Compositional Design of a Generic Design Agent

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

Vrije Universiteit Amsterdam, Faculty of Sciences, Department of Artificial Intelligence
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 a design agent, to be used in an Internet environment. The design agent 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. The generic architecture has been designed using the compositional development method DESIRE, and has been used to develop a prototype design agent for automated agent design.

Keywords: design model(s), design automation, software design, artificial evolution, computational model

Design is a task often performed by one or more specialised (human) agents. Architects are, for example, specialised agents: their area of expertise is the design of buildings in given surroundings. Design agents negotiate with other agents on requirements and generate one or more designs (design object descriptions) on the basis of these requirements and additional information received from other agents. Design agents make these design object descriptions (together with all other (intermediate) results of the design process) available to other agents in the course of the design process, and react to input provided by other agents as a result.

The World Wide Web provides a rich potential for distributed design in which designers and other parties interact. The number of possibilities is immense: designers can navigate through component libraries available on the Web[20], they can choose between a large number informal textual and formal types of knowledge representation (and be supported in the process), they can interact with other parties interested in the same Web sites as they themselves[36], they can perform calculations, to name a few.

This paper introduces a generic architecture for a design agent that can be integrated in a Web-based distributed design environment. This architecture is based on the Generic Agent Model GAM introduced in [11] and the Generic Design Model described in [13]. Both of these models have been devised on the basis of experience, and tested in a number of different domains: applications of GAM can be found, for example, in [8, 9, 10]; applications of GDM can be found, for example, in [16, 17]. To tune a generic model to a specific domain of application more refined knowledge is needed: both more specific knowledge of the task at hand (by specialisation), and more specific factual knowledge of the domain (by instantiation). For example, domain-specific knowledge is needed 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

The architecture for the design agent has been modelled using the compositional development method for multi-agent systems DESIRE[8]. This compositional development method is briefly introduced in Section 1. The generic agent model employed is described in Section 2. The generic model of design[13] employed in which strategic reasoning and dynamic management of requirements are explicitly modelled, is described in Section 3. The integration of the two models is discussed in Section 4. Although both the Generic Agent Model GAM and the Generic Design Model GDM have been tested in different application domains, the architecture for a generic design agent described in this paper as yet has only been tested in the application domain of automated design of Internet agents. Therefore this application domain is used to illustrate applicability of the proposed approach. The specific design agent for this application is described in Section 5. Section 6 compares the generic agent model GAM and the generic design model GDM to other agent and design models. Section 7 discusses the results and sketches a perspective of the use of the design agent architecture in Web-based environments for Electronic Commerce applications: a current focus of research.

1 Compositional Design of Agents

The design agent described in this paper has been developed using the compositional development method DESIRE for multi-agent systems (Design and Specification of Interacting Reasoning components; cf. [8]). Within this method knowledge of the following three types is distinguished:

- process composition
- knowledge composition
- the relation between process composition and knowledge composition

The development of a multi-agent system is supported by graphical design tools. Translation to an operational system is straightforward; in addition to the graphical design tools the software environment includes implementation generators with which specifications can be translated into executable code of a prototype system. The three types of knowledge are discussed in more detail below.

1.1 Process Composition

Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of (is composed of) lower level processes.

1.1.1 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 defined. 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.

1.1.2 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).

1.2 Knowledge Composition

Knowledge composition identifies the knowledge structures at different levels of (knowledge) abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction levels may correspond to the process abstraction levels, but this is often not the case.

1.2.1 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 graphically be represented on the basis of conceptual graphs and logically 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 if-then rules.

1.2.2 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.

1.3 Relation between Process Composition and Knowledge Composition

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

2 A Generic Agent Model

Agents are often designed to perform their own specific tasks, for example the design of an artefact. In addition, a number of generic agent tasks can be identified. This section describes the Generic Agent Model GAM introduced in [11], 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 [63], which distinguishes the following agent abilities:

- autonomy
- reactiveness
- pro-activeness
- social abilities.

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.

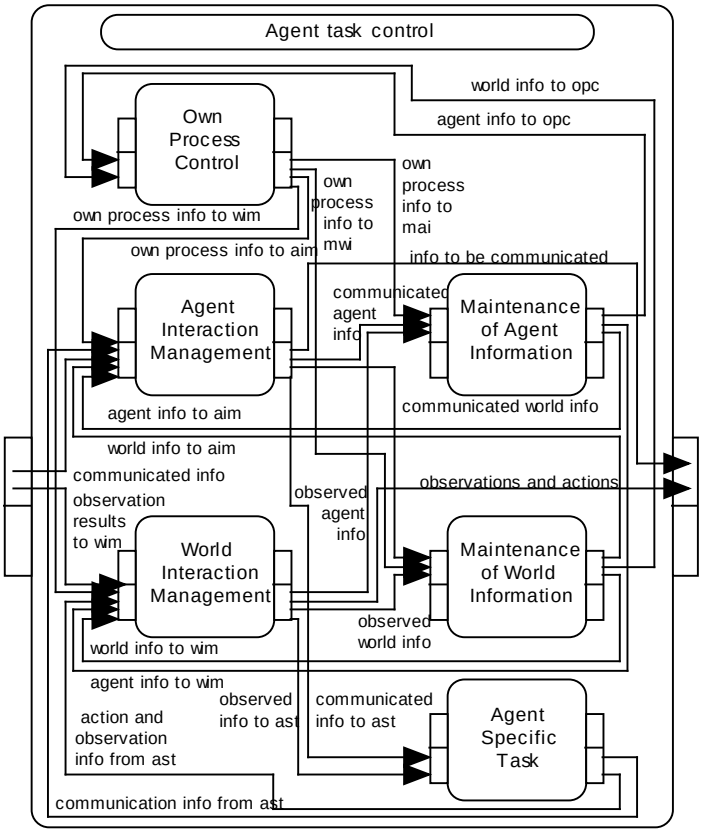

Figure 1 Generic Agent Model

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, OPC). Reactiveness and social ability are related to the ability to be able to communicate with other agents (agent interaction management, AIM) and to interact with the external world (world interaction management, WIM). 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, MWI) and other agents (maintenance of agent information, MAI). The generic agent model also includes an empty generic component to model the agent specific task (AST). In addition, a component cooperation management (CM) can be used within GAM, but this was not used in this paper. The tasks related to the generic abilities and agent specific tasks may be modelled by components within an agent as depicted in Figure 1. In addition to the sub-components, the model includes information links that specify which information is exchanged between components; these information links are named.

The exchange of information within GAM can be described as follows. Observation results received by the agent 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 observation initiatives (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 observed world information from the component world interaction management, through the information link observed world info and communicated world information (through the link communicated world info) from the component agent interaction management. Information from maintenance of world information, is transferred to 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 communicated information on other agents from the component agent interaction management, through the information link communicated agent info and observed agent information (through the link observed agent info) from the component world interaction management. Information, maintained in the component maintenance of agent information, becomes input 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.

As an illustrative example of the internal behaviour of an agent based on GAM, a typical pattern to process incoming observation results is the following:

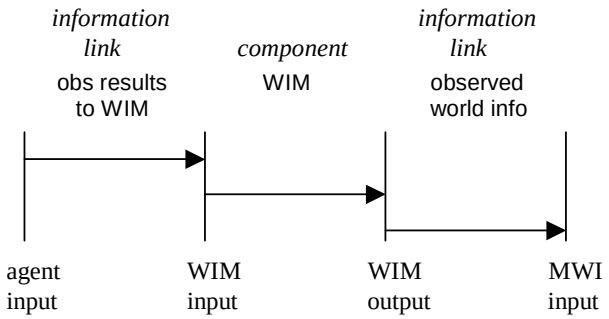

Figure 2. Typical pattern to process incoming observation results.

The information about the observation, for example of the form

observation_result(car_present, pos)

received at the agent's input interface, is transferred to the component WIM. Within WIM the world information

car_present

is extracted from the observation, and prepared to be stored. This world information then is transferred to MWI, where it is stored as beliefs about the world. In a similar pattern incoming communication is processed, starting, for example with

communicated_by(car_present, pos, agent_A)

at the agent's input interface. In this case within the component AIM the content (world) information is extracted from the communication information, possibly taking into account credibility of the other agent. Analogously other patterns can be described on the basis of Figure 1.

3 A Generic Model of Design

The generic model of a design agent is based on both the generic agent model discussed in Section 2, 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 agent is described.

A *Generic Design Model* GDM, 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 [13]. This model is based on a logical analysis of design processes [14] and on analyses of applications, including elevator configuration[16] and design of environmental measures[17]. The model not only provides an abstract description of a design process comparable to a *design model*, e.g., [26, 51], 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 set of requirements. This set 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.

During design the considered sets of requirements may change as may the design object descriptions: they evolve during design. The strategy employed for the co-ordination 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 [49]), or the result of interaction with an outside party (e.g., a client or a designer).

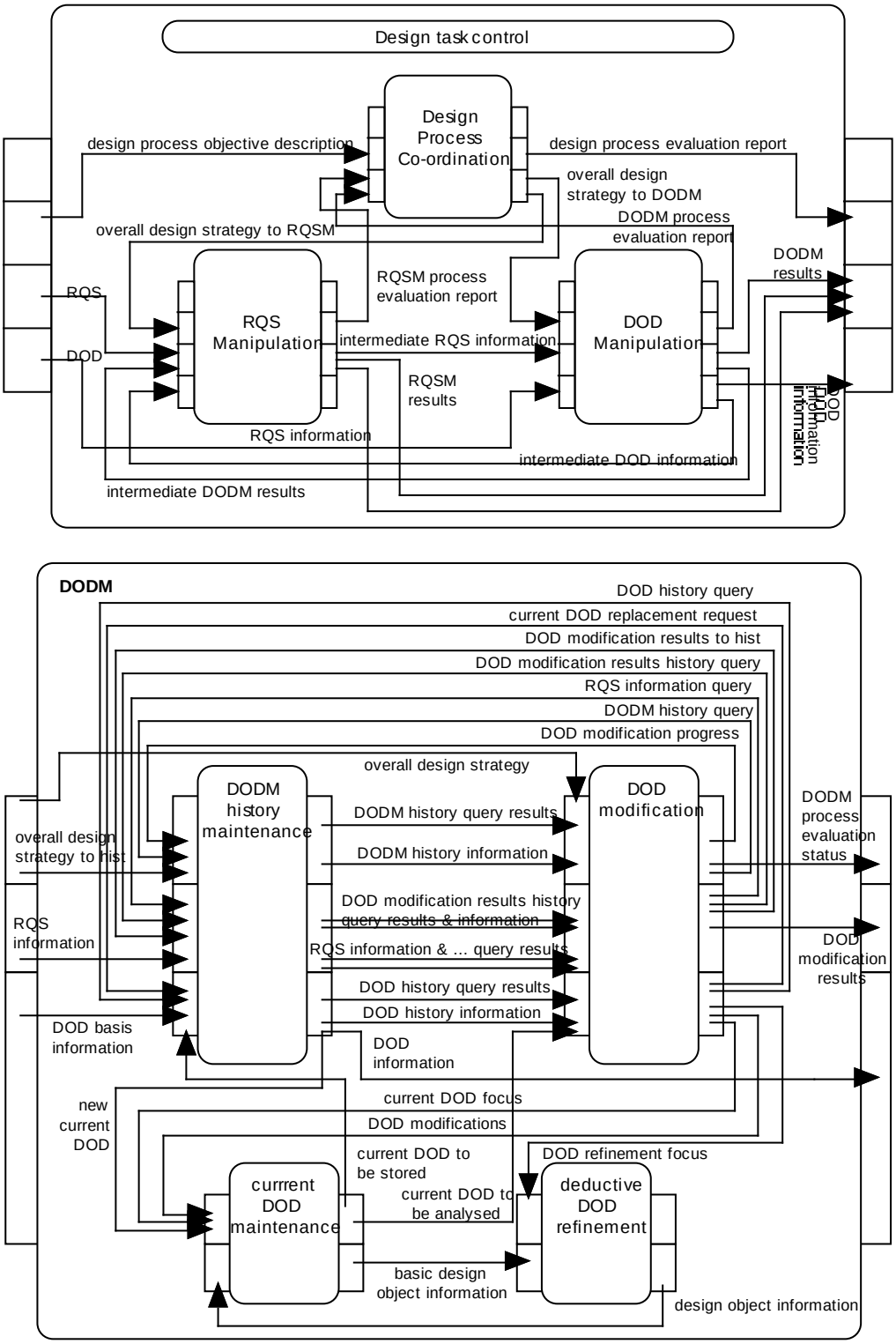

Figure 3 Generic Design Model

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. Four processes and information exchange are shown at the second level for DODM.

The four processes (see Figure 3) related to the process *requirement qualification set manipulation (RQSM)* 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 (DODM)* 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 co-ordination* is composed in a similar manner.

The processes in a manipulation process are often activated in a typical pattern. Figure 4 shows such a typical example in which the process design process co-ordination issues a design strategy for the requirement qualification set manipulation process in the situation that no information from design object determination manipulation is present and an initial RQS is already stored in the RQSM history maintenance process (e.g. during the initialisation phase). Within the RQSM process, first RQS modification is activated, and based on the given design strategy, it consults the history, maintained by RQSM history maintenance. Historical information is transferred to the RQS modification process, and/or the contents of a specific RQS is to be used as the current RQS via the process current RQS maintenance. The RQS modification process typically first checks for specific properties in the current RQS by means of the deductive RQS refinement process, after which modifications to the current RQS can be processed by the process current RQS maintenance. The resulting, modified, RQS is stored in the history, after which the RQS modification process can again consult the history and decide with which RQS to continue. Eventually, the RQS modification process decides when to stop modification, and report on progress achieved while working on the given design strategy.

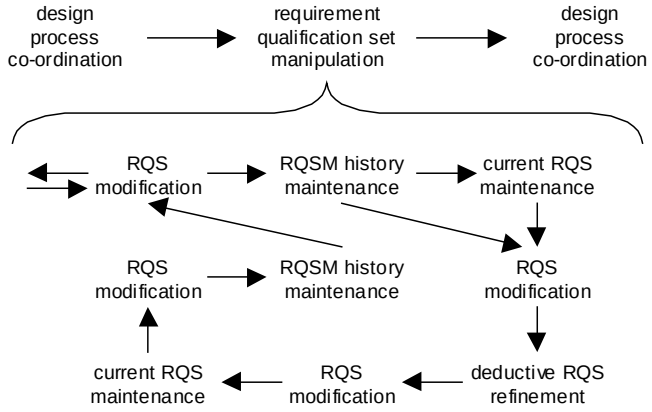

Figure 4. An example of process flow in requirement qualification set manipulation.

4 Design of the Generic Design Agent

As stated in the introduction of this paper, the architecture of the design agent is based on the two models described above (the generic agent model and the model of design). The design of this architecture is addressed in this section, starting with the requirements imposed on this process and the construction of the model.

4.1 Requirements on a design agent

Requirements on a generic model for a design agent can be divided into two categories: requirements on the 'agent' properties of the design agent, and requirements on the integration of the 'design properties' in the design agent.

A generic design agent needs to be:

- *capable of bi-directional communication.* A design agent has to be able to bi-directionally communicate about information needed by, or resulting from, a design activity.
- *capable of world interaction.* A design agent has to be able to interact in the material world to observe (or provide) information needed by (or resulting from) a design activity.
- *capable of co-operation.* A design agent has to be able to co-operate (and, e.g., to negotiate) on a design activity.
- *capable of agent own process control.* A design agent has to be able to monitor and plan its own processes, including the design process.

Please note that the agent's own process control does *not* cover control of the processes *inside* the design process but only determines design process objectives.

The desired properties of the design model are:

- Explicit distinction between the manipulation of design object descriptions, manipulation of sets of qualified requirements, and co-ordination of the design process.
- Explicit representation and manipulation of design process objectives.
- Explicit representation and manipulation of (sets of) qualified requirements.
- Explicit representation and manipulation of design object descriptions.

Requirements on the integration of a process of design within an agent model are:

- *The design process is to be modelled within the agent as one of its (possible) capabilities.* The ability to perform a design process is to be modelled as (one of) the agent's specific task.
- *Information needed for the design process can be acquired via communication or world interaction.* Information on which a specific design process is based (design process objectives, sets of qualified requirements, design object descriptions) can be acquired by two means: by communication, or by observation in the material world.
- *Information resulting from the design process can be made available via communication or world interaction.* All types of information resulting from a design process (design object description information, requirement qualification set information, process results, design process evaluation status) can be made available by two means: by communication, or by actions in the material world.

4.2 Constructing a generic model for a design agent

The generic model for a design agent has been constructed by combining two existing generic models: a generic design model (Section 3) and a generic agent model (Section 2). The resulting model is described on the basis of: the process composition, knowledge composition, and the relation between process composition and knowledge composition.

The design process has been positioned within the agent specific task. The interfaces of sub-processes in the agent model are adapted to accommodate design process related information. Two modelling options are possible to facilitate information exchange at different meta-levels between processes within the agent and the design component within the agent specific task. The design component can be modified to translate information needed

for, or provided by, the design process, or the interface of the agent specific task can be modified to accommodate information at a number of meta-levels. A (preferred) minimal change to the 'agent' part of the design agent is achieved by adopting the first solution: by using information links to transfer information between meta-levels. Some modifications are, however, needed to information types in levels distinguished in the interface of the design process. Information links are defined within agent specific task and the design process, and task control knowledge is added in the component agent specific task for activation of the design process.

The knowledge composition of the generic design agent includes information types, knowledge bases, and levels of knowledge abstraction. The information types in the generic model for the agent and the generic model for design are also in the model for the design agent. In addition, information types are used to 'connect' the information from the generic model for the agent and the generic model for design.

The information types related to the Agent Specific Task and Design have been specified, and the relation between other processes (in both the agent and design) and knowledge structures were not modified. The generic model for a design agent, has been refined for the application domain addressed: design of compositional systems. The design process within the generic design agent is refined for the design of compositional systems in Section 5.

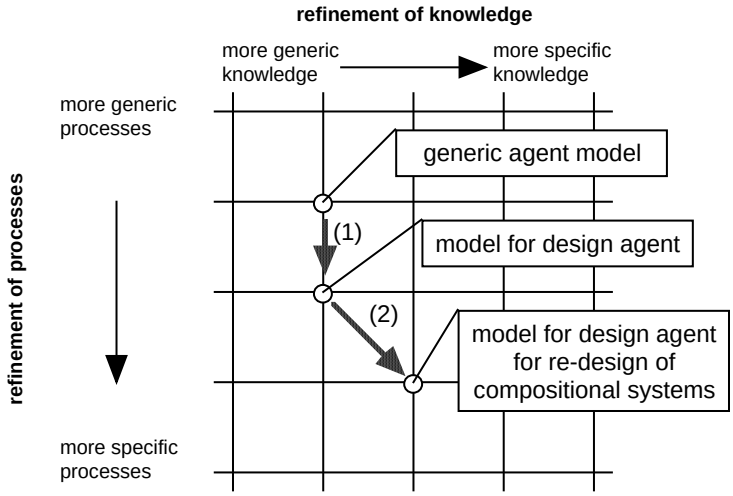

Figure 5 Overview of refinement relations

4.3 Overview of the constructed design agent models

The generic model for a *design agent* is generic with respect to its domain of application, yet is specific with respect to the processes distinguished within the agent (as compared to the generic *agent* model). The model for a design agent for design of compositional systems is specific with respect to the domain of *design processes*, and specific with respect to the distinguished knowledge structures. These transitions in two dimensions of genericity (process vs. knowledge) are depicted in Figure 5. The numbered arrows correspond to the two phases during the refinement of the generic agent model. The 'degree of genericity' of the three models placed in the matrix in Figure 5 differs, as can be inferred from the position of these three models.

The construction process for the model of the generic design agent was straightforward. The positioning of the design process within the agent was relatively simple: minimal changes to the agent model implies that additional components are placed *within* existing components of the agent. The only serious work was encountered in the mappings between information at one meta-level for agent processes and information at three meta-levels for the process of design. The second step in the process, refining the model to a specific application domain will be addressed in Section 5.

5 Application: a Specific Design Agent for Agent Design

The generic model of a design agent described above can, in principle, be used for any domain of application. To obtain a proof of concept this model has been applied to a specific domain, namely the domain of Internet agent design. For this domain a prototype application has been designed and implemented. The refinement of the process composition is described in Section 5.1 (specialisation). The refinement of the knowledge composition by specific knowledge for the application domain is addressed in Section 5.2 (instantiation).

5.1 Refinement of the Design Model

In this section a refinement of the design model (devised for the design of compositional systems) is described: the process of requirement qualification set modification and the process of design object description modification are defined in more detail (specialisation of the two processes)

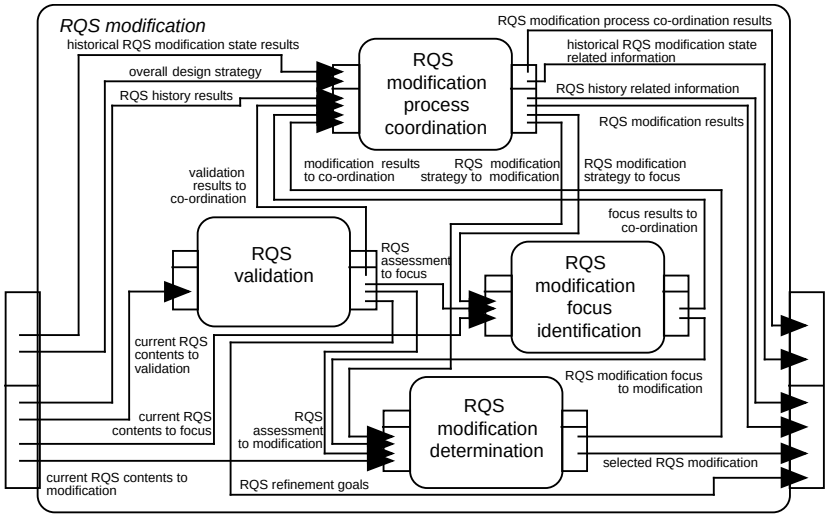

Figure 6 Specialisation of RQS modification

5.1.1 Specialisation of requirement qualification set modification

The process RQS modification determines modifications to a requirement qualification set (RQS). To this purpose a number of sub-processes are distinguished as shown in Figure 6. The process RQS modification process co-ordination is responsible for the co-ordination of the entire process within RQSM: this process determines whether, when and by which means a specific RQS is to be modified.
The global phases within RQS modification resemble a process control model (e.g., controlling a chemical process). A process control task usually relies on a feedback loop within which sub-tasks such as analysis, planning, and execution are distinguished.

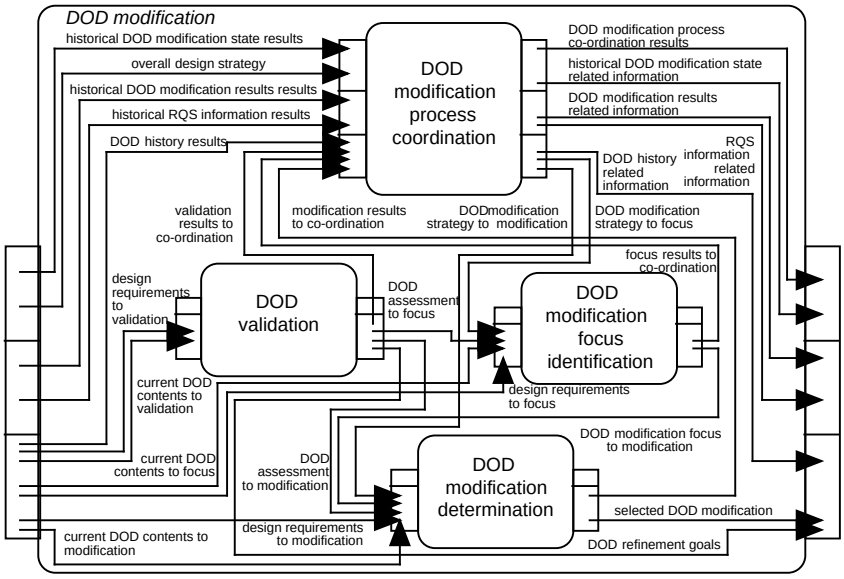

Figure 7 Specialisation of DOD modification

Similarly, within RQS modification analysis is performed by RQS validation, planning is performed by RQS modification focus identification and RQS modification determination, and execution is performed by effectuating modifications to a RQS, resulting in a new RQS in current RQS maintenance.

5.1.2 Specialisation of design object description modification

The process DOD modification determines modifications to a design object description (DOD) such that a DOD is constructed that adheres to the design requirements given to DODM. To this purpose a number of sub-processes are distinguished, as shown in Figure 7. The process DOD modification process co-ordination is responsible for the co-ordination of the entire process within DODM: this process determines whether, when and by which means a particular DOD is to be modified.

The global phases within DOD modification also resemble a process control model: analysis, planning, execution. Similarly, within DOD modification analysis is performed by DOD validation (including assessing requirements in the current DOD), planning is performed by DOD modification focus identification and DOD modification determination, and execution is performed by effectuating modifications to a DOD, resulting in a new DOD in current DOD maintenance.

5.2 Refinement of the knowledge composition

In this section the instantiation of the model by application dependent knowledge is addressed. Ontologies and knowledge are defined to describe requirements (Section 5.2.1) and design object descriptions (Section 5.2.2) in the domain of Internet agent design. A trace of the design process is also presented in Section 5.2.2.

5.2.1 Requirements and knowledge on properties of agents

For this prototype system an ontology of requirements on agents has been developed. Moreover, knowledge has been identified 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.

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.

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. In Figure 6 the ability of bi-directional communication and its refinements are depicted. For a description of other agent abilities see Brazier, Jonker, Treur and Wijngaards (1998).

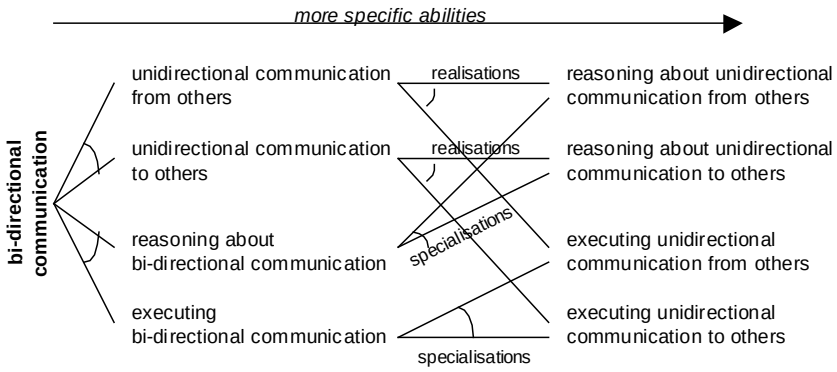

Figure 8 Refinements of the ability of bi-directional communication

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 related to the effectuation of the ability).

Figure 8 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

```
if      is_qualified_requirement_selected_as_focus( QR: qualified_requirement_name )
  and     holds( is_qualified_requirement(      QR: qualified_requirement_name,
                                                Q: requirement_qualification,
                                                R: requirement_name )
  and     holds( refers_to_requirement( R: requirement_name,
                                                has_property( A: agent_name,
                                                        is_capable_of_bidirectional_communication_
                                                        with( A2: agent_name )) ),
                                                pos )
  and     refinement_alternative( specialisations )

then    addition_to_current_RQS(
        is_qualified_requirement(   new_name( QR: qualified_requirement_name, a ),
                                                Q: requirement_qualification,
```

```
                                        new_name( R: requirement_name, a ) ) )
and         addition_to_current_RQS(
    refers_to_requirement(      new_name( R: requirement_name, a ),
                                        has_property( A: agent_name,
                                                is_capable_of_unidirectional_communication_
                                                from( A2: agent_name ) ) )
and         addition_to_current_RQS(
    is_qualified_requirement(   new_name( QR : qualified_requirement_name, b ),
                                        Q: requirement_qualification,
                                        new_name( R: requirement_name, b ) ) )

and         addition_to_current_RQS(
    refers_to_requirement(      new_name( R: requirement_name, b ),
                                        has_property( A: agent_name,
                                                is_capable_of_unidirectional_communication_
                                                        to( A2: agent_name ) ) )
and         addition_to_current_RQS(
    is_qualified_requirement(   new_name( QR: qualified_requirement_name, c ),
                                        Q: requirement_qualification,
                                        new_name( R: requirement_name, c ) ) )
and         addition_to_current_RQS(
    refers_to_requirement(      new_name( R: requirement_name, c ),
                                        has_property( A: agent_name,
                                        is_capable_of_combining_unidirectional_
                                        communication_from_and_to( A2: agent_name ) ) );
```

Top-level requirements are refined into more specific requirements during a design process. The result is the construction of a specific hierarchy of requirements, which adheres to the requirements ontology and refinement knowledge. Figure 9 shows an example of (part of) such a requirements refinement hierarchy. The current prototype design agent makes extensive use of the requirements ontology, generic models and design object building blocks. The design process is fairly linear, in the sense that few options are generated and selected. The most refined requirements are almost directly operationalisable by building blocks for design object descriptions. A specific design requirement, currently in focus in DOD modification, is broken up (i.e., refined) into smaller properties: assessment points. These assessment points can be tested for, and when not yet realised, building blocks related to an assessment point can be added to the current design object description.

The generation of options for sets of qualified requirements and design object descriptions involving explicit strategic knowledge can be incorporated in the design model, as described by (Brazier, Langen, and Treur, 1998).

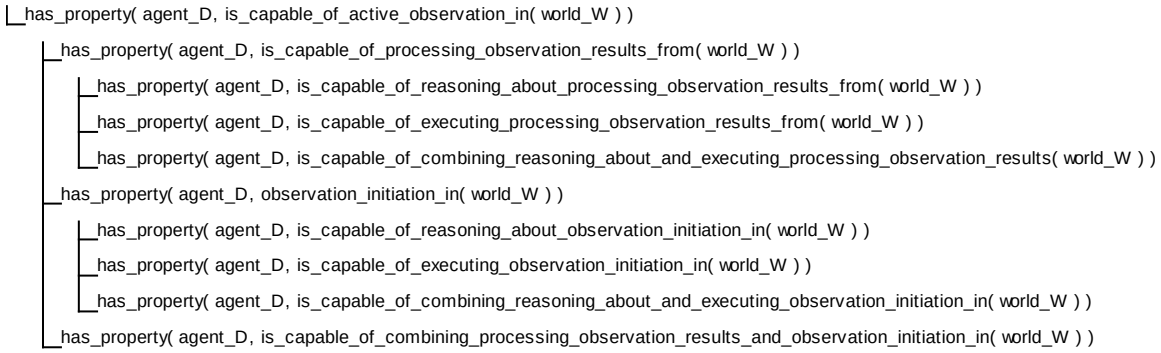

Figure 9 Requirement refinement hierarchy constructed by the prototype design agent.

The implication of designing (parts of) a multi-agent system, is that a multi-agent system is the object of design, and as such should be formally represented in a design object description. In this application 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 compositional formal specification language underlying DESIRE forms an adequate basis for such a design object description representation.

5.2.2 Example trace of the design of an agent

In this section, as an illustration an example trace of the application of the generic design agent to Internet agent design is shown. Within this trace it is shown how design object descriptions and domain knowledge related to design object descriptions are represented for the application domain.

Prerequisites for design

The Design Agent receives the following initial requirements for a new agent:

```
is_qualified_requirement( qr_m1, hard, r_m1 );
refers_to_requirement( r_m1, has_property(
                                    mas_S,
                                    is_capable_of_distributed_information_gathering(
                                        agent_A, agent_D, world_W ) ) );

is_qualified_requirement( qr_m2, hard, r_m2 );
refers_to_requirement( r_m2, has_property(
                                    agent_D,
                                    is_capable_of_information_information_gatherering_for(
                                        agent_A, scientific_publications ) ) );
```

These requirements state that the new agent should gather information. The new agent's specific subject of expertise is that, it should be capable of gathering information on scientific publications (e.g. on the Internet). The design agent commences a design process on the basis of these requirements.

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. Knowledge on refinements of the ability of bi-directional communication can be formally represented (see the example knowledge in Section 5.1). Meta-reasoning is employed to decide which refinement alternative should be employed for which ability.

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

Manipulation of requirements

On the basis of the given requirements, more refined requirements can be formulated. For the first qualified requirement qr_m1, refinement knowledge is applied which results in the property refinement graphs of which an example is depicted in Figure 9. Requirements on these refined properties are used to construct a design object description.

The representation of requirements on compositional systems has been briefly shown. Representations of design object descriptions for a compositional agent is presented below. Moreover, knowledge that can be used to derive properties of the design, for example the required properties, is presented. The description of the compositional system is augmented with a description relating existing structures to generic models. This provides valuable information for the identification of abilities and properties.

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, attributes, and relations is used. Part of the top level of the multi-agent system can be represented as follows:

```
is_top_level(c_00 );
corresponds_with(c_00, agent_D );
has_characterisation( c_00, generic, agent );
corresponds_with( lm_01, active_observations );
has_subcomponent( c_00, c_01 );
has_subcomponent( c_00, c_04 );
has_information_link( c_00, lm_01 );
```

has_source_component(lm_01, c_01);
has_destination_component(lm_01, 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.

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

Modifications within the agent D

During the design process several descriptions of agents are proposed. For example, an agent D 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'.

Assessment points are more specific properties of a design object description, that are related to the current design requirement in focus, yet are simpler to realise ('satisfy') than the design requirement in focus. The design requirement that refers to the property is capable of bi-directional communication is related to a number of assessment points, among which: a component is present for bi-directional communication, and private information links are present for bi-directional communication. A strict order is imposed on the realisation of the properties to which assessment points refer so that consequences of the realisation of the properties to which one assessment point refers can aid the realisation of the properties to which another assessment point refers. Below two knowledge elements are presented which relate assessment points which need to be realised to modifications of the current design object description.

To realise the assessment point a component is present for bi-directional communication for C: component_name, a new component is introduced (as a sub-component of the C: component_name), this component is characterised as being a component for bi-directional communication, and this component is made 'awake' by task control in C: component_name.

if assessment_point_to_be_realised(
has_property(A: component_name,
a_component_is_present_for_bidirectional_communication_
with(A2: component_name)))
and current_DOD_contents(has_task_control(
A: component_name,
T: task_control_kb_name), pos)
then addition_to_current_DOD(is_component(
new_name(A: component_name, AIM)))
and addition_to_current_DOD(has_subcomponent(
A: component_name,
new_name(A: component_name, AIM)))
and addition_to_current_DOD(has_characterisation(
new_name(A: component_name, AIM),
generic,
bidirectional_communication))
and addition_to_current_DOD(has_characterisation(
new_name(A: component_name, AIM),
specific,
bidirectional_communication_with(A2: component_name)))
and addition_to_current_DOD(has_part(
T: task_control_kb_name,
makes_awake(new_name(A: component_name, AIM))));

The second knowledge element requires that the sub-component which is to realise bi-directional communication is already present to formulate information links. To realise the assessment point private information links are present for bi-directional communication with for C: component_name, two information links are introduced, both private information links of C: component_name. These information links connect A: component_name with the sub-component which is to realise bi-directional communication, and these information links are made 'awake' by task control in C: component_name.

```
if   assessment_point_to_be_realised(
          has_property( C: component_name,
               private_information_links_are_present_for_bidirectional_communication_
                    with( C2: component_name ) ) )
     and  current_DOD_contents(   is_component(
                                  C: component_name ),
                                  pos )
     and  current_DOD_contents(   has_interface_information_type(
                                  C: component_name,
                                  input_interface,
                                  Cin: information_type_name ),
                                  pos )
     and  current_DOD_contents(   has_interface_information_type(
                                  C: component_name,
                                  output_interface,
                                  Cout: information_type_name ),
                                  pos )
     and  current_DOD_contents(   has_subcomponent(
                                  C: component_name,
                                  D: component_name ),
                                  pos )
     and  current_DOD_contents(   has_characterisation(
                                  D: component_name,
                                  bidirectional_communication ),
                                  pos )
     and  current_DOD_contents(   has_characterisation(
                                  D: component_name,
                                  bidirectional_communication_with( C2: component_name ) ),
                                  pos )
     and  current_DOD_contents(   has_interface_information_type(
                                  D: component_name,
                                  input_interface,
                                  Din: information_type_name ),
                                  pos )
     and  current_DOD_contents(   has_interface_information_type(
                                  D: component_name,
                                  output_interface,
                                  Dout: information_type_name ),
                                  pos )
     and  current_DOD_contents(   has_task_control(
                                  C: component_name,
                                  T: task_control_kb_name ),
                                  pos )
then      addition_to_current_DOD( is_information_link(
                                  new_name( C: component_name, incoming_comm ) ) )
     and  addition_to_current_DOD( has_information_link(
                                  C: component_name,
                                  new_name( C: component_name, incoming_comm ) ) )
     and  addition_to_current_DOD( has_source_component(
                                  new_name( C: component_name, incoming_comm ),
                                  C: component_name ) )
     and  addition_to_current_DOD( has_source_information_type (
                                  new_name( C: component_name, incoming_comm ),
                                  Cin: information_type_name ) )
     and  addition_to_current_DOD( has_destination_component(
                                  new_name( C: component_name, incoming_comm ),
                                  D: component_name ) )
     and  addition_to_current_DOD( has_destination_information_type (
                                  new_name( C: component_name, incoming_comm ),
                                  Din: information_type_name ) )
     and  addition_to_current_DOD( has_part(
                                  T: task_control_kb_name,
```

```
                                    makes_awake( new_name( C: component_name, incoming_comm
) ) ) )
and   addition_to_current_DOD( is_information_link(new_name( C: component_name,
outgoing_comm ) ) )
and   addition_to_current_DOD( has_information_link(
                                    C: component_name,
                                    new_name( C: component_name, outgoing_comm ) ) )
and   addition_to_current_DOD( has_source_component(
                                    new_name( C: component_name, outgoing_comm ),
                                    D: component_name ) )
and   addition_to_current_DOD( has_source_information_type (
                                    new_name( C: component_name, incoming_comm ),
                                    Dout: information_type_name ) )
and   addition_to_current_DOD( has_destination_component(
                                    new_name( C: component_name, outgoing_comm ),
                                    C: component_name ) )
and   addition_to_current_DOD( has_destination_information_type (
                                    new_name( C: component_name, incoming_comm ),
                                    Cout: information_type_name ) )
and   addition_to_current_DOD( has_part(
                                    T: task_control_kb_name,
                                    makes_awake( new_name( C: component_name, outgoing_comm
) ) ) );
```

A ‘new’ agent D is shown in Figure 10 in which both abilities are incorporated, as required.

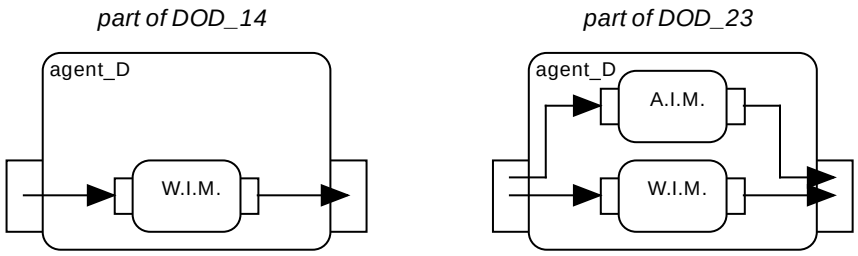

Figure 10 Possible design object descriptions (focused on composition of agent_D).

Knowledge is employed 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.

Identification of an ability

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

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

```
if   is_component( C: component_name )
     and   has_characterisation(      C: component_name,
                                      agent )
     and   has_interface_information_type(  C: component_name,
                                            output_interface,
                                            Cout: information_type_name )
     and   is_component(              D: component_name )
```

and has_subcomponent(C: component_name,
D: component_name)
and has_characterisation(D: component_name,
world_interaction_management)
and has_interface_information_type(D: component_name,
output_interface,
Dout: information_type_name)
and is_information_link(Lout: information_link_name)
and has_information_link(C: component_name,
Lout: information_link_name)
and has_source_component(Lout: information_link_name,
D: component_name)
and has_source_information_type(Lout: information_link_name,
Dout: information_type_name)
and has_destination_component(Lout: component_name,
C: component_name)
and has_destination_information_type(Lout: information_link_name,
Cout: information_type_name)
and has_task_control(C: component_name,
TC: task_control_kb_name)
and makes_awake(TC: task_control_kb_name,
[D: component_name, Lout: information_link_name])
and is_component(W: component_name)
and has_characterisation(W: component_name,
external_world)
and has_interface_information_type(W: component_name,
input_interface,
Win: information_type_name)
and is_information_link(Lobs: information_link_name)
and has_source_component(Lobs: information_link_name,
C: component_name)
and has_source_information_type(Lobs: information_link_name,
Cout: information_type_name)
and has_destination_component(Lobs: information_link_name,
W: component_name)
and has_destination_information_type(Lobs: information_link_name,
Wout: information_type_name)
then has_ability(C: component_name,
is_capable_of_executing_observation_initiation_in(W:component_name));

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

if has_ability(C: component_name,

is_capable_of_reasoning_about_unidirectional_communication_from(C2:component_name))
and has_ability(C: component_name,
is_capable_of_executing_unidirectional_communication_from(C2: component_name))
and has_ability(C: component_name,
is_capable_of_combining_reasoning_about_and_executing_
unidirectional_communication_from(C2: component_name))
then has_ability(C: component_name,
is_capable_of_unidirectional_communication_from(C2: component_name));

When the 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.

6 Comparison to Related Work on Agents and Design

The approach presented in this paper combines two generic models: the Generic Agent Model GAM and the Generic Design Model GDM. In this section for both of these models it is discussed how it relates to other literature.

6.1 Relation of the Generic Agent Model GAM to existing agent architectures

In the agent literature, various agent architectures can be found, often specialised to a particular type of application. The design of most of these agent architectures is not formally specified in detail; usually they are only available in the form of an implementation, and at the conceptual level some informal pictures and natural language explanations. In general, the aim for the development of these agent architectures in the first place is to have a working piece of software for a specific type of application. The design of the Generic Agent Model GAM introduced in {11} and used in this paper has a different aim. GAM was developed as a unified model for weak agency, formally specified in an implementation- and domain-independent manner at a high level of abstraction. Therefore it is possible to specialise and instantiate the agent model GAM to obtain conceptual, formal specifications of more specific models for a variety of (implemented, but not formally specified) agent types and agent behaviours. Thus it serves as a unified conceptual description which enables comparison of these agent architectures at a conceptual but yet formally defined level.
In this sub-section it is discussed how the generic agent model GAM can be refined to obtain a formally specified design model for four other existing agent architectures: Touring Machines[29, 30], INTERRAP[43, 44], ZEUS[45], and ADEPT[37]. A summary is given in Table 1. Note that in the comparison also the component cooperation_management of GAM is incorporated, which was not used in this paper.

The Touring Machines architecture described in [29, 30] distinguishes three layers: a reactive layer, a planning layer, and a modelling layer; all layers process concurrently. The reactive layer can be formally specified as an instantiation of the the components world interaction management and agent interaction management in the generic agent model GAM. If reactions on combined input from observation and communication have to be modelled, two information links between world interaction management and agent interaction management are added for direct information exchange, avoiding modelling this information as beliefs. The planning layer can be specified as a refinement of component own process control; also the Control Rules are part of this refinement of own process control. The modelling layer can be obtained by instantiation of the components maintenace of world information and maintenance of agent information, where models of the agent's environment are maintained. The specific approach to control by Control Rules (in the form of Censors and Suppressors) entails that all incoming and outgoing information has to be filtered by the Control Rules within own process control. This means that, although in principle all layers are meant to be connected independently to the outside world, in order to do the filtering, in practice these connections come together in the Control Rules component within own process control. This confirms analyses of this agent architecture available in the literature; e.g., see [30].

Within the INTERRAP architecture[43, 44], the following components play a role: *World Interface* (Sensors, Communication, and Actors), *Agent KB* (Social Model (SM), Mental Model (MM), World Model (WM)), *Agent Control Unit* (Cooperative Planning Layer (CPL), Local Planning Layer (LPL), Behaviour-Based Layer (BBL)). A formal design specification of the World Interface can be obtained as an instantiation of the components agent interaction management (communication) and world interaction management (sensors, actors) within GAM. A design specification of Agent KB's Social Model can be obtained as an instantiation of the component maintenance of agent information and the World model of maintenance of world information. The Mental Model can be obtained as a refinement within own proces control, as far as mental concepts referring to the agent itself are concerned. If also mental concepts such as joint intentions are involved, these can be included within cooperation management. The Local Planning Layer can be obtained as a refinement of own process control, the Cooperative Planning Layer of cooperation management, and the Behaviour-Based Layer of the components agent interaction management and world interaction management. The INTERRAP model has a much richer structure than the generic agent model GAM, especially in control aspects. Control differs from the Touring Architecture in that only the Behaviour-Based Layer is connected to the outside world, and the Local Planning Layer (within own process control) becomes involved as soon as the Behaviour-Based Layer indicates that the situation is assessed as beyond its competence. Similarly, own process control can indicate that the situation is beyond its (individual) competence and involve the Cooperative Planning Layer (in cooperation management).

For the refinement of GAM this means that it is specified that the appropriate control information is exchanged between world interaction management and agent interaction management, own process control and cooperation management.

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

The architecture ADEPT (Advanced Decision Environment for Process Tasks; see [37]) represents business processes by a hierarchy of cooperative agents. The hierarchy ensures that communication overhead between agents and the autonomy of the agents are balanced. Within this model, agents have the following modules: a communication module, an interaction management module (IMM), a situation assessment module (SAM), a service execution module (SEM), a self model (SM), acquaintance models (AM). These modules have been specified as a refinement of GAM as follows: the module IMM as a refinement of the component cooperation management, the modules SAM and SM as components within a specialisation of the component own process control, the module SEM can clearly be described as a specialisation of the component maintenance of agent information.

	WIM	AIM	MWI	MAI	OPC	CM	AST
Touring Machines	Reactive Layer	Reactive Layer	Environment Model	Agent Models	refinement for Planning Layer, Control Rules	-	-
INTERRAP	Sensors, Actors, Behaviour-Based Layer	Communicat-ion , Behaviour-Based Layer	World Model	Social Model	refinement for Mental Model, Local Planning Layer	Social Mental Model, Cooperative Planning Layer	-
ZEUS	-	Mailbox, Message Handler	Resource Database	Acquaintance Models	refined for Planner and Scheduler, Task/Plan Database, Execution Monitor	Coordination Engine	
ADEPT	-	CM	-	AM	SAM SM	IMM	SEM

Table 1 Overview of refinements of GAM to designs for various agent architectures

6.2 Comparison of the Generic Design Model GDM to existing design approaches

Design processes occur in many areas, including engineering design and software design. In engineering design the focus is on finding a configuration of certain physical elements that, combined in one artefact, perform the required functions, see for example[47, 38, 3]. Similarly in software design, a configuration of program-components has to be found that, combined into one program (i.e., the artefact), performs the required functions. In both areas (required) function is related to the structure of the artefact. Also, in both areas structured artefacts are employed and properties can be formulated to reflect the functionality or behaviour of an artefact.

A substantial amount of research has focused on defining models of design as a basis for knowledge-based design systems; e.g., [31, 57, 58, 18, 24, 26, 32, 55, 1, 56, 60, 46, 19]. While modelling the required functionality (or properties) of the design

object description is claimed to be an important aspect of a process of design (for an overview see [62]) not many approaches have included actual reasoning about these properties in the context of requirement manipulation. Some of these models recognise manipulation of requirements or strategies as an important part of (re-) design. Relevant literature on models for (re-)design is addressed below.

In [38] the process of design consists of synthesis and selection (or analysis) processes, where the selection (or analysis) process validates results of the synthesis process. In [47] the process of design consists of explanation of the problem description, conceptual design, detailed design, and manufacturing.

Models of processes of design provide a structured description of a design activity. Models of design differ in their underlying formalisations. Design models are represented in structures such as blackboard architectures (e.g., [4]); algorithms (e.g., [2]), SOAR[53], task models or problem solving methods[18, 13, 61], or agent architectures[27, 6, 40].

One approach employed to model design tasks, is to design as the application of the problem solving method "Propose & Revise" [41] in which a tentative solution is generated and modified. Propose-and-Revise is based on the problem-solving methods "Propose-Critique-Modify" and "Propose-Verify-Redesign"[23, 24, 33]. Within the context of parametric design several experiments with Propose-and-Revise have been performed by [64].

A perspective on engineering design as a synthesis process is described by Alberts [3]. Original requirements and basic generic elements are input of the design process, and final requirements and product descriptions are output of the design process. This perspective on engineering design includes the manipulation of requirements (and the manipulation of a product description) but does not explicitly include objectives on the design process itself.

A model of design proposed by Ohsuga[46] features both the manipulation of a design object description as well as strategic knowledge on the management of this process. Two kinds of knowledge are identified in this model: knowledge applied directly to the model being designed, and knowledge to guide and control the exploration or search process. An extension of this model investigates the manipulation of sets of requirements in interaction with users[54]. An experience-based approach is taken, allowing users to explore the space of requirements.

Another model in which both the manipulation of requirements and the manipulation of design object descriptions are discerned is proposed by Smithers[50]. From his viewpoint of design as exploration, both the exploration of possible sets of requirements as well the exploration of possible design object descriptions are explicitly modelled[52].

Models for design processes incorporate ontologies: ontologies for design objects and ontologies for requirements. Ontologies can be briefly characterised as descriptions of concepts in the world. Ontologies for design objects can be shared across domains and incorporated in design processes[3, 35, 7]. Within the Ontolingua project[34] an ontology for a process of design has been proposed which is geared towards the representation of design object descriptions. Likewise, ontologies can be employed to represent requirements. Examples of ontologies that represent requirements (e.g., required properties of compositional systems) are: properties of diagnostic systems [5] and [25], properties of propose-and-revise problem-solving methods[28].

7 Discussion

To design an agent capable of designing, insight is required in both desired characteristics of the agent, and desired characteristics of the design process. The architecture of the generic design agent presented in this paper is based on an existing generic agent model, and an existing generic model of design. It combines results from the area of Multi-Agent Systems and the area of AI and Design.

The generic *agent* model employed has been developed on the basis of experience with agent models of different kinds; for example, models for information gathering agents, co-operative agents for project co-ordination, BDI-agents, negotiating agents, broker agents, and agents capable of simulating animal behaviour. The generic *design* model has been developed and evaluated on the basis of experience with design applications in a number of domains; for example, design of sets of measures for environmental policy, aircraft design, and elevator design. These two generic models: a generic agent model and a generic design model, have been combined to form a generic model for a design agent.

This design agent can be integrated in a distributed Web-based design environment for any given application

domain. The environment needs to include at least one agent that can communicate requirements, and one agent (possibly the same agent) to which design object descriptions can be communicated.

For the acquisition of requirements, the design agent DA can co-operate with another agent RA that communicates design requirements. This agent RA may simply be an interface agent for a user, a client of DA, that literally acquires requirements from the user. RA may, alternatively, be an intelligent support agent for Requirements Engineering, that interacts with clients about their requirements, and in turn communicates the acquired requirements to DA. For the agent DA it makes no difference whether requirements are communicated directly by a client, or by another type of agent.

Comparably, a design object description may be communicated directly to the (interface agent of the) client, or, for example, to a manufacturing agent MA, that controls an automated manufacturing system. This role of MA can also be integrated in DA, which would make DA an agent both for design and manufacturing. Based on the prototype application of a design agent for compositional systems, described in this paper, another application was made in which the manufacturing indeed was integrated in the agent DA: after having designed an agent, it is also able to (pro-)create the designed agent, resulting in deliberate evolution of the running multi-agent system. For more details about this application, see [12].

The design agent DA can also cooperate with other agents during the design process. For example, to acquire information about useful design components that can be used in a design object description, DA can co-operate with component broker agents as described in [20]. Moreover, DA can co-operate with other design agents or human designers, for example in a setting as described in [36] on different aspects of the design process.

Electronic Commerce necessarily involves interaction between human users in different types of organisations, and very dynamic, automated environments, in which the parties involved are not known beforehand, and often change. In such environments human users can be supported by Personal Assistant. These Personal Assistant Agents, in turn may make use of existing broker agents and other task-specific agents. Co-operation between these (human and computer) agents is to the advantage of all. To cope with the dynamic character of the environment, frequently new agents need to be created, or existing agents need to be modified, for specific purposes. Such frequent modification of an environment necessitates almost continuous maintenance.

Recently a few applications of broker agents have been addressed for this area; see, for example[21, 22, 39, 42, 48, 59]. However, these applications have been implemented without an explicit design at a conceptual level, and without taking into account the dynamic requirements imposed by the domain of application and the maintenance problem implied by this dynamic character.

On the basis of the approach introduced in this paper, a generic multi-agent Electronic Commerce environment will be developed in which a broker agent can dynamically reconfigure (parts of) the multi-agent system by adding or modifying Personal Assistant agents, broker agents and additional agents. More specifically, the aim is to develop a multi-agent broker architecture with a number of co-operative broker agents, Personal Assistant agents, and task specific agents. Each broker agent can dynamically configure and implement new agents or modify existing agents as part of the multi-agent system as follows:

- if new users (clients) subscribe to a broker agent, Personal Assistant agents tuned to the requirements imposed by this user, may be created, or existing Personal Assistant agents may be modified, due to changed requirements.
- if required in view of the load of an existing broker agent, new broker agents can be added to distribute the load (and avoid overload of the existing broker agent), or existing broker agents can be modified.
- if opportune, or requested, new agents may be created to perform specific tasks, fulfilling certain dynamically imposed requirements; for example, for searching the Internet for specific types of information, or shadowing information at a specific site.

A principled approach to the design of the architecture is of crucial importance: a generic conceptual architecture of a broker agent is needed to support the (re)design process needed for dynamic creation or modification of agents based on dynamically imposed requirements. An approach in which conceptual design is the basis for structure-preserving (formal) detailed design and operational design, can provide the means to model, specify and implement the flexible structures required.

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 software environment. This research has been (partially) supported by NWO-SION within project 612-322-316: 'Evolutionary design in knowledge-based systems' (REVISE).

References

1 **Alberts L K, Wognum P M and Mars N J I**. Structuring design knowledge on the basis of generic components. In: Gero, J.S. (Ed.), *Artificial Intelligence in Design (AID'92)*, Kluwer, Dordrecht (1992), pp. 639-656.

2 **Alberts L M, Bakker R R, Beekman D and Wognum P M** Model-based redesign of technical systems. In: *Proceedings of the 4th international workshop on principles of diagnosis* (1993).

3 **Alberts M** *YMIR: an ontology for engineering design*. PhD thesis, University of Twente, The Netherlands (1993).

4 **Ball N R and Bauert F** The Integrated Design Framework: supporting the design process using a blackboard system. In: Gero, J.S. (Ed.), *Artificial Intelligence in Design (AID'92)*, Kluwer academic publishers (1992), pp. 327-348.

5 **Benjamins V R** *Problem Solving Methods for Diagnosis*. PhD Thesis, University of Amsterdam, Amsterdam, The Netherlands (1993).

6 **Berker I and Brown D C** Conflicts and Negotiation in Single Function Agent Based Design Systems. In: Brown, D.C., Landes, S.E., and Petrie, C.J. (Eds.), *Concurrent Engineering: Research and Applications, Journal*, Special Issue: Multi Agent Systems in Concurrent Engineering, Technomic Publishing Inc., Vol 4, No 1 (1996), pp. 17-33.

7 **Borst W N, Akkermans J M and Top J L** Engineering ontologies. In: *International Journal of Human-Computer Studies*, special issue on using explicit ontologies in KBS development, Vol 46 (1997), pp. 365-406.

8 **Brazier F M T, Dunin-Keplicz B M, Jennings N R, and Treur J** Formal Specification of Multi-Agent Systems: a Real World Case In: Lesser V (ed) *Proceedings First International Conference on Multi-Agent Systems ICMAS'95* (1995) pp 25-32 MIT Press. Extended version in: Huhns M and Singh M (eds) *International Journal of Co-operative Information Systems IJCIS* Vol 6 No 1 (1997) pp 67-94 (Special issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems)

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

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

11 **Brazier F M T, Jonker C M and Treur J** Compositional Design and Reuse of a Generic Agent Model. Applied Artificial Intelligence Journal, vol. 14 (2000), pp. 491-538.

12 **Brazier F M T, Jonker C M, Treur J and Wijngaards N J E** *Deliberate Evolution in Multi-Agent Systems* Technical Report Vrije Universiteit Amsterdam, Department of Artificial Intelligence (1998)

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

14 **Brazier F M T, Langen P H G van and Treur J** 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

15 **Brazier F M T, Langen P H G van and Treur J** Strategic Knowledge in Compositional design Models In: Gero J S and Sudweeks F (eds) *Artificial Intelligence in Design '98* Kluwer Academic Publishers, Dordrecht (1998), pp 129-148

16 **Brazier F M T, Langen P H G van, Treur J, Wijngaards N J E and Willems M** 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* Vol 44 (1996), pp 469-520

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

18 **Brown D and Chandrasekaran B** *Design problem solving: knowledge structures and control strategies*, San Mateo, CA, Morgan Kaufmann (1989).

19 **Brown D C and Birmingham W P** Understanding the Nature of Design. In: *IEEE Expert*, Vol 12, No 2 (1997), pp. 14-16.

20 **Charlton C** A Web Broker for Mechanical Component Selection In: P Rodgers and A Huxor (eds) *Distributed Web-based AI design Tools: AID'98 Workshop 2 Notes* (1998)

21 **Chavez A and Maes P** Kasbah: An Agent Marketplace for Buying and Selling goods In: *Proc of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* PAAM'96 The Practical Application Company Ltd, Blackpool (1996), pp 75-90

22 **Chavez A, Dreilinger D, Gutman R, and Maes P** A Real-Life Experiment in Creating an Agent Market Place In: *Proc of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* PAAM'97 The Practical Application Company Ltd, Blackpool (1997), pp 159-178.

23 **Chandrasekaran B** Generic Tasks in Knowledge-Based Reasoning: High Level Building Blocks for Expert System Design. In: *IEEE Expert* (1986), pp. 23-30.

24 **Chandrasekaran B** Design problem solving: a task analysis. In: *AI Magazine*, Vol 11 No 4 (1990), 59-71.

25 **Cornelissen F, Jonker C M and Treur J** Compositional Verification of Knowlegde-Based Systems: A Case-Study for Diagnostic Reasoning. 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 (1997), pp. 65-80.

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

27 **Dunskus B V, Grecu D L, Brown D C and Berker I** Using Single Function Agents to Investigate Conflict. In: *Artificial Intelligence in Engineering Design and Manufacturing (AIEDAM)*, Special Issue: Conflict Management in Design, Vol 9 No 4 (1995), pp. 299-312.

28 **Fensel D and Motta E** Structured development of problem solving methods. In: Gaines, B.R., Musen, M.A. (Eds.). *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-based Systems workshop (KAW'98)*, Calgary: SRDG Publications, Department of Computer Science, University of Calgary (1998), pp. 20.

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

30 **Ferguson I A** Integrated control and coordinated behaviour. In: [64] (Wooldridge and Jennings, 1995a), (1995). pp. 203-218.

31 **French R and Mostow J** Toward Better Models of the Design Process. In: *AI Magazine*, Vol 6 No 1 (1985), pp. 44-57.

32 **Gero J S**Design prototypes: a knowledge representation schema for design. In: *AI Magazine*, Vol 11 No 4 (1990), 26-36.

33 **Goel A and Chandrasekaran C** Functional representation of design and redesign problem solving. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, Detroit MI, Morgan Kaufmann Publishers, Los Altos CA (1989), pp. 1388-1394.

34 **Gruber T R** A translation approach to portable ontology specifications. In: *Knowledge Acquisition*, Vol 5 No 2 (1993), pp. 199-220.

35 **Gruber T R and Olsen G R** An Ontology for Engineering Mathematics. In: Doyle, J., Torasso, P., and Sanewall, E. (Eds.), *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann (1994), pp. 241–245.

36 **Huxor A, Rogers P, Clarkson J, and Caldwell N** Knowledge-based Systems as Navigation Aids to

Online Content and Users In: P Rodgers A Huxor (eds) *Distributed Web-based AI design Tools: AID'98 Workshop 2 Notes* (1998).

37 **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** ADEPT: Managing Business Processes using Intelligent Agents. In: Proc. BCS Expert Systems 96, Conference (ISIP Track), Cambridge, UK (1996), 5-23.

38 **Koller R** (1985). *Für den Machinenbau.* Springer-Verlag, Berlin.

39 **Kuokka D and Harada L** On Using KQML for Matchmaking In: V Lesser (ed) *Proc of the First International Conference on Multi-Agent Systems ICMAS'95* MIT Press Cambridge MA (1995), pp 239-245

40 **Lander S E** Issues in Multi-agent Design Systems. In: *IEEE Expert*, Vol 12 No 2 (1997), pp. 18-26.

41 **Marcus S and McDermott J** SALT: A Knowledge-Acquisition Language for Propose and Revise Systems. In: *Journal of Artificial Intelligence*, Vol 39 No 1 (1989), pp. 1-37.

42 **Martin D, Moran D, Oohama H, and Cheyer A** Information Brokering in an Agent Architecture In: *Proc of the Second International Conference on the Practical Application of Intelligent Agents and Multi-agent Technology* PAAM'97 The Practical Application Company Ltd, Blackpool (1997), pp 467-486.

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

44 **Müller J P, Pischel M, and Thiel M** Modelling reactive behaviour in vetically layered agent architectures. In: [64] (Wooldridge and Jennings, 1995a), pp. 261-276

45 **Nwana H S, Ndumu D T and Lee L C** 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 (1998), 377-391. Also in Applied AI, vol. 13, pp. 129

46 **Ohsuga S** Strategic Knowledge Makes Knowledge Based Systems Truly Intelligent. In: Candy, L., and Hori, K. (Eds.), *Proceedings of the First International Workshop on Strategic Knowledge and Concept Formation*. Lutchi Research Centre (1997), pp. 1-24.

47 **Pahl G and Beitz W** *Engineering Design.* Springer Verlag, New York (1984). Originally: Konstruktionslehre, 1977, in German.

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

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

50 **Smithers T** Design as Exploration: Puzzle-Making and Puzzle-Solving. In: *Proceedings of the workshop on exploration-based models of design and search-based models of design at second international conference on AI in Design* (1992).

51 **Smithers T** On knowledge level theories of design process In Gero J S and Sudweeks F (eds) *Artificial Intelligence in Design '96* Kluwer Academic Publishers, Dordrecht (1996), pp 561-579.

52 **Smithers T and Troxell W** Design is intelligent behaviour, but what's the formalism? In: *AIEDAM*, Vol 4 No 2 (1990), pp. 89-98.

53 **Steier D** Automating Algorithm Design within a General Architecture for Intelligence. In: Lowry, M.R., and McCartney, R.D. (Eds.), *Automating Software Design*, AAAI Press (1991), pp. 577-602.

54 **Sumi Y** Supporting the Acquisition and Modelling of Requirements in Software Design. In: Candy, L., and Hori, K. (Eds.), *Proceedings of the First International Workshop on Strategic Knowledge and Concept Formation*, Lutchi Research Centre (1997), pp. 205-216.

55 **Takeda H, Veerkamp P J, Tomiyama T and Yoshikawa H** Modelling Design Processes. In: *AI Magazine*, Vol 11 No 4 (1990), pp 37-48.

56 **Tham K W and Gero J S** PROBER – A design system based on design prototypes. In: Gero, J.S. (Ed.), *Artificial Intelligence in Design (AID'92)*, Kluwer, Dordrecht (1992), pp. 657-675.

57 **Tomiyama T and Yoshikawa H** Extended General Design Theory. In: H. Yoshikawa and E.A. Warman (Eds.), *Proceedings of the IFIP WG 5.2 Working Conference on Design Theory for CAD*, North-Holland (1987), pp. 95-125.

58 **Treur J** A logical analysis of design tasks for expert systems. In: *International Journal of Expert Systems*, Vol 2 (1989), 233-253.

59 **Tsvetovatyy M and Gini M** Toward a Virtual Marketplace: Architectures and Strategies In: *Proc of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* PAAM'96 The Practical Application Company Ltd, Blackpool (1996), pp 597-613.

60 **Vescovi M and Iwasaki Y** Device design as functional and structural refinement. In: Faltings, B. (Ed.), *Working Notes of the IJCAI'93 Workshop on AI in Design*, Chambéry (1993), pp. 55-60.

61 **Wielinga B J and Schreiber A Th** Configuration design problem solving. In: *IEEE Expert*, Vol 12 No 2, Special issue on AI and design (1997), pp. 49-56.

62 **Winsor J and Maccallum K** A review of functionality modelling in design. In: *The Knowledge Engineering Review*, Vol 9 No 2 (1994), pp. 163-199.

63 **Wooldridge M J and Jennings N R** *Intelligent Agents: Theory and Practice* In: Knowledge Engineering Review Vol 10 No 2 (1995), pp 115-152.

64 **Zdrahal Z and Motta E** An In-Depth Analysis of Propose & Revise Problem Solving Methods. In: Gaines, B.R., and Musen, M.A. (Eds.), *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-based Systems workshop (KAW'95*), Calgary: SRDG Publications, Department of Computer Science, University of Calgary (1995), pp. 38/1-38/20.