to_be_added_in(knowledge_element(K:KB, A:CONJ, C:CONJ), C:COMPONENT)
```

Conclusions drawn using this knowledge are transferred to the component to which the reference is made, which actually executes the maintenance. Similarly, retraction of knowledge (if a functionality is not desired anymore) is modelled. After these changes the agent will show the functionalities related to the modified knowledge.

7. Discussion

Applications of broker agents (addressed in, e.g., [5], [6], [9], [10], [11], [12]), often are not implemented in a principled manner: without an explicit design at a conceptual level, and without support maintenance. The broker agent architecture introduced here was designed and implemented in a principled manner, using the compositional development method for multi-agent systems DESIRE [1]. An application to the design of an intelligent Website in insurance can be found in [7]. Due to its compositional structure it supports reuse and maintenance; a flexible, easily adaptable architecture results. Moreover, within the agent model facilities have been integrated that provide automated support of the agent's own maintenance. Therefore, the agent is not only easily adaptable, but, because adaptation is automated, it shows adaptive behaviour to meet new requirements (either in reaction to communication with a maintenance agent, or fully autonomous).

The approach to automated maintenance introduced here has a high potential. In future research this potential will be explored further. In current research the broker agent model is applied within a project on electronic commerce, in co-operation with the Internet application company Crisp, and in a project on intelligent Web-sites in co-operation with the insurance company Delta Lloyd and the software company Ordina Utopics. The approach introduced here addresses dynamic modification of the agent's own knowledge. Dynamic modification of the agent's own process composition (e.g., adding/deleting components or information links) is another possibility, which may result in complete re-design of the agent; this is addressed in see [4].

Required properties or functionalities of agents can be formalised, and the relation between required properties and underlying assumptions can be established in a formal manner. An example of a result of such a formal analysis is the relation between basic functionalities (required properties) and available knowledge (assumptions) discussed in Section 5 (see *Figure* 2). In this paper the result of formal analysis was used in the agent model; the formal analysis itself was done by us as designers. To support this, a compositional verification method for multi-agent systems has been developed and successfully applied to verify the behaviour of a multi-agent system for one-to-many negotiation (see [2]), and to give a formal analysis of pro-activeness and reactiveness (see [8]). One of the more ambitious aims of our future research is to explore possibilities to include these formal analyses themselves in an agent model, and not only the results obtained by them.

Acknowledgements

Frances Brazier, Mehrzad Kharami and Pascal van Eck made useful comments on an earlier draft of the text. Mehrzad and Pascal also supported experiments with some variants of the broker agent model.

References

[1] BRAZIER, F.M.T., DUNIN-KEPLICZ, B., JENNINGS, N.R., TREUR, J., 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, 1995, 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.

[2] BRAZIER, F.M.T., CORNELISSEN, F. GUSTAVSSON, R., JONKER, C.M., LINDEBERG, O., POLAK, B., TREUR, J., Compositional Design and Verification of a Multi-Agent System for One-to-Many Negotiation. In: Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98, IEEE Computer Society Press, 1998, pp. 622-629.

[3] BRAZIER, F.M.T., JONKER, C.M., TREUR, J., Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal,* vol. 14, 2000, pp. 491-538.

[4] BRAZIER, F.M.T., JONKER, C.M., TREUR, J., WIJNGAARDS, N.J.E., Compositional Design of a Generic Design Agent. *Design studies journal*, 2001, in press. Extended abstract: Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98, IEEE Computer Society Press, 1998.

[5] CHAVEZ, A., MAES, P., Kasbah: An Agent Marketplace for Buying and Selling goods. In: Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'96, The Practical Application Company Ltd, Blackpool, 1996, pp. 75-90.

[6] CHAVEZ, A., DREILINGER, D., GUTMAN, R., MAES, P., A Real-Life Experiment in Creating an Agent Market Place. In: Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'97, The Practical Application Company Ltd, Blackpool, 1997, pp. 159-178.

[7] JONKER, C.M., LAM, R.A., AND TREUR, J., A Multi-Agent Architecture for an Intelligent Website in Insurance. In: M. Klusch, O. Shehory, and G. Weiss (eds.), *Cooperative Information Agents III, Proceedings of the Third International Workshop on Cooperative Information Agents, CIA'99.* Lecture Notes in Artificial Intelligence, vol. 1652. Springer Verlag, 1999, pp. 86-100. Extended version in: *Journal of Applied Intelligence,* 2001, in press.

[8] JONKER, C.M., TREUR, J., 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, Springer Verlag, 1998. Extended version in: *International Journal of Cooperative Information Systems,* 2001, in press.

[9] KUOKKA, D., HARADA, L., On Using KQML for Matchmaking. In: V. Lesser (ed.), Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95, MIT Press, Cambridge, MA, 1995, pp. 239-245.

[10] MARTIN, D., MORAN, D., OOHAMA, H., CHEYER, A., Information Brokering in an Agent Architecture. In: Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'97, The Practical Application Company Ltd, Blackpool, 1997, pp. 467-486.

[11] SANDHOLM, T., LESSER, V., Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Network. In: V. Lesser (ed.), Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95, MIT Press, Cambridge, MA, 1995, pp. 328-335.

[12] TSVETOVATYY, M., GINI, M., Toward a Virtual Marketplace: Architectures and Strategies. In: Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'96, The Practical Application Company Ltd, Blackpool, 1996, pp. 597-613.

[13] WOOLDRIDGE, M., JENNINGS, N.R., Agent theories, architectures, and languages: a survey. In: [14], pp. 1-39.

[14] WOOLDRIDGE, M., JENNINGS, N.R. (eds.), Intelligent Agents, Proc. of the First International Workshop on Agent Theories, Architectures and Languages, ATAL'94, Lecture Notes in AI, vol. 890, Springer Verlag, 1995.