An Agent Architecture for Dynamic Re-design of Agents

Frances M.T. Brazier, Catholijn M.Jonker, Jan Treur, and Niek J.E. Wijngaards

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

Abstract. This paper presents a generic architecture for an agent capable of designing and creating new agents. The design agent itself is based on an existing generic agent model, and includes a refinement of a generic model for design, in which strategic reasoning and dynamic management of requirements are explicitly modelled. This model is refined for the design of agents, or (parts of) multi-agent systems. It includes an explicit formal representation at a logical level of (1) requirements that can be formulated for agents and multi-agent systems, and (2) design object descriptions of a (part of a) multi-agent system. The generic architecture has been formally specified in DESIRE, and has been tested in a prototype application.

1 Introduction

Agents that are able to dynamically design and create new agents, or to dynamically modify existing agents can be very useful. For example, Internet agents that are capable of dynamically creating new agents to assist them in information gathering, or agents that are capable of creating interface agents tuned to specific users, are agents of this type. To design an agent capable of (re)designing and creating agents, the following aspects must be addressed:

- an agent model as a basis for the design agent
- a model of the design task used by the design agent
- explicit representation within the design agent of requirements on agents to be (re)designed and knowledge to derive refinements of these requirements
- explicit representation within the design agent of agent design object descriptions, and knowledge to derive properties of design object descriptions
- a model and implementation of the execution of the creation action that actually creates (while the multi-agent system is running) the designed agent on the basis of the design

In this paper a generic architecture is introduced for an agent capable of designing and creating new agents, which was modelled using the compositional development method for multi-agent systems DESIRE (Brazier, Dunin-Keplicz, Jennings and Treur, 1995). The design agent itself is based on an existing generic agent model (Section 2.1), and includes a refinement of a generic model for design (Brazier, Langen, Ruttkay and Treur, 1994), in which strategic reasoning and dynamic management of requirements are explicitly modelled (Section 2.2). In this paper this model is refined for the design of agents, or (parts of) multi-agent systems. It includes an explicit formal representation at a logical level of (1) requirements that can be formulated for agents and multi-agent systems (Section 3), and (2) design object descriptions of a (part of a) multi-agent system (Section 4).

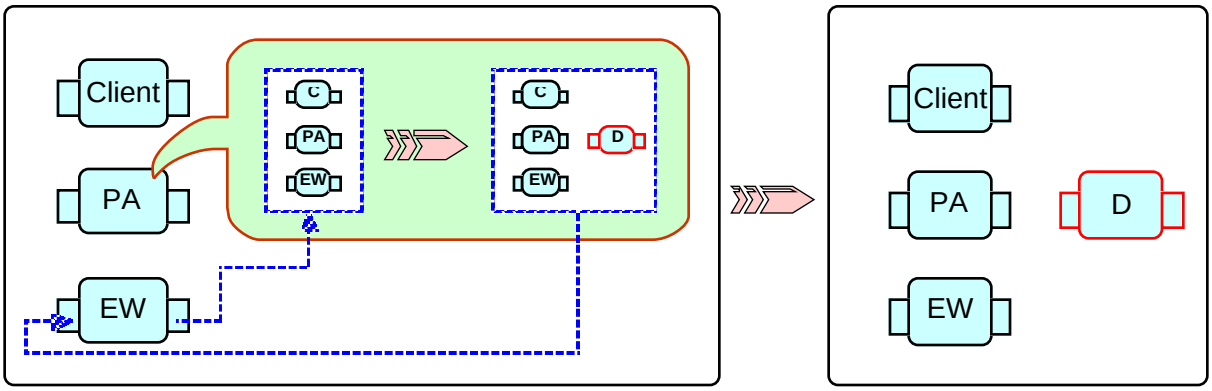


Figure 1 Redesign of a multi-agent system: agent PA modifies the structure of the multi-agent system

Moreover, additional knowledge is included of different types, for example, knowledge that can be used to derive whether a given design object description satisfies given properties (e.g., requirements), and knowledge that can be used to derive how to refine requirements into more specific requirements. After an agent has been designed by the design agent, this design is effectuated by execution of a creation action of the design agent in the external world. After this creation action the multi-agent system functions with the additional agent.

In Figure 1 a sketch of the redesign process is depicted. The left box contains the multi-agent system before modification (consisting of the agents Personal Assistant, Client and External World), the right box after modification (with an additional agent D included). The Personal Assistant (PA) plays the role of the design agent. It has internal representations of the multi-agent system before modification and designs a modification of the system by adding an agent D to the system. After this design process it effectuates the design by execution of the creation action in the External World, which represents all material aspects, including the material aspects of the agents.

2 A Generic Model of a Design Agent

The generic model of a design agent proposed in this paper is a refinement of a generic agent model (Section 2.1) and includes a refinement of a generic model of design (Section 2.2).

2.1 A Generic Agent Model

Agents are often designed to perform their own specific tasks, for example the design of an artifact. In addition, a number of generic agent tasks can be identified. This section describes a generic agent model in which such generic agent tasks are modelled. This model abstracts from the specific domain of application and can be (re)used for a large variety of agents. The model is based on the abilities associated with the notion of weak agency (Wooldridge and Jennings, 1995). Instead of designing each and every new agent individually from scratch, a generic agent model can be used to structure the design process: the acquisition of a specific agent model is based on the generic structures in the model.

The characteristics of weak agency provide a means to reflect on the tasks an agent needs to be able to perform. Pro-activeness and autonomy are related to an agent's ability to reason about its own processes, goals and plans and to control these processes (own process control). Reactivity and social ability are related to the ability to be able to communicate with other agents (agent interaction management) and to interact with the external world (world interaction management). The ability to communicate with other agents and to interact with the external world often relies on the information an agent has of the world (maintenance of world information) and other agents (maintenance of agent information). The generic agent model also includes an empty generic component to model the agent specific task. The tasks related to the generic abilities and agent specific tasks may be modelled by components within an agent as depicted in Figure 2. In addition to the sub-components, the model includes information links that specify which information is exchanged between components; these information links are named.

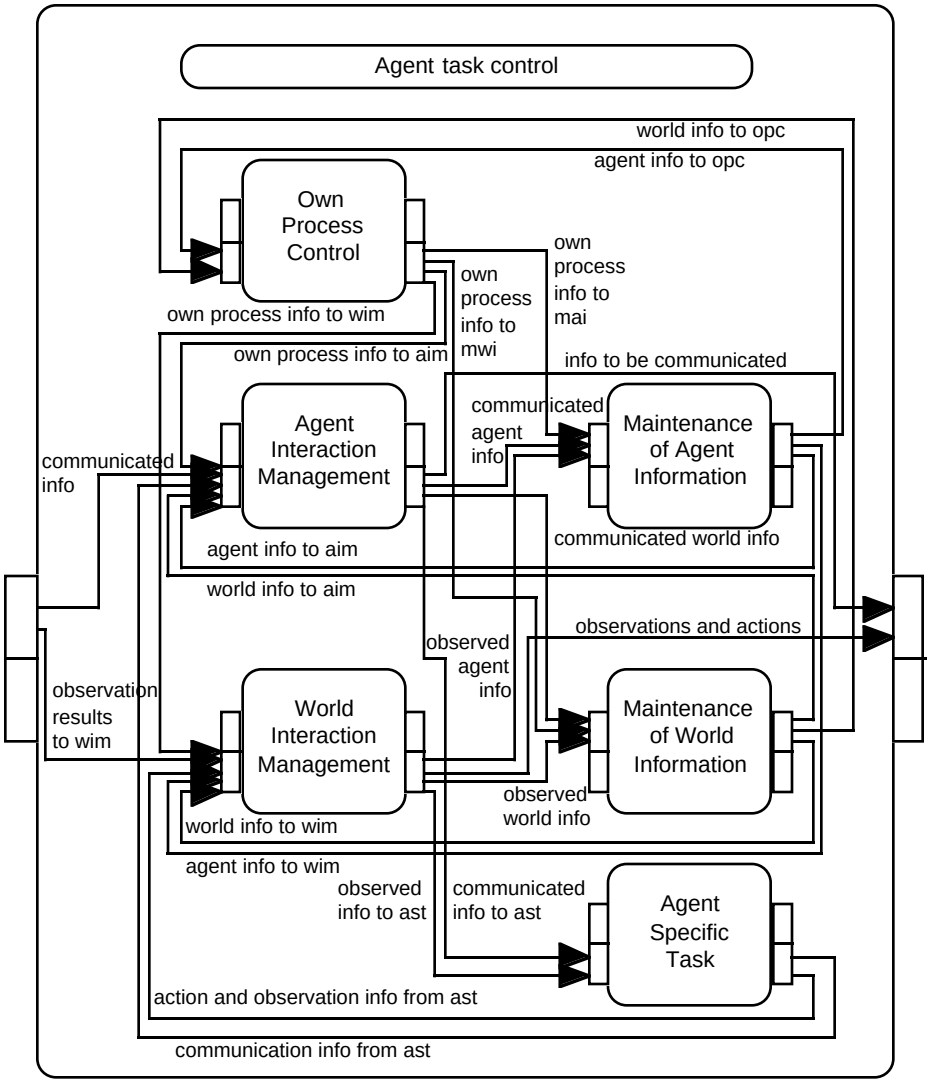


Figure 2 A Generic Agent Model

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

The component maintenance of world information receives meta-information on observed world information from the component world interaction management, through the information link observed world info and meta-information on communicated world information (through the link communicated world info) from the component agent interaction management. Epistemic information

from maintenance of world information, epistemic world info, is transferred to input belief info on world of the components world interaction management, agent interaction management and own process control, through the information links world info to wim, world info to aim and world info to opc.

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

2.2 A Generic Model of Design

The generic model of a design agent is based on both the generic agent model discussed in Section 2.1, and a generic model of the design task, used to model the agent specific task component. In this section the structure of this agent specific task component for a design task is discussed.

A *generic model of design*, in which reasoning about requirements and their qualifications, reasoning about design object descriptions and reasoning about the design process are distinguished, has been introduced in (Brazier, Langen, Ruttkay and Treur, 1994). This model is based on a logical analysis of design processes (Brazier, Langen and Treur, 1996) and on analyses of applications, including elevator configuration (Brazier, Langen, Treur, Wijngaards and Willems, 1996) and design of environmental measures (Brazier, Treur and Wijngaards, 1996). The model not only provides an abstract description of a design process comparable to a *design model* , e.g., (Coyne, Rosenman, Radford, Balachandran and Gero, 1990; Smithers, 1994), but also a generic structure which can be refined for specific design tasks in different domains of application. Refinement of the generic task model of design, by specialisation and instantiation, involves the specification of knowledge about applicable requirements and their qualifications, about the design object domain, and about design strategies.

An initial design problem statement is expressed as a set of initial requirements and requirement qualifications. *Requirements* impose conditions and restrictions on the structure, functionality and behaviour of the *design object* for which a structural description is to be generated during design. *Qualifications* of requirements are qualitative expressions of the extent to which (individual or groups of) requirements are considered hard or preferred, either in isolation or in relation to other (individual or groups of) requirements. At any one point in time during design, the design process focuses on a specific subset of the set of requirements. This subset of requirements plays a central role; the design process is (temporarily) committed to the current requirement qualification set: the aim of generating a design object description is to satisfy these requirements. Other qualifications of requirements may play a heuristic role.

During design the considered subsets of the set of requirements may change as may the requirements themselves. The same holds for design object descriptions and design object knowledge: they evolve during design. The strategy employed for the coordination of

requirement qualification set manipulation and design object description manipulation may also change during the course of a single design process. Modifications to the requirement qualification set, the design object description and the design strategy, may be the result of straightforward implications drawn from knowledge available to a design support system. Modifications may also be the result of specific knowledge on appropriate default assumptions (see also (Smith and Boulanger,1994), or the result of interaction with an outside party (e.g., a client or a designer). Figure 3 shows two levels of composition of the generic model for design. Three processes are shown at the top level, together with the information exchange.

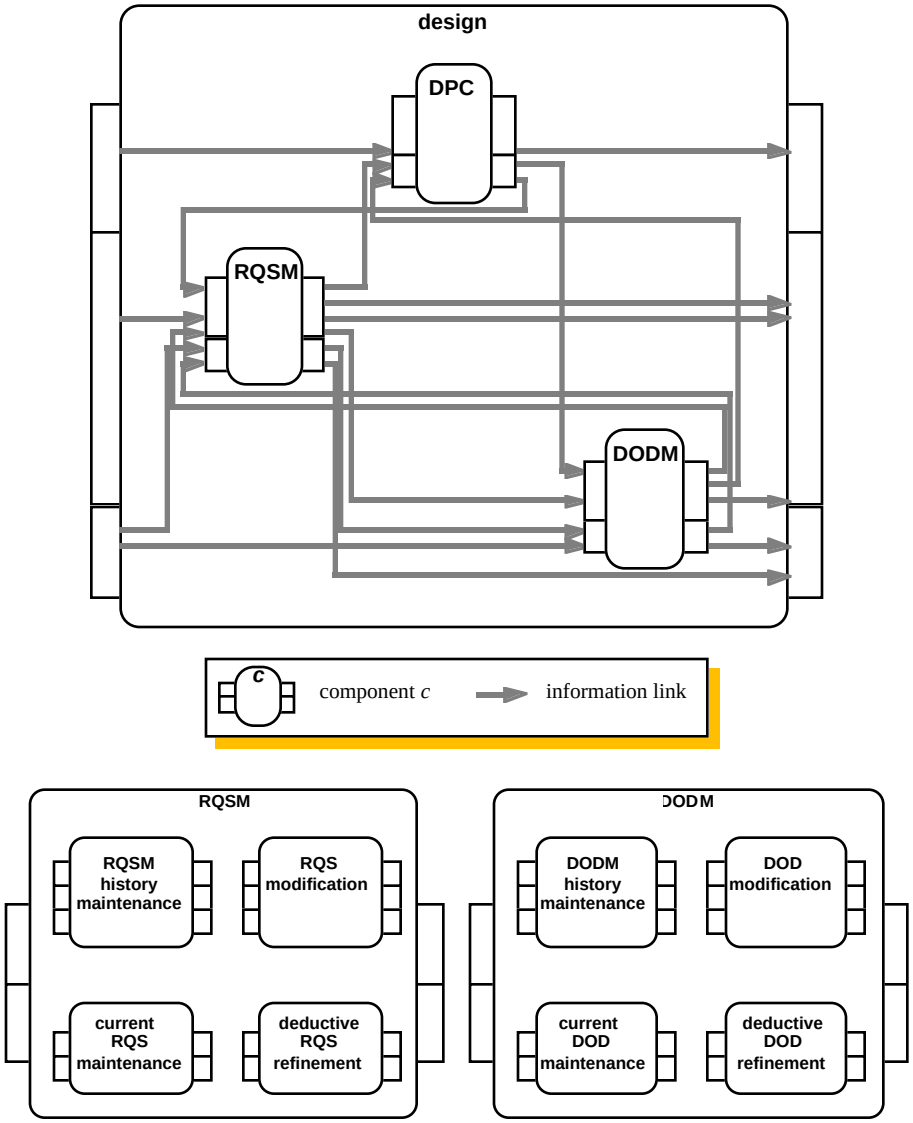


Figure 3 A generic model of design

The four processes (see Figure 3) related to the process *requirement qualification set manipulation* are:

- RQS modification: the current requirement qualification set is analysed, proposals for modification are generated, compared and the most promising (according to some measure) selected,
- deductive RQS refinement: the current requirement qualification set is deductively refined by means of the theory of requirement qualification sets,
- current RQS maintenance: the current requirement qualification set is stored and maintained,

- RQSM history maintenance: the history of requirement qualification sets modification is stored and maintained.

The four processes related to the process of *manipulation of design object descriptions* are:

- DOD modification: the current design object description is analysed in relation to the current requirement set, proposals for modification are generated, compared and the most promising (according to some measure) selected,
- deductive DOD refinement: the current design object description is deductively refined by means of the theory of design object descriptions,
- current DOD maintenance: the current design object description is stored and maintained,
- DODM history maintenance: the history of design object descriptions modification is stored and maintained.

The process *design process coordination* is composed in a similar manner.

3 Representation of requirements within a design agent

The generic model of a design agent introduced in Section 2, can in principle be used for any domain of application. The next step in this paper is to show how it can be used in the specific domain of multi-agent system design. In this section a formal representation of requirements on multi-agent systems is shown. Moreover, knowledge is presented that can be used to reason about these requirements, to derive more specific requirements by refining the original requirements. These more specific requirements play a crucial role in the design process: they guide the direction in which solutions are sought.

In this section examples are given of the representation and manipulation (refinement) of requirements on abilities of agents. A prototype design system for agent design has been developed on which the examples are based. The examples are simplified to the extent that only processes and information exchange are shown; issues such as control over processes and definitions of information are not addressed in this paper.

Example Situation description

Figure 1 depicts the initial multi-agent system. Agent C represents the (human) client; agent PA represents a personal assistant. The client can ask certain questions and the personal assistant provides answers to these questions. For the sake of the example, consider the situation in which the client poses a specific question for information. The personal assistant receives this request, and realizes that it does not have the information asked for. However, the personal assistant is able to design other agents to solve specific types of problems. To this end a number of requirements need to be formulated and information on the structure of the multi-agent system needs to be acquired on the basis of which a new agent can be designed to search for information to answer the question of the client.

Requirements are formulated in terms of abilities and properties of agents and the external world. Abilities and properties can be assigned to

- individual agents,
- the external world,
- an individual agent in relation to the agents and the world with which it interacts,
- the world in relation to the agents with which it interacts, and

- a multi-agent system as a whole.

Example Prerequisites for re-design

The design agent (i.e. agent_A) formulates the following initial requirements for the new agent:

```
is_requirement( r_m1, has_property( mas_S, agent_solves_subproblem_for( agent_D, information_gatherer, agent_A ) ) );
is_requirement( r_m2, agent_task_explication( agent_D, information_gatherer, searching( internet, scientific_publications ) ) );
```

These requirements state that the new agent should solve subproblems for the existing agent A by gathering information (which takes place in the external world). As specific subject of expertise for this new agent, it should be able to search the internet for (and understand annotations of) scientific publications. Not only requirements are needed to design a new agent, but also knowledge of the structure of the existing multi-agent system. To this end the design agent A makes an explicit observation in the external world EW and observes the structure of the existing multi-agent system. The agent's design task commences on the basis of these requirements and the structure of the multi-agent system commences the agent its design task by manipulating requirements and by manipulating agent representations.

Abilities of agents such as co-operation, bi-directional communication, and world interaction are often needed for agents to jointly be able to perform a certain task. The next section describes the ability of bi-directional communication in relation to the requirements on the multi-agent system. For a description of other agent abilities see Brazier, Jonker, Treur and Wijngaards (1997).

3.1 Generic ability of bi-directional communication

The ability of bi-directional communication can be refined, both with respect to its *specialisation* (refinement of the ability into more specific abilities) and with respect to its *realisation* (refinement of the ability into more fine-grained abilities related to reasoning about the ability, and more fine-grained abilities abilities related to the effectuation of the ability).

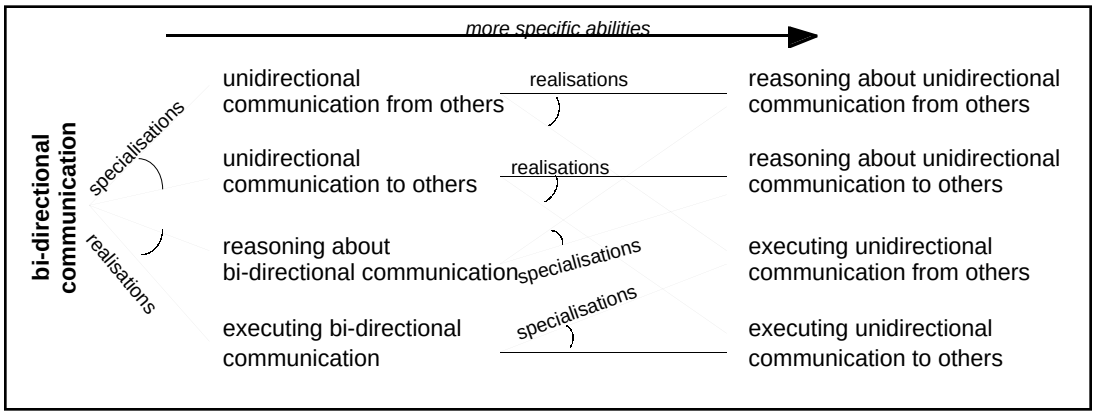


Figure 4 Refinements of the ability of bi-directional communication

Figure 4 shows the refinement relationships for the ability of bi-directional communication. The more specific abilities related to bi-directional communication are the ability to communicate to others (unidirectional communication to others) and the ability to receive communication from others (unidirectional communication from others). The abilities related to the realisation of the

ability of bi-directional communication are the ability to *reason* about bi-directional communication, and the ability to *execute* bi-directional communication.

These more specific abilities are further refined, and related to the ability to reason about unidirectional communication from others, the ability to reason about unidirectional communication to others, the ability to execute unidirectional communication from others, and the ability to execute unidirectional communication to others.

Knowledge on refinements of the ability of bi-directional communication can be formally represented as shown below. Meta-reasoning is employed to decide which refinement alternative should be employed for which ability.

Example Representation of requirements refinement knowledge

Below two formal rules are presented which correspond to two refinements shown in Figure . The format of a rule is as follows: the first condition specifies which requirement has been selected to be refined. The second condition specifies the required ability or property, and the third condition concerns which refinement alternative should be considered (which is decided elsewhere). The conclusions provide possible refinements for the requirement in focus.

```
if    is_requirement_selected_as_focus( R: requirement_name )
and   is_requirement( R: requirement_name, has_ability( A: agent_name, bi_directional_communication( A2: agent_name ) ) )
and   refinement_alternative( specialisations )
then  is_possible_refinement_for( R: requirement_name,
            has_ability( A: agent_name, unidirectional_communication_from( A2: agent_name ) ) )
and   is_possible_refinement_for( R: requirement_name,
            has_ability( A: agent_name, unidirectional_communication_to( A2: agent_name ) ) );

if    is_requirement_selected_as_focus( R: requirement_name )
and   is_requirement( R: requirement_name,
            has_ability( A: agent_name, unidirectional_communication_from( A2: agent_name ) ) )
and   refinement_alternative( realisations )
then  is_possible_refinement_for( R: requirement_name,
            has_ability( A: agent_name, reasoning_about_unidirectional_communication_from( A2: agent_name ) ) )
and   is_possible_refinement_for( R: requirement_name,
            has_ability( A: agent_name, executing_unidirectional_communication_from( A2: agent_name ) ) );
```

3.2 Manipulating required abilities

On the basis of the requirements given to the design process, additional, more refined, requirements can be determined. The assumption underlying the refinement of requirements into more specific requirements is that more specific requirements can be used to focus the design process.

Example Manipulation of requirements

On the basis of the given requirements, more refined requirements can be formulated. For the first requirement r_m1, refinement knowledge is applied which results in the following refinement graph:

has_ability(mas_S, agent_solves_subproblem_for(agent_D, information_gatherer, agent_A))
- has_ability(agent_A, bi-directional_communication_with(agent_D))
  - has_ability(agent_A, unidirectional_communication_from(agent_D))
    - has_ability(agent_A, executing_unidirectional_communication_from(agent_D))
    - has_ability(agent_A, reasoning_about_unidirectional_communication_from(agent_D))
  - has_ability(agent_A, unidirectional_communication_to(agent_D))
    - has_ability(agent_A, executing_unidirectional_communication_to(agent_D))
    - has_ability(agent_A, reasoning_about_unidirectional_communication_to(agent_D))
- has_ability(agent_D, bi-directional_communication_with(agent_A))
  - has_ability(agent_D, unidirectional_communication_from(agent_A))
    - has_ability(agent_D, executing_unidirectional_communication_from(agent_A))
    - has_ability(agent_D, reasoning_about_unidirectional_communication_from(agent_A))
  - has_ability(agent_D, unidirectional_communication_to(agent_A))
    - has_ability(agent_D, executing_unidirectional_communication_to(agent_A))
    - has_ability(agent_D, reasoning_about_unidirectional_communication_to(agent_A))
- has_ability(agent_D, active_observation_in(world_W))
  - has_ability(agent_D, processing_observation_results_from(world_W))
    - has_ability(agent_D, reasoning_about_processing_observation_results_from(world_W))
    - has_ability(agent_D, executing_processing_observation_results_from(world_W))
  - has_ability(agent_D, observation_initiation_in(world_W))
    - has_ability(agent_D, reasoning_about_observation_initiation_in(world_W))
    - has_ability(agent_D, executing_observation_initiation_in(world_W))
- has_property(world_W, processing_active_observation_by(agent_D))

These refined requirements are used to construct a design object description.

Within the above example the ability of active observation in the world is introduced. This ability is refined into two specialised abilities: the ability of observation initiation and the ability of processing observation results. Refinement with respect to realisation would have resulted in the following refined abilities: ability of reasoning about observation initiation in the world, ability of executing observation initation in the world, ability of reasoning about processing observations results from the world, and ability of executing processing observation results from the world.

4 Representation of an agent design within a design agent

The formal representation of requirements on multi-agent systems has been shown in Section 3. In this section formal representations of design object descriptions for multi-agent systems are presented. Moreover, knowledge that can be used to derive properties of the design, for example the required properties, is presented.

4.1. Compositional design object description

The implication of designing (parts of) a multi-agent system, is that the multi-agent system itself is the object of design, and as such should be represented in a design object description. In this

paper the design object description is assumed to be a compositional object description. The assumption underlying this decision is that a compositional structure facilitates the process of (re-)design.

The description of the compositional system is augmented with a description relating existing structures to generic models. This provides information useful for documentation purposes and it also provides valuable information for the identification of abilities and properties.

Example Representation of an agent design

The design agent needs a representation of a multi-agent system including agents and the external world. To this purpose, a representation based on objects and attributes is used. Part of the top level of the multi-agent system can be represented as follows:

```
is_top_level( “c_00” );
has_value( "c_00", corresponds_with, "mas_S" );
has_value( "c_01", corresponds_with, "agent_A" );
has_value( "c_04", corresponds_with, "world_W" );
has_characterisation( “c_00”, generic, multi_agent_system );
has_characterisation( “c_01”, generic, agent );
has_characterisation( “c_4”, generic, external_world );
has_value( “lm_01", corresponds_with, "active_observations” );
has_value( "c_00", subcomponent, "c_01" );
has_value( "c_00", subcomponent, "c_04" );
has_value( “c_00”, information_link, “lm_01” );
has_value( “lm_01”, source_component, “c_01” );
has_value( “lm_01”, destination_component, “c_04” );
```

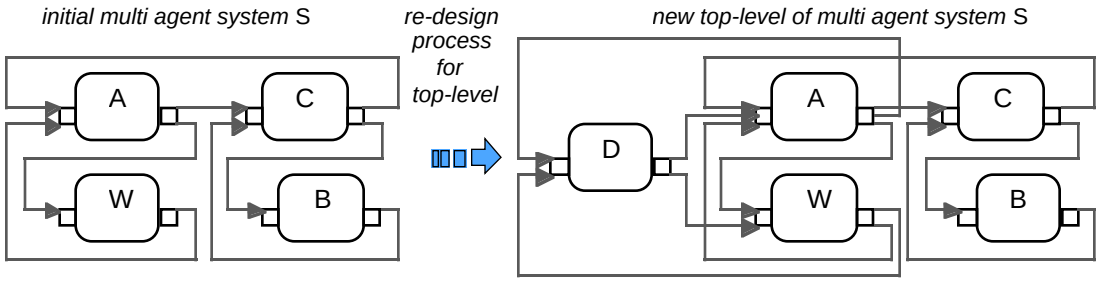
Unique identifiers are assigned to components and links so that names of links and components can be reused in several parts of the composition.

4.2. Modification of the top-level of the multi-agent system

The compositional structure of the design object guides the re-design process. Implications of modifications to the compositional structure of a multi-agent system are first explored at the top-level, then one level lower, et cetera.

Example Modification of the top-level of the multi-agent system

The result of modifying the top-level of the multi-agent system is shown below: on the basis of the initial description of the multi-agent system and refined requirements, a new multi-agent system is proposed which contains agent D.



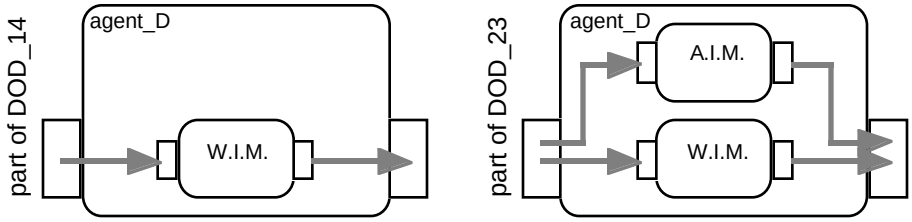
Note that although the agent D has been added and information links are present between D, A and W, the agent D is an empty component at this point in the design process.

4.3 Modifications within the agent D

When modifying the description of the agent D, several possible intermediate descriptions are explored during the re-design process. The description of an agent is constructed by modifying previous design object descriptions.

Example Modifications within the agent D

During the re-design process several descriptions of agents are proposed. For example, an agent D (part of design object description no. 14) may be proposed. Structural analysis shows that this particular agent D does have the ability of 'observation initiation', yet lacks the ability of 'bi-directional communication'.



A 'better' agent D (part of design object description no. 23) is shown in which both abilities are incorporated, as was required.

Knowledge is needed to analyse any given design object description, to establish whether particular abilities or properties hold. Particular goals, corresponding to the abilities and properties in the current requirements are used to focus this reasoning process.

Example Identifying an ability

As an example of knowledge with which an ability can be identified, consider the follow rules.

The first rule states that if, in addition to having the necessary task control knowledge to activate the world interaction process and links, the component with identifier I_agent has the generic structure of an agent, includes a component for world interaction management, that is linked to the output interface of the agent, and the agent is linked to the external world, then the agent has the ability of executing observation initiation.

```
if    has_characterisation( I_agent: ID, generic, agent )
  and     has_value( I_agent: ID, subcomponent, I_wim: ID )
  and     has_characterisation( I_wim: ID, generic, world_interaction_management )
  and     has_value( I_agent: ID, information_link, I_out: ID )
  and     has_value( I_out: ID, source_component, I_wim: ID)
  and     has_value( I_out: ID, destination_component, I_agent: ID )
  and     has_value( I_agent: ID, task_control, I_tc: ID )
  and     makes_awake( I_tc: ID, [ I_wim: ID, I_out: ID ] )
  and     has_characterisation( I_world: ID, generic, external_world )
  and     has_value( I_link: ID, source_component, I_agent: ID )
  and     has_value( I_link: ID, destination_component, I_world: ID )
  then    has_ability( I_agent: ID, executing_observation_initiation_in( I_world: ID ) );
```

The rule below shows how the knowledge on refinement of abilities can also be used to conclude that a more generic ability holds.

```
if    has_ability( I_agent: ID, reasoning_about_unidirectional_communication_from( I_agent2: ID ) )
  and     has_ability( I_agent: ID, executing_unidirectional_communication_from( I_agent2: ID ) )
  then    has_ability( I_agent: ID, unidirectional_communication_from( I_agent2: ID ) );
```

When the re-design process has finished, the results include a set of requirements (based on the initial requirements) and a design object description, for example with label dod_55, which fulfills the set of requirements.

The 'size' of the resulting design object description can be 'tuned'. In this situation only the differences between the initial and new multi-agent system are of importance. This includes adding agent D, communication from agent D to agent A and vice versa, interaction from agent A to W and vice versa, plus modifications within agent A and W (to be able to handle agent D).

5 Creation action: realisation of a designed agent

After the design process within the component agent specific task of agent_A has been completed, the agent decides to effectuate the modifications.

5.1 Preparation of the effectuation of the new design

As discussed in (Jonker and Treur, 1997) effectuation of the modification of the design can be modelled by changing the material representation of the multi-agent system within the external world.

Example Changing the material representation of the multi-agent system.

The resulting design object description, dod_55, contains the complete set of modifications that are to be made to the multi-agent system mas_S (including the creation of a new agent). The design object properties that together form dod_55, are represented by statements such as:

has_DOD_characteristic(“dod_55”, has_value(“c_05”, corresponds_with, “agent_D”));

has_DOD_characteristic(“dod_55”, has_value(“c_00”, subcomponent, “c_05”));

These statements reside at a meta-level with respect to design object description statements. The second argument of each statement expresses relationships within the design object description.

The changes in the material representation of the multi-agent system are transferred from the agent specific task of agent A to the world interaction management task and to the external world. The modification action itself is derived by the component world interaction management.

Example Effectuation action for modifying the multi-agent system.

Within the world interaction management of agent_A, an action is formulated to effectuate the modification of the current multi-agent system:

to_be_performed(modify_according_to(“dod_55”));

5.2 Execution of the creation action in the external world

To be able to execute the creation action in the external world, the external world needs to have certain properties. These properties are related to how “equipped” the world is to handle interaction with agents. There are two generic properties needed for the interaction of agents with the external world: the processing of observations and the processing of actions. Observation of the external world was needed to inform agent A of the current material representation of the multi-agent system, see Section 4.1.

The property of processing actions can be refined into the properties:

- the external world can receive initiated actions, and the related information, and
- the external world can perform actions (effectuation of the physical effects of actions).

To change the number of agents and their characteristics, the external world has to adapt the executable specification of that system while the system is running. This implies that the parts of the system that are affected by the modifications need to be interrupted, their information states stored, after which the executable specification of those parts need to be modified, and the modified system need to be reactivated with the correct information states.

Example Result of the effectuation action.

The external world world_W effectuates the creation action and modifies the multi-agent system according to the given modifications.

As an agent in the multi-agent system, agent D and receives a request from agent A: would it like to find out more about YYY?. The agent D gathers information on subject YYY by initiating observations in the world W, and interpreting the observation results. Once the answer is found, agent D reports its findings to agent A. Agent A can then which finally answer the question of the client.

6 Discussion

Research within multi-agent systems research has focussed on the behaviour of individual agents and their interaction. The dynamic creation of new agents within an existing multi-agent system, on the basis of the identification of newly required functionality and behaviour, is an area on which little research has focussed. Most of the research in the area of dynamic agent creation is based on a genetic programming approach; e.g., (Cetnarowicz, Kisiel-Dorohinicki and Nawarecki, 1996; Numaoka, 1996). The approach taken in this paper is that to create new agents, an existing agent must be capable of designing a new agent on the basis of a model for design and then be capable of bringing this agent to life.

To design an agent capable of designing another agent, insight is required in the type of agent to be designed. In this paper a compositional approach to agent design has been followed. An agent's abilities are related to the tasks an agent is able to perform. These abilities are the means with which both the existing agents' abilities are expressed. In addition, the properties of the multi-agent system and the external world are of importance. As such, this work is related to the properties distinguished with respect to problem solving methods (Benjamins, Fensel and stratman, 1996; Breuker, 1997; Fensel, Motta, Decker, and Zdrahal, 1997). Within the field of Knowledge Engineering properties of problem solving methods are used to support knowledge engineers during the design process: providing a means to describe existing generic components that may be used, modified or refined during a design process, depending on their applicability in a given situation. The Knowledge Engineering community has not focussed on abilities and properties of agents and their interaction, as was done in this paper.

The architecture of the design agent is based on an existing generic agent model, and includes a refinement of a generic model of design. It combines results from the area of Multi-Agent Systems and the area of AI and Design. The approach described has been formalised: the initial multi-agent system described in this system has been specified and implemented, using the automated implementation generator within the DESIRE software environment, as have the design agent, the new agent and its creation within the new multi-agent system. The formal agent model presented in this paper includes formalisations of agent design descriptions and requirements on agents within an agent, and formalisations of agent design knowledge.

One aspect of the approach described in this paper is that a design agent not only designs another agent and the implications for the integration of the agent in an existing system at a conceptual level, the design agent also actually creates the new agent dynamically. In fact, a design agent could re-design (parts of) itself in the same manner. The integration of re-design on a conceptual and logical level and run-time modification of the system at the implementation level is an important distinguishing aspect of the approach presented in this paper. This is in contrast to, on the one hand, conceptual and logical approaches for which no direct connection to executable code exists, and, on the other hand, to approaches that address agent creation at an implementation level.

Acknowledgements

The authors wish to thank Pieter van Langen for his contributions to the generic model of design, and Lourens van der Meij and Frank Cornelissen for their support of the DESIRE modeling and specification tools. This research has been (partially) supported by NWO-SION within project 612-322-316: "Evolutionary design in knowledge-based systems" (REVISE).

References

Benjamins, R., Fensel, D., and Straatman, R. (1996). Assumptions of Problem-Solving Methods and their Role in Knowledge Engineering. In: Wahlster, W. (ed.). *Proceedings European Conference on AI (ECAI '96)*, Wiley and Sons, Chichester.

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: Lesser, V. (Ed.), *Proceedings First International Conference on Multi-Agent Systems, ICMAS'95*, MIT Press, pp. 25-32. Extended version in: Huhns, M. and Singh, M. (Eds.), *International Journal of Co-operative Information Systems, IJCIS* vol. 6 (1), special issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems, pp. 67-94.

BRAZIER, F.M.T., JONKER, C.M., TREUR, J. and WIJNGAARDS, N.J.E. (1997). The Role of Abilities of Agents in Re-design. Submitted to: KAW'98, Gaines, B. and Musen, M. (Eds). 11th knowledge acquisition workshop. URL: `http://www.cs.vu.nl/~niek/kaw98/BrazierJonkerTreurWijngaards.html`.

Brazier, F.M.T., Langen, P.H.G. van, Ruttkay Zs., and Treur, J. (1994). On formal specification of design tasks. In Gero, J.S., and Sudweeks, F. (eds.) *Artificial Intelligence in Design '94* Kluwer Academic Publishers, Dordrecht, pp. 535-552.

Brazier, F.M.T., Langen, P.H.G. van, and Treur, J. (1996). A logical theory of design. In: *Advances in Formal Design Methods for CAD*, J.S. Gero (ed.), Chapmann & Hall, New York, 1996, pp. 243-266.

Brazier, F.M.T., Langen, P.H.G. van, Treur, J., Wijngaards, N.J.E. and Willems, M. (1996). Modelling an elevator design task in Desire: the VT example. In: Schreiber, A.Th., and Birmingham, W.P. (Eds.), Special Issue on Sisyphus-VT. *International Journal of Human-Computer Studies*, 1996, **44**, pp. 469-520.

Brazier, F. M. T., Treur, J., and Wijngaards, N. J. E. (1996b). Interaction with experts: the role of a shared task model. In: Wahlster, W. (ed.). *Proceedings European Conference on AI (ECAI '96)*, pp. 241–245. Wiley and Sons, Chichester.

Breuker, J.A. (1997). Problems in indexing problem solving methods. In: FENSEL, D. (ed.) *Proceedings of the Problem Solving Methods for Knowledge Based Systems workshop*, IJCAI'97, pp. 19-35.

Cetnarowicz, K., Kisiel-Dorohinicki, M., and Nawarecki, E. (1996). The Application of Evolution Process in Multi-Agent World to the Prediction System. In: Tokoro, M. (Ed.) *Proceedings of the Second International Conference on Multi-Agent Systems*, ICMAS'96, AAAI PRESS, MENLO PARK CA, PP. 26-32.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., and Gero, J.S. (1990). *Knowledge-based design systems* Addison-Wesley Publishing Company, Reading.

Fensel, D., Motta, E., Decker, S., and Zdrahal, Z. (1997). Using Ontologies for Defining Tasks, Problem-Solving Methods and their Mappings. In: Plaza, E. and Benjamins, R. (Eds.). *Knowledge Acquisition, Modeling and Management*, proceedings of the 10th European workshop, EKAW'97, Lecture Notes in Artificial Intelligence, **1319**, Springer, Berlin, pp. 113-128.

Jonker, C.M., and J. Treur (1997). Modelling an Agent's Mind and Matter. In: Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'97, Lecture Notes in AI, vol. 1237, Springer Verlag, pp. 210-233.

Numaoka, C. (1996). Bacterial Evolution Algorithm for Rapid Adaptation. In: Van de Velde, W. and Perram, J.W. (Eds.) *Proceedings of the 7th European Workshop on Modelling*

Autonomous Agents in a Multi-Agent World, MAAMAW'96, Lecture Notes in Artificial Intelligence, **1038**, Springer, pp. 139-148.

Smith, I.F.C., and Boulanger, S. (1994). *Knowledge representation for preliminary stages of engineering tasks* Knowledge Based Systems, **7**, pp. 161-168.

Smithers, T. (1994). On knowledge level theories of design process. In Gero, J.S., and Sudweeks, F. (eds.) *Artificial Intelligence in Design '96* Kluwer Academic Publishers, Dordrecht, pp. 561-579

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