

Knowledge level model of an individual designer as an agent in collaborative distributed design*

Frances M.T. Brazier¹, Lilia V. Moshkina², Niek J.E. Wijngaards¹

¹ Intelligent Interactive Distributed Systems Group
Department of Artificial Intelligence, Division of CS,
Faculty of Sciences, Vrije Universiteit Amsterdam
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

Email: {frances,niek}@cs.vu.nl

Fax: +31 - 20 - 444 7653

² Georgia Institute of Technology
College of Computing
801 Atlantic Drive
Atlanta, GA 30332-0280, USA
Email: lilia@cc.gatech.edu

Abstract. In this paper a knowledge-level model of an individual designer as an agent is described, in which reflective reasoning about elements of situatedness, and reasoning from the point of view of other participants, are explicitly modelled. This model is based on existing models of single agent design. An individual designer in a specific distributed design process, namely website design, is used to illustrate the model.

Keywords. distributed design, multi-agent systems, reflection, cooperation, situatedness.

1. Introduction

Collaborative distributed design is becoming more prevalent in current practice. Multiple participants (customers, designers, manufacturers and other stakeholders) work together, but not necessarily at the same time, at the same location, or with the same resources. Although the technology to support exchange of information between participants is available, more content related support is not. Existing knowledge level models of design focus on design as a single agent process. Knowledge-level models of collaborative distributed design are needed. Namely the situation in which an individual designer contributes to a collaborative distributed design process differs significantly from the situation in which an individual designer completes an entire design project on his/her own. In a distributed design process an individual designer reasons about several elements of a situation including design partners, design culture, and (shared) understanding of the design problem, that are irrelevant to single agent design.

This paper focuses on the implications of distributed design for individual design processes. An individual designer is considered to be an agent, in a multi-agent setting. The types of reasoning and knowledge that need to be included in knowledge level models of individual agents that collaborate in a distributed design process are analysed and specified.

* CONCEPT VERSION. Appeared in *Artificial Intelligence in Engineering*, volume 15, 2001, pp. 137-152.

In single agent design, reflective reasoning is employed to reason about the progress of the design process, and e.g., to reason from a domain-specific point of view. An artefact to be designed may be designed from several viewpoints: a heating-system viewpoint can co-exist with an electrical-system viewpoint. A single design agent can choose which viewpoint to work from at each point during a process.

In distributed collaborative design, individual designers need to be able to reason reflectively about other designers. The knowledge used for reflective reasoning is often incomplete and imprecise (this holds for design in general as well, see Section 2.1) so conclusions can be assumed to hold, until the opposite is discovered through interaction with other agents. This is a form of hypothetical reasoning.

The main purpose of this study is to devise a formal knowledge-level model of an individual agent capable of reasoning about other agents (their knowledge, experience and results), and about the need for interaction (and content) during a design process. In Section 2 relevant literature is briefly discussed. Section 3 discusses a number of differences between distributed design settings and single design agents. In Section 4 an example of a distributed design process is introduced. This example domain is used to illustrate the types of knowledge and reasoning processes included in the knowledge level model. Section 5 presents a knowledge-level model of a co-operative design agent, based on existing generic models of agents. Section 6 depicts types of reasoning and knowledge required for distributed design extracted from the formal specification of the knowledge level model. In Section 7 knowledge fragments illustrate an implementation of the model described in Section 6 for the example described in Section 4. Section 8 discusses the results and proposes future work.

2. Research on distributed design

Distributed design is clearly related to a large number of disciplines. This paper focuses on additional types of reasoning and knowledge that need to be included in knowledge level models of individual agents that collaborate in distributed design processes. Results from three related areas of research are addressed below: single agent design, concurrent engineering and multi-agent systems.

2.1 Single Agent Design

In the past few decades, research on *design* has led to a number of theories and models of design as a task and as a process (e.g. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]). Interesting enough, formal knowledge-level models for a process of design which includes the notion of manipulating requirements as well as manipulating an artefact description have surfaced mainly in the last decade [9, 10, 12, 13, 14]. In addition, design methods have been developed (e.g. [15, 16, 17]), as have design support systems (e.g. [18, 19, 20, 21, 22]). These theories, models, methods, and systems, however, do not explicitly model reflective reasoning required for multi-agent distributed design.

A compositional *generic design model* (*the GDM* -see Section 5.2) in which explicit reasoning about requirements and their qualifications, reasoning about design object descriptions and reasoning about the design process are distinguished, is based on a logical (formal) analysis of design processes [12] and on analyses of applications, including elevator configuration [23] and design of environmental measures [24]. Within this knowledge-level model different levels of reflection are distinguished enabling explicit representation of design strategies and design

process progress monitoring. One of the advantages of distinguishing different levels of reflection is to explicitly model strategic reasoning in design [25]. Additional types of reflective reasoning are, however, involved when designers need to reason about other designers' reasoning.

2.2 Concurrent engineering

A variant of distributed design is *concurrent engineering*. In the field of concurrent engineering, research and development has resulted in methods and means that enable customers, engineers and manufacturers to co-operate in complex engineering tasks (such as the development of computer software, consumer electronics, cars, aircraft and ships) often at one physical location.

As a result of the technological advances made in the last decades (notably the Internet) designers in different parts of the world work around the clock, separately or together, on the same project. The co-ordination of these projects in virtual environments, in particular the co-ordination of conflicting (partial) designs, interests, models, requirements (e.g., new requirements imposed during design), etc., requires extensive knowledge of the design process, of the available expertise and skills, of dependencies and, in particular, of the consequences of modification. For this reason, models supporting a common understanding of agents about a specific task (e.g. [26]) and models of distributed project management (e.g. [27, 28]) are becoming increasingly important. A number of tools and services have been designed to support specific aspects of the co-ordination process; for example [29, 30, 31, 32, 33, 34, 35, 36, 37].

2.3 Multi-agent systems

Research in *multi-agent systems* is currently mainstream AI. Different notions of agency have been proposed (e.g. [38, 39, 40, 41, 42]). One notion of agents in which weak agency is distinguished from strong agency has been proposed by Wooldridge and Jennings [39]: weak agency is characterised by autonomy, social ability, reactivity, 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 [43]). The characteristics of weak agency defined by Wooldridge and Jennings [39] 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. Reactivity and social ability are related to the ability to interact with the material world and to communicate with other agents. The ability to communicate and co-operate with other agents and to interact with the material world often relies on an agent's ability to acquire and maintain its own knowledge of the world and other agents.

By studying the ways that single agents and groups of single agents interact, theories and models of co-operation have been and are still being developed that can be used for advanced human-computer interaction and for the development of multi-agent systems. A typical example of research in multi-agent systems that is relevant to the design community is research on information brokering and information gathering agents [44, 45, 46, 47].

The agent metaphor offers a means to model situations with distributive activity on a conceptual level. A number of researchers, but not many, have combined the research areas of design and multi-agent systems. For example, Grecu and Brown describe a system for parametric spring design [48], built from small knowledge-based systems called Single

Function Agents (SiFAs). The deliberately restricted capabilities of each SiFA enforce the interaction among SiFAs; only by co-operation are they able to carry out a design task. Different types of SiFAs exist, each having a single target, which is to select, estimate, evaluate, criticise or praise parameter values. The ability of SiFAs to learn reduces the number of conflicts during a design process.

Campbell, Cagan and Kotovsky present a theory of engineering design, A-design, that models an engineering process as a complex adaptive system of interacting software agents [49]. In A-design, configuration agents create conceptual designs; instantiation agents fill these in with actual components from a catalogue. Fragment agents and subsystem agents come into play after evaluation. Fragment agents can remove 'bad' components, whereas subsystem agents extract 'good' assemblies (consisting of multiple components) and store them in the catalogue for future use. Manager agents maintain these four types of agents; e.g. the number of agents of a specific type can be adjusted, depending on contributions of agents to 'good' and 'bad' designs.

McAlinden, Florida-James, Chao, Norman, Hills and Smith [50] describe how design agents can be integrated to facilitate information and knowledge sharing. In this approach, a central product model of the STEP standard is used, as well as ACL and knowledge-based ontologies. Their aim is to incorporate existing and legacy systems without delay in a design project.

The aforementioned research on distributed design [48, 49, 50] does not include explicit representations for reflective reasoning. Being able to reason about, or even from, the viewpoint of another agent is a means with which, e.g., conflicts can be prevented. In the literature on reflection such as [50, 51, 52, 53, 54] 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 [55, 39, 56, 57] 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.

3. Distributed design vs. single agent design

As stated above existing models and theories do not explicitly capture the distributed nature of design projects, in particular not with respect to reasoning about (the knowledge of) other participants and the types of interaction required. As, in practice, many design projects involve collaboration between groups of designers such types of reasoning need to be made explicit. The situation within which an individual agent participates in a distributed design process differs considerably from the situation in which a single agent performs a complete design process on his/her own.

3.1 Elements that define a situation

Each of the designers in a distributed design process is an individual, with his/her own goals, commitments, perspectives on the artefact being designed, etc. Each individual designer interacts with other designers to collectively produce a satisfactory solution to a design problem. The decisions taken by individual designers in a distributed design setting are not only influenced by the specific design knowledge available to an individual designer, but also by the individual agent's interpretation of his/her own *situation*. Each individual designer has

his or her own view of the world and other agents. Each individual designer needs to be able to reason explicitly about the situation in which the design process is to be performed.

In the RoboCup experiments [58, 59] agents are situated in a dynamic environment containing other agents and a playing field. Soccer players are all capable of the same tasks while this is not, in general, the case in distributed design where each designer has his or her own expertise.

Situatedness influences the thoughts and actions of a designer [60, 61]. In distributed design processes a form of shared understanding among designers is assumed (or acquired in some way): at the very least some designers are able to “understand” other designers’ designs to some extent. Such shared information forms the basis of collaborative endeavours of co-operating individual design agents.

Reasoning from different *viewpoints* is a necessary part of most design processes. Domain specific contexts are often the grounds for such viewpoints. As an example, consider the design of a building, in which it is useful to distinguish different systems involved: e.g. electrical, sewage, and heating. Models of single agent design most often incorporate reasoning from different viewpoints, including strategic reasoning about viewpoints, and reasoning from the perspective of other viewpoints. In distributed design, however, an individual designer also has to be able to reason about, e.g., other designer’s knowledge and expected actions. An individual design agent often reasons hypothetically about other designers’ reasoning processes, by reasoning from the point of view of another designer, and testing whether those conclusions conflict with the current version of the individual designer’s current design solution. The change in the aspects that determine the situation in which an agent function require different types of reflective reasoning [62].

A number of specific elements that determine the situation in an individual designer functions, can be distinguished. These elements can be categorised by their source: the client, the designer him- or herself, and the design environment. Elements of outside of the design problem or design process (such as recent experiences) are not taken into account in this approach; their influence on an individual designer, however, cannot be disregarded.

The context of a given design problem is an obvious element: a designer often has a given set of requirements, (possibly) a given description of an existing object to be re-designed, and design process objectives: these are goals (or constraints) a designer has to adhere to while performing his or her design task. Such goals can e.g. limit the time spent on the design task. Designers may differ in the way in which they interpret these types of information. Each individual designer has his/her own view on the design process objectives: which goals and constraints have to be achieved by the design process. Design solutions are specific to (groups of) individual designers: some may share a (partial) solution, other may share a different solution. The context of design requirements is also unique for an individual designer: some design requirements may be shared with other designers, other requirements are specific to the individual designer. The notion of progress of the design process is relative to an individual designer.

Design knowledge is directly related to a designer him or herself: each individual designer has limited design knowledge and experience. Knowledge about his or her limitations can aid a designer in avoiding dead-ends. Each designer has his or her own history of (un)succesful

designs (e.g., a case-base, statistics on useful approaches) and his/her interpretation of the processes themselves.

The design environment is also clearly of importance. Each designer has his or her knowledge of, and experiences with, other designers, i.e. the design partners. Design resources are not available to all designers, each has access to specific resources, some of which may be shared. The culture of a group, sometimes expressed as the collection of norms and values, may also influence interaction among members of a group. An individual designer is part of a design culture in which he/she defines his/her own role. Each designer may consciously choose to adhere to, or violate, norms and values within the design culture, e.g., the shared view on the progress of the (overall) design process. Some knowledge may be available to all designers and forms a shared context of design knowledge.

The elements that are relevant to an individual designer in a distributed setting are summarized in Table 1.

Element in situation	Explanation
<i>Related to client:</i>	
Given problem	is divided into two parts: given sets of design requirements; given initial design object descriptions.
Design process	Client's objectives on the design process (e.g., in terms of time, money, resources used, number of results).
<i>Related to designer:</i>	
Design knowledge	design knowledge available to the designer.
Design experience	data (e.g., based on experience, case bases) available to the designer.
<i>Related to environment:</i>	
Design partners	design partners; their capabilities and trustworthiness.
Design resources	resources available to a designer (possibly shared).
Design Culture	culture (including norms and values) among (groups of) designers.
Design problem	shared information on the design process objectives; (partial) solutions to the design problem at hand, consisting of sets of (partial) design requirements, and (partial) artefact descriptions.
Design progress	current, and previous, state of the design process, as performed by designers in isolation and in co-operation.
Design knowledge	knowledge available to all designers about other designers, design processes, etc.

Table 1. Description of elements that (partially) define a situation for distributed design.

3.2 Reasoning about a designer's situation

An individual designer that participates in a distributed design process has to be able to reason about the situation in which design is required. An individual designer needs to interact with other agents and the environment, but when and how is, in general, up to the individual designer. An individual designer may, for example, decide not to inform a project manager about an expected delay, given his/her own expectations with respect to the consequences.

The expected impact of the result of an interaction on the (re-)design process influences whether an interaction is effectuated. For example, a routine interaction may result in

information that does not much affect the progress of the design process, and can be postponed. Yet another routine interaction about, e.g., the cost of the roof of a building, may significantly influence the progress of the design process, as subsequent choices are influenced, and earlier solutions possibly discarded.

An individual designer needs to be aware of both the complexity of the interactions, and the expected impact these interactions may have on the progress of the design process. An explicit decision needs to be made about which interaction is performed in which manner.

An individual designer also has his or her own opinion of the level of expertise of other design partners (most often based on experience and hearsay); knows about their social skills, their ability of consensus making in negotiations, etc. This knowledge influences the way in which an individual designer reasons about the other design partners, and as a result the way in which an individual designer interacts with these other design partners (e.g., if previous experience with a highly skilled designer in another field increases an individual designer's knowledge of this field, he/she may be aware of alternatives, that may not be devised by another designer in the same field in a similar context).

4. Example of an individual design agent in a distributed design process

In this paper an example of distributed website design is used to illustrate the types of reflective reasoning required by an individual design agent involved in a distributed design process. This example entailed a distributed design process, where several participants needed to interact with each other to be able to design an artefact. The design process took place in the domain of website design¹, not physical artefact design. The interaction required is, however, comparable to the design of a physical artefact. First, an overview of the case study is given, then a more detailed description.

4.1 Assistive Technology Website Design Project

The design project under analysis is a project to design a complex and extensive website on disability-related resources providing up-to-date, thorough information on assistive technologies (technologies intended to provide assistance to people with disabilities), adaptive environments and community resources for the disabled. In addition to general requirements imposed on website design (e.g., consistency, relevancy of information, clear organization, etc.), the satisfaction of the following requirements was imposed:

- ? the site must be accessible by people with disabilities;
- ? vendors of assistive technologies must be able to modify related information with ease and accuracy via a Vendor Data Entry Interface (VDEI);
- ? the information must be accurate and up to date; and
- ? the data entry process must be secure, and the information verified before committing to permanent changes.

¹ The authors express their gratitude to the development team of the Assistive Technology Website Design Project at the Georgia Institute for Technology for their co-operation in the research by providing the necessary information about the design process.

Due to a variety of skills required in the project execution, a number of different specialists were needed: a Project Manager, a Web Designer, a Database/Systems Administrator, a Assistive Technology Specialist, a HTML designer, and a Dynamic Web Designer.

4.2 Design of Vendor Data Entry Interface

The design process focussed on in this paper spanned over 6 weeks from inception to completion of the design of part of the above specified website. This particular design process was performed by the Dynamic Web Designer and has been analyzed in detail. This process was chosen for a number of reasons:

- ? first of all, expert knowledge about the process was fully available;
- ? the individual design process required specific skills: dynamic pages development, and was a primary responsibility of a single designer yielding an individual designer's perspective in a distributed context; and
- ? finally, as certain requirements and parts of the website were shared by a number of designers (e.g., both the Database Administrator and the Dynamic Web Designer were involved in designing and modifying the Vendor database), the design process required extensive interaction among team members.

In this process, three phases are distinguished: acquisition of requirements, design of the VDEI, and evaluation of the VDEI.

The focus of the analysis is on types of reflections/viewpoints exhibited during the Vendor Data Entry Interface design process. The purpose of Vendor Data Entry Interface is to provide vendors of assistive technology products and devices as well as employees of the organization supporting the website with an easy way to modify vendor and product-related information on the web. The following are general requirements:

- ? vendor and product-related information must be accurate and up to date;
- ? the interface must be easy to use: or, stated differently, the interface should not require special skills from the users;
- ? the VDEI must be consistent with the rest of the website.

Early in the project it was decided that the VDEI would be designed as a dynamically generated website, with data being read from and written to a database at the time of user interaction. This decision affected security and accuracy aspects, and resulted in a number of more specific requirements:

Security requirements:

- ? only authorized users must be able to change, add or delete information on the website;
- ? vendors (a category of authorized users) must be able to make changes to product, company or contact info pertaining only to the company they represent;
- ? only users given administrative privileges must be able to access the administrative section (the highest level of privileges);

Information accuracy requirements:

- ? product, company and contact info must be accurate and up to date;

The decision to design the VDEI as a dynamically generated website also implied that four of the aforementioned specialists would have to interact closely in the design process: the Project Manager, the Database Administrator, the Web Designer, and the Dynamic Web Designer.

4.2.1 Requirements acquisition

Initial design process objectives (e.g. how much time was allotted for the design, what infrastructure was available, etc) and initial requirements were communicated to the Dynamic Web Designer at the beginning of the project. After a preliminary analysis of the received information it became clear that additional requirements acquisition was necessary, in particular the following information was missing: current overall design description, specific Vendor Data Entry Interface Requirements, and a Vendor database description.

The requirements acquisition process included a number of specific subtasks (the following order is not imposed):

- ? identifying an appropriate team member from whom to request information (to include requesting information about specific team members);
- ? requesting information from other designers/manager, be it requirements or their specification, details of existing parts of the design, or any other project-related information;
- ? clarifying information in case the information received is vague, unclear or contradictory;
- ? negotiating with designers or clients over specific requirements or specifications , and
- ? confirming overall strategies and tactics with project manager or any other involved designer; see the example below.

The following situation can serve as an example of obtaining confirmation from the Project Manager for a local strategy of the Dynamic Web Designer. During the requirements acquisition stage the Dynamic Web Designer determined that the dynamic site to be developed would entail two parts: 1) dynamic, and 2) static HTML. She decided that in order to satisfy the consistency requirement, the best design strategy would be that of integration: the two parts, dynamic and static, would be developed separately by the Dynamic Web Designer and the Web Designer, respectively, and integrated once both design processes completed. As this strategy would involve co-operation with another designer and potentially influence the overall design process, the Dynamic Web Designer had to confirm this strategy with both the Project Manager and the Web Designer.

Acquired requirements had to be further refined. For example, the security and information accuracy requirements described above, as a result of analysis and repeated information/ clarification/ confirmation requests, gave rise to the following *data verification requirement*:

- ? an administrator must verify the data entered by internal employees or vendors before committing to permanent changes.

The following functionality was requested to support this requirement:

- ? changes made by vendors and internal employees (unless given administrative privileges) must not be visible until verified by the administrator;
- ? once a change/addition/deletion is made, an e-mail must be sent to the administrator with a link to a page containing the changes;
- ? after being verified by the administrator, the changes must become permanent and visible; otherwise, they will be reversed.

4.2.2 Design of the VDEI

Once the requirements acquisition was completed, the Dynamic Web Designer proceeded with the design of the actual Vendor Data Entry Interface. Three major sections were identified

based on the intended user base: vendor, internal employee, and administrator sections. In addition, a data verification process that spanned across the three sections had to be designed. Throughout the design process modifications to the database structure (initiated by the Dynamic Web Designer) had to be implemented which involved intensive interaction with the Database Administrator. This interaction included database changes specification and requests, negotiations about the changes, database structure and changes clarification, and changes confirmation.

When the design of the data verification process and corresponding parts of the three sections were almost complete, the Dynamic Web Designer received information contradicting a previous assumption, namely the fact that frequent modifications were to be expected whereas the previous assumption was that such changes would be quite rare. After a behavioural simulation of a hypothetical administrator it became clear that the existing data verification functionality requirements and the data verification process design were no longer adequate and had to be modified. A new data verification functionality requirement had to be composed and confirmed with the Project Manager, and resulted in major changes to the existing data verification process and the already designed web pages.

Below is the new data verification functionality requirement:

- ? after login the administrator must have an option to verify additions, modifications and deletions by following an appropriate link on the main menu;
- ? the names of the companies for which unverified changes exist (be it company, product or contact info) must be displayed once a link is followed;
- ? the administrator must be able to choose a company from the list, and view all the changes made for this company, and then verify or roll back the changes.

4.2.3 Evaluation of the resulting VDEI

During the testing and debugging stage of the Vendor Data Entry Interface design, the Dynamic Web Designer was informed that the integration of the dynamic and static HTML pages would be performed by an HTML designer, and was requested to provide integration instructions. However, it was soon discovered that the HTML designer would be unable to perform the integration due to the lack of programming skills, and another designer was found.

4.3 Initial observations

On the basis of the description of the case study, a number of observations can be drawn. First of all, the Dynamic Web Designer needed to interact with other designers on various issues, ranging from obtaining approval for strategic decisions to providing detailed design information. The complexity of the interaction ranges from merely informing other agents to extensive negotiations. Secondly, the Dynamic Web Designer showed reflective reasoning not only on the process performed by the Dynamic Web Designer (e.g., when to do what), but also reasoning about other designers' capabilities, and even reasoning from the point of view of another designer. Lastly, the design process performed by the Dynamic Web Designer is (or quickly became) a *re-design process*: an existing (partial) set of requirements and corresponding website description were modified because of changes in the context of the design problem of the Dynamic Web Designer.

5. Existing models of agents and design

The main purpose of this study is to understand which additional types of reflective reasoning are required in distributed design compared to single agent design. In this paper an existing knowledge-level DESIRE model of a design agent for single agent design [63] is used to model the types of reflective reasoning encountered in the case study of distributed design described above. This model is enhanced with a component which manages co-operation between agents as proposed in [28] for project co-ordination. The resulting model (tested for a design process for aircraft industry) is described below.

DESIRE is a formal knowledge-level modelling and specification framework for knowledge-intensive (multi-agent) systems [64, 65]. 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.

5.1 Co-operative design agent model

The existing model distinguishes seven main processes within an agent, as depicted in Figure 1 below. 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. maintains information about other agents (component Maintenance of Agent Information),
4. interacts with the external world (component World Interaction Management),
5. maintains information about the external world (component Maintenance of World Information),
6. participates in project co-ordination (component Co-operation Management) and
7. designs an artefact (within component Agent Specific Tasks is a component Design).

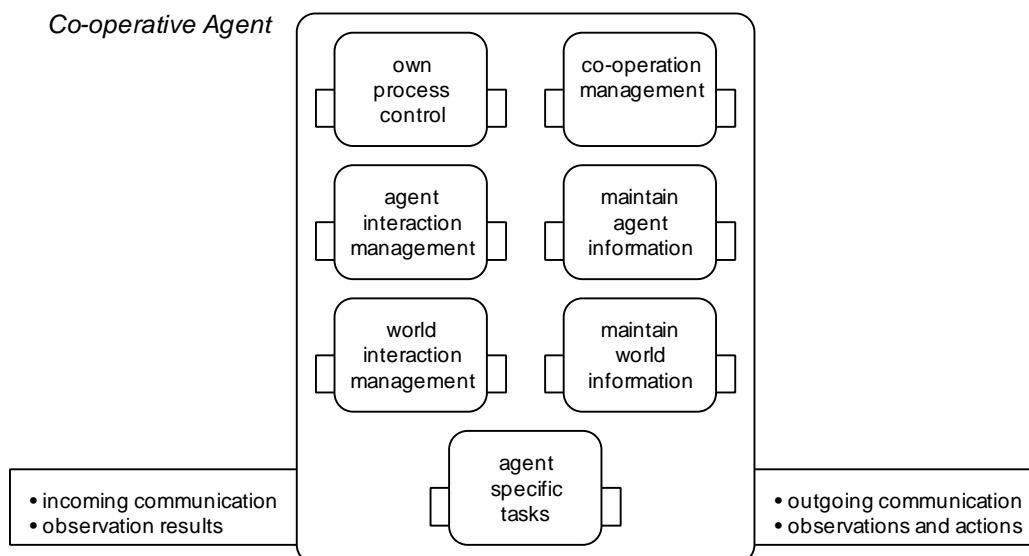


Figure 1. Process abstraction levels for a generic co-operative agent.

This model for a co-operative agent includes components for management of its own processes, interaction with other agents including co-operations, interaction with the external (material) world, and performing an agent's specific tasks. In this model, a co-operative agent has input information consisting of incoming communication from other agents, and results from observations in the external world. As output information, a co-operative agent yields outgoing communication to other agents and observations and actions in the external world.

This generic agent model supports the notion of weak agent [39]. Autonomy and pro-activeness with respect to the agent is supported by the component Own process Control. Social abilities, and reactiveness and pro-activeness with respect to other agents is supported by the components Co-operation Management, Agent Interaction Management and Maintenance of Agent Information. Reactiveness and pro-activeness with respect to the external world is supported by the components World Interaction Management and Maintenance of World Information. Interactions of the agent with the external (physical) world are explicitly modelled in this model. Extending this agent model with the notion of intelligent stance can be achieved by extending the component Own Process Control with stronger mentalistic mechanisms, such as BDI models [66].

This generic structure can be refined for specific types of agents and design tasks in different domains of application. Refinement of the generic model, by specialisation and instantiation, involves the specification of more specific sub-processes, knowledge about applicable requirements and their qualifications, about the design object domain, and about design strategies.

The refinement of the component design in the generic knowledge-level model of a design agent is based on a generic model of single agent design [12]. In this model an initial design problem statement is expressed as a set of initial requirements and requirement qualifications. *Requirements* impose conditions and restrictions on the structure, functionality and behaviour of the *design object* for which a structural description is to be generated during design. *Qualifications* of requirements are qualitative expressions of the extent to which (individual or groups of) requirements are considered hard or preferred, either in isolation or in relation to other (individual or groups of) requirements. At any one point in time during design, the design process focuses on a specific subset of the set of requirements. This subset of requirements plays a central role; the design process is (temporarily) committed to the current requirement qualification set: the aim of generating a design object description is to satisfy these requirements.

Figure 2 shows one level of composition of the processes distinguished within the component design, and the types of input and output involved. The refinements of these three processes are not further depicted (see [63]) for a description of a formal specification including details on additional process composition, information flow, control flow, generic information types and generic knowledge bases). The left hand side describes the input information to the design process; the right hand side describes the output information. The design process is shown to be composed of three sub-processes: design process co-ordination, requirement qualification set manipulation, and design object description manipulation. The process Design Process Co-ordination co-ordinates the design process by issuing information related to overall design strategies on the basis of progress reports of the manipulation components and given design process objectives. The process Requirement Qualification Set Manipulation manipulates sets of requirements, on the basis of an overall design strategy, information from Design Object

Description Manipulation, and given sets of qualified requirements. The process Design Object Description Manipulation manipulates descriptions of design objects, on the basis of an overall design strategy, information from Requirement Qualification Set Manipulation, and given design object descriptions.

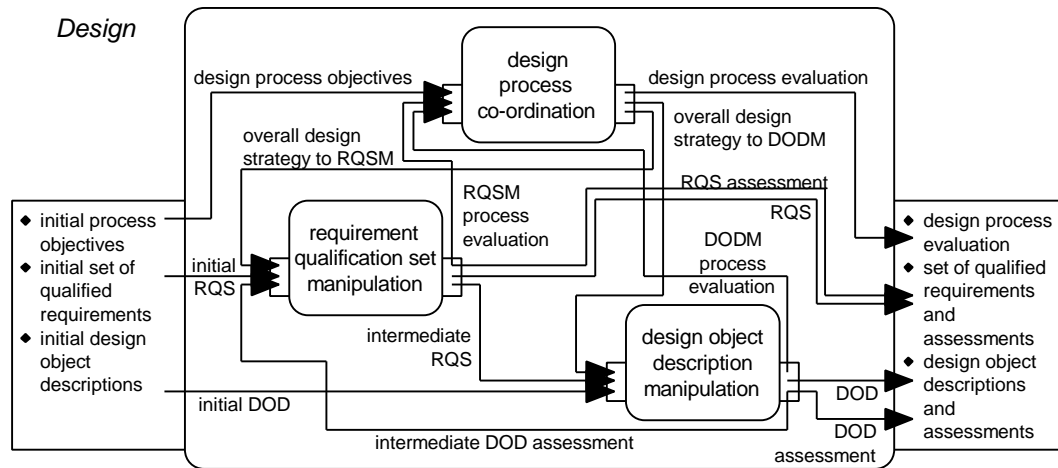


Figure 2. Composition of processes of the design process in the generic model of design.

In addition to a composition of processes, Figure 2 also shows information flow between processes. Each information link is named. For each application, explicit control on the activation of information links is explicitly specified (which may vary from strict dictatorial control to entirely loose control: all information links may work in parallel). Below is a more detailed description of the information dependencies between the three subprocesses.

- ? The process Design requires, as input, information objectives for the overall design process (design process objectives), a given RQS (RQS) and a given DOD (DOD). The process Design produces an evaluation of the overall design process (design process evaluation), an evaluation of resulting requirement qualification sets (RQS assessment), an evaluation of resulting design object descriptions (DOD assessment), sets of qualified requirements (RQS) and design object descriptions (DOD).
- ? The process Design Process Co-ordination requires information on objectives for the overall design process (design process objective), and evaluations of the manipulation processes (manipulation process evaluation). The process Design Process Co-ordination produces an evaluation of the overall design process (design process evaluation), and strategies for RQS Manipulation and DOD Manipulation (overall design strategy).
- ? The process RQS Manipulation requires a strategy (overall design strategy), an evaluation of resulting design object descriptions (DOD assessment), and a given RQS (RQS). The process RQS Manipulation produces an evaluation of the status of its own process (RQSM process evaluation), an evaluation of resulting qualified requirement sets (RQS assessment), and contents of sets of requirement qualifications (RQS).
- ? The process DOD Manipulation requires an overall design strategy (overall design strategy), information on the requirement qualification set for which a design object description is to be constructed (RQS), and possibly an existing design object description (DOD). The process DOD Manipulation produces an evaluation of the status of its own process (DODM process evaluation), an evaluation of resulting design object descriptions, including information on the satisfaction of design requirements for specific design object descriptions (DOD assessment), and design object descriptions (DOD).

6. Model for design agent in distributed design

The single agent model of a co-operative agent described above in Section 5 needs to be modified to address the following desiderata:

- ? a design agent has knowledge and information of the distributed design situation (Section 6.1).
- ? a design agent is able to (strategically) *reason about* the distributed design situation (Section 6.2).
- ? a design agent is able to *reason with* knowledge from the distributed design situation (Section 6.3).

An example application of an agent formally specified on the basis of the model of the design agent, described in Sections 5 and 6, is given in Section 7. The example includes fragments of knowledge structures which are identified, and described, in Section 6.

6.1 Modelling elements that determine a situation

Knowledge and information described in Section 3.1 can be modelled in the components of the knowledge-level model of a design agent.

Elements related to the client:

- ? *Given problem.* Information about the given problem can be modelled as world information, and is thus available to all components.
- ? *Design process.* Design process objectives and communications from the client are modelled as ‘goals’ and messages that can be communicated among, and within an agent. The components Own Process Control, Co-operation Management and Design mainly use this information and contain knowledge about the design process.

Elements related to the designer him- or herself:

- ? *Design knowledge.* Information on design knowledge, and the design knowledge itself, is modelled within the agent’s specific Design component.
- ? *Design experience.* Information and knowledge on design experience is modeled within the agent’s specific Design component. If design experience includes, e.g., experience in interacting with other designer, then it also modelled in communication, negotiation, and world interaction histories; stored and maintained in the components Maintenance of Agent Information, and Maintenance of World Information.

Elements related to the environment of the designer:

- ? *Design partners.* Information about design partners is modelled as agent information, and is thus available to all components. Knowledge about the design partners (e.g., how they might react to certain stimuli) is modelled within the components Design, Agent Interaction Management, and Co-operation Management.
- ? *Design resources.* Information about design resources is modelled as world information and agent information, depending on the kind of resources (e.g., money and time versus case-libraries) and is available to all components. Knowledge about design resources is modelled within the components Own Process Control, Design, and Co-operation Management.
- ? *Design culture.* Information about design culture is modelled as agent information, and is available to all components. Knowledge about design culture is modelled within the

components Own Process Control, Co-operation Management, and Agent Interaction Management.

- ? *Design process.* Information about the design process, usually shared with some or all of the other designers, is modelled as agent information. Knowledge about the design process is modelled within the components Own Process Control, Co-operation Management, and Design.
- ? *Design problem.* Information on design solutions, i.e. (partial) sets of design requirements and (partial) artefact descriptions, is modelled as world information, and is thus available to all components. Knowledge about design solutions is mainly modelled within the component Design, but some knowledge (more global, descriptive) on design solutions is modelled in Co-operation Management.
- ? *Design progress.* Information on design progress, of the progress of the designer itself and progress achieved in co-operation with other designers, is modelled as agent information, and is available to all components. Knowledge about design progress is modelled within the components Own Process Control, Co-operation Management, and Design.
- ? *Design knowledge.* Information on design knowledge, from the point of view of another agent, or domain specific point of view, is modelled within the component Design. The design knowledge itself is also modelled within the component Design.

Each of the above mentioned elements can be viewed from different perspectives. Domain specific viewpoints have been discussed before (e.g., electrical system vs. heating system vs. sewage system in a building). Viewpoints related to the environment of the designer are usually dependent on (groups of) agents. E.g., a design agent may have information on, or even knowledge of, how another design agent interprets design requirements. Being able to distinguish viewpoints of other design agents, facilitates reasoning about these agents. Reflective reasoning (about another design agent) may aid the individual designer to, e.g., predict potential conflicts, and pro-actively avoid the occurrence of these conflicts.

To accurately model interactions of a design agent with other designers and/or the external world, the generic design model placed in the agent's specific task is modified. The modified model of design includes information on design interactions and results from design interactions. Figure 3 shows the modified interface of the design process, on the input side additional information on design interaction results is now available to the design process, and on the output side information on design interactions is a possible result of the design process.

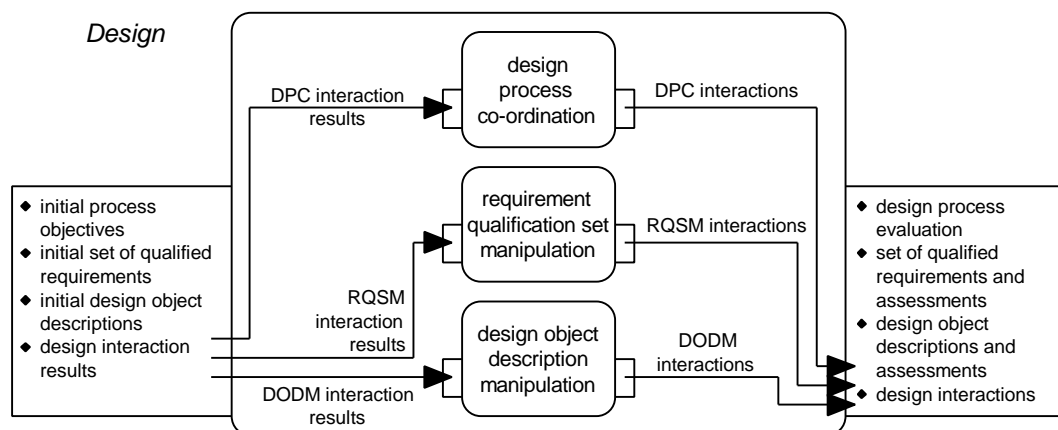


Figure 3. The interface of the design model has been extended to include information on Design Interaction Results and Design Interactions.

Each of the three subprocesses of the design process is capable of conducting design interactions, albeit with different subjects. For this end, the design interactions information is constructed from three separate information types, namely DPC interaction, RQSM interactions and DODM interactions, which originate from the processes design-process co-ordination, requirement qualification set manipulation and design object description manipulation, respectively. Information links with the same name as the information type transferred, are shown in the right half of Figure 3. Similarly, all three processes need to obtain results of their interactions, which are provided by the three information links in the left part of Figure 3: DPC interaction results, RQSM interaction results, and DODM interaction results. These names of information types are the same as the information type transferred by a link. The information type design interaction results is composed of three information types: DPC interaction results, RQSM interaction results, and DODM interaction results.

Each of the three subprocesses of the design process is capable of producing information on design interactions, and understanding information on design interaction results. The six additional information links are depicted in Figure 3; the information links among the three subprocesses provided sufficient information flow to exchange information on results, or intended, design interactions. Additional details of the formal specification of the designer agent are not shown.

6.2 Reasoning about a situation

A design agent needs to be able to reason about its own situation. This is *reflective reasoning* about the agent itself, and the situation it is in. Some of this reasoning is on a strategic level, as it concerns reasoning about one's own situation.

Within a design process based on the generic model of design, strategic reasoning about, e.g., when to apply which knowledge is already possible. This entails a representation of (or, information on) the available design knowledge, so that explicit decisions can be made about when to apply which knowledge to which situation. E.g., when two different predictive models can be used to simulate the air flow in a building, which differ in accuracy and cost, such strategic reasoning aids in building a rationale of the design process, and in choosing a predictive model to use.

Strategic reasoning by an individual designer about elements of its situation in a distributed design process is realised in a similar manner. Within the model of a design agent, strategic reasoning takes place in several components, including Own Process Control, Co-operation Management, and Design.

An example of strategic reasoning in a design agent is a situation in which, e.g., the design process has a need for resolving a conflict in design requirements with another designer. The sub-process Co-operation Management may conclude on the basis of past experiences in negotiating with that specific designer, that it is not worthwhile to start a negotiation at all, as the other designer is very stubborn. This can be indicated to the design process as, e.g., a non-executed, to-be-assumed as failed, negotiation. The design process can continue work on the basis of this information.

6.3 Reflective reasoning

Reasoning from another point of view, is a useful ability for a designer, which entails not only reasoning from a domain specific point of view (e.g., different system views in a building), but also reasoning from the point of view of another designer. Reasoning from the viewpoint of another design agent can be used for, e.g.,

- ? validating a design object description against the given design requirements using a perspective on design requirement assessment knowledge
- ? comparing a design object description against a perspective on a design object description. (and the same for sets of qualified requirements)
- ? predictions: if the other agent would work on this part of the artefact description, what would he or she add or modify?

In the example domain the Dynamic Web Designer, reasons both from the perspective of another (co-designer) agent, and from the perspective of users of the resulting (to-be-designed) system. Reasoning from the perspective of a user, can be modelled in the same manner as reasoning from the perspective of another agent.

The knowledge used to reason from the point of view of another agent is not as complete or accurate as the knowledge available to that other agent. The reasoning from a point of view is a form of hypothetical reasoning: the outcome is assumed to be correct, but may be falsified after interacting with the actual agent.

To be able to reason from the perspective of another agent, facilities need to be available within the design model. First of all, the actual knowledge bases need to be available (beforehand, or dynamically). Secondly, decisions need to be made concerning when to reason with which information from which viewpoint (see the previous section). And finally, the agent has to actually reason with the knowledge from a knowledge base. These facilities are provided by the model of design described above in section 5.

This same generic design model also provides facilities for storing versions of sets of qualified requirements, design object descriptions, and even storing design rationale. These facilities are used to store : (1) history of a design process (traces), and (2) design rationale.

7. Example of design agent in a collaborative setting

To illustrate behaviour of a design agent, the knowledge-level model described in Section 6 for which a formal specification has been written, was instantiated and implemented for the Vendor Data Entry Interface (VDEI) design in Section 4. The emphasis in the trace was on strategic considerations within the design agent, therefore almost no domain specific knowledge related to the specific Dynamic Web design process domain was included. The Dynamic Web Design agent has been modelled and specified within the DESIRE framework. A trace of the automatically generated prototype implementation of the design agent shows that the original trace, as related by the Dynamic Web Designer, can be yielded by the design agent.

Specific examples of reasoning from viewpoints and reasoning about interaction, along with corresponding traces and knowledge pieces from the detailed model, are presented below.

7.1 Examples of reflective reasoning

Two different examples for reflective reasoning are given: reasoning from the point of view of co-designers, and reasoning from the point of view of intended users of the website to be designed.

7.1.1 Reasoning from other designer's perspective

In the example described in Section 4.2, the Dynamic Web Designer reasoned from the point of view of three other designers: the Web Designer, the HTML Designer and the Database Administrator.

Web Designer's viewpoint

One of the first explicit shifts in viewpoints occurred during the determination of an integration strategy (refer to the integration decision in Section 4.2), when the Dynamic Web Designer first realized that the dynamic site will have two parts: a static HTML part and a dynamic functional part. The Dynamic Web Designer knew that the Web Designer worked on the static HTML of the overall website, and decided that the best strategy would be to design static and dynamic parts separately, and later integrate them to insure that the site is consistent and work is not duplicated.

The Dynamic Web Designer reasoned about her knowledge of static HTML, and available Web Designer's info, and decided that if the Web Designer doesn't have sufficient knowledge of dynamic pages design, it would be rather difficult for him to integrate the dynamic part with the static part (there was a conscious, explicit shift in reasoning in viewpoints). Concluding, the Dynamic Web Designer made an assumption that the integration will be the Dynamic Web Designer's responsibility. Furthermore, the Dynamic Web Designer analysed her own knowledge, and made an assumption that the integration should be rather easy to implement.

Some fragments of knowledge employed by the Dynamic Web Design agent are shown below. These knowledge elements are taken from the process co-ordination component inside the component Requirement Qualification Set Manipulation.

The first knowledge fragment shown in Table 2 specifies a local strategy, based on (partial) results in designing the artefact description, i.e., discovering that the description of the dynamic web pages consists of both dynamic html pages and static html pages. The proposed strategy states that the Web Designer is to integrate the static and dynamic HTML parts.

<pre> if analysis_from_DODM_results(current_dod, contains_two_viewpoints(V: Viewpoints)) and design_strategy(DS: design_strategy_name, own_viewpoint(OV: Viewpoint)) and V: Viewpoint ? OV: Viewpoint and agent_info(A: Agent_Name, is_concerned_with_viewpoint(V: Viewpoint)) then RQSM_goal(not_work_on(V: Viewpoint)) and proposed_RQSM_strategy(new(RS): RQSM_Strategy_Name, integrate(OV: Viewpoint, V: Viewpoint), when_stable(OV: Viewpoint) by_agent(A: Agent_Name)); </pre>	<p>This knowledge fragment specifies that the current artefact description contains two viewpoints (V = {static_html, dynamic_html}). The overall design strategy issued by Design Process Co-ordination states the viewpoint to be used by the dynamic_web_designer is OV = {dynamic_html}. The third condition states that the viewpoint which is not the other viewpoint is important (V = {static_html}). The fourth condition uses the available information on design partners to retrieve the name of an agent who is concerned with the static_html viewpoint (A =</p>
--	--

self-performed-integration assumption was false, and another designer - HTML Designer - would be performing the integration.

The Dynamic Web Designer had to analyze the HTML Designer's level of expertise with respect to static and dynamic HTML and deduce whether this knowledge would be sufficient to perform the integration. The Dynamic Web Designer tried to reason from the HTML Designer's viewpoint while composing integration instructions to imagine how the HTML Designer would view it. However, based on the comments on the instructions received from the HTML Designer, the Dynamic Web Designer realized that the HTML Designer's knowledge differed from the assumed knowledge and was limited to static HTML only.

The fourth knowledge fragment, shown in Table 5, illustrates the decision to shift reasoning to the point of view of another co-designer.

if and and and then	DODM_strategy(DS: DODM_Strategy_Name, P: Purpose, intended_for(A: Agent_Name)) agent_info(A: Agent_Name, is_concerned_with_viewpoint(V: Viewpoint)) agent_info(A: Agent_Name, related_knowledge_base(K: Knowledge_Base_Name)) knowledge_base_has_purpose(K: Knowledge_Base_Name, P: Purpose) method_to_apply(analysis, P: Purpose, knowledge_base_to_use_from_viewpoint(K: Knowledge_Base_Name, V: Viewpoint));
This knowledge fragment specifies that a local strategy with a specific purpose for designing an artefact description intended for a specific agent (DS = {dodm_43}, P = {build_integration_instructions}, A = {html_designer}) is best realised by reasoning from the perspective of that agent, on the condition that a knowledge base (K = {kb_4}) which can be used for the intended purpose is available to reason from the point of view of that agent.	

Table 5. Knowledge fragment 4: a rule from DODM Process Co-ordination.

DBA's viewpoint

At multiple occasions in the trace it was necessary to have changes made to the database by the Database Administrator, based on the requests from the Dynamic Web Designer. In order to make a request for changes understandable for the Database Administrator, the Dynamic Web Designer had to analyse the Database Administrator's characteristics and her own knowledge of databases in order to make a decision on what information the Database Administrator would need to successfully implement the changes. As a result of this reasoning, the requests included information on which fields for which tables had to be changed, what data types the fields should be, etc.; rationale for changes was also included; however, the actual SQL code was left for the Database Administrator to write.

7.1.2 Reasoning from user's perspective

In the example described in Section 4.2, the Dynamic Web Designer reasoned from the point of view of two intended users of the website to be designer: administrators and employees of vendors of products.

Administrator's viewpoint

While working on the data verification process design, and on the design of the administrative section of the website in general, the Dynamic Web Designer had to simulate an

administrator's behaviour and follow the designed data verification procedure from the viewpoint of a hypothetical administrator - someone who may not have any web design knowledge at all.

This resulted in automating and simplifying a substantial number of steps in the procedure which could be hard to do or confusing for a person with limited internet knowledge. Furthermore, a case of similar reasoning from an administrator's viewpoint - simulation of the effect produced by multiple changes on the data verification process performed by an administrator - resulted in a change of a requirement and major modifications to the website (refer to the description of the data verification requirement modification in Section 4.2). The ease-of-use and ease-of-maintenance requirements implied that such a shift of viewpoints would be necessary in the design.

The fifth knowledge fragment, shown in Table 6, illustrates the decision to reason from the point of view of a group of intended users.

<pre> if RQS_has_subset(current_RQS, S: RQS_Subset_Name) and finished_working_on_RQS_subset(S: RQS_Subset_Name, successful(functional_requirements)) and not finished_working_on_RQS_subset(S: RQS_Subset_Name, successful(behavioural_requirements)) and RQS_subset_has_intended_group_of_user(S: RQS_Subset_Name, U: User_Group_Name) and knowledge_base_has_purpose(K: Knowledge_Base_Name, simulate_behaviour_of(U: User_Group_Name)) then method_to_apply(analysis, simulate_behaviour_of(U: User_Group_Name), knowledge_base_to_use(K: Knowledge_Base_Name, from_viewpoint(U: User_Group_Name)); </pre>
<p>This knowledge fragment specifies that if the current set of design requirements has a subset, from which the functional requirements have been satisfied, but the behavioural requirements have not yet been satisfied; then an analysis should be made by behavioural simulation from the point of view of the intended users of the part of the artefact description that the subset of requirements has required ($S = \{ \text{data_verification_procedure} \}$, $U = \{ \text{administrators} \}$, $K = \{ \text{kb_23} \}$).</p>

Table 6. Knowledge fragment 5: a rule from DODM Process Co-ordination.

Vendor and Internal Employee's viewpoints.

Reasoning from vendors and internal employees' viewpoints was similar to that of an administrator's; but in this case not only absence of web design knowledge had to be assumed, but also limited knowledge of how the system works in general as well. The Dynamic Web Designer had to design the website so that people with such limited knowledge could use it effectively, and thus had to look at the website from their viewpoint.

8. Discussion and future work

Existing knowledge-level models of design do not address the types of reasoning and knowledge involved in distributed design. This paper has shown that such models need to include explicit knowledge of other participants and the design environment. In addition a knowledge level model of an individual designer needs to include sufficient knowledge to be

able to reason reflectively about other agents and the design environment. This includes being able to reason from the point of view of other agents during a design process.

The main purpose of this study was to understand which types of reflective reasoning required in distributed collaborative design compared to single agent design. The environment of an individual designer in a distributed design process includes, for example, knowledge of co-operation partners, design culture, and design problem. To manage these elements, an individual designer needs to be able to reflect on its own behaviour and knowledge in relation to (elements of) its environment.

In this paper two existing generic models of agents have been combined: a generic model of a design agent (for single agent design), and a generic model of a co-operative agent. The resulting model has been extended to include reasoning about other participant's knowledge and other participants reasoning during the design process, but also to include reasoning about the need for interaction. Such reflective reasoning requires additional types of input and generates additional types of requests during a design process. These information types have been depicted. Reflective reasoning about viewpoints in itself is included in the model of an individual design agent, and thus required no additional modifications of the model.

Reasoning within the resulting model of a design agent has been illustrated for one specific example domain, namely the distributed design of a website. A trace of the simulated behaviour corresponds to the behaviour encountered in practice.

This paper does not address several other aspects of distributed design, such as:

- ? reasoning about negotiation strategies, including norms and values in a culture, deliberate norm transgression;
- ? specific ontologies used by an individual designer and ontologies shared among a group of designers;
- ? learning about design partners, their behaviour, and their reactions to conflict resolution negotiations;
- ? sharing information and knowledge;
- ? (distributed) planning of projects, tasks, and activities;
- ? project management for parts of the local, and overall, design process.

The research reported in this paper is a step towards a knowledge-level model and a formal theory for a distributed design process. Existing models of design do not address aspects of distribution and their effect on design processes. Such models and theories are needed to increase both the quality and quantity of distributed design systems.

Acknowledgements

The authors express their gratitude to Pieter van Langen for material on distributed design research and Ashok Goel for initiating collaboration with Lilia Moshkina. The research reported in this paper has been financially supported by the NLnet foundation.

References

1. Alexander C. Notes on the Synthesis of Form. Cambridge: Harvard University Press, 1964.

2. Archer LB. An overview of the structure of the design process. In: Moore GT, editor. *Emerging Methods in Environmental Design and Planning*. Cambridge: MIT Press, 1970. p. 285-307.
3. Mostow J. Toward better models of the design process. *AI Magazine* 1985; 6(1):44-57.
4. Bijl A. An approach to design theory. In: Yoshikawa H, Warman EA, editors. *Proceedings IFIP WG5.2 Working Conference on Design Theory for CAD*. Amsterdam: Elsevier (North-Holland), 1987. p. 3-25.
5. Tomiyama T, Yoshikawa H. Extended general design theory. In: Yoshikawa H, Warman EA, editors. *Proceedings IFIP WG5.2 Working Conference on Design Theory for CAD*. Amsterdam: Elsevier (North-Holland), 1987. p. 95-125.
6. Brown DC, Chandrasekaran B. *Design Problem Solving: Knowledge Structures and Control Strategies*. Research Notes in Artificial Intelligence. London: Pitman, 1989.
7. Maher M L. (1990). Process models for design synthesis. *AI Magazine* 1989;4(1):49-58.
8. Takeda H, Veerkamp PJ, Tomiyama T, Yoshikawa H. Modelling design processes. *AI Magazine* 1990;11(4):37-48.
9. Logan BS, Smithers T. Creativity and design as exploration. In: Gero JS, Maher ML, editors. *Modelling Creativity and Knowledge-Based Creative Design*, Hillsdale: Lawrence Erlbaum, 1992. p. 149-188.
10. Brazier FMT, Langen PHG van, Ruttkay Zs, Treur J. On formal specification of design tasks. In: Gero JS, Sudweeks F, editors. *Proceedings Artificial Intelligence in Design (AID'94)*. Dordrecht: Kluwer Academic Publishers, 1994. p. 535-552.
11. Wielinga BJ, Akkermans JM, Schreiber ATH. A formal analysis of parametric design. In: Gaines BR, Musen MA, editors. *Proceedings of the Ninth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW '95)*. Calgary: University of Calgary, Department of Computer Science, SRDG Publications, 1995. Volume 2, p. 37/1-37/15.
12. Brazier FMT, Langen PHG van, Treur J. A logical theory of design. In: Gero JS, editor. *Advances in formal design methods for CAD*. London: Chapman and Hall, 1996. p. 243-266.
13. Smithers T. On knowledge level theories of design process. In: Gero JS, Sudweeks F, editors. *Artificial Intelligence in Design '96 (AID '96)*. Dordrecht: Kluwer Academic Publishers, 1996. p. 561-579.
14. Smithers T. Towards a knowledge level theory of design process. In: Gero JS, Sudweeks F, editors. *Artificial Intelligence in Design '98 (AID '98)*, Dordrecht: Kluwer Academic Publishers, 1998.
15. Pahl G, Beitz W. *Engineering design*. New York: Springer-Verlag, 1984.
16. Pugh S. *Total Design: Integrated Methods for Successful Product Engineering*. Reading: Addison-Wesley, 1990.
17. Suh NP. *The Principles of Design*. Oxford Series of Advanced Manufacturing. Oxford: Oxford University Press, 1990.
18. Goel A, Chandrasekaran B. Functional representation of design and redesign problem solving. In: Sridharan NS, editor. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI '89)*. Los Altos: Morgan Kaufmann, 1989. p. 1388-1394.
19. Coyne RD, Rosenman MA, Radford AD, Balachandran M, Gero JS. *Knowledge-Based Design Systems*. Reading: Addison-Wesley, 1990.
20. Gero JS. Design prototypes: a knowledge representation schema for design. *AI Magazine* 1990;11(4):26-36.
21. Candy L, Edmonds E. Artefacts and the designer's process: implications for computer support to design. *Revue Sciences et Techniques de la Conception* 1994;3(1):11-31.
22. Runkel JT, Balkany A, Birmingham WP. Generating non-brittle configuration-design tools. In: Gero JS, Sudweeks F, editors. *Proc. Artificial Intelligence in Design '94 (AID '94)*. Dordrecht: Kluwer Academic Publishers, 1994. p. 183-200.
23. Brazier FMT, Langen PHG van, Treur J, Wijngaards, NJE, Willems M. Modelling an elevator design task in Desire: the VT example. In: Schreiber ATH, Birmingham WP, editors. *Special Issue on Sisyphus-VT*. In: *International Journal of Human-Computer Studies (IJHCS)* 1996;44:p. 469-520.
24. Brazier FMT, Treur J, Wijngaards, NJE. Interaction with experts: the role of a shared task model. In: Wahlster W, editor. *Proceedings European Conference on AI (ECAI'96)*, Chichester: Wiley and Sons, 1996. p. 241-245.
25. Hori, K. Special issue on strategic knowledge and concept formation. *Knowledge Based Systems* 1998;11.
26. Brazier FMT, Jonker CM, Treur J, and Wijngaards, NJE. On the use of shared task models in knowledge acquisition, strategic user interaction and clarification agents. *Int. J. Human-Computer Studies* 2000;52(1):77-110.
27. Jennings NR. Controlling Co-operative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions. *Artificial Intelligence Journal* 1995;74(2):195-240.

28. Brazier FMT, Jonker CM, and Treur J. Modelling project coordination in a multi-agent framework. In: Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96. Los Alamitos: IEEE Computer Society Press, 1996. p. 148-155.
29. Bahler D, Dupont C, Bowen J. Anaxiomatic approach that supports negotiated resolution of design conflicts in concurrent engineering. In: Gero JS, Sudweeks F, editor. Artificial Intelligence in Design, Dordrecht: Kluwer Academic Publishers, 1994. p. 363-379.
30. Balasubramanian S, Norrie DH. A multiagent architecture for concurrent design, process planning, routing, and scheduling. *Concurrent Engineering: Research and Applications* 1996;4(1):7-16.
31. Cutkosky M, Engelmores R, Fikes R, Gruber T, Genesereth M, MarkW, Tenenbaum J, Weber J. PACT: An experiment in integrating concurrent engineering systems. Special Issue on Computer Support for Concurrent Engineering, *IEEE Computer* 1993;26:28-37.
32. Klein M. Conflict management as part of an integrated exceptionhandling approach. *AIEDAM* 1995;9:259-267.
33. Petrie C. Design space navigation as a collaborative aid. In: Gero JS, editor. Artificial Intelligence in Design '94, Proceedings AID '94. Dordrecht: Kluwer Academic Publishers, 1994. p. 611-623.
34. Goldmann S. 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, 1996.
35. Gupta L, Chionglo J, Fox M. A constraint-based model of communication and coordination in concurrent design projects. In: Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96. Los Alamitos: IEEE Computer Society Press, 1996.
36. Maurer F, Dellen B, Bendeck F, Goldmann S, Holz H., Kötting B, Schaaf M. (2000). Merging project planning and web-enabled dynamic workflow techniques. In: *IEEE Internet Computing* 2000:65-74.
37. Haymaker J, Ackermann E, Fischer M. Meaning Mediating Mechanism. In: Gero JS, editor. Artificial Intelligence in Design '00, Proceedings of the Sixth International Conference on AI in Design, AID'2000. Dordrecht:Kluwer Academic Publishers, 2000. p. 691-715.
38. Nwana HS. Software agents: an overview. *The Knowledge Engineering Review* 1996;11(3):205-244.
39. Wooldridge MJ, Jennings NR. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 1995;10(2):115-152.
40. Jennings NR, Wooldridge MJ (editors). *Agent Technology; Foundations, Application, and Markets*. Berlin: Springer Verlag, 1998.
41. Shoham Y. Agent-oriented programming. *Artificial Intelligence* 1993;60:51-92.
42. Bradshaw JM (editor). *Software Agents*. AAAI Press / MIT Press, 1997.
43. Dennett DC. *The Intentional Stance*. Cambridge: MIT Press, 1987.
44. Levy AY, Sagiv Y, Srivastava D. Towards efficient information gathering agents. In: *Software Agents, proceedings of the AAAI 1994 spring symposium*. AAAI Press, 1994. p. 64-70.
45. Sycara K, Zeng D. Multi-agent integration of information gathering and decision support. In: Wahlster W, editor. *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96)*. Wiley and Sons, 1996. p. 549-553.
46. Knoblock CA, Ambite JL. Agents for Information Gathering. In: Bradshaw JM, editor. *Software Agents*. AAAI Press / MIT Press, 1997. p. 347-373.
47. Wong H-C, Sycara K. A Taxonomy of Middle-agents for the Internet. In: *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000)*. 2000.
48. Grecu DL, Brown DC. Learning by single function agents during spring design. In: Gero JS, Sudweeks F, editors. *Artificial Intelligence in Design '96 (AID '96)*. Dordrecht: Kluwer Academic Publishers, 1996. p. 409-428.
49. Campbell MI, Cagan J, Kotovsky K. A-Design: theory and implementation of an adaptive, agent-based method of conceptual design. In: Gero JS, Sudweeks F, editors. *Artificial Intelligence in Design '98 (AID '98)*. Dordrecht: Kluwer Academic Publishers, 1998. p. 579-598.
50. McAlinden LP, Florida-James BO, Chao K-M, Norman PW, Hills W, Smith P. Information and knowledge sharing for distributed design agents. In: Gero JS, Sudweeks F, editors. *Artificial Intelligence in Design '98 (AID '98)*. Dordrecht: Kluwer Academic Publishers, 1998. p. 537-556.
50. Weyhrauch RW. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence* 1980;13:133-170.
51. Davis R. Metarules: reasoning about control. *Artificial Intelligence* 1980;15:179-222.
52. Maes P, Nardi D (editors). *Meta-level architectures and reflection*. Elsevier Science Publishers, 1998.
53. Attardi G, Simi M. Proofs in Context. In: Fribourg L, Turini F, editors. *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, volume 883, 1994. p. 410-424.
54. Clancey WJ, Bock C. Representing control knowledge as abstract tasks and metarules. In: Bolc L, Coombs MJ, editors. Computer Expert Systems. Heidelberg: Springer-Verlag, 1988, pp. 1-77.
 55. Fisher M, Wooldridge M. Specifying and Verifying Distributed Intelligent Systems. In: Filqueiras M, Damas L, editors. Progress in AI. Proc. EPAI'93. Springer Verlag, Lecture Notes in AI, volume 727, 1993. p. 13-28
 56. Cimatti A, Serafini L. Multi-agent Reasoning with Belief Contexts II: Elaboration Tolerance. In: Lesser V, editor. Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95, MIT Press, 1995. p. 57-64.
 57. Wagner G. A Logical and Operational Model of Scalable Knowledge- and Perception-based Agents. In: van der Velde W, Perram JW, editors. Agents breaking away, Proc. MAAMAW'96. Springer Verlag, Lecture Notes in AI, volume 1038, 1996. p. 26-41.
 58. Asada M, Veloso MM, Tambe M, Noda I, Kitano H, Kraetzschmar GH. Overview of RoboCup-98. AI Magazine 2000;21(1):9 - 19.
 59. Stone P, Veloso M, Riley P. CMUNITED-98 Simulated Team. AI Magazine 2000;21(1):20 - 28.
 60. Gero JS. Conceptual Designing as a Sequence of Situated Acts. In: Smith I, editor. Artificial Intelligence in Structural Engineering. Berlin: Springer, 1998. p. 165-177.
 61. Maher ML, Simoff S, Cicognani A. Understanding Virtual Design Studios. London: Springer-Verlag, 2000.
 62. Schön, DA. The Reflective Practitioner: how professionals think in action. Aldershot (England): Arena, 1995.
 63. Brazier FMT, Jonker CM, Treur J, and Wijngaards, NJE. Deliberate Evolution in Multi-Agent Systems. In: Gero JS, editor. Artificial Intelligence in Design '00, Proceedings of the Sixth International Conference on AI in Design, AID'2000. Dordrecht: Kluwer Academic Publishers, 2000. p. 633-650.
 64. Brazier FMT, Dunin-Keplicz BM, Jennings NR, Treur J. Formal specification of Multi-Agent Systems: a real-world case. In: Lesser V, editor. Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95. Cambridge MA: MIT Press, 1995. p. 25-32. Extended version in: Huhns M, Singh M, editors. International Journal of Co-operative Information Systems, special issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems 1997;6:67-94.
 65. Brazier FMT, Jonker CM, Treur J. Principles of Compositional Multi-agent System Development. In: Cuenca J, editor. Proceedings of the 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98. 1998. p. 347-360.
 66. Brazier FMT, Dunin-Keplicz BM, Treur J, Verbrugge LC. Modelling Internal Dynamic Behaviour of BDI agents. In: Meyer J-JCh, Schobbes PY, editors. Formal Models of Agents, Selected papers from final ModelAge Workshop. Springer Verlag, Lecture Notes in AI, volume 1760, 1996. p. 36-56.

Vitae

prof.dr. Frances M.T. Brazier

is a full professor of Computer Science at the Vrije Universiteit Amsterdam where she chairs the new Intelligent Interactive Distributed Systems research group. Design is a typical example of an interactive distributed process for which intelligent support at many different levels is required. Intelligent information retrieval, the topic of Brazier's PhD thesis, is another. Both are areas in which Brazier has been involved for many years.

dr. Niek J.E. Wijngaards

is an assistant professor at computer science at the Vrije Universiteit amsterdam. He received his PhD in 1999, on the topic of self-modifying agent systems using a re-design process. His current research goals include support for the design of multi-agent systems, such as distributed multi-agent design: an example of a service and an application for an Agent Operating System.

Lilia V. Moshkina, MA MSc

Lilia Moshkina is a Ph.D. student in Computer Science at Georgia Institute of Technology, USA. She obtained her B.A. and M.A. degrees in English and French from Gorlovka Institute of Foreign Languages, Ukraine, and her M.S. degree in Computer Information Systems from Georgia State University, USA. Her current research interests lie at the intersection of Artificial Intelligence, Cognitive Science and Psychology, to include multi-agent systems and reflection, robotic and agent personality, direct brain-machine interfaces, and user modeling and predictive interfaces.