Compositional Specification of a Reusable Co-operative Agent Model

Brazier, F.M.; Cornelissen, F.J.; Jonker, C.M.; Treur, J.

published in
International Journal of Cooperative Information Systems
2000

DOI (link to publisher)
10.1142/S0218843000000120

document version
Publisher's PDF, also known as Version of record

Compositional Specification and Reuse of a Generic Co-operative Agent Model

Frances M.T. Brazier, Frank Cornelissen, 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,frankc,jonker,treur}. Email: {frances,frankc,jonker,treur}@cs.vu.nl

Abstract
In this paper one of the informally described models of agent co-operation (Jennings, 1995) has been used to develop and formally specify a generic model of a co-operative agent (GCAM). The compositional development method for multi-agent systems DESIRE supported the principled design of this model of co-operation. To illustrate reusability of the generic model, two application domains have been addressed: collaborative engineering design, and Call Center support.

Keywords: agent, co-operation, reuse, generic model, compositional

Introduction

In current engineering practice well-structured hierarchical management and decentralised project (virtual) organisation are often combined. The combination of traditional management structures and virtual organisations result in dynamic organisational structures, liable to considerable change during the life span of a project. Distributed project co-ordination is essential.

The types of interaction encountered in such distributed real-life design situations show how intricate such processes can be. Although the problem of distributed project co-ordination has been recognised only a few attempts have been made to support the development of support systems; see, for example (Dunskus, Grecu, Brown and Berker, 1995; Petrie, 1994; Goldman, 1996; Gupta, Chionglo and Fox, 1996; Maurer, 1996). Jennings (1995) has proposed an informal multi-agent model for cooperative problem solving. Essential elements of this model are the dynamic organisation and management of joint activities, susceptive to change due to unexpected events. As described by Jennings, the model, however, does not provide enough detail to support analysis, modelling and implementation of design co-ordination systems in specific domains. To acquire a more precise description of this model more detailed analysis is required. The DESIRE development method and environment provides support at this level; see (Brazier, Jonker and Treur, 1998) for the underlying principles, and (Brazier, Dunin-Keplicz, Jennings, and Treur, 1995) for a real-world case study. In this paper a generic model for a co-operative agent (GCAM) is presented: a model developed on the basis of an existing generic agent model (GAM) and Jennings' model. To assess the value of the generic model of a co-operative agent, the model is applied to the analysis of a real distributed design process in the domain of aircraft design, and then used to design a new system in a completely different domain, namely the domain of call centre support.

The compositional approach to the development of multi-agent systems, DESIRE, is described in Section 2. The use of generic models is briefly discussed in Section 3. An existing generic agent model is described in Section 4. This model is extended and refined to model co-operation in Section

5. In Section 6 a real-life design project is analysed for a situation in which traditional management and virtual organisations are combined: the design of part of the interior of a specific aircraft. Section 7 describes the application of the generic model of the co-operative agent model to the design of a multi-agent system to support distributed scheduling for a call centre.

2. Compositional Development of Multi-Agent Systems

The compositional development method DESIRE for multi-agent systems (DEsign and Specification of Interacting REasoning components, cf. (Brazier, Dunin-Keplicz, Jennings, and Treur, 1995), supports the design and development of multi-agent systems from conceptual design through to (prototype) implementation, supported by an environment that includes graphical design tools and automated support for the translation of a specification to an operational system; 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.

2.1. 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.

2.1.1. 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.

2.1.2. 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).

2.2. 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.

2.2.1. 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.

2.2.2. 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.

2.3. 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.

3 Generic Models and Reuse

Within the development method DESIRE , instead of designing each and every new agent application from scratch, existing generic models can be used. Generic models can be distinguished for specific types of agents, agent tasks and 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*

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, at a lower level of process abstraction are distinguished. 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 4 a generic agent model for weak agency is presented. The generic model for co-operative agents presented in Section 5 of this paper is a refinement of this generic agent model. The generic model for co-operative agents can be used for a wide variety of more specific agent models through refinement and composition. *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.

4 A Generic Agent Model

For the design of a generic agent model the following main aspects are considered: process composition, knowledge composition, and relations between knowledge and process composition, as discussed in Section 2. In this section a compositional generic agent model (GAM), supporting the weak agency notion (cf. (Wooldridge and Jennings, 1995a)) is briefly presented. At the highest abstraction level within an agent, a number of processes can be distinguished that support interaction with the other agents (see Figure 1).

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 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: 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. Figure 1 depicts how the agent is composed of its components.

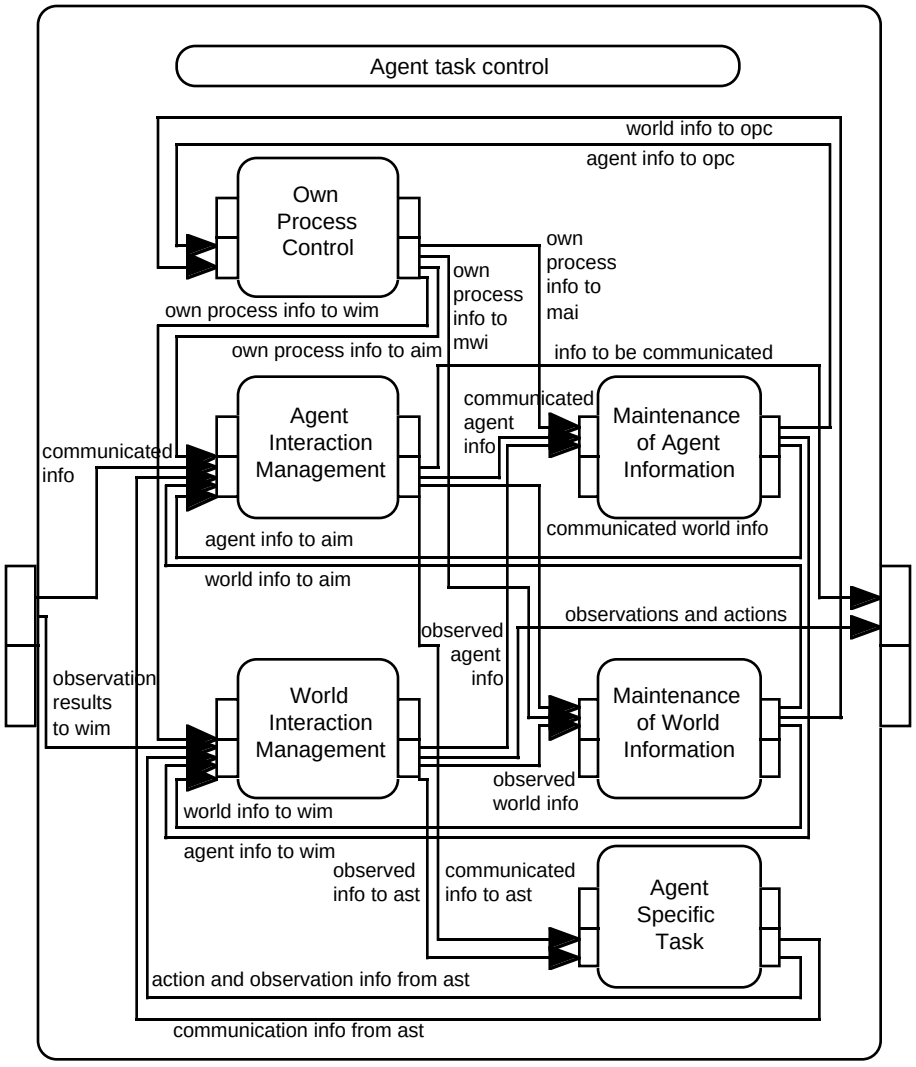

Figure 1 Composition at the highest level within the broker agent

5 A Compositional Generic Model of Co-operation

To successfully develop a support system for co-operation in a complex, dynamic and not always predictable environment, a well-defined and transparent model of co-operation is required: a model that is robust and flexible enough to cope with unexpected events. To this aim in (Jennings, 1995) a model for co-operative problem solving using joint intentions was introduced, based on experience in industrial applications. The model describes both the phase of organising (creating) a joint project and the phase of performing (executing) the joint project, in which managing unexpected difficulties is included. In (Jennings, 1995) details of this model are described for an implementation in one specific environment with limited focus on possible applicability in other environments. In this section, the generic co-operative agent model GCAM is described in terms of specifications at the conceptual level as a refinement of the generic agent model GAM discussed in Section 4.

In Jennings' model of co-operation, agents are capable of organising projects. An agent decides to organise a project to reach a given goal. With respect to the current state of the world, an agent determines a set of activities to reach this goal and the temporal dependencies between the activities. The organising agent then identifies other agents capable of performing the activities. In interaction with these agents, the organising agent determines which agents are willing and able to participate in the project. On the basis of this information, the activities to be performed, the order in which the activities are to be performed and the deadline, the organising agent tries to put together a project team and a project schedule (called a *recipe*). The creation of this recipe is an iterative process requiring interaction with the other agents on their own schedules (related to other projects). When completed, the recipe is communicated to all participants, and the project commences.

Once committed, each participating agent (including the organiser) receives the final recipe, and is committed to the relevant time interval in the recipe. Each agent has the same obligation towards the project: each member monitors the progress of the project and is equally responsible for its success. If a team member discovers a problem that endangers the project, he/she informs all relevant participants. One of the agents (e.g., the project manager) can then take the initiative to modify the project plan, to create a new project for the same goal or to inform all relevant participants that the goal is unattainable or that it is no longer necessary to reach the goal.

5.1 Refinement of the Generic Agent Model

The generic agent model GAM presented in Section 4 can be extended to include the component co-operation management . This component exchanges information with the components own process control, agent interaction management, maintenance of agent information, world interaction management, and maintenance of world information. The information links required for this purpose are depicted in Table 1.

information link	*source*	*destination*
self info for co-operation	own process control	co-operation management
communicated info to CM	agent interaction management	co-operation management
observed world info to CM	world interaction management	co-operation management
world info to CM	maintenance of world information	co-operation management
other agents info for co-operation	maintenace of agent information	co-operation management
co-operation info to opc	co-operation management	own process control
self info request	co-operation management	own process control
info to be communicated	co-operation management	agent interaction management
required observations	co-operation management	world interaction management
required world info	co-operation management	maintenance of world information
other agents info request	co-operation management	maintenace of agent information

Table 1 Information links needed for the component co-operation management

The extension of the generic model to include the component co-operation management is shown below in Figure 2.

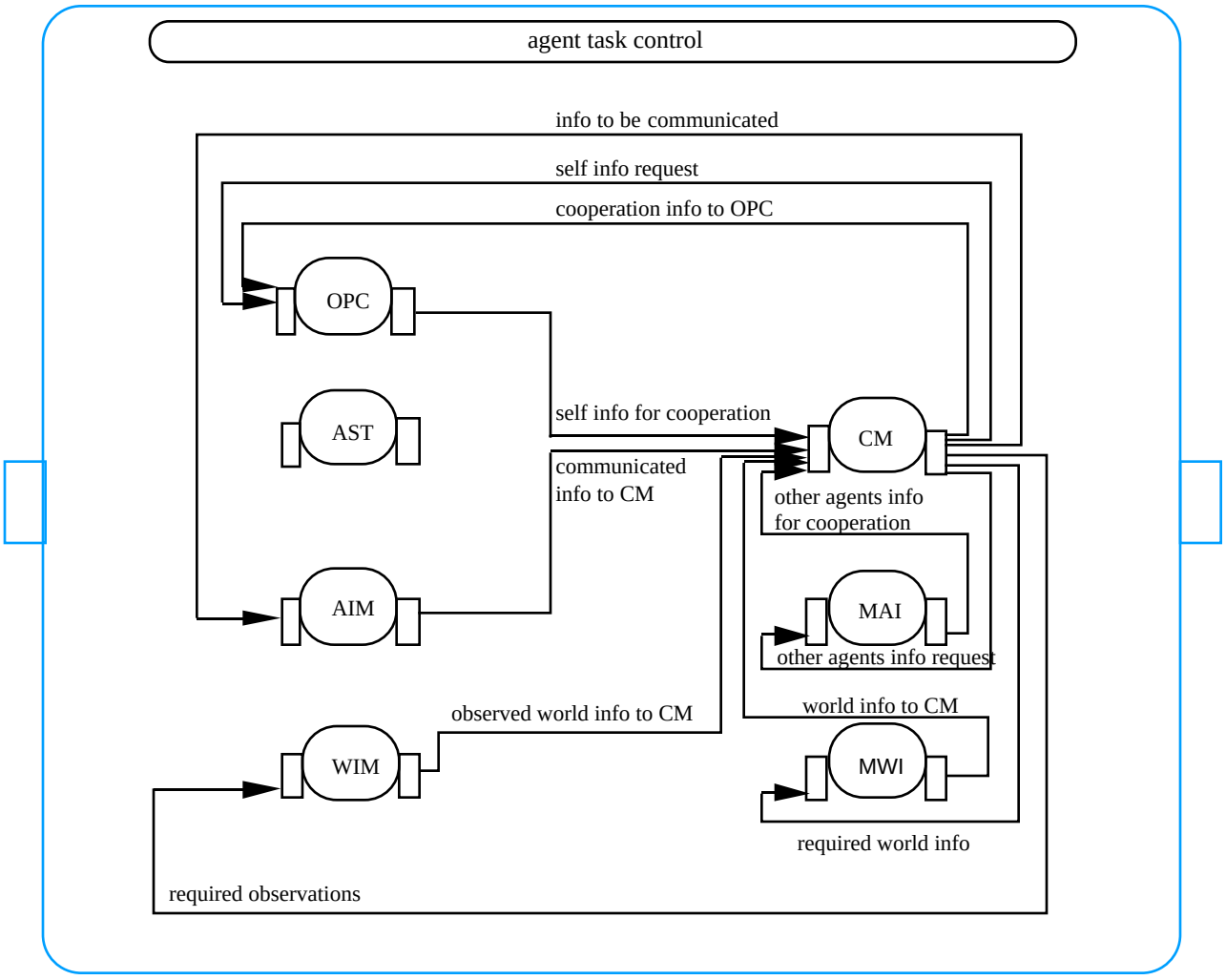

Figure 2 Component and interaction structure at the top level of the generic agent

Each of the processes depicted in Figure 2 can be described in more detail (see Figure 3). Refinements of the components responsible for the processes own process control, agent interaction management, maintenance of agent information, co-operation management, world interaction management, and maintenance of world information are presented below in the following sections.

5.2 Own Process Control (OPC)

The agent component OPC is a composed component responsible for determining, planning, scheduling and monitoring an agent's activities. Furthermore, it is responsible for maintaining all relevant information on an agent's activities and its status. These sub-processes are performed by OPC's sub-components: determine goals and commitments (DPC), assess information (AI), evaluate own processes (EOP), plan and schedule (PS) and maintain own activities (MOA).

Determine Goals and Commitments (DGC)

DGC determines goals of an agent on the basis of its motivations, priorities, and deadlines and its role within a system. Selection of a goal depends on motivation: motivation is a necessary precondition for goal selection. Selection of a goal implies individual commitment to the goal.

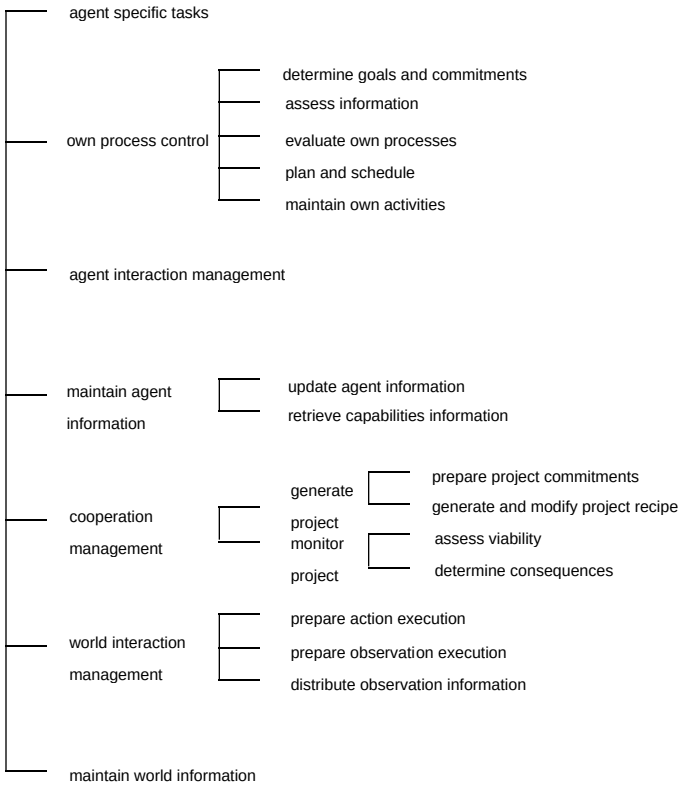

Figure 3 Process abstraction levels for a co-operative agent

Assess Information (AI)

The AI component maintains all relevant information on an agent's activities: which information is based on its own observations; which on own assumptions; which has been received by communication, and from which source; and which information has been derived, and is based on which other information.

Evaluate Own Processes (EOP)

This component is responsible for the evaluation of the progress of an agent's activities with respect to its individual commitments. It involves monitoring relevant activities (its own and other agents) and analysing monitoring information. For instance, the progress of an activity can be compared to the scheduled duration and finishing time of the activity. If a sub-activity is taking more time than scheduled, the schedule may have to be adapted. Furthermore, it is possible that due to the delay, some goal cannot be reached before the indicated deadline. During analysis EOP may, for example, deduce that the motivation for a goal has disappeared: this goal is then removed.

Plan and Schedule (PS)
The component PS is responsible for planning and scheduling an agent's activities, upon request for participation in a project by another agent or on the basis of information received from EOP or DGC. The component PS uses domain-knowledge to find a set A of activities, called a plan, that meets the following criteria: (1) execution of the plan will lead to the fulfilment of a goal G, (2) the plan can be scheduled without contradicting prior commitments, (3) the plan matches the priority and the deadline of the goal. If no such plan and schedule can be found, not even by requesting the help of other agents, this must be communicated to EOP. Another goal can then be selected by DGC. If an agent cannot reach the goal G itself while respecting the priority and deadline, but the goal may possibly be reached with the help of others, then all relevant information is transferred to CM, which will try to create a project to reach the goal.

Maintain Own Activities (MOA)
This component stores an agent's own schedule, which actions an agent can perform (domain dependent) and which commitments an agent has made to which goals. Commitments can be made with respect to other agents and projects.

5.3 Agent Interaction Management (AIM)

The component AIM manages communication with other agents, in particular with team members of a project. It receives information from CM which it transfers to (possible) participants in a project. Furthermore, it receives (communicated) information from other agents which it transfers to other relevant components. For example, upon receiving a new recipe, AIM determines the subset of recipe-elements that concern its own activities. This subset is passed on as "own process" information to OPC. The whole recipe is transferred to CM.

5.4 Maintenance of Agent Information (MAI)

Upon request MAI provides other agents or other sub-components with names of agents capable of performing certain specified activities. Two sub-components are responsible for the performance of this process: update agent information (UAI) and retrieve capabilities information (RCI).

Update Agent Information (UAI)
UAI maintains models of other agents known to an agent itself. A model of another agent consists of statements that express how co-operative the other agents is, its availability (that it normally has no time to help other agents, or normally is able to help), punctuality with respect to deadlines, et cetera. UAI stores and updates its knowledge by maintaining which activities other agents are capable of performing, the projects in which they participate and the goals to which they are committed.

Retrieve Capabilities Information (RCI)
RCI provides, for each activity, the names of all agents known to be capable of performing an activity and the available meta-information concerning the exhaustiveness of the information.

5.5 Co-operation Management (CM)

The component CM is a composed component responsible for all processes concerning projects, project commitments and co-operation. The interaction between the components of CM and CM's environment is organized through the links depicted in Figure 2.

The component cooperation management needs the following types of information:

- goal, deadline, and necessary activities: to create a new project
- capabilities of other agents: to find participants for a project
- commitments of other agents: to build a joint recipe
- observation information: to monitor existing projects
- communicated project information: to monitor existing projects

The component cooperation management provides the following types of information:

- recipe elements relevant for possible participants
- joint recipe relevant for all participants
- monitoring information relevant for all participants

If a new project is to be created the relevant information enters the component GP through the link (see Figure 4) required project (which transfers the information also transferred by link self info for co-operation in Figure 2). The information GP needs on other agents enters GP through link info on other agents (see also links other agents info for co-operation and communicated info to CM in Figure 2) and this information is requested through link required info on other agents (see also links info to be communicated and self info request in Figure 2). The commitments made in the created project and the information on the joint project are transferred through link commitments to output (see also links info to be communicated and co-operation info to OPC in Figure 2). The generated project is transferred to MP to be monitored through link own generated project.

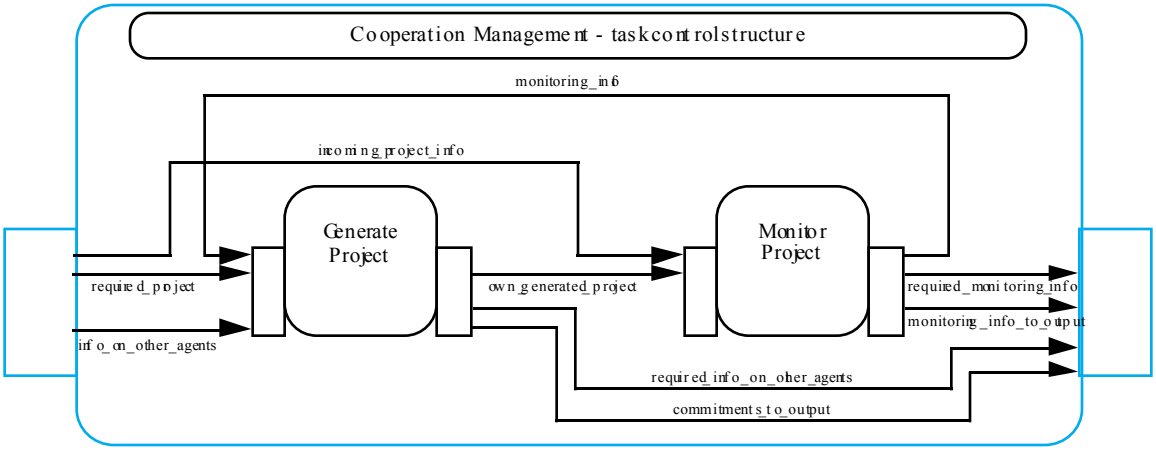

Figure 4 Composition of the component co-operation management

If GP is activated because a monitored project is to be reconsidered, the relevant information is transferred through link monitoring info. The links incoming project info (see also link communicated info to CM in Figure 2) and own generated project transfer the necessary information on projects that MP has to monitor. For this purpose MP needs information which is requested through the link required monitoring info (see also links required observations and required world info in Figure 2) and enters MP through link incoming project info (see also links observed world info to CM and world info to CM in Figure 2). If necessary the resulting monitoring information is transferred through the links monitoring info and monitoring info to output (see also links info to be communicated and co-operation info to OPC in Figure 2).

Generate Project (GP)
Given the goal G, motivation M, priority p, deadline T, all possible sets A of activities with which goal G can be reached, and an agent's own capabilities, the component GP has two main processes: to prepare project commitments, and to generate and modify project recipes.

Links are defined to regulate the interaction between GP's components and its environment, see Figure 5. Recipes enter PPC through the links recipe to be repaired (see also link monitoring info in Figure 4) and recipe to be prepared see also link required project in Figure 4). To prepare the commitments PPC requests information on other agents. These requests are transferred through the link needed info on other agents info on other agents (see also link required info on other agents in Figure 4), the answers enter through link info on other agents to PPC (and link info on other agents in Figure 4). The information produced by PPC is transferred to GMR through link prepared project. While making the recipe GMR interacts with other agents through the links info for agents (and link commitments to output in Figure 4) and info on participants (see also link info on other agents in Figure 4). The final recipe is transferred to the participants through link info for agents..

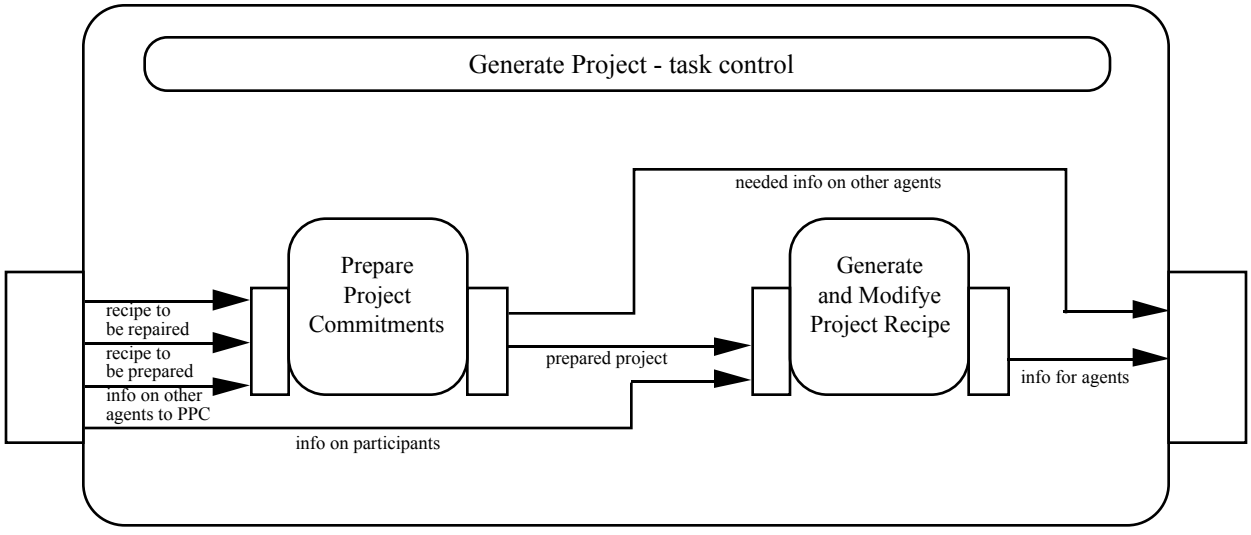

Figure 5 Composition of the component generate project

The component **Prepare Project Commitments (PPC)** determines a preferred set A of activities with which goal G can be reached. Using domain-knowledge the dependencies between the activities in A are determined, for example by Critical Path Methods. This (partial) ordering of the activities in

A (see Figure 6) is important in the development of a recipe R for goal G. Given this dependency-graph PPC determines which agents can and are willing to perform activities to help reach goal G. The dependency-graph for A, the information (G, M, p, T), the relevant capabilities of the willing participants (including the agent's own relevant capabilities) and the corresponding names of the agents, are transferred to GMR.

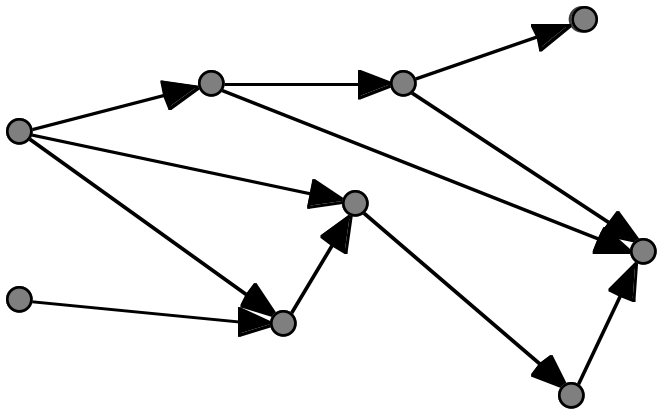

Figure 6 Dependency graph for a set of activities

Using PPC's information, the component **Generate and Modify project Recipe (GMR)** designs a recipe R that conforms to the interdependencies between the activities in A (thus leading to G's fulfilment). The recipe R is interactively designed by iteratively generating and communicating proposed recipe elements to agents interested in participation. For one activity more than one agent may be approached; afterwards a choice can be made among the agents that responded positively. Criteria for such a choice can be, for example, the starting time or whether the agent is already involved in other activities.

A recipe element consists of an activity of A, a willing participant capable of performing that activity, a priority p and a deadline T for that activity. At any stage a recipe element is only selected if all activities on which it depends have already been scheduled. Therefore the generic model is instantiated with a 'wave model' for the generation and communication of recipe-element proposals to other agents (see Figure 7). With a wave the processing of a (sub)set of activities that are still to be scheduled is meant: generation of recipe elements scheduling these activities and simultaneously communicating these recipe elements to the agents involved. Such a wave is only finished when all proposed recipe elements have been confirmed by the agents executing them (possibly after modification). A wave only contains activities for which all activities they depend on already were scheduled by recipe elements generated and confirmed (by the agents involved) in one of the previous waves. To start with, the first wave contains all tasks that can be executed immediately, without depending on previous tasks.

For example, in a third wave, only tasks occur that temporally depend on at least one task (treated in the second wave) that depends on another task (that is treated in the first wave) and do not depend on

longer chains of tasks. In order to minimize the number of participants it might be necessary to send several recipe elements of one wave to the same agent.

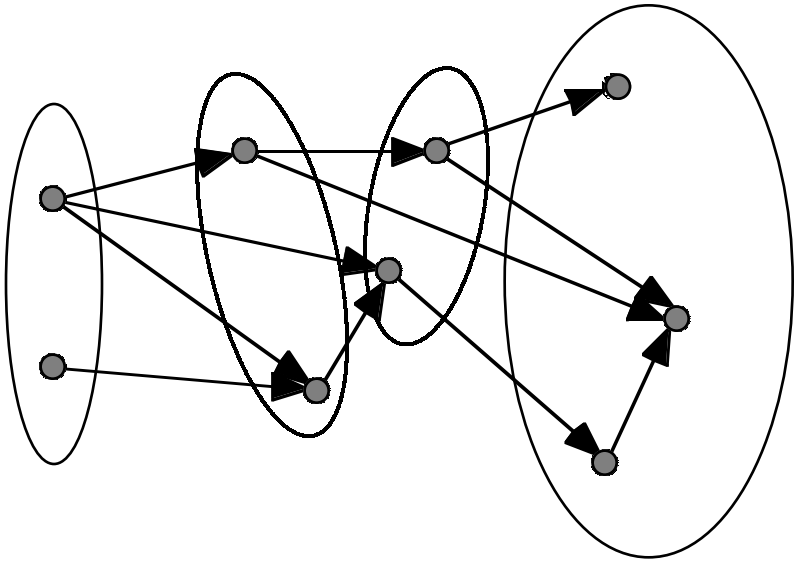

Figure 7 Four waves in a dependency graph

The willing participants accept, adapt or reject the proposed recipe elements. Acceptance or adaptation of a recipe element implies that an agent commits to this element. GMR adjusts the partial recipe depending on the replies from participating agents. A recipe may be found that is acceptable to all participants and that will reach goal G before its deadline. The duration of the recipe and team building is estimated on the basis of the number of activities involved, the number of willing participants and the time needed for communicating requests and responses. The time required for communication (depending on the situation) is assumed to be known. In addition, communication is assumed to be error free. The resulting recipe is communicated to all participants.

Monitor Project (MP)

The component MP is responsible for the detection of the need for alterations to the project or the need to stop the project. MP monitors the progress of the project. In order to perform its task MP has two sub-components: assess viability and determine consequences.

The components and the links for interaction within MP are depicted in Figure 8. Information on the project to be monitored and the necessary monitoring information enters AV through link project info. The request for monitoring information and the resulting assessment information is transferred to CM's output interface through links assessment info to output and monitoring info to output (see Figure 4). Through link assessment info to DC this information is also transferred to DC, which uses it to determine changes to the joint project. Information on changes is transferred through the links info on project changes and monitoring info to output (see Figure 4) to CM's output interface. From CM's output interface the information is transferred to AIM through the link info to be communicated (see Figure 2).

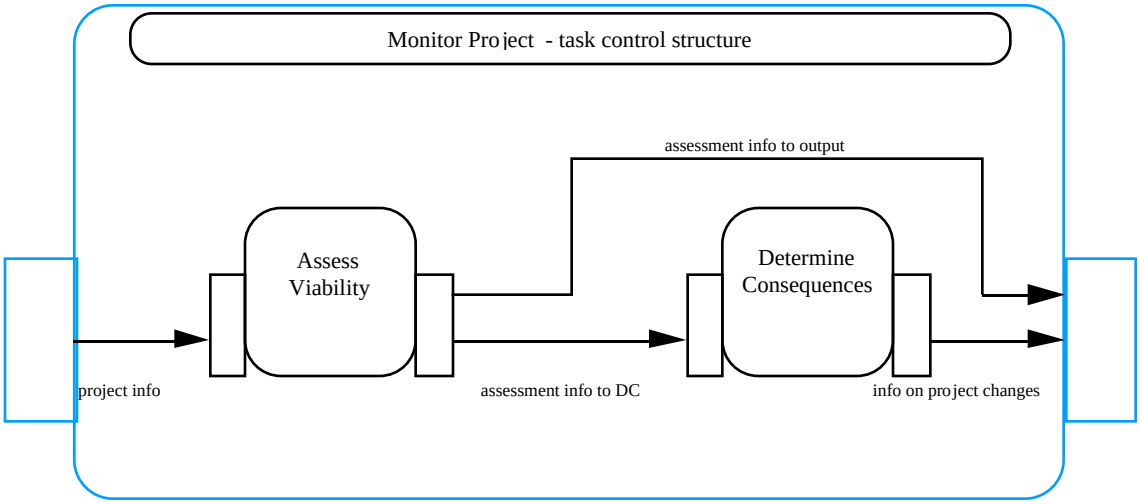

Figure 8 Components and Communication within the monitor project component

Assess Viability (AV) monitors the viability and validity of the recipe. To check the validity of the project recipe, AV uses the same considerations as the sub-component evaluate own processes of the component own process control. To monitor the process it uses information received from OPC, WIM and MWI (its other components). It can also actively formulate requests for observational information from WIM, MWI or information of other agents via MAI and AIM.

Determine Consequences (DC) interprets AV's monitoring results. The component DC issues requests to find new recipes or to adapt existing recipes, to the component project generation of CM and issues corresponding messages to the participants. DC also determines when a goal G should be withdrawn (for example, because the goal is unattainable, the goal has been reached, or because the motivation for the goal no longer exists) and prepares and issues a message to that effect to each participant.

5.6 Maintenance of World Information (MWI)

MWI contains the current world state as known to the agent. MWI stores all information obtained by monitoring the world (also the material aspects of all agents including the agent itself).

5.7 World Interaction Management (WIM)

The component WIM is responsible for the execution of observations and actions. An important sub-task of this component is the observation of the effects on the world of the processes executed by the other agents and by the agent itself.

Prepare Action Execution (PAE)

This component prepares the execution of actions determined by AST by communicating to the world which actions should be taken.

Prepare Observation Execution (POE)

WIM prepares specific observations. The observational information is transferred via DOI to those sub-components that analyse this information.

Distribute Observation Information (DOI)
Upon request, observational information is transferred from DOI to other components (including MWI). DOI can also take the initiative to inform other components (including MWI) of (domain-dependent) important changes in the world.

6 Application to Project Co-ordination in Distributed Design

Co-ordination of complex engineering projects often entails co-ordination of individuals but also co-ordination of groups, often somehow related to departments and/or project groups. These entities, whether departments, project groups or individuals, may be modelled as agents: each with their own responsibilities and autonomy. In this section the generic model of a co-operative agent GCAM presented above in Section 5 is used as a building block to model the co-operative design of aircraft interior.

6.1 Design and co-ordination

Co-ordination of a design project entails co-ordination of all phases of design such as initial design, feasibility studies, design definition, and validation. In essence, design entails co-ordination of (1) modification of requirements, (2) modification of a design object description, and (3) the design strategy. In this section a simplified example of the co-ordination of a routine design project is addressed: the design of aircraft interior. Agents refer to individuals (or groups of individuals) with a specific task in the project. Requirements are specified at the level of detail required for verification, including specification of the verification procedures.

A Design Project Manager (DPM) is assigned the task of co-ordinating all design activities for the interior of an aircraft, for example the design of the toilet unit, luggage bins, wardrobe, gallies, side panels, and the floors, often in close collaboration with the financial department. The responsibility for the design of each of the individual units is delegated to a unit manager (UM) who, in turn co-ordinates the design of more specific aspects of that unit to specific engineers. The design project manager interacts with a number of specialists: financial specialists, styling specialists, logistic specialists, tooling specialists, et cetera, to co-ordinate the project as a whole. At this level, co-ordination is clearly hierarchically organised. Although relatively well-defined, the frequency and content of interaction and co-operation is not as easily specified.

Detailed design at the level of one of the units, however, will be used to illustrate the approach. The unit manager considered receives requirements for the aircraft as a whole, together with technical specifications for a specific unit, in our example the toilet unit. He/she is responsible for the integrated design of the unit, but also for interaction with other unit managers and the project manager, in particular with respect to control and configuration management. The unit manager co-ordinates detailed design of the unit: he/she examines (partial) designs produced by design engineers, electrical engineers, and systems engineers, identifies inconsistencies, and interacts with the designers/engineers to find solutions. The unit manager is responsible for the provision of

information within his/her unit group; for example, the most recent version of the integrated design, relevant guidelines and decisions taken within the project management group.

The engineers, in turn, co-ordinate their own design processes. Design engineers, for example, interact not only with electrical engineers and systems engineers, but also with other experts, such as product specialists, purchasing department, tooling specialists and styling specialists, when necessary. When and how other specialists are involved, is left up to the discretion of the individual engineers: they themselves define virtual organisations

For the sake of simplicity, the above example of design will be modelled for one unit, with one engineer of each signature. These engineers will most often represent a group of engineers responsible for the tasks assigned in this model. The patterns of communication between engineers are, however, comparable.

Interaction between agents is modelled by information links, controlled by the agent from which the links originate. The double-arrowed lines in Figures 8 and 9 depict the information links that specify the exchange of these types of information between agents.

To describe the interaction between agents two scenarios will be sketched. First the creation of a project is sketched from the perspective of a design project manager in Section 6.2. A system trace is presented for the creation process, sketching the activation of agents, components of agents and the information communicated through time. As an example of project execution, the design of a unit is described from the perspective of a design engineer in Section 6.3, part of which is presented in a system trace.

6.2 Communication during project creation

In this section a scenario for project creation is described. An agent decides to organise a project to reach a given goal: in the example the goal of the design project organised by the Design Project Manager (DPM) is to design the interior of the aircraft, in particular the design of the toilet unit. The DPM, for instance, considers dependencies between activities such as co-ordination, design of the construction, design of electrical systems, design of other systems, styling, and tooling. The organising agent DPM then identifies other agents capable of performing the activities (e.g., a Unit Manager UM, a Design Engineer DE, an Electrical Engineer EE, a Systems Engineer SE, a Styling Specialist SS, a Tooling Specialist TS, et cetera). Each of these agents is modelled as a specific instantiation of the generic co-operative agent model GCAM described in Section 5. The communication patterns are depicted in Figure 9. A trace of a process of project formation is shown in Table 2.

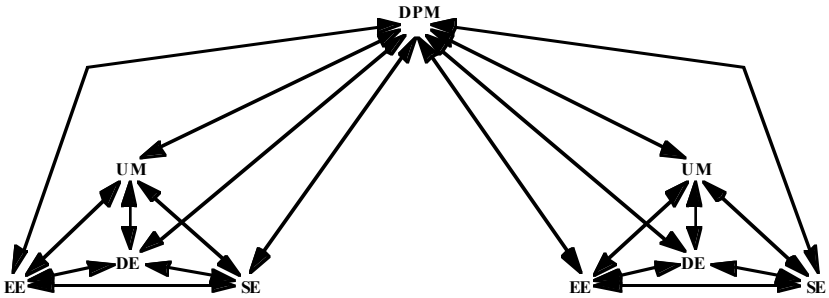

Figure 9 Communication during project creation

Project creation scenario

Table 2 depicts how, in this scenario (for each agent) each of the instantiated components of the generic co-operative agent model GCAM is involved in the process of project creation. The component OPC of the design project manager (DPM) has the goal to design an aircraft (1). To reach this goal, DPM needs help. Thus, his component GP (part of CM) is activated to generate the project. Immediately, PPC (part of GP) is activated to determine which activities are needed to reach the goal and which possible team members for the project (2) can be found. For this purpose DPM requests possible participation from design engineers, electrical engineers, systems engineers and unit managers. The requests are initiated by DPM's AIM component (3). Each of these agents receives the request through its own AIM component (4), and each considers the request for possible participation in its own component OPC (5). Each agent's AIM component returns an answer to the request (6). DPM receives the agents' responses (through the AIM component) (7). The replies are forwarded to the PPC component, which does the administration of project activities and commitments. This information is transferred to GMR, the component responsible for the creation of the final recipe. Both PPC and GMR interact frequently with possible participants (iterating steps 3 through 9).

The OPCs of the willing participants check to see if the activities assigned to them fit in their own schedules (12). Information on the success or failure of their scheduling is transferred by their AIM component (13) to the AIM component of DPM (14), which forwards it to GMR (15). By iterating steps 10 through 15, GMR creates a final recipe. The resulting recipe includes the global goal (i.e., aircraft to be designed given global requirements and specifications) and recipe elements. A recipe element related to the design of a unit includes the following information:

- general requirements and specifications
- the specific requirements and specifications for the unit to be designed (based on the initial design of the whole aircraft),
- one unit manager (UM),
- one design engineer (DE),
- one electrical engineer (EE), and
- one systems engineer (SE).

The resulting recipe is communicated to each of the unit managers by AIM (16). The CM component of DPM makes sure that the resulting recipe will be monitored by its sub-component MP (16).

After the unit groups have been formed the unit managers schedule the design process of their unit, following a similar pattern. For example, the unit manager responsible for the design of the toilet unit (i.e., toilet basin, counter top, sink and cabinet combination, et cetera) decides that the design engineer involved should make an initial design for the electrical engineer and the systems engineer. To ensure that the electrical engineer and the systems engineer can start as quickly as possible, the unit manager initially gives the toilet basin the highest priority compared to the top counter, the sink and the cabinet combination.

time point	agent	agent component	sub component	sub-sub-component
1.	DPM	OPC		
2.	DPM	CM	GP	PPC
3.	DPM	AIM		
4.	other	AIM		
5.	other	OPC		
6.	other	AIM		
7.	DPM	AIM		
8.	DPM	CM	GP	PPC
9.	DPM	CM	GP	GMR
10.	DPM	AIM		
11.	other	AIM		
12.	other	OPC		
13.	other	AIM		
14.	DPM	AIM		
15.	DPM	CM	GP	GMR
16.	DPM	AIM CM	MP	

Table 2 System trace: project creation

6.3 Communication during project execution

The unit manager receives requirements and specifications from the design project manager. This information is forwarded to the design engineer, the electrical engineer and the systems engineer. The communication patterns between team members and other experts are depicted in Figure 10.

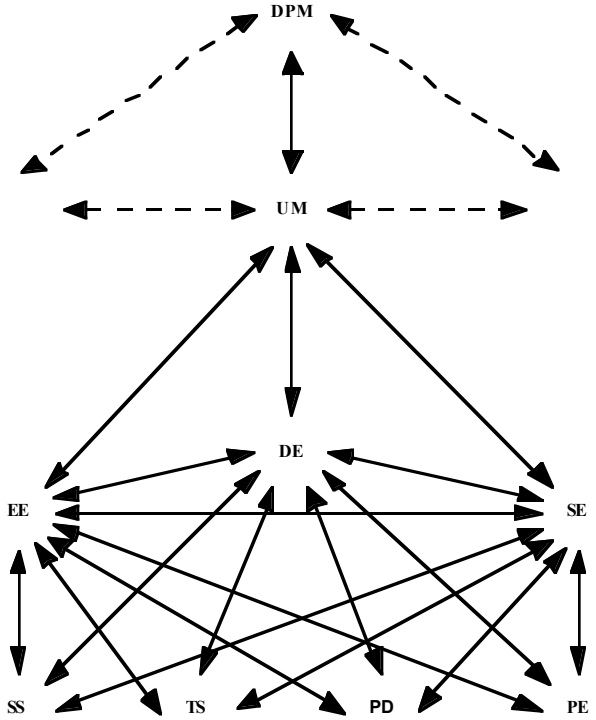

Figure 10 Communication patterns between design agents

The communication between team members includes both object and meta-level information. Object level information includes information on, e.g., the design object description, the initial cable routing, switch dimensions and positions, the initial design, product information. Meta-level information includes requests for information, evaluation information on the design object description, conflicts between routing of cables and the initial design, and information on the design process (e.g., planning and scheduling). Requirements such as (fire) safety requirements, are not specified explicitly but are assumed to be known to the managers and engineers.

In addition, the unit manager provides each engineer with relevant guidelines and planning information (e.g., deadlines and priorities). Guidelines, such as, 'Use aluminum instead of stainless steel if at all possible', may evolve during the design process at unit management level. Such guidelines are forwarded immediately to the engineers - often causing modifications to existing (partial) designs.

The design engineer first analyses the information on the position of the unit. The initial contours of the unit and planes within the unit are identified. This initial sketch is given to the other engineers. This sketch roughly indicates where electrical, air-conditioning and water systems should be positioned. The electrical engineer and the systems engineer start working on a first draft of their systems, roughly following the priorities provided by the unit manager.

Expectations of the time involved in manufacturing guide the design strategy and thus scheduling of sub-tasks. The unit manager had initially given the toilet basin highest priority. The design engineer,

however, expects the counter-top, sink and cabinet combination to be more complex. She informs the unit manager of her intention to work on the counter-top, sink and cabinet combination first, and the reasons for this decision. One of the reasons is the fact that the requirements differ considerably from previous designs, implying that extensive interaction with suppliers, product specialists and tooling specialists is required. The unit manager agrees with the argumentation and informs the other engineers of the change in priority.

The design engineer designs and positions the cabinets, the sink and the counter top. Different options are explored: properties of material, appearance, functionality, et cetera, are analysed in interaction with specialists. An example of the types of interaction involved is illustrated for the requirement that the overflow in the sink should not be immediately visible. This requirement mandates, in our example,

- interaction with the purchasing department to determine whether sinks exist for which the overflow is closer to the user than to the wall (so that it cannot be seen),
- interaction with the product specialist to determine whether and how a sink can be made to fulfil this requirement (if possible using standard components),
- interaction with the tooling specialist to determine whether specific tooling is required in the production process.

The design engineer discusses the different options with the systems engineer (position of the drain is of importance), and the electrical engineer (the position of the sensor to activate the water flow is of importance), and proposes a solution. If the unit manager agrees, the solution is accepted.

A similar pattern of communication is required for the counter top and the cabinets, in which case the styling expert is consulted for input on the precise shape of the combination. The process sketched above is described below in more detail, with a system trace as shown in Table 3.

Project execution: design scenario

Table 3 shows how, in this scenario (for each agent) each of the instantiated components of the generic co-operative agent model GCAM is involved in the process of project execution. During the design process for the counter top and the cabinets the component AST of the design engineer makes a partial initial design (1) which is communicated by its AIM component (2) to the unit manager (UM), the electrical engineer (EE) and the systems engineer (SE).

The AIM components of the electrical and systems engineer (3) forward the initial design to their own AST components (4). The AIM components (5) of these agents then send the initial designs of their systems to the unit manager and the design engineer. Their AIM components (6) transfer these designs to the respective AST components (7). The electrical engineer sends an initial design of the electrical cable routing, the system design sends an initial design of all other systems.

The design engineer's AST component positions the electrical cable routing in her current design and discovers a problem: the cable routing directly crosses mounting points of the cabinet (7). Using her AIM component (8), the design engineer informs the electrical engineer and the unit manager of this problem. The information arrives in their AIM components (9) and is transferred on to their AST

components (10). The AST component of the electrical engineer solves the problem by re-routing the cable.

time point	agent	agent component
1.	DE	AST
2.	DE	AIM
3.	EE SE	AIM AIM
4.	EE SE	AST AST
5.	EE SE	AIM AIM
6.	DE UM	AIM AIM
7.	DE UM	AST AST
8.	DE	AIM
9.	UM EE	AIM AIM
10.	EE UM	AST AST
11.	EE	AIM
12.	DE UM	AIM AIM
13.	DE	AST
14.	DE	AIM
15.	EE SE UM	AIM AIM AIM
16.	EE SE	AST AST
	SE	AIM
18.	EE	AIM
19.	DE	AIM
20.	DE	AST

Table 3 System trace: project execution

The solution is communicated by the AIM component (11) to the design engineer and the unit manager; they receive the solution in their AIM component (12). With the solution, the AST component of DE can resume its work (13).

To finalise the design of the counter top, sink and cabinet combination the AST component of the design engineer needs more detailed information on switches, light points, sensors, et cetera, from the electrical engineer, thus a request is communicated by the design engineer's AIM component. The design engineer also needs more detailed information on pipes and drains (size, mounting specifications, screws, et cetera) from the systems engineer, again a request is communicated by the AIM component. In both cases the unit manager is informed as well (14). The AIM components of the EE, SE and UM receive the request (15).

The AST component of EE has to reschedule some of its sub-processes to provide this information as soon as possible (16). This is important for the acquisition of the necessary materials and tooling. After rescheduling, the information is communicated by EE's AIM to the design engineer (18).

In the mean time, the AST component of SE (16) is able to provide the information immediately, SE's AIM component (17) sends the information to DE. The design engineer receives the information from EE and SE, and via AIM (19) and AST (20) proceeds to design the toilet basin, requiring interaction with both the electrical engineer and the systems engineer.

7 Application to Distributed Scheduling for Call Center Support

In this section another application of the generic co-operation model is described: a distributed multi-agent scheduling system to support a Call Center. An increasing number of Call Centres are now providing 24 hour service to their customers. One of the areas of industry in which this phenomenon has become manifest is finance. To increase service level, the Rabobank, one of the largest banks in the Netherlands, for example, now provides its bank relations 24 hour a day telephone service.

This section describes a prototype system, developed in close co-operation with the Rabobank, that automatically schedules procedures on the basis of client requests forwarded by a call centre. This system, a multi-agent system, uses a generic model of co-operation based on joint intentions to model the two types of automated agents involved: the work manager and personal assistants. Interaction with the other agents involved: the client, the call centre employee and all other employees, is also explicitly modelled.

More detail on the Rabobank itself with respect to this application is provided in Section 7.1. Section 7.2 describes the multi-agent approach to this problem. Section 7.3 describes the work manager. The conceptual model of the personal assistant is described in Section 7.4. The role of the employee is discussed in Section 7.5.

7.1 Problem Description

The Rabobank is one of the largest banks in the Netherlands with a co-operative structure with autonomous branch offices, each responsible for specific geographical areas. These local organisations (local banks) all service both the consumer market and industry.

7.1.1 The Problem

The Rabobank's aim is to achieve a stronger position in the financial market by using its resources more efficiently and effectively, and binding potential clients directly to the bank. As, in today's society, clients and potential clients are more inclined to switch between service providers than in the past, depending on the service level provided and the cost involved, the task of client advisors has become more proactive: to focus both on finding new clients and satisfying existing client needs.

7.1.2 The Organisational Solution

Part of the solution the Rabobank has adopted is 24 hour a day availability together with new procedures aimed at binding clients directly to the bank. Clients' requests and questions can be divided into three categories:

- simple questions that can be answered directly by teller personnel, e.g. a question about the current advertised interest rates for the different types of savings accounts the bank offers.
- simple questions and requests that can be handled right away without any further contact with the client but require further processing, e.g. a request for new cheques.

- complex questions and requests which need the attention of a client advisor, e.g. an inquiry about a mortgage.

The assumption behind 24 hour a day service is that clients will be less likely to shop around and take their business elsewhere, if their requests are taken seriously. Operators of the call centre have been trained to deal with the relatively simple client requests. These simple requests (the first two types) amount to about 70 percent of the calls. This approach reduces the number of simple requests client advisors need to address, leaving more time for other activities such as, for example, more complex client requests or client acquisition. Operators schedule appointments for clients with complex requests with a qualified client advisor. It is important that the appointment takes place as soon as possible.

As local banks are autonomous, the agendas of client advisors of the local banks are not directly available to the call centre. The overall procedure employed is as follows:

1. A client or a potential client calls the local bank and the call is redirected to the call centre.
2. If the request of the client is relatively simple, the operator deals with it right away.
3. If the request of the client is more complex and needs to be serviced by a client advisor, the computer system of the operator contacts a computer system at the local bank with a request for service.
4. The computer system at the local bank determines if this request can be serviced and suggests a number of possible appointments with the client. The client can choose one of these appointments. The computer system at the local bank can do this by selecting an appropriate procedure to service the request and schedule the activities of that procedure in the agendas of the employees of the bank

Within the Rabobank procedures have been defined for most types of client requests. These procedures are all specified in a process definition language, defining the workflow within the organisation. In this section a simplified version of the procedure for dealing with requests related to financing consumer expenditure is used to illustrate the types of activities (and the relations between activities) to be scheduled. The procedure for dealing with requests related to financing consumer expenditure, as specified by the Rabobank, consists of 22 activities of which 8 are completely automated (including, for example, information retrieval, calculations and provision of standard contract conditions). In short three types of loans can be provided for consumer expenditure: personal loans, revolving credit, and student loans. This paper focuses on three (groups of) activities within this procedure:

- Client advise (requires activities such as the acquisition of information on credit rating and financial status (including current income and expenditure), analysis of available information, decision with respect to maximum loan, overview of possible options)
- Written agreement (requires additional information from the client and possibly other sources, possibly requiring approval by authorised persons depending on factors such as amount and risk involved)
- Administrative transaction.

Employees are fully responsible for their own agendas. They can refuse or change appointments in their agendas. Changes to an employee's agenda should take the profile and wishes of an employee into account, e.g., the employee's preferences for specific types of activities, the employee's capabilities and authorisation with respect to specific activities, the employee's preferences for allocation of specific activities to specific times of day, or the employee's availability (e.g., due to holidays or illness).

7.2 A Multi-Agent Perspective

The problem description clearly defines the problem as a distributed problem: one call centre services several local banks as depicted below in Figure 11.

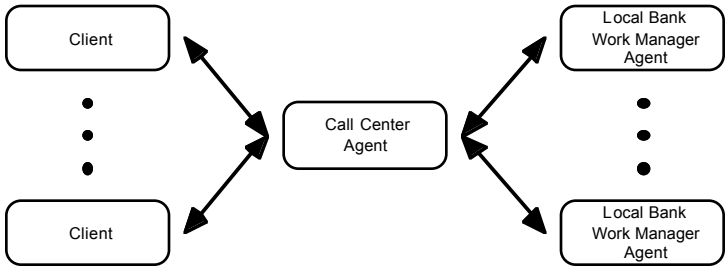

Figure 11 One call centre for several local banks

As described above in Section 7.1, the clients, the local banks and the call-centre are autonomous, distributed entities: entities responsible for their own internal processes in interaction with (and in response to) other entities. As described the entities involved fulfil the characteristics of weak agency proposed by (Wooldridge and Jennings, 1995): autonomy (all agents are in full control of their own processes), social ability (all agents are able to communicate and co-operate with other agents), pro-activeness (all agents are able to initiate processes independently and take the initiative to initiate new processes when necessary) and reactiveness (all agents are able to respond to new incoming information. In fact, to model the activities involved, a more detailed analysis of the actors involved is required: in this example the local banks not only have a Work Manager but also Employees. A multi-agent system has been designed to provide this functionality.

To perform the tasks distinguished above in Section 7.1 appropriately, the system as a whole, needs to satisfy the following requirements:

1. The system needs to be able to cope with changes made by employees. An employee can change his/her agenda, s/he can:
   a) refuse to perform a specific activity.
   b) refuse to perform it within a specific period of time.
   c) reschedule his/her agenda.
   d) delay some of the activities.

2. The system needs to know :
   a) the capabilities of the employee.
   b) the preferences of the employee (e.g., time periods during which the employee does or does not want to perform specific kinds of activities).
   c) the availability of the employee (e.g., holidays).

3. The system may only reschedule the agenda of an employee in a way that respects the profile of the employee.

4. The system needs to be able to interact with the each and every employee's agenda.

The prototype multi-agent system designed to support the call centre consists of the seven agents Client, Call Centre Agent, Work Manager, two Personal Assistants and two Employees, and includes a global clock to ensure that the schedules of the Personal Assistants are synchronised with the schedules of the Work Manager. More detailed descriptions of a Work Manager, a Personal Assistant and (an interface for) an Employee, are provided in Sections 7.3, 7.4 and 7.5.

7.3 Work Manager

To successfully develop a support system for co-operation in a complex, dynamic and not always predictable environment, a well-defined and transparent model of co-operation is required: a model that is robust and flexible enough to cope with unexpected events. In the generic model of a co-operative agent GCAM specified in Section 5, agents are capable of organising and monitoring projects to reach given goals. This section describes how this generic model has been refined to obtain an instantiated model of the Work Manager. How the generic co-operative agent model GCAM was tuned (specialised and/or instantiated) to this application is discussed for each of its components.

As described in Sections 7.1 and 7.2, a Work Manager is free to decide whether or not to accept a request communicated by a Call Centre Agent. If, after interpretation of a request, a Work Manager decides to accept the request, this request is translated into a goal for the Work Manager to adopt. This part of the process of the Work Manager is modelled within the Work Manager's component own process control. To achieve the adopted goal, co-operation with Personal Assistants is required. Within the component cooperation management the Work Manager selects an appropriate plan (the procedure to which the Bank's description of the application domain referred) for this goal and determines a schedule for the activities in the procedure. The Work Manager asks Personal Assistants of those Employees selected to execute the schedule (this communication is managed by its component agent interaction management) whether or not they can commit to specific activities. If a proposed schedule is accepted by all relevant Personal Assistants, the Work Manager selects a set of possible appointments with the client and communicates this set to the Call Centre Agent. If not, the Work Manager needs to either adapt its schedule or choose another procedure, depending on the information communicated by the Personal Assistants. This process is discussed in more detail in this section.

7.3.1 Agent Interaction Management

The component agent interaction management is composed of two components: incoming communication management and outgoing communication management. A Work Manager can communicate with either the Call Centre Agent or a Personal Assistant.

Management of incoming communication

Within the component incoming communication management incoming communication is analysed and communicated information identified. For example, the following types of information can be identified:

- a new request (from a Call Centre Agent)
- commitment: a Personal Assistant commits to activity A, with deadline D, priority P, earliest starting time E, and latest starting time L.
- conditional commitment: a Personal Assistant commits to the request under the condition that the Work Manager relieves it of the commitment to activity A' that has a lower priority P' than activity A.
- refusal: a Personal Assistant cannot commit to activity A, with deadline D, priority P, earliest starting time E, and latest starting time L.
- a progress report on already scheduled activities (from a Personal Assistant); e.g., a reported delay.

Monitoring information communicated by a Personal Assistant in the form of a progress report specifies whether the Personal Assistant expects its employee to be able to perform a specific activity A (within a given time slot). A Personal Assistant can report, e.g., that a commitment A cannot be kept because no start has been/will be made at time L, the deadline D will not be met, or the necessary information/material regarding A is not available.

Depending on the type of information received, the implications for information to be provided to appropriate component(s) within the Work Manager are identified. For example, communicated information on a new request from a Call Centre Agent is needed by the Work Manager's component own process control, whereas the other two types of communicated information listed above are needed by the component cooperation management.

Management of outgoing communication

The component outgoing communication management prepares the following types of outgoing information:

- appointment proposals (to a Call Centre Agent)
- commitment requests (to a Personal Assistant)
- commitment confirmations (to a Personal Assistant)

The information to be communicated is provided by the component cooperation management, and transferred to agent interaction management through the information link cooperation info to AIM. Preparation of communication includes, for example, labelling outgoing communication so that the agents that receive the information can refer to this information in their reply.

7.3.2 Own Process Control

Within the component own process control requests communicated by the Call Centre Agent are analysed and the decision whether or not to accept a request (and as a consequence adopt a goal to respond to the request as one of the Work Manager's own goals) is made. The agent's own characteristics are explicitly represented within this component. Examples of specific agent characteristic are: that a request for a certain type of client (e.g., for a known client) is to be given priority, or that requests for credit card services are processed with higher priority than student loan services.

7.3.3 Maintenance of Agent Information

Within the component maintenance of agent information the Work Manager maintains information on the capabilities and preferences of the other agents. One example of information maintained by the Work Manager is the information a Work Manager has on the activities for which a given Personal Assistant can be approached.

7.3.4 Co-operation Management

As discussed in Section 5, the component cooperation management consists of two components; one for the generation of projects and one for the monitoring of existing projects (see Figure 3). The refinement of each of its components to the application domain is discussed in more detail in this section.

Generate Project

As presented in Section 5, the component generate project is composed of two components: prepare project commitments, and generate and modify project recipes (see Figure 5). Within the component generate project the component prepare project commitments receives the Work Manager's own goals. The component prepare project commitments's aim is to determine procedures that can be followed to achieve a given goal. This is performed using knowledge relating requests to procedures, for example of the following form:

if own_goal(appointment_request(service(credit_card)))
then selected_procedure(procedure(cp2))

To determine which activities are required to execute the procedure, knowledge is used that relates procedures to activities, and knowledge that defines duration of activities, and temporal relations between them; for example knowledge of the form:

if selected_procedure(procedure(cp2))
then selected_activity(activity(a1), duration(5))
and selected_activity(activity(a2), duration(3))
and precedence(activity(a1), activity(a2))

Within the component generate project the component generate and modify project recipe receives the selected activities, their duration and temporal relations between them. It determines which Personal Assistant is capable of taking responsibility for a given activity (using agent information maintained in maintenance of agent information) and proposes a schedule. This process involves intensive interaction with the Personal Assistants and may iterate a number of times (according to the wave model

introduced in Section 5), until the proposed schedule is accepted by all participating Personal Assistants.

Monitor Project

Progress of the procedure is monitored within the component monitor project. For example, this component determines when a goal should be withdrawn (for example, because the goal is unattainable, or the goal has been reached) and prepares and issues communication to that effect to each participating Personal Assistant.

7.4 Personal Assistant

All Personal Assistants are also modelled as a refinement of (the part of) the generic model of a co-operative agent GCAM. This section describes how this generic model has been refined to obtain an instantiated model of the Personal Assistant. For each of the components of the generic co-operative agent model GCAM it is discussed how this component was tuned (specialised and/or instantiated) to this application. Only four components are used: own process control, agent interaction management, cooperation management, and maintenance of agent information. A Personal Assistant communicates with both the Work Manager and the Employee.

7.4.1 Interaction with the Work Manager

A Personal Assistant (PA) receives requests from a Work Manager for:

1. a commitment to a specific activity A before a certain deadline
2. (possibly with) additional information on the importance of the activity (priority P), the earliest starting time (E) and the latest starting time (L).
3. cancellation of a commitment
4. monitoring information on a specific activity A.

Requests for commitment and cancellation

Incoming communication in the form of requests for commitment (from the Work Manager), is analyzed and the relevant communicated information is identified and classified (comparable to the process described in Section 7.3.1). An identified request for commitment is transferred from the component agent interaction management to the component own process control. The component own process control decides whether or not to accept a request as a goal for the Personal Assistant (see Section 4 for further explanation). If a request is accepted, this information is transferred to component cooperation management. The component cooperation management is composed as discussed in Section 5. The component prepare project commitments determines whether or not the Employee the Personal Assistant represents is capable of performing the activity, and the component generate and modify schedule determines whether a new schedule can be generated in which the requested commitment can be awarded. If a new schedule can be generated, this schedule is forwarded to the component monitor project. The component monitor project uses information about the schedule, commitments to identify contradictions and to take appropriate action.

If a new schedule cannot be generated by the component generate project without changing existing commitments (with information acquired by the component prepare project commitments), information about the nature of the conflict is transferred to the component agent interaction management: commitment

can only be acquired if a commitment with lower priority is cancelled, otherwise given the current priorities and schedule, commitment is not possible. The component agent interaction management manages the communication on the issue with the Work Manager.

Requests for monitoring information

A Personal Assistant also receives requests for monitoring information. These requests are identified by the component agent interaction management and transferred to the component cooperation management. Within the component cooperation management, the component monitor project is responsible for monitoring the execution of a procedure, and providing the necessary information (through the output interface of the component cooperation management) to the component agent interaction management to communicate to the Work Manager. To monitor a procedure, the component monitor project requires information on the current status of a procedure. This is transferred to the output interface of the component cooperation management, and from there to agent interaction management, which manages the communication with the Employee.

7.4.2 Interaction with the Employee

The Personal Assistant requests information about current commitments and schedules from the Employee. These requests are devised by the component monitor project within the component cooperation management. In addition, the component receives information communicated by its Employee without having initiated interaction. An Employee may provide information on changes in his/her schedule without having been explicitly requested to do so. This information is identified in agent interaction management and transferred to the component cooperation management. Within the component cooperation management the component monitor project detects possible new conflicts.

7.5 The Employee

Within the multi-agent system the Employee only interacts with his/her Personal Assistant, as shown in Figure 11. Interaction with the Employee is modelled by the interface in the prototype system. The interface presents the contents of the agenda as received from the Personal Assistant. The interface allows for the Employee to make changes to the agenda. These changes are communicated to the Personal Assistant. In addition, the Employee can make changes to the Employee profile maintained by the Personal Assistant.

More details of this multi-agent system for distributed agenda scheduling in the context of Call Centre support can be found in (Brazier, Jonker, Jüngen and Treur, 1998).

8 Discussion

Multi-agent literature focuses on modelling interaction between agents, most frequently based on informal models of interaction; see (Wooldridge and Jennings, 1995). In this paper one of the informally described models of agent co-operation (Jennings, 1995) has been used to develop and formally specify the generic model of a co-operative agent GCAM. The compositional development method for multi-agent systems DESIRE supported the principled design of this model of co-operation.

To illustrate reusability of the generic model GCAM, two application domains have been addressed: collaborative engineering design, and Call Centers. Collaborative, concurrent engineering projects are complex. The co-ordination of these projects in virtual environments, in particular the co-ordination of conflicting (partial) designs, interests, models, requirements (e.g., new requirements imposed during design), et cetera, requires extensive knowledge of the design process, of the available expertise and skills, of dependencies and, in particular, of the consequences of modification. Recently a number of tools and services have been designed to support specific aspects of the co-ordination process; for example, (Bahler, Dupont and Bowen, 1994; Cutkosky, Engelmore, Fikes, Gruber, Genesereth, Mark, Tenenbaum and Weber, 1993; Klein, 1995; Petrie, 1994; Goldman, 1996; Gupta, Chionglo and Fox, 1996; Maurer, 1996). In Section 6 a multi-agent perspective to project co-ordination was presented in which each agent is obtained as an instantiation of GCAM.

Another application of the co-operation model has been developed in the domain of Call Center support. More and more organisations offer a 24 hour a day telephone service using a call centre to co-ordinate the service provided. Without support to really support clients, by, for example, being able to schedule appointments with a client, such a service is of limited value: only simple questions can be answered. This paper has presented a multi-agent system, introduced to increase the value of 24 hour a day service by supporting call centres in making appointments and scheduling activities of employees in preparation of such appointments. This multi-agent system architecture has been applied to the banking domain, in co-operation with (and partially funded by) the Rabobank, one of the largest banks in the Netherlands. In this system scheduling is a co-operative distributed effort: each Employee is represented by its own Personal Assistant agent (that also maintains the Employee's agenda), and a Work Manager agent co-ordinates the schedules, and the client's requirements (through the Call Centre Agent). Each of the agents was developed as a refinement of the generic co-operative agent model GCAM.

As shown in this paper compositional DESIRE models specify processes and knowledge at different levels of abstraction. Information exchange between processes and process sequencing are explicitly defined at each of the levels distinguished. Different levels of abstraction within the knowledge composition structure information types and knowledge bases. Reuse of generic models within DESIRE is supported by their transparent compositional structure. This paper shows how the compositional generic specifications of the model GCAM can be used in a variety of situations, instantiated for the specific domain of application. By formally specifying not only the knowledge involved, but also the types of interaction and co-ordination patterns required during these types of projects, more detailed insight is acquired in the required support for project co-ordination.

The compositional approach to agent design followed in this paper has some aspects in common with object oriented design methods; e.g., (Booch, 1994; Coleman, Arnold, Bodoff, Dollin, Gilchrist, Hayes, and Jeremaes, 1994; Rumbaugh, Blaha, Pelerlani, Eddy, and Lorensen, 1991). However, there are differences as well. Examples of approaches to object-oriented agent specifications can be found in (Aridor and Lange, 1998; Kendall, Murali Krisna, Pathak, and Suresh, 1998). A first interesting point of discussion is to what the difference is between agents and objects. Some tend to classify agents as different from objects. For example, Jennings and Wooldridge (1998a) compare objects with agents on the dimension of autonomy in the following way:

> ‘An object encapsulates some state, and has some control over this state in that it can only be accessed or modified via the methods that the object provides. Agents encapsulate state in just the same way. However, we also think of agents as encapsulating behavior, in addition to state. An object does not encapsulate *behavior*: it has no control over the execution of methods – if an object x invokes a method m on an object y, then y has no control over whether m is executed or not – it just *is*. In this sense, object y is not autonomous, as it has no control over its own actions. In contrast, we think of an agent as having *exactly* this kind of control over what actions it performs. Because of this distinction, we do not think of agents as invoking methods (actions) on agents – rather, we tend to think of them *requesting* actions to be performed. The decision about whether to act upon the request lies with the recipient.’.

Some others consider agents as a specific type of objects that are able to decide by themselves whether or not they execute a method (objects that can say ‘no’), and that can initiate action (objects that can say ‘go’).

A difference between the compositional design method DESIRE and object-oriented design methods in representation of basic functionality is that within DESIRE declarative, knowledge-based specification forms are used, whereas method specifications (which usually have a more procedural style of specification) are used in object-oriented design. Another difference is that within DESIRE the composition relation is defined in a more specific manner: the static aspects by information links, and the dynamic aspects by (temporal) task control knowledge, according to a prespecified format. A similarity is the (re)use of generic structures: generic models in DESIRE, and patterns (Alexander, 1977; Gamma, Helm, Johnson, and Vlissides, 1995; Fowler, 1997; Grand, 1998) in object-oriented design methods, although their functionality and compositionality are specified in different manners, as discussed above.

Other approaches include the use of an agent co-ordination language, an informal architecture for the design of hierarachies of co-operative agents and an informal architecture for a generic agent, as described below.

COOL (Barbuceanu & Fox, 1995) is an agent co-ordination language that focuses on the specification of co-ordination between agents (such as the co-ordination required in a supply chain, (Barbuceanu & Fox, 1996)). It uses finite state machines to describe the flow of communication between agents based on a fixed number of speech acts, such as propose, accept, and reject. The main difference between DESIRE and COOL is that COOL focuses on a specific way of modelling co-ordination between agents and not on the architecture of an agent itself, while DESIRE provides generic models to specify agent architectures, without prescribing specific protocols for specific functionality, such as, for example interaction between agents. The method used by COOL to describe the co-ordination between agents could be used to model the co-ordination in the Cooperation Management component of the DESIRE agent. The designer is however free to chose another coordination method.

The ADEPT architecture (Advanced Decision Environment for Process Tasks; see (Jennings, Faratin, Norman, O'Brien, Wiegand, Voudouris, Alty, Miah, and Mamdani, 1996)) is, in some ways, comparable to the model presented in this paper, although it was not formally specified to our

knowledge. It models business processes by a hierarchy of co-operative agents. The hierarchy ensures that communication overhead between agents and the autonomy of the agents are balanced. Within this architecture, 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 correspond to the components within the generic DESIRE agent model: the module IMM may be viewed as the component cooperation management, the SAM may be viewed as part of the component own process control, the SEM is clearly related to the Agent Specific Task. The SM is also part of the component own proces control and the module AM can be viewed as the component maintenance of agent information.

Also the ZEUS architecture of a generic agent (Nwana, Ndumu and Lee, 1998) is, to a certain degree, comparable to the generic DESIRE agent model, but was also not specified (at least to our knowledge). The ZEUS model 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 correspond to the component agent interaction management within the generic DESIRE agent model. The Co-ordination Engine is modelled by the component cooperation management. The Execution Monitor with the Planner and Scheduler, and the Task/Plan Database together provide the functionality provided by the component own process control. The Acquaintance Model is comparable to the 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. The ZEUS agent does not include models for specific types of tasks, but focuses on reusable components for interaction based on standard interface protocols.

Acknowledgements

Discussions on the co-operation model with Nick Jennings were very useful to the authors. For the analysis of project co-ordination in collaborative aircraft design, co-operation with Cor van der Lee,

from the former Fokker Aircraft, was crucial. Part of the research has been funded by Rabobank, Telemachos Planning and the Ministry of Economic Affairs. Close collaboration on the Call Centre application with domain experts within the Rabobank, and with Frederik-Jan Jungen (Cambridge Technology Partners), Pieter van Langen (Rabofacet), Patrick Frants (MSc student) and Guy Kornblum (Telemachos Planning) in particular, has been of utmost importance to this application. The contribution of all mentioned persons is greatly appreciated.

References

Alexander, C. (1977). A Pattern Language. Oxford University Press.

Aridor, Y., and Lange, D.B. (1998). Agent Design Patterns: Elements of Agent Application Design. Proc. of the Second Annual Conference on Autonomous Agents, Agents'98, ACM Press, pp. 108-115.

Barbuceanu, M. and Fox, M.S. (1995). COOL: A Language for Describing Coordination in Multi Agent Systems. In: V. Lesser (ed.), Proc. of the First International Conference on Multi-Agent Systems, ICMAS'95, MIT Press, Cambridge, MA, pp. 17-24.

Barbuceanu, M. and Fox, M.S. (1996). Coordinating Multiple Agents in the Supply Chain. In: Proceedings of the Fifth Workshops on Enabling Technology for Collaborative Enterprises, WET ICE'96, IEEE Computer Society Press, pp. 134-141.

Booch, G. (1994). Object-Oriented Analysis and Design (2nd ed.). Addison-wesley. reading, MA.

Bradshaw, J. (1997) (e.d). Software Agents. AAAI Pres, 1997.

Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R. and Treur, J. (1995). Formal specification of Multi-Agent Systems: a real-world case. In: V. Lesser (ed.), Proc. 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., Jonker, C.M., Jungen, F.J., Treur, J. (1998). Distributed Scheduling to Support a Call Centre: a Co-operative Multi-Agent Approach. 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, pp. 555-576. Also in: Applied Artificial Intelligence Journal, vol. 13, 1999, pp. 65-90. Special Issue with selected papers from PAAM'98.

Brazier, F.M.T., C.M. Jonker, J. Treur (1996). Modelling Project Coordination in a Multi-Agent Framework. In: Proceedings of the Fifth Workshops on Enabling Technology for Collaborative Enterprises, WET ICE'96, IEEE Computer Society Press, 1996, pp. 148-155

Brazier, F.M.T., Jonker, C.M., Treur, J. (1997). Formalisation of a cooperation model based on joint intentions. In: (Müller, Wooldridge and Jennings, 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.

Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., and Jeremaes, P. (1994). Object-Oriented Development: the FUSION method. Prentice Hall International: Hempel Hempstead, England.

Fowler, M. (1997). Analysis Patterns: Reusable Object Models. Addison Wesley.

Gamma, E.R., Helm, R., Johnson, R., and Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Grand, M. (1998). Patterns in Java: Volume 1. John Wiley and Sons.

Genesereth, M.R., Fikes, R.E. (1992). Knowledge Interchange Format - version 3 - reference manual. Technical Report Logic Group, Logic-92-1, Stanford University, Stanford, CA.

Jennings, N.R. (1995). Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, Artificial Intelligence Journal 74 (2), pp 195-240.

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.

Kendall, E.A., Murali Krisna, P.V., Pathak, C.V., and Suresh, C.B. (1998). Proc. of the Second Annual Conference on Autonomous Agents, Agents'98. ACM press.

Müller, J.P., Wooldridge, M.J., and Jennings, N.R. (eds.) (1997). 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

Norman, T.J., Jennings, N.R., Faratin, P. and Mamdani, E.H. (1997). Designing and implementing a multi-agent architecture for business process management. In: (Müller, Wooldridge and Jennings, 1997), pp. 261-275.

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, pp. 377-391.

Rumbaugh, J., Blaha, M., Pelerlani, W., Eddy, F., and Lorensen, W. (1991). Object-Oriented Modelling and Design. Prentice Hall, Eaglewoods Clifs, NJ.

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

Wooldridge, M.J., and Jennings, N.R. (eds.) (1995b). Intelligent Agents (Proc. of the First International Workshop on Agent Theories, Architectures and Languages, ATAL'94), Lecture Notes in Artificial Intelligence, Vol. 890, Springer Verlag, Berlin.