STRATEGIES FOR INTEGRATING MULTIPLE VIEWPOINTS AND LEVELS OF DETAIL*

FRANCES M.T. BRAZIER, SANDER VAN SPLUNTER, AND NIEK J.E. WIJNGAARDS
Intelligent Interactive Distributed Systems Group, Faculty of Sciences, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081 HV, Amsterdam, The Netherlands
Email: {frances,sander,niek}@cs.vu.nl
URL: http://www.iids.org/

Abstract. Automated design requires explicit representation of strategic knowledge. This paper focuses on strategic knowledge needed to reason with and about viewpoints during the design of a software agent. Reasoning with and about viewpoints entails not only deciding which viewpoint to consider when and in which context, but also at which level of detail. In this paper an information retrieval agent is used to illustrate how these types of knowledge can be used to design software agents.

1. Introduction

Design is a complex process, a process in which many different types of knowledge and (knowledge of) requirements play a role. The perspective on a process of design taken in this paper is that of 'exploration of design space', where the design space consists of two subspaces: one for the possible sets of qualified requirements and one subspace for possible designs. Viewpoints provide a means to focus a design process within and between each of these two spaces. The overall design strategy of the design process determines the types of exploration strategies employed (Brazier, Langen, Ruttkay and Treur, 1994; Brazier, Langen and Treur, 1998; Logan and Smithers, 1992; Löckenhof and Messer, 1994).

This paper focuses on strategies used for "exploration": (1) within a

* Concept version, to appear in Proceedings of workshop on Strategic Knowledge and Concept Formation, December 2001.

viewpoint, and (2) for the co-ordination of viewpoints during different phases of a design process. A description and categorisation of these strategies is provided for the domain of design considered in this paper namely fully automated design of software agents. An information retrieval agent is introduced for the purpose of illustration. Note that these agents may be designed for many different types of tasks, of which information retrieval, diagnosis, and design, are examples. Autaomted design of internet agents is an area in which recently some progress has been booked (Reticular Systems, 1999; Brazier and Wijngaards, 2001; O'Hare, 1996).

An assumption in this paper is that explicit representation of strategies within a process of design provides additional structure to the process of design (Rist, 1995). Acquiring, representing and applying strategic knowledge within design processes has been studied by, e.g., (Gruber, 1990; Strelnikov and Dmitrevich, 1991; Rist, 1995; Ohsuga, 1997; Hori, 1998).

Section 2 discusses research on viewpoints in general, and viewpoints with respect to agent design, in particular. Section 3 introduces an agent factory, capable of automated design of intelligent agents. Section 4 distinguishes four viewpoints and introduces the concept of a design focus for the agent factory. Strategies for reasoning about, and within, viewpoints at different levels of detail are described in Section 5. Section 6 focuses on the results and areas for further research.

2. Research on Perspectives: Viewpoints and Levels of Detail

Design of an artefact can be viewed from different perspectives, e.g. from a specific point of view during a specific phase of detail in a design process.

2.1 LEVELS OF DETAIL

During a design process an initial conceptual design becomes more detailed. This also holds for automated design of an agent. For automated design of an agent two levels of design are distinguished: conceptual design and detailed design. The result of detailed design is a blueprint for the agent, and contains sufficient information for the agent to be created. The design of an agent is an iterative process: during detailed design choices may be made that necessitate changes to the conceptual design, etc.

2.2 VIEWPOINTS

Reasoning from different *viewpoints* is a necessary part of most design processes. Domain specific contexts are often the grounds for such viewpoints. Models of design most often incorporate reasoning from different

viewpoints, including strategic reasoning about viewpoints, and reasoning from the perspective of other viewpoints (Schön, 1983).

A common vision is that viewpoints are strongly related to the different fields of expertise involved. For example in a specific example of aircraft design engineering, electrical engineering, systems engineering, unit management, styling specialisms and tooling expertise (Brazier, Jonker and Treur, 1996) were involved. These domains of expertise each impose their own restrictions (requirements for the design artefact to fulfill) and involve specific domain expertise / knowledge (domain models). In addition requirements imposed on the design process as a whole often have implications for the requirements imposed within the different viewpoints, and as such require the necessary co-ordination.

A single design agent, for example, needs strategic, reflective knowledge to know which viewpoint to work from at each point during a process. Reflection is required continually. This holds also for multi-agent design. In the literature on reflection such as (Weyhrauch, 1980; Davis, 1980, Maes and Nardi, 1998, Attardi and Simi, 1994, Clancey and Bock, 1988) a restricted number of types of reflective reasoning are modelled. Non-trivial combinations of different types of reflective reasoning, however, have not been studied extensively. In literature (Fisher and Wooldridge, 1993; Wooldridge and Jennings, 1995; Cimatti and Serafini, 1995; Wagner, 1996) on multi-agent systems, most often the types of reflective reasoning agents are capable of performing is limited. For example, in the literature mentioned no explicit reflective reasoning about communication is modelled.

For a fully automated design process reflective knowledge includes *strategic knowledge* for choice of viewpoint, and co-ordination between viewpoints, and also strategic knowledge on how to reason within each viewpoint. The need to be able to express different viewpoints within software design has been recognised by (Finkelstein, Kramer, Nuseibeh, Finkelstein and Goedicke, 1992). Their approach, however is does not include reflective strategic reasoning about choices during design.

2.3. VIEWPOINTS ON AGENTS

For automated design, the choice of a conceptual framework may facilitate, or obstruct, the use of specific viewpoints. This is related to the granularity of the description of an artefact (e.g., see (Stefik, 1995)): it is impossible to deal with parts of an artefact which cannot be described; this holds for all tasks, not just design, e.g. diagnosis, classification, and configuration. In this paper agents are modelled using a conceptual framework developed to model knowledge intensive multi-agent systems.

DESIRE is a formal knowledge-level modelling and specification

framework for knowledge-intensive (multi-agent) systems (Brazier, Dunin-Keplicz, Jennings and Treur, 1995, 1997; Brazier, Jonker and Treur, 1998). Both conceptual models and detailed formal specifications are supported by the framework. The compositional nature of the models, and the separation between processes and knowledge makes it possible to build knowledge intensive systems from reusable components. Automated prototype generation on the basis of detailed formal specifications facilitates verification and validation of knowledge intensive systems.

The DESIRE framework, used to design and develop multi-agent sytems, distinguishes different types of knowledge needed to specify the design of an agent:

1. knowledge composition
2. process composition
3. information exchange
4. control

These types of knowledge can be mapped onto viewpoints with which the design of an agent can be specified; a knowledge perspective, a process composition perspective, an information exchange perspective and a control perspective. These perspectives can be described at both a conceptual level and a detailed level. Requirements, often expressed in terms of properties and structure of an artefact, may be related to one, or more, viewpoints and levels of detail.

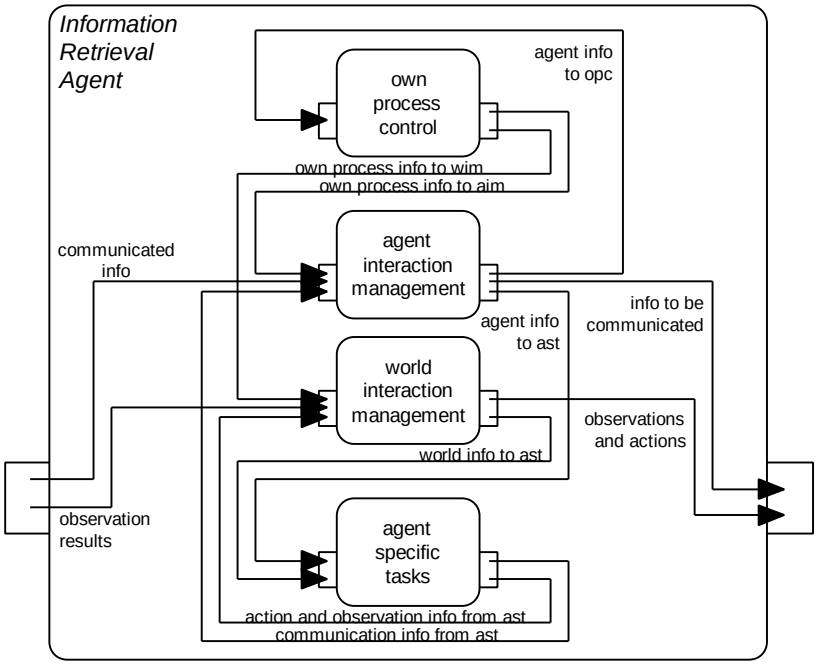

Figure 1. Architecture of a simple information retrieval agent.

The architecture of a simple information retrieval agent is shown in Figure 1. This architecture models an agent that:

1. reasons about its own processes (component Own Process Control),
2. communicates with other agents (component Agent Interaction Management),
3. interacts with the external world (component World Interaction Management),
4. maps requests for information onto queries (within component Agent Specific Tasks is a component Information Retrieval).

In this architecture, information exchange is modelled by information links between interfaces of components. The names of the information links denote the types of information transferred. The control of activation of components and information links can be fully specified. For example, a complex sequential control mechanism can be employed. Alternatively, all four components can run in parallel, and all information links can be made 'awake', that is, information is transferred once it becomes available.

A more complex agent architecture includes components for management of cooperation with other agents, explicit maintenance of information about other agents, and explicit maintenance of information about the external world, as defined in the *generic agent model* (Brazier, Jonker and Treur, 2000).

3. Agent Factory

Agents are constructed from building blocks by an automated agent factory (Brazier and Wijngaards, 2001). Adapting an agent entails adapting the configuration of its templates and components. Adaptation may be performed by an external service (e.g., by an agent factory) or by an agent itself (e.g., as self-modification of an agent (Brazier and Wijngaards, 2001b)). Section 3.1 describes characteristics of an agent factory and Section 3.2 describes knowledge needed in the agent factory. Section 3.3 describes an example of combining building blocks. Section 3.4 discusses the relation of research on the agent factory to previous research.

3.1 CHARACTERISTICS OF AN AGENT FACTORY

An agent factory builds and adapts agents. The agent factory is based on five underlying assumptions: (1) agents have a compositional structure, (2) re-usable parts of agents can be identified, (3) two levels of descriptions are used: conceptual and detailed, (4) properties and knowledge of properties are available, and (5) no commitments are made to specific languages and/or ontologies.

On the basis of these assumptions agents can be designed and adapted. Whether the need for adaptation arises in an agent itself, or in another agent is not relevant in this context. Limited interaction with a client of an agent factory is of relevance, as this implies that an agent factory has to be able to (fully) automatically (re-)design agents.

The design of an agent within the agent factory is based on configuration of building blocks. Building blocks may include cases and partial (agent) designs (cf. generic models / design patterns). This approach relates to design patterns (e.g., Gamma, Helm, Johnson and Vlissides, 1994; Peña-Mora and Vadhavkar, 1996; Riel, 1996) and libraries of software with specific functionality (e.g., problem-solving models (Schreiber, Akkermans, Anjewierden, de Hoog, Shadbolt, van de Velde, and Wielinga, 1999) or generic task models (Brazier, Jonker, and Treur, 1996)).

In the agent factory building blocks can be templates with open slots, fully specified components, and/or a combination of both. Building blocks are defined at both levels of detail: conceptual and detailed. As a result a structural preserving mapping can be made, if wished, between building blocks at conceptual level and building blocks at detailed level. (Note that this is not always ideal.) A detailed description for an agent most often includes operational detail (e.g. code). For each conceptual description, a number of detailed descriptions may be devised and vice versa. These detailed descriptions may differ in the operational language (e.g., C, C++, Java), but

also in, for example, the efficiency of the operational code. A mapping between building blocks relates a building block containing a conceptual description to a building block containing a detailed description. The mapping relates open slots of the conceptual building block to the open slots in the detailed building block.

Building blocks themselves are configurable. Templates and/or components cannot be combined indiscriminately. The *open slot* concept is used to regulate the ways in which templates and components may be combined. An open slot in a template or component has associated properties at both levels of detail that prescribe the properties of the entity to be 'inserted'.

Specific 'glue' may be needed to aid the insertion of a building block in an open slot. Glue, which exists at both conceptual and detailed levels of design, is used to transform certain information to the correct format/ontology.

3.2 KNOWLEDGE NEEDED IN AN AGENT FACTORY

An agent factory capable of automatically re-designing an agent needs to combine knowledge on its domain (i.e., design of intelligent agents), its process (i.e., re-design processes), and the application of its domain to its process.

Knowledge on the domain, i.e. design of intelligent agents, requires knowledge:

- of agents
- at two levels of design: conceptual and detailed
- per level of detail: properties on structure, function, behaviour, and non-functional aspects
- of relations between levels of detail
- of relations among properties
- of building block combinations
- of glue

Knowledge on the process, i.e. re-design, entails knowledge on:

- manipulation of sets of qualified requirements
- manipulation of design object descriptions
- co-ordination of re-design process

Knowledge on the process of re-design in the domain of intelligent agents entails knowledge on:

- refinement and modification of qualified requirements
- retrieval of building blocks (including intelligent matching)
- assessment of qualified requirements on the basis of a partial description of an agent
- resolution of conflicts among qualified requirements

A number of models are used in the (re-)design of intelligent agents. A generic model of a design process (Brazier, Langen, Ruttkay and Treur, 1994) has

been used to *design* the agent factory. This generic model of design has an associated logical theory of design (Brazier, Langen and Treur, 1996). Within this model and theory of design, design strategies (Brazier, Langen and Treur, 1998) and design rationale can be modelled (Brazier, Langen and Treur, 1997), and conflict management can be explicitly described (Brazier, Langen and Treur, 1995).

Second, a number of issues are related to the *design of agents*. The characteristics that play a role are described in (Brazier, Jonker and Treur, 1998). A generic model of an agent (Brazier, Jonker and Treur, 2000), based on a notion of weak agency proposed by Wooldridge and Jennings (1995) has been used: weak agency is characterised by autonomy, social ability, reactiveness, and pro-activeness. In contrast the notion of strong agency is based on the characteristics of mentalistic and intentional notions (related to the notion of intentional stance by Dennet (1987)). Models of co-operation and co-ordination between agents have been proposed (Brazier, Jonker and Treur, 1996).

3.3 BUILDING BLOCKS

The architecture of a simple information retrieval agent described above is available as a building block (template) within the agent factory. The conceptual building block for this simple information retrieval agent contains the architecture description shown in Figure 1. A number of the components and information types contain open slots that need to be filled in.

The components with open slots are shown in Figure 2. The component Own Process Control (opc) has an open slot, in which specific functionality needs to be included for the specific simple information retrieval agent. Likewise, the component World Interaction Management (wim) has an open slot, which needs to include knowledge about the specific means with which the agent can interact with resources in the external world. The component Agent Specific Task (ast) has an open slot which needs to include specific knowledge on transforming requests from a user into queries on information sources, and results of queries on information sources into answers to users.

The fourth component of the agent, agent interaction managent (aim), does *not* have an open slot. The template of the simple information retrieval agent is based on the assumption that the simple information agent only communicates with one other agent, e.g. its owner.

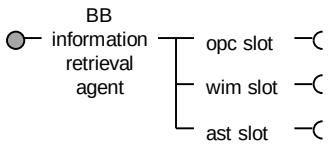

Figure 2. Component open slots of building block.

In the example used in this paper, the open slots in the template for the simple information retrieval agent are filled in as shown below in Figure 3.

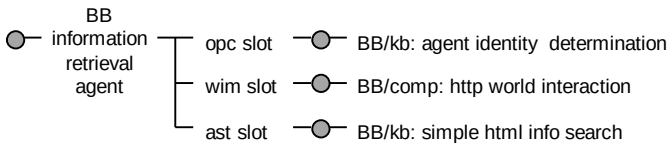

Figure 3. Building block with extended component slots.

The control inside this building block is pre-defined, no control slot is available for extension. A number of information types used by the agent need to be extended, see Section 4. A detailed building block for the information retrieval agent template is available (in Java). The structure of the code may mirror the architecture of the agent, but it may also be quite different, e.g. for reasons of efficiency. The open slots in the detailed building block are related to the open slots in the conceptual building block.

3.4 RELATION TO PREVIOUS RESEARCH

The agent factory, in essence, re-designs descriptions of agents. Previous research (Brazier, Jonker and Treur, 2000; Brazier, Jonker, Treur and Wijngaards, 2000) focussed on automated redesign of multi-agent systems at a conceptual level without the use of building blocks at two levels of detail. The automated servicing service is an extension of this work in two aspects.

The first distinction with previous work is that the agent factory is not primarily focussed on re-designing agents on the basis of first principles on a conceptual level, as described in (Brazier, Jonker, Treur and Wijngaards, 2000). The agent factory uses building blocks to construct, and adapt, agents. Building blocks are reusable parts of agents, ranging from skeletons for larger parts of agents (i.e., templates) to specific functionality (i.e., components).

The second distinction with previous work is a broadening of the scope of the re-design process. The agent factory modifies not only the conceptual description of an agent, but also its detailed description (operational code). This necessitates knowledge about the relationship between the conceptual description and detailed description in a template or component.

4. Multiple Viewpoints

The four viewpoints distinguished in Section 2.4 are relevant at both the conceptual and the detailed level of design. Section 4.1 describes the viewpoints, and Section 4.2 describes how a design focus makes use of viewpoints and levels of detail.

4.1 VIEWPOINTS

The four viewpoints identified in the agent factory are described in this section. These viewpoints are perspectives that can be taken on a (partial) design description during a design process (both conceptual and detailed design).

4.1.1 Process Composition Viewpoint

The process composition viewpoint focuses on the components of processes in the description of an agent.

- process - subprocess relation
- task models / process models
- input and output interfaces of a process
- information links between input and output interfaces of a process
- …

The process composition view on the simple information retrieval agent shown in Figure 1, focuses on information on, e.g., the composition relation between processes, as shown in Figure 4.

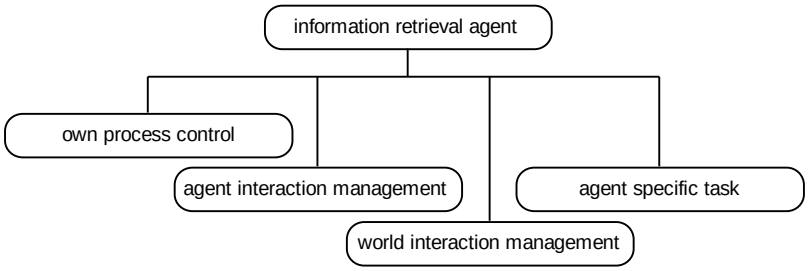

Figure 4. Process composition for information retrieval agent.

4.1.2 Information Exchange Viewpoint

The information exchange viewpoint focuses on information types and their relations in the description of an agent.

- input information type of a process
- output information type of a process
- meta-relations among information types

- meta-level relations among interfaces of a process
- information type composition relations
- information links, including their source and destination information types and -processes.
- …

4.1.3 Knowledge Viewpoint

The knowledge viewpoint focuses on knowledge in the description of an agent:

- knowledge bases
- numerical algorithms
- neural networks
- facts-bases
- information-type mappings (used in information links)
- …

4.1.4 Control Viewpoint

The control viewpoint focuses on control in the description of an agent. Information in this viewpoint includes:

- activation of (sub)processes
- activation of information links
- activation of knowledge-bases & algorithms
- fine-grained control on inference engines
- …

4.2 DESIGN FOCUS

This paper assumes that a need for modification has been determined by an earlier step in a design process and that the requirements on which "exploration" within the design object space are based, are known. A specific design focus can be defined by:

1. a level of abstraction (either conceptual or detailed),
2. a viewpoint (or combination of viewpoints),
3. modification context (structural, functional, behavioural, and non-functional properties of (parts of) the agent).

Strategic knowledge is needed to determine which focus is chosen at which level of detail: conceptual or detailed. Choices made at each level of abstraction influence choices at the other level of abstraction.

Strategic knowledge is needed to determine when to focus on which viewpoint. Each viewpoint shows, and hides, specific details of the agent.

Strategic knowledge is needed to determine demarcation of the part of an agent in which modification is to take place. An example of this is 'divide and conquer': by choosing, for example, to modify the world interaction management ability of an agent, the agent's specific task and the control of the

top-level of the agent, for example, can be safely ignored.

An example of a design focus is a combination of a conceptual level of detail, a process composition viewpoint, and an agent's specific task. Possible modifications for this example design focus include adding or removing an agent specific task by adding or removing (composed) processes.

In addition strategic knowledge is needed to know how to combine these factors and when to switch between foci.

5. Strategies

Strategies provide a means to consider, and co-ordinate a design process: in particular to consider and co-ordinate design foci as defined above.

The description of strategies in this section is structured by the definition of a design focus. A distinction is made between local strategies for selecting a level of abstraction, a viewpoint, and a modification context, versus more global strategies on design foci based on these single strategies. Section 5.1 describes strategies for selecting a level of abstraction, a viewpoint, and a modification context. Section 5.2 describes strategies for changing design foci. Section 5.3 illustrates the aforementioned strategies for the re-design of the example information retrieval agent.

5.1 SINGLE FACTOR STRATEGIES

The 'single factor' strategies provide options for each of the three parts of the design focus based on a given situation. These strategies are needed to delimit the search space of design foci. Section 5.1.1 describes strategies for a level of abstraction, section 5.1.2 describes strategies for a viewpoint, and section 5.1.3 describes strategies for a modification context.

5.1.1 Level of abstraction selection strategies

Strategies for selecting a level of abstraction may include:

- If no information is present within the modification context for a specific viewpoint, consider selecting another level of abstraction.
- ”translate” modifications (and their implications) at one level of abstraction to modifications at the other level of abstraction.
- always select the conceptual level of abstraction first, as it provides most structure for the design process.
- select the detailed level first if very specific techniques are required for which detailed building blocks are available, and later select the conceptual level to find a model of an agent within which the building block combinations at detailed level can be incorporated.
- Only select the detailed level if all modifications have been made at the

conceptual level (strict hierarchical approach).

5.1.2 Viewpoint selection strategies

Strategies for selecting a viewpoint may include:

- modification implication driven viewpoint selection: the number of (expected) violations of requirements for one specific viewpoint is important. The view with the most unfulfilled requirements is selected.
- in inconsistency driven viewpoint selection, the viewpoint with the most inconsistencies, is selected first.
- in interdependency driven viewpoint selection, the impact of modification within one viewpoint for another viewpoint is the selection criterion.
- in complexity driven viewpoint selection, the viewpoint with the most constraints for other viewpoints is chosen first.
- a pre-defined order of viewpoints to achieve a specific goal, e.g. by first taking a process composition viewpoint, then an information exchange viewpoint, then a knowledge viewpoint and finally a control viewpoint.

5.1.3 Modification context selection strategies

Strategies for selecting a modification context may include:

- uninformed traversal of compositional structure: breadth first, depth first, bottom-up, top-down, smallest before larger, from outside to inside, …
- more informed: complexity oriented, size of the modification context…
- task-order based: traverse order in which subtasks of a larger task are executed; if that order can be determined: backward or forward traversal.
- expected complexity of modification: select simplest, or most complex, modification context.

5.2 DESIGN FOCUS SELECTION STRATEGIES

Strategies are needed to move from one design focus to another, to eventually satisfy the need for modification. Changing a design focus involves reasoning about

- the given situation,
- the alternatives,
- the need for modification,
- the approach taken, and
- the previous design foci.

The given situation provides information on whether, e.g., a given modification has succeeded, whether the need for modification has been resolved, etc.

A new design focus is determined by a more global strategy which may be (and often is) based on options determined for each of the factors involved (level of detail, viewpoint and modification context)

The need for modification has an obvious influence on the selection of

design foci. A need for modification which is, e.g., to change the name of an agent requires less extensive modifications to an agent than, e.g., to change an agent's specific task, is directly or indirectly based on the requirements for the design process as a whole.

A strategy for resolving a need for modification usually involves a number of steps (a plan), each of which involves a design focus and a subsequent modification. The results of each step influence the choice. When determining the next design focus to use, it is of importance whether the design process is able to pursue multiple design foci in parallel. The assumption for this paper is that design foci are pursued in sequence, not in parallel.

In general, a number of reasons for changing a design focus can be determined:

- *Finished reasoning*. The simplest situation is when the modification process has finished attempting all modifications within the current design focus: no more progress can be made in the current design focus and switching to another design focus may be fruitful.
- *Forced interruption*. Instead of waiting for the modification process to finish attempting all modifications within the current design focus, the modification process may be forced by specific time constraints to change to another design focus. The interval for these changes can be set randomly, or can be foreordained. Furthermore the means for enforcing a switch of design focus must be known; e.g., is the modification process able to wrap up is current line of reasoning, or does it has to stop immediately, or can the last `representative` modification result be returned.
- *Motivated interruption*. Interruption of the modification process within a design focus may be motivated. A modification process focussed on a design focus decides to stop making modifications within that design focus, without having finished its modification alternatives. Alternatively, it could be decided that a modification in another design focus may yield more progress.

In this paper, the motivated changes of a design focus are most interesting, as these employ explicit, informed, strategies.

A design focus consists of three parts: a selected level of detail, a selected viewpoint, and a selected modification context. The change to a design focus with respect to a previous design focus can be measured as the extent to which the three parts of the design focus remain unchanged: the amount of *persistence of a design focus*. Four qualitative degrees of persistence of design focus are distinguished: maximum persistence, reasonable persistence, minimum persistence, and complete change. An assumption is that the need for modification does not change; as such a change usually implies a major change of design focus. Each of the degrees of persistence of design focus is described below.

5.2.1 Maximum persistence of design focus

A maximum persistence of design focus means that the new design focus is exactly the same as the previous design focus: the new design focus has the same level of detail, the same viewpoint, and the same modification context.

Strategies for choosing such a new design focus are, for example,

- the selected modification method has not yet yielded satisfactory results, an alternative modification method may yield better results.

5.2.2 Reasonable persistence of design focus

A reasonable persistence of design focus means that the new design focus is almost the same as the previous design focus: two of the three parts remain unchanged and one part is changed.

Strategies for changing the level of detail without changing the selected viewpoint and the selected modification context are, for example,

- the strategies described in section 5.1.1.
- a modification has succeeded within the current design focus, and the success of a modification at the other level of detail needs to be verified.

Strategies for changing the viewpoint without changing the selected level of detail and the selected modification context are, for example,

- the strategies described in section 5.1.2.
- A modification to a design involves changes within a number of viewpoints.

Strategies for changing the modification context without changing the selected level of detail and the selected viewpoint are, for example,

- the strategies described in section 5.1.3.
- another modification context needs to be focussed on within the selected level of detail and viewpoint.
- a generic model is applied (e.g., the generic agent model) and its sub-components are further specialised with other (generic) models.

5.2.3 minimum persistence of design focus

A minimum persistence of design focus means that the new design focus loosely resembles the previous design focus: one of the three parts of remains unchanged and two parts are changed.

Strategies for changing the level of detail and viewpoint without changing the selected modification context are, for example,

- a combination of strategies described in section 5.1.1 and section 5.1.2.
- implications of a change within the modification context need to be verified at the other level of detail from another point of view.

Strategies for changing the level of detail and modification context without changing the selected viewpoint are, for example,

- a combination of strategies described in section 5.1.1 and section 5.1.3
- having switched from the other level of detail and having analysed the

implications of the modification, switch back to the original level of detail and switch to the next modification contextwithin the same viewpoint.

Strategies for changing the viewpoint and modification context without changing the selected level of detail are, for example,

- a combination of strategies described in section 5.1.2 and section 5.1.3.
- One part of the agent has been succesfully modified, subsequent modifications need to take place in other parts of the agent from a different point of view, e.g. when working from the outside of an agent's architecture to the inside.

5.2.4 Complete change of design focus

A complete change of design focus means that the new design focus does not resemble the previous design focus at all: different levelof detail, viewpoint, and modification context are selected.

Strategies for complete changing a design focus are, for example,

- a combination of strategies described in section 5.1.1, section 5.1.2, and section 5.1.3.

5.3 EXAMPLE STRATEGIES

In this section an example is given of strategies employed in the agent factory. An overview of the re-design of the example information retrieval agent is given in section 5.3.1. Strategies employed for inserting a component building block into an open slot are given in section 5.3.2.

5.3.1 Re-design of information retrieval agent

A prototype of the agent factory automatically designs an information retrieval agent. Building blocks are combined at a conceptual and a detailed level of design culminating in a simple information retrieval agent. The composition of building blocks of an existing simple information retrieval agent has been described in Section 3.3. In this section strategies are discussed with which an information retrieval agent is re-designed.

The existing information retrieval agent is capable of searching for information on html-pages, available via the http-protocol. A new requirement imposed on this agent is that the agent needs to be capable of using the ftp-protocol to retrieve information. The origin of this requirement lies, in this example, outside of the agent; its owner may have posed this requirement. A more complex agent may deduce, on the basis of analysis of its own behaviour, that it is lacking certain capabilities and contact an agent factory to be re-designed.

The information retrieval agent needs to be changed in a number of ways. The agent needs to know (1) that multiple world interaction protocols exist, (2) its own preferences between protocols, (3) how to compose queries and

(4) how to process query results.

This implies that the components in the slots in own process control, world interaction management and agent specific task need to be modified.

The World Interaction Management slot of the simple information retrieval agent, which originally contained only the http protocol, needs to be modified. The http component needs to be replaced by a building block with three components: one for http world interaction, one for ftp world interaction, and one to determine which protocol to choose. Figure 5 depicts the three components in the open slot in world interaction management.

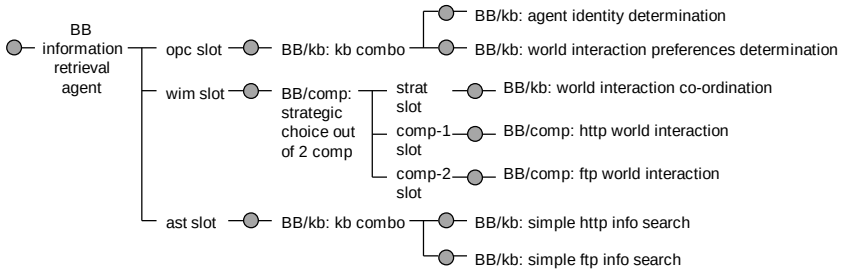

Figure 5. Process composition of the new information retrieval agent

Figure 5 shows the new process composition based on the simple information agent retrieval template. The component open slots of the simple agent information retrieval template have been filled in as follows:

- The own process control slot now contains a building block with two components: a knowledge-base to determine an agent's identity and capabilities, and a knowledge-base to determine preferences on world interaction protocols.
- The world interaction management slot contains a building block with a composed component for strategic process co-ordination of independent processes. This new building block contains three open slots. The first open slot is for strategic knowledge to co-ordinate the processes. The second open slot is is for one of the protocols for world interaction: the http world interaction. The last component slot is for the other protocol for world interaction: ftp world interaction.
- The agent specific task slot contains a building block with a number of components: a component to determine how information can be acquired via html/http, and a component to determine how information can be acquired via ftp.

The new process hierarchy of the information retrieval agent is shown below in Figure 6.

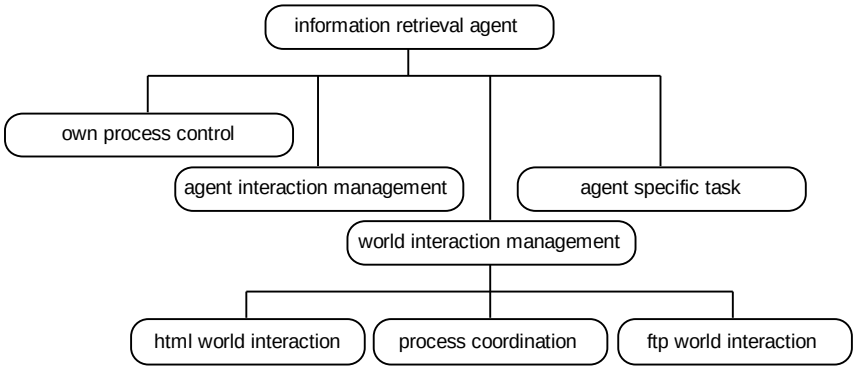

Figure 6. New process hierarchy of the information retrieval agent.

A number of information types also need to be extended to incorporate the terminology related to ftp. The information type observation results information, shown in Figure 7, needs to be extended to include the information type ftp results. In addition, the information type observation results information also needs to include language elements to express results from both http and ftp actions.

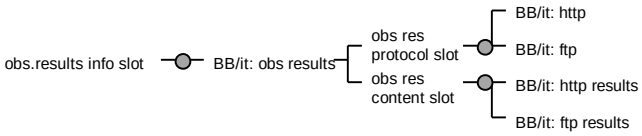

Figure 7. Extension of the information type observation results information with ftp observation results.

The information type observations, shown in Figure 8, now also includes the information type ftp observations. In addition, the information type observations has been extended to include language elements to express both http and ftp observation actions.

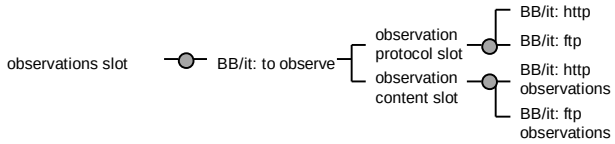

Figure 8. Extension of the information type observations with ftp observations.

The information type actions, shown in Figure 9, has been extended to include the information type ftp actions. In addition, the information type actions has been extended to include language elements to express both http and ftp actions.

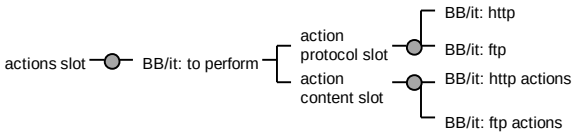

Figure 9. Extension of the information type actions with ftp actions.

The information type own characteristics, shown in Figure 10, has been extended with the information type protocol preferences. In addition, the information type own characteristics has been extended to include language elements to express preferences on http and ftp protocols.

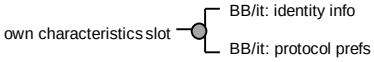

Figure 10. Extension of the information type own characteristics with protocol preferences.

The transformation of the information retrieval agent capable of interacting via http into an information retrieval agent capable of interacting via http and ftp, required the modifications described above. Subsequent additions of world interaction protocols are straightforward - the relevant building blocks are added next to the existing building blocks for html/http and ftp.

5.3.2 Strategies for building block insertion: a trace

To illustrate the use of strategies in parts of the example described in the previous section a trace of the process is discussed in more detail.

Assume the design process has reached the point where both the conceptual and detailed building blocks for interacting with the world using ftp have been found in its library. One of the requirements specified explicit strategic co-ordination of the use of the http world interaction or ftp world interaction components.

In this example strategies are described with which a component building block is inserted into an open slot at two levels of abstraction. Most structure is defined at the conceptual level, so this is the first level of abstraction the design process focuses on. The first design focus is on the process composition point of view, in the context of the world interaction management open slot of the agent building block ("wim slot").

> Design_focus_1(conceptual level, process composition viewpoint, wim slot)
> The library is queried for a component that can be filled in with both protocols and exert strategic process co-ordination over these protocols. The conceptual building block named ‘BB/comp:strategic choice out of two components’ is returned. This building block replaces the building

block ‘BB/comp:http world interaction’ in the slot of wim.

Affixing a component building block into an open slot entails not only defining that the building block is inserted into the open slot, but also information flow and control flow need to be taken care of. First, it is decided to connect the information flow of the wim-slot to the inserted building block. For this a change in design focus is necessary; the information exchange viewpoint is useful here.

> Design_focus_2(conceptual level, information exchange viewpoint, wim slot)
> Communication links are made from the input-side of the wim-slot to the input-side of the inserted building block, and from the output-side of the inserted building block to the output-side of the wim-slot. The definition of the new building block states that it needs all information on the input, and produces all information on the output, of the component with the relevant open slot. Linking the information is therefore quite straightforward.

If the strategy for switching between viewpoints is strictly regimented, it may now also need to change focus to the knowledge viewpoint.

> Design_focus_3(conceptual level, knowledge viewpoint, wim slot)
> No knowledge modifications need to be made.

The building block is inserted in the wim slot, and information exchange has been arranged within the wim-slot. To also arrange the transfer of control, a new design focus is needed from the control point of view.

> Design_focus_4(conceptual level, control viewpoint, wim slot)
> The control over the inserted component needs to be set. The default option for the inserted component is to be awake, in order not to conflict with its strategic co-ordination of its control over its sub-components.

The building block has been completely inserted at the conceptual level. It is not yet known whether the associated detailed building block may also be inserted into the detailed agent-building block, at the detailed level.

A similar order of viewpoints is chosen as before, only now at the detailed level of abstraction. The first focus at this level of detail is from the process composition point of view, in the context of the world interaction management open slot.

> Design_focus_5(detailed level, process composition viewpoint, wim slot)
> Modifications to the Java code:

```
// the open slot of wim is emptied (http bb/component
// is removed)
BB_agent.getSlot(“wimslot”).clean_slot();
// the detailed ‘BB/comp:strategic choice out of two
// components’ is created (using class name 'BB_Strat_23').
BB_strat  = new BB_Strat_23();
// insert BB_strat in the wim-slot
BB_agent.getSlot(“wimslot”).fillCompSlot(BB_strat);
```

After inserting the building block in the slot, the information exchange between the slot and the building block needs to be arranged. The information exchange viewpoint is chosen, the remaining parts of the design focus are unchanged.

Design_focus_6(detailed level, information exchange viewpoint, wim slot)
Modifications to the Java code:

```
Wim_slot = BB_agent.getSlot(“wimslot”)
// create a link from input-side of wim slot to input-side
// of inserted component
Wim_slot.addInputLinkTo(Wim_slot.getInsertedComp());
// create a link from output-side of inserted component
// to output-side of wim slot
Wim_slot.addOutputLinkTo(Wim_slot.getInsertedComp());
```

Having inserted the building block, and arranged its information exchange, the last step is to arrange for the newly inserted building block to be activated when necessary. To this end the control viewpoint is adopted in the new design focus.

Design_focus_7(detailed level, control viewpoint, wim slot):
Modifications to the Java code:

```
Wim_slot = BB_agent.getSlot(“wimslot”)
// the control over the inserted component is set on awake
Wim_slot.transferControlTo(Wim_slot.getInsertedComp());
Wim_slot.getInsertedComp().setControl(“awake”);
```

The open slot of the component world interaction management of the agent building block has been filled on both the conceptual level and the detailed level.

6. Discussion

Automated design requires explicit representation of strategic knowledge. Reasoning required to (re-)design a software agent includes strategic

knowledge on how and when to focus on specific aspects/characteristics of an agent. Viewpoints provide a means to structure/group such aspects/ characteristics. Given these viewpoints specific knowledge is required on how and when to focus on each of the viewpoints, and when to co-ordinate results. In itself, nothing new: such strategic reasoning is common in design. Making such strategies explicit is, however, non-trivial. In particular when also considering different levels of detail and the interaction between these levels. In the automated agent factory in which agents are configured two levels of detail are used. A simplification of many complex design situations, but complex enough to require specific strategies during design.

This paper presents a classification of strategies needed for automated agent design in which reasoning with and about viewpoints is modelled by design foci. Each design focus defines a viewpoint, a level of detail and a modification context. Strategies are defined for each of these factors, and for the choice of design foci at the global level.

To automatically re-design a software agent, four viewpoints are of importance, knowledge of which is needed at both conceptual and detailed level. This paper illustrated strategic knowledge needed to re-design a simple information retrieval agent. It is clear that a better understanding of "good" strategies is needed. More research may help.

Fully automated design of software agents of any type requires knowledge of the domains of application and design strategies. A compositional approach helps but is not sufficient: libraries of generic agent architectures, domain specific components, strategic components, and ontologies are prerequisites to success. Some success is being booked for specific types of internet agents.

Acknowledgements

The authors wish to thank the graduate students Hidde Boonstra, David Mobach, and Oscar Scholten for their explorative work on the application of an agent factory for an information retrieving agent. This work was supported by NLnet Foundation, http://www.nlnet.nl.

References

Attardi, G. and Simi, M.: 1994, Proofs in Context, *in* L. Fribourg and F. Turini (eds), *Logic Program Synthesis and Transformation-Meta-Programming in Logic*, Proceedings of the Fourth International Workshop on Meta-Programming in Logic, META'94. Springer Verlag, Lecture Notes in Computer Science, **883**, pp. 410-424.

Bahler, D., Dupont, C. and Bowen, J.: 1994, Anaxiomatic approach that supports negotiated resolution of design conflicts in concurrent engineering, *in* J. S. Gero, F. Sudweeks (eds), *Artificial Intelligence in Design*, Dordrecht: Kluwer Academic Publishers, pp.

363-379.

Balasubramanian, S. and Norrie, D. H.: 1996, A multiagent architecture for concurrent design, process planning, routing, and scheduling, *Concurrent Engineering: Research and Applications*, **4**(1), pp. 7-16.

Brazier, F. M. T. and Wijngaards, N. J. E.: 2001, Automated servicing of agents. D. Kudenko & E. Alonso (eds), *Proceedings of the AISB-01 Symposium on Adaptive Agents and Multi-agent systems, at the Agents & Cognition AISB-01 conference*, the society for the study of artificial intelligence and the simulation of behaviour, ISBN 1.902956.17.0, pp. 54 - 64.

Brazier, F. M. T. and Wijngaards, N. J. E.: 2001b, Designing Self-Modifying Agents. To appear in proceedings of Computational and Cognitive Models of Creative Design, the fifth international roundtable conference.

Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N.R. and Treur, J.: 1995;1997, Formal specification of Multi-Agent Systems: a real-world case, *in* V. Lesser (ed), *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95*, Cambridge MA: MIT Press, pp. 25-32. Extended version in M. Huhns and M. Singh (eds), *International Journal of Co-operative Information Systems*, special issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems; **6**, pp. 67-94.

Brazier, F. M. T., Jonker, C. M. and Treur J.: 1996, Modelling project co-ordination in a multi-agent framework, in *Proceedings Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96*, Los Alamitos: IEEE Computer Society Press, pp. 148-155.

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

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

Brazier, F. M. T., Jonker, C. M., Treur, J. and Wijngaards, N. J. E.: 2000, Deliberate Evolution in Multi-Agent Systems, *in* J. Gero (ed.), *Proceedings of the Sixth International Conference on AI in Design, AID'2000*. Kluwer Academic Publishers, 2000, pp 633-650.

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

Brazier, F. M. T., Langen, P. H. G. van and Treur, J.: 1995, Modelling conflict management in design: an explicit approach, *Artificial Intelligence for Engineering Design, Analysis and Manifacturing, (AIEDAM), in* I. F. C. Smith (ed.), Special Issue on Conflict Management in Design, **9**(4), 353-366.

Brazier, F. M. T., Langen, P. H. G. van and Treur, J.: 1997, A compositional approach to modelling design rationale, *Artificial Intelligence for Engineering Design, Analysis and Manifacturing, (AIEDAM), in* pp. W. H. Chung and R. Banares-Alcantara (eds), Special Issue on Representing and Using Design Rationale, **11**(2), 125-139.

Brazier, F. M. T., Langen, P. H. G. van and Treur, J.: 1998, Strategic Knowledge in Compositional Design Models, *in* J. S. Gero and F. Sudweeks (eds), *Proceedings of the Fifth International Conference on Artificial Intelligence in Design, AID'98*, Kluwer Academic Publishers, Dordrecht, pp. 129-147.

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

Campbell, M. I., Cagan, J. and Kotovsky, K.: 1998, A-Design: theory and implementation of an adaptive, agent-based method of conceptual design, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '98 (AID '98)*, Dordrecht: Kluwer Academic Publishers, pp. 579-598.

Cimatti, A. and Serafini, L.: 1995, Multi-agent Reasoning with Belief Contexts II: Elaboration Tolerance, *in* V. Lesser (ed), *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95*, MIT Press, pp. 57-64.

Clancey, W.J. and Bock, C.: 1988, Representing control knowledge as abstract tasks and metarules, *in* L. Bolc and M. J. Coombs (eds), *Computer Expert Systems*, Heidelberg: Springer-Verlag, pp. 1-77.

Cutkosky, M., Engelmore, R., Fikes, R., Gruber, T., Genesereth, M., Mark, W., Tenenbaum, J. and Weber, J.: 1993, PACT: An experiment in integrating concurrent engineering systems, *Special Issue on Computer Support for Concurrent Engineering, IEEE Computer*, **26**, pp. 28-37.

Davis, R.: 1980, Metarules: reasoning about control, *Artificial Intelligence*, **15**, pp. 179-222.

Dennett, D. C.: 1987, *The Intentional Stance*, MIT Press, Cambridge.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M.: 1992, Viewpoints: a framework for integrating multiple perspectives in system development, *International Journal of Software Engineering and Knowledge Engineering*, **2**(1), pp. 31-58, http://www.doc.ic.ac.uk/~ban/pubs/ijseke92.pdf

Fisher, M. and Wooldridge, M.: 1993, Specifying and Verifying Distributed Intelligent Systems, *in* M. Filqueiras and L. Damas (eds), *Progress in AI. Proc. EPAI'93*, Springer Verlag, Lecture Notes in AI, **727**, pp. 13-28.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: 1994: *Design Patterns: Elements of reusable object-oriented software*. Reading, Massachusetts: Addison Wesley Longman.

Goldmann, S.: 1996, Procura: a project management model of concurrent planning and design, *in Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96*, Los Alamitos: IEEE Computer Society Press.

Grecu, D. L. and Brown, D. C.: 1996, Learning by single function agents during spring design, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '96 (AID '96)*, Dordrecht: Kluwer Academic Publishers, pp. 409-428.

Gruber, T. R.: 1990, Acquiring strategic knowledge from experts, *in* J. H. Boose and B. R. Gaines (eds), *The Foundations of Knowledge Acquisition*, Knowledge Based Systems, **4**, Academic Press Limited, pp. 115-133.

Gupta, L., Chionglo, J. and Fox, M.: 1996, A constraint-based model of communication and co-ordination in concurrent design projects, *in Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96*, Los Alamitos: IEEE Computer Society Press.

Haymaker, J., Ackermann, E. and Fischer, M.: 2000, Meaning Mediating Mechanism, *in* J. S. Gero, (ed), *Artificial Intelligence in Design '00*, Proceedings of the Sixth International Conference on AI in Design, AID'2000, Dordrecht:Kluwer Academic Publishers, pp. 691-715.

Hori, K.: 1998, Special issue on strategic knowledge and concept formation, *Knowledge Based Systems*, **11**.

Jackson, M. A.: 1975, *Principles of Program Design*, Academic Press.

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

Klein, M.: 1995, Conflict management as part of an integrated exceptionhandling approach. *AIEDAM*; **9**, pp. 259-267.

Löckenhoff, C. and Messer, T.: 1994, Configuration, *in* J. A. Breuker and W. van de Velde (eds), *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, chapter 9, pp. 197-212.

Logan, B. S. and Smithers, T.: 1992, Creativity and design as exploration, *in* J. S. Gero and M. L. Maher (eds), *Modelling Creativity and Knowledge-Based Creative Design*, Hillsdale: Lawrence Erlbaum, pp. 149-188.

Maes, P. and Nardi, D. (eds): 1998, *Meta-level architectures and reflection*. Elsevier Science Publishers.

Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kötting, B. and Schaaf, M.: 2000, Merging project planning and web-enabled dynamic workflow techniques, *IEEE Internet Computing*, pp. 65-74.

McAlinden, L. P., Florida-James, B. O., Chao, K-M., Norman, P. W., Hills, W., Smith, P.: 1998, Information and knowledge sharing for distributed design agents, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '98 (AID '98)*, Dordrecht: Kluwer Academic Publishers, pp. 537-556.

O'Hare, G. M. P.: 1996, Agent Factory: An Environment for the Fabrication of MultiAgent Systems, *in* G. M. P. O'Hare and N. R. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, Wiley Interscience, pp. 449-484.

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

Peña-Mora, F. and Vadhavkar, S.: 1996, Design Rationale and Design Patterns in Reusable Software Design, *in* J. S. Gero and F. Sudweeks (eds.), *Artificial Intelligence in Design (AID'96)*, Dordrecht: Kluwer Academic Publishers, pp. 251-268.

Petrie, C.: 1994, Design space navigation as a collaborative aid, *in* J. S. Gero, (ed), *Artificial Intelligence in Design '94, Proceedings AID '94*, Dordrecht: Kluwer Academic Publishers, pp. 611-623.

Riel, A. J.: 1996, *Object-Oriented Design Heuristics*. Reading Massechusetts: Addison Wesley Publishing Company.

Reticular Systems Inc: 1999, AgentBuilder: An integrated toolkit for constructing intelligent software agents. *White Paper*, http://www.agentbuilder.com, February 1999.

Rist, R. S.: 1995, Program structure and design, *Cognitive Science*, **19**, pp, 507-562.

Rumbaugh, J., Jacobson, I. and Booch, G.: 1999, *The unified modeling language reference* manual. Reading, Massachusetts: Addison Wesley.

Schön, D. A.: 1983, *The Reflective Practioner: how professionals think in action*, Aldershot (England): Arena.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W. and Wielinga, B.: 1991, *Knowledge Engineering and Management, the CommonKADS Methodology*. MIT press.

Stefik, M.: 1995, *Introduction to Knowledge Systems*, Morgan Kaufmann Publishers, Inc, San Francisco, California.

Strelnikov, Y. N. and Dmitrevich, G. D.: 1991, Formal description and comparison of interactive design strategies, *Artificial Intelligence in Engineering*, **6**(4), pp. 186-195.

Wagner, G.: 1996, A Logical and Operational Model of Scalable Knowledge- and Perception-based Agents, *in* W. van der Velde and J. W. Perram (eds), *Agents breaking away*, *Proc. MAAMAW'96*, Springer Verlag, Lecture Notes in AI, **1038**, pp. 26-41.

Weyhrauch, R. W.: 1980, Prolegomena to a theory of mechanized formal reasoning, *Artificial Intelligence*,**13**, pp. 133-170.

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