Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2004), *18*, 199–209. Printed in the USA.
Copyright © 2004 Cambridge University Press 0890-0604/04 $16.00
DOI: 10.1017/S0890060404040132

The role of trust in distributed design

NIEK J.E. WIJNGAARDS, HIDDE M. BOONSTRA, AND FRANCES M.T. BRAZIER
Intelligent Interactive Distributed Systems Group, Faculty of Sciences, Vrije Universiteit Amsterdam,
1081 HV Amsterdam, The Netherlands

(RECEIVED May 27, 2003; ACCEPTED June 28, 2004)

Abstract

Automated support of design teams, consisting of both human and automated systems, requires an understanding of the role of trust in distributed design processes. By explicating trust, an individual designer's decisions become better understood and may be better supported. Each individual designer has his or her private goals in a cooperative design setting, in which requirement conflicts and resource competitions abound. However, there are group goals that also need to be reached. This paper presents an overview of research related to trust in the context of agents and design, a computational knowledge-level model of trust based on the seven beliefs distinguished by Castelfranchi and Falcone, and an example of the use of the trust model in a specific design process, namely, Website design from the perspective of a single designer. The results are discussed in the context of distributed design in open systems.

Keywords: Agents; Distributed Design; Knowledge Level; Trust Model

1. INTRODUCTION

In distributed design, individual designers work together to solve a specific design problem. Such group design processes differ from individual design processes (Dwarakanath & Blessing, 1996) in a number of ways: explication of design steps is often needed to facilitate interaction between the different parties involved, resulting in a larger number of alternatives being explored (see Cross et al., 1996). Partial or complete automated support for such distributed design processes necessitates an understanding of such distributed processes, to be able to build models on which new systems can be based.

Each individual designer reasons explicitly about the situation in which the design process is performed: about his/her interpretation of a specific situation (Gero, 1998; Maher et al., 2000; Gero & Kannengiesser, 2002). Each designer has his or her own view of the world and other agents, and their environments, including assessments of their expertise, reliability, experience, trust, and so forth. Such assessments require reflection (Schön, 1983). A model of the types of reflection of individual designers in distributed design is presented in Brazier et al. (2001*a*).

This paper focuses on the role of trust in distributed design. Distributed design involves a number of participants, each with their own characteristics: for example, expertise, experience, goals, and attitudes (e.g., Busby, 2001). Information acquired from different participants may be valued differently in terms of accuracy and trustworthiness, depending on the context in which it is acquired. Human participants in a distributed design setting often know whom they trust, and whose abilities they value, and when. This knowledge is not often made explicit while it does influence distributed design processes (i.e., the way in which members of a design team assess and incorporate each others' designs, objectives, and evaluations). These trust relations need to be made explicit to be able to acquire the models with which complete or partial design support systems can be developed.

Agents are a useful metaphor for designers. Agents, in this context, are defined as social, reactive, proactive, and autonomous entities (Wooldridge & Jennings, 1995). This is supported by work done, for example, by Lawson (1997) and Cross et al. (1996) in design: autonomy, cooperation, and competition are the basic characteristics of designer agents to which they refer:

Reprint requests to: Frances M.T. Brazier, Intelligent Interactive Distributed Systems Group, Faculty of Sciences, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. E-mail: fmt.brazier@few.vu.nl

- A designer agent needs autonomy to exert control over its own processes, deliberate about cooperative work, its attitude, and so forth.
- A designer agent needs to be cooperative to jointly work with other designer agents, employing shared ontologies, shared protocols, shared communication languages, shared agreements, a shared model of design, and so forth.
- A designer agent needs to be competitive to deliberate about its own goals in relation to the goals of other agents and the aims of the design project, make (somewhat) selfish decisions to prevent overload, persuade other designers, and so forth.

Cooperation between agents is essential for the single function agents of Grecu and Brown (1996). These agents, however, have very limited knowledge. Relatively few examples of the use of agents in design processes exist. In A-design (Campbell et al., 1998), agents are used to model evolutionary computation: designer agents are specialized in either creating, assessing, or removing solutions. As such, their role in a distributed process is not that of a human designer. This is the case for the agents used by McAlinden et al. (1998) to support information and knowledge handling in a design project. Collaboration between these agents is, however, minimal.

Cooperation in distributed or concurrent human design processes is subject to automation by facilitation (Boujut & Laureillard, 2002), for example, by introducing tools to monitor progress and understanding (Hill et al., 2002), and to analyze participation (Simoff & Maher, 2000). Collaboration is often made explicit in agent-based design systems from an engineering perspective (e.g., Wilson & Shi, 1996; Lees et al., 2001; Anumba et al., 2002; Liu & Frazer, 2002; Zha, 2002), focusing on task coordination without explicitly incorporating trust. An overview of design processes involving collaborative agents is provided by, for example, Wang et al. (2002).

This paper first discusses the current state of art with respect to research on trust in Section 2. A computational model of trust, based on Castelfranchi and Falcone (2000) is introduced. This model is used in a knowledge-level analysis of trust relations in distributed Website design. Two specific design projects (Us Media) are analyzed, in particular with respect to task delegation from the perspective of a single designer. Formalization of the knowledge involved (including trust) provided a means to computationally simulate the processes involved, making it possible to evaluate the results. Section 3 presents this work. Section 4 discusses the results and indicates areas in which further research is required.

2. TRUST

This section presents a brief overview of research on trust in agent systems and design. Section 2.1 explores the notion of trust. Section 2.2 discusses research on trust related to agents and design. Section 2.3 discusses trust models. Section 2.4 introduces a new computational trust model based on Castelfranchi and Falcone's (2000) trust model. This model is used to analyze the role trust plays in a distributed Website design project described in Section 3.

2.1. Defining trust

Trust is a complex, subjective concept with many definitions from various fields of research such as psychology, management and communications, sociology, economics, and political sciences (McKnight et al., 1998; McKnight & Chervany, 2001). McKnight and Chervany (2001) and McKnight et al. (2002) specify a conceptual typology of high-level trust and distrust concepts on the basis of literature research, combining trust definitions from various research areas. Five conceptual trust types are distinguished and can be used to guide researchers in choosing their definition of trust. The five types defined by McKnight et al. (2002) and the relations between these types are depicted in Figure 1.

- Disposition to trust: the general extent to which trust is placed in others.
- Institution-based trust: the truster believes favorable conditions are in place.
- Trusting beliefs: the extent to which the truster believes characteristics of the trustee.
- Trusting intentions: the truster is *willing* (intending) to depend on the trustee.
- Trust-related behavior: the truster *depends* on the trustee.

Trust is clearly about consequences related to risk and actions (e.g., Luhmann, 2000). McKnight et al. (1998) distinguish more precisely three essential elements in trust: potential negative consequences, dependency, and feelings of security. In this paper, trust is viewed as a combination of: trusting beliefs, trusting intentions, and trust-related behavior. This is in accordance with Gambetta (2000): the subjective probability that one or more trustees will perform a particular action.

2.2. Trust and agents

Whenever agents (human or automated) cooperate, compete, perform transactions, or engage in other interactions,

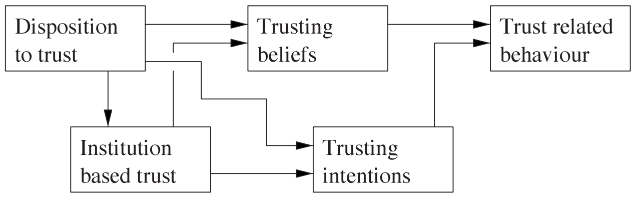

Fig. 1. Trust types and their relations (McKnight & Chervany, 2001).

trust plays a role (Deutch, 1962; Castelfranchi & Falcone, 1998; Cambetta, 2000; Falcone et al., 2001). In essence, all cooperative tasks include trust (Gambetta, 2000), as an individual reduces autonomy by increasing dependence on other individuals. Although not often recognized explicitly (e.g., Schön, 1983; Valkenburg & Dorst, 1998; Coates et al., 2000; Brazier et al., 2001*b*; Chao et al., 2002; Wang et al., 2002), this is also the case in distributed design (Brazier & Wijngaards, 2002). In distributed design, trust plays a role in delegation (e.g., Milewski & Lewis, 1997), and trust plays a role in assessing information sources (Hertzum & Mark Pejtersen, 2000; Hertzum, 2002). Trust in other agents, trust in information, and trust in organizations, all play a role.

This paper focuses on an individual designer's trust in other agents. An individual designer in a distributed setting is a reflective designer (e.g., Schön, 1983; Atman et al., 1999; Adams et al., 2003) whose professional skills also involve interpersonal and project management skills (Lewis & Bonollo, 2002), sometimes assuming different roles (e.g., Sonnenwald, 1996). Different aspects of trust identified in the literature (e.g., for an overview, see Wang et al., 2002) include reputation (e.g., Craig & Zimring, 2000; Lang et al., 2002), knowledge about collaborators (e.g., Denton, 1996; Busby, 2001), delegation and coordination (e.g., Wilson & Shi, 1996; Tambe, 1997; Valkenburg & Dorst, 1998; Jensen et al., 2000; Pynadath & Tambe, 2002), risk (e.g., Valkenburg & Dorst, 1998), expectations (Gero & Kannengiesser, 2002), and judgements (Holt, 1997).

The trust an agent has in other agents is, in general, based on a number of factors including its own direct and indirect experiences (e.g., observations or deduction), on other agents' experiences, on an agents' reputation (Aberer & Despotovic, 2001; Mui et al., 2002). In general, trust is not transitive (i.e., recommendations cannot be passed on), and trust is context dependent and dynamic. It has also been noted that trust and distrust are most often reciprocal by nature (Lawson, 1997; Falcone & Castelfranchi, 2001).

In some situations an agent will trust an agent fully, in others it may not. That an individual's reputation (Baya & Leifer, 1996; Lang et al., 2002) influences design processes is recognized. It clearly also influences the trust other designers have in other individual agents.

The trust agents have in each other changes over time. Agents continually update beliefs in other agents (Beth et al., 1994; Barber & Kim, 2001; Birk, 2001; Witkowski et al., 2001), themselves, and their environment. They continually need to deal with the "trust dilemma": the trade-off between positive or negative results of trusting another agent versus positive or negative results of not trusting another agent. Trust and risk are correlated concepts; trust is used to rationalize decisions involving risk (e.g., McKnight et al., 2002).

In small environments direct interaction-derived reputation-based trust mechanisms may suffice (Birk, 2000; Witkowski et al., 2001) to model the role trust plays. In open, dynamic environments in which incentives may differ it is questionable whether agents will always be truthful with respect to the information they provide about other agents (Beth et al., 1994; Schillo et al., 1999; Abdul-Rahman & Hailes, 2000; Jurca & Faltings, 2002).

2.3. Trust models

Existing models of trust express acquisition and representation of trust (e.g., Castelfranchi & Falcone, 2000), communication of trust (e.g., Aberer & Despotovis, 2001), and reasoning about trust (e.g., Marsh, 1994). Different trust models have been devised for different purposes, a number of which are discussed in this section.

Our knowledge-level approach to modeling individuals' involvement in distributed design processes yields insight in their aggregate behavior. To explore the role of trust, trust models are sought that support this approach. This implies that applicable trust models need to be knowledge-level models: they need to be defined in terms of intentionalistic notions such as beliefs, desires, and intentions. Explicit representation of the multifaceted nature of trust is also of importance for reasoning with and about trust. To facilitate experimentation, applicable trust models also need to be operationalizable.

Trust models such as Bell–LaPadula (Bell & LaPadula, 1973) distinguish different levels of trust and the relations between them. Such models require explicit knowledge of the levels of trust within a domain, and the role of individual agents in this hierarchy. The Bell–LaPadula model is designed for the military, in which such roles are clearly distinguished. Such relations are less easily defined for open distributed systems in which large numbers of agents operate. The Bell–LaPadula model is symbolic and not a knowledge-level model; trust is not expressed as a single value. It is operationalized: a number of implementations of Bell–LaPadula exist.

Marsh (1994) models trust in a simple but expressive way. In his model an agent's trust in another agent is based on three types of trust: basic, general, and situational trust. A value is assigned to each type (a continuous value between -1 inclusive and 1 exclusive). Basic trust expresses the disposition of an agent towards trust in general. General trust, the trust of an agent in a specific other agent unrelated to the situation or context, may be influenced by the value of basic trust. Situational trust, the trust of an agent in a specific other agent in a specific situation or context, is based on an estimation of general trust combined with the utility and importance of the situation. The threshold used to determine whether an agent trusts another agent sufficiently to cooperate with the other agent, is based on perceived risk, competence, and importance. A decision to cooperate is made when the situational trust is higher than the threshold. Marsh's trust model is to some extent a knowledge-level model, and different types of trust are distinguished, facilitating reasoning about and with trust. It is, however, not operationalized.

TrustBuilder (Winslett et al., 2002) is a trust management system. Its purpose is to negotiate which resources agents may access, based on both specific agent's and Trustbuilder systems' policies and credentials. Both agents and the TrustBuilder system have credentials: for example, proof of membership of an organization, a credit card number, or specific authorization certificates. Both also have policies describing the parties to whom they will disclose those credentials. During negotiation a TrustBuilder system and an agent exchange policies and credentials until the TrustBuilder system is satisfied, or until one of the parties is unable or unwilling to provide the requested credentials. TrustBuilder does not determine how credentials (the basis for trust) are acquired. Trustbuilder is symbolic and not a knowledge-level model. The trust valuation does not reflect its multifaceted basis. It is operationalized.

Aberer and Despotovic (2001) combine a trust model with a peer-to-peer trust storage model, storing complaints of agents about interactions with other agents in a global distributed model. Trust values are either 0 or 1, that is, dishonest or honest. An agent that is considering interaction with another agent asks its neighbors (peers) if they trust the other agents: in this model this translates to having knowledge of complaints and/or complaints about complaining agents. If the number of complaints found does not exceed an agent's threshold for acceptance it may decide to initiate interaction. This simple model may be suitable for straightforward applications such as trading communities (e.g., Ibazar). This trust storage model is a knowledge-level model, and explicitly involves the basis for trust, facilitating reasoning about and with trust. It is also operationalized. However, its assumption that trust is based on complaints is not easily (or generally) applied to a distributed design setting: more aspects are of importance than conflicts (which can be considered as a kind of complaint).

Ramchurn et al. (2003) describe a trust model based on confidence and reputation in the context of negotiation and contracts. Confidence information is acquired on the basis of an agent's personal experiences; reputation information is gathered from other agents. Trust in a specific context has a value between 0 and 1, and is computed by weighing confidence information stronger than reputation information. This is a knowledge-level model, in which confidence and reputation information are combined into a single trust value. It is not multifaceted. It is unclear whether this model has been operationalized.

Castelfranchi and Falcone (2000) describe a more elaborate trust model. They distinguish seven types of beliefs related to trust. These types of beliefs can be described as follows:

1. *Competence belief:* a belief that the other agent has the abilities to do the tasks.
2. *Disposition belief:* a belief that the other agent is inclined do what it says it will do.
3. *Dependence belief:* a belief that it is better to rely/depend on the other agent than to approach a task without the other agent.
4. *Fulfilment belief:* a belief that the goal will be achieved due to the other agent's contribution.
5. *Willingness belief:* a belief that the other agent has decided and intends to do an action (to achieve the goal).
6. *Persistence belief:* a belief that the other agent is stable in its intentions (related to reliability).
7. *Self-confidence belief:* a belief that the other agent knows that it can do an action.

Based on these beliefs and their subjective certainty a degree of trust is calculated. The decision to delegate, rely, or bet on an other agent is based on a comparison of the degree of trust with the risk factor and possibly a self-trust estimation. This trust model is a knowledge-level model, is operationalized, and combines the values of its seven beliefs into a single value of trust.

2.4. Our model

The seven beliefs related to trust, distinguished by Castelfranchi and Falcone (1998, 2000), described above, provide a means to structure trust encountered in distributed design. However, the computational model Castelfranchi and Falcone presented (1998), combines these beliefs in a single-valued expression of trust, thereby reducing the options for agents to explicitly reason about different aspects of trust. A different computational model is needed to model trust relations in distributed design.

The model designed for this purpose is based on the seven beliefs, but uses a different mechanism to express trust. A threshold (vector) is used to express the threshold values for each of the seven beliefs: the values needed for one agent to trust another in a given context. These thresholds are determined for each agent with respect to each other agent. The thresholds change over time, as do the beliefs agents have in each other. Comparing agent A's threshold values with the beliefs agent A has in agent B, provides enough information to determine to which extent agent A trusts agent B. (Note that agent A may be equal to agent B.) The use of thresholds is similar to Marsh's (1994) approach: our beliefs are similar to Marsh's situational trust, and our threshold is compared with our beliefs, akin to comparing Marsh's threshold with Marsh's situational trust.

The proposed model uses simple values to express knowledge-level beliefs and thresholds: discrete values from -2 to $+2$ have been used to facilitate computation and to have a sufficiently explicit representation of "degrees of belief." Knowledge-level valuations are mapped into a discrete domain, on the assumption that five-valued predicates are sufficient to elicit information from humans (e.g., the Likert scale; Likert, 1932).

The mechanism(s) with which the valuations of the aforementioned beliefs are determined are not discussed in this paper. These beliefs can be based on, for example, agents' reputations, based on personal experiences, or experiences by other agents (cf. Aberer & Despotovic, 2001). If beliefs have continuous values then they need to be mapped to discrete values when used in this trust model, requiring explicit choices in the assessment of the beliefs in agents in specific contexts. This is not discussed in this paper, because in our experiment, a knowledge-level analysis yielded five values per belief.

As stated above the belief of an agent in another specific agent in a specific context is expressed as a vector of seven values corresponding to the seven beliefs distinguished by Castelfranchi and Falcone:

```
Current_Belief_in( agent, context,
                               < 2, 1, 1, 2, 2, 2, 1 > )
```

The threshold expressing the levels of belief required for each individual type of belief in a specific context, is also expressed as a vector of seven values: the values express the minimum value for which each specific belief is deemed to be acceptable (in combination with the other six).

```
Threshold( context, < 1, 2, 1, 2, 2, 2, 2 > )
```

Evaluating the beliefs about agents with respect to a threshold corresponds to calculating the difference between the minimally required values and the actual belief values in a specific context, yielding discrete values between -4 and $+4$, inclusive. These extreme values arise when, for example, the threshold specifies a value of $+2$, yet the belief value is -2, or vice versa. The evaluation has negative values for threshold violations, zeros for exact matches, and positive values for exceptional satisfactions:

```
Evaluation( agent, context,
                  < 1, −1, 0, 0, 0, 0, −1 > )
```

Comparing evaluations of agents and choosing the best alternative may involve more complex algorithms. For the domain of application discussed in the next section, namely distributed Website design, a simple heuristic is used. This heuristic is based on counting the number of violations, exact matches, and exceptional satisfactions. This also makes it possible to choose between "worst" candidates, a feature that may be relevant in situations in which work simply has to be performed. In the case of a tie, a random choice is made. The algorithm is shown in the pseudocode in Figure 2. The next section demonstrates the use of this trust model.

3. DISTRIBUTED WEBSITE DESIGN

Distributed Website design, in which a number of team members each with their own expertise and experience, collectively design and build a Website, is an example of a distributed design process in which trust plays an important role. In this section, the trust model presented above is used to analyze two specific design processes (Us Media) and to evaluate the trust model presented above. A computational simulation of these processes demonstrates how trust and trust thresholds influence the design process' results. Section 3.1 describes the Website design application. Section 3.2 depicts the specific beliefs and trust thresholds involved for task delegation. Section 3.3 illustrates the effect of different trust thresholds.

3.1. Design application

Two specific distributed Website design projects are described in this section: one for the design of a Website for a nonprofit organization, the other is the design of a project for a com-

```
Let v be the set of violaters
Let m be the set of minimal compliers
Let s be the set of superb compliers

For each evaluation e:
        add to v if e has one or more negative elements.
        add to s if e has one or more positive elements and no negative elements.
        add to m if e has no positive elements and no negative elements.

If s has one element,
        return that element as the best evaluation
else if s has more than one evaluation
        return the evaluation with most positive elements
else if m has one evaluation
        return that evaluation
else if m has more than one evaluation
        return an evaluation
else
        return the evaluation in s with the least number of negative elements
```

Fig. 2. Pseudocode for evaluation selection.

pany (Us Media). The same team is responsible for both projects, and the design process is similar.

In general, the Websites this design team produces include separate sections for news, addresses, and an interactive forum. The nonprofit organization requires a Website to promote physics, and is aimed at high school students. The Website is to contain information about the field of physics, news items on progress in science, explanatory articles, example exercises, addresses of professors, support and ideas for school projects, and information about BS and MS physics programs. This customer's emphasis is on the functionality of the Website: on the information to be communicated and shared, and to a lesser extent on "look and feel." An extended example of such a Website can be found at www.natuurkunde.nl (in Dutch).

The other site is for a company. Its overall goal is to support and attract clientele. The Website is to contain information about the products, addresses of dealers and repair shops, and discussion about technologies. This customer's emphasis is more on the look and feel, than on the functionality provided by the Website.

The design team includes graphical designers, logical designers, code designers (programmers), and HTML designers. A Website consists of models for its graphical and logical layout, and (executable) specifications of its functionality and HTML. Briefly summarized, graphical designers design the overall "look & feel" of the Website, logical designers design the "flow": the sequences of pages a client encounters, code designers write functional code, and HTML designers implement the layout in HTML pages.

Usually, the graphical and logical designers interact with a customer, and subsequently delegate work to the code designers and HTML designers. No strict hierarchy is enforced, and each participant is able to delegate work to other participants. In all cases, the participant to whom a job has been delegated reports back to the participant who assigned the job. Each designer usually adds his or her own requirements and results [(partial) designs] to the overall project information, maintained by a version control system (e.g., CVS).

Work delegated to a designer (by the customer or by other designers) includes (partial) Website descriptions, specifications/requirements (possibly qualified), and design project goals. The designer reports include information on partial models of the Website, specifications (requirements) of Website models, the extent to which specifications have been fulfilled, and possible problems encountered, including conflicting specifications (assessments of requirements with respect to Website descriptions), and whether project goals have been fulfilled, for example, finishing work on time with the expected quality.

3.2. Role of trust in Website design

The role the trust model, described in Section 2.3, plays in Website design, is illustrated in this section for task delegation within the design projects described above. Beliefs about other designers are evaluated with a trust threshold for different contexts. The resulting analysis was implemented in a computational simulation, using these beliefs and trust thresholds. The trace can be found at www.iids.org/research/distributed_design/aiedam2004/. The agents in this trace represent their human counterparts. Figure 3 shows part of this trace, in which the graphical designer agent deliberates about delegating work to one of the two HTML designer agents.

In this example, the graphical designer needs one of the two HTML designers to implement the layout of the news section of the Website. The graphical designer's beliefs about the two HTML designers are expressed as the graphical designer's agent's context-specific valuations of its belief predicates concerning the fitness of two agents. In this context the graphical designer needs an HTML designer who is capable of implementing the graphical design requirements of the news section of the Website in 1 day. These beliefs are expressed as follows for the graphical designer agent:

```
Current_Belief_in( htmler_1, newsDesign,
                    < 1, 1, 1, 1, 1, 2, 2 > )

Current_Belief_in( htmler_2, newsDesign,
                   < 2, 1, 1, 2, 2, 2, 1 > )
```

The semantics of the order of the beliefs (and trust thresholds) in the notation that is used is

```
< Competence, Disposition, Dependence,
    Fulfilment, Willingness, Persistence,
                              Self-confidence >
```

The interpretation of the belief about the first HTML designer agent is given below; this agent has the following:

- Competence 1: which indicates that he/she has all the competencies needed.
- Disposition 1: usually does what he/she promises.
- Dependence 1: is a better alternative than using myself.
- Fulfilment 1: usually designs HTML satisfying requirements on time.
- Willingness 1: is normally willing to work on this task.
- Persistence 2: is very reliable.
- Self-confidence 2: is well aware of his/her layouting abilities.

The graphical designer agent employs the following trust threshold for this context:

```
Threshold( newsDesign,
                   < 1, 2, 1, 2, 2, 2, 2 > )
```

The rationale for this threshold is that the graphical designer wants to be convinced that the designer who is to

```
"Graphic Designer" (t=6)
input:
       Requirement Qualification Set:
       Design Process Evaluations:
              dpe-gd-1(fin-gd-1(dsgn-gd-1(addressbook)))
       Design Process Objectives:
       Assessment:
       DesignObjectDescription:
reasoning:
       Checking incoming customer requirements
       Checking for missing designs ****
       Designing [news]
              Delegating news [1, 2, 1, 2, 2, 2, 2](t=1)
                     Current_Belief in [HTMLer 1] = [1, 1, 1, 1, 1, 2, 2]
                     Current_Belief in [HTMLer 2] = [2, 1, 1, 2, 2, 2, 1]
                     [HTMLer 1] [news] <0, -1, 0, -1, -1, 0, 0> (3, 4, 0)
                     [HTMLer 2] [news] <1, -1, 0, 0, 0, 0, -1> (2, 4, 1)
              Selected [HTMLer 2]
       Checking for done state
output:
       Requirement Qualification Set:
              rqs-gd-5(html-gd-2(news))
       Design Process Evaluations:
              dpe-gd-2(fin-gd-2(dsgn-gd-2(news)))
       Design Process Objectives:
              dpo-gd-2(udod-gd-2(dod-gd-2))
       Assessment:
       DesignObjectDescription:
       dod-gd-2(dsgn-gd-2(news))
```

Fig. 3. An excerpt from a trace illustrating the graphical designer agent's reasoning about his beliefs and trust in the HTML designer agents.

do the HTML work has a high likelihood of success for this task. The knowledge-level concepts with which this is modeled and on which these values are based, are as follows:

- Competence of $\geq +1$: is reasonably competent for this context.
- Disposition of $\geq +2$: always does what he/she agrees upon.
- Dependence of $\geq +1$: is a better alternative than myself.
- Fulfilment of $\geq +2$: will definitely successfully complete the task.
- Willingness of $\geq +2$: is really willing to take on the task.
- Persistence of $\geq +2$: is very reliable.
- Self-confidence of $\geq +2$: has full self-confidence in him/herself for this task.

In this example the following evaluation of the two HTML designer agents is obtained:

```
Evaluation( HTMLer_1, newsDesign,
                   < 0, −1, 0, −1, −1, 0, 0 > )

Evaluation( HTMLer_2, newsDesign,
                   < +1, −1, 0, 0, 0, 0, −1 > )
```

In other areas of application comparing evaluations of the parties involved and choosing the best alternative may involve more complex algorithms. In this domain of application, a simple heuristic is used, based on counting the number of violations, exact matches, and exceptional satisfactions. In the running example, this amounts to

```
Evaluation-Summary( htmler-1, newsDesign,
                              < 3, 4, 0 > )

Evaluation-Summary( htmler-2, newsDesign,
                              < 2, 4, 1 > )
```

On the basis of this algorithm no satisfactory candidates are found (as the model of both HTML designers violates at least two threshold values), so the most promising HTML designer has to be chosen. HTML designer 2 is preferable over HTML designer 1, as the former has the most exact matches and exceptional satisfactions.

3.3. Role of trust in work delegation

The design of the news section of the Website entails collaboration between designers of HTML and designers of code. The code designers provide the code needed to implement the functionality and flow depicted in HTML pages, based on the logical requirements with which they are provided. This design of code and HTML for the news section is used to illustrate the effect of changing trust threshold on results of the design process.

3.3.1. *First project*

The following situation occurs in the design of the Website for the nonprofit organization (the physics Website). Code designer 2 is responsible for adding functionality to the HTML that the HTML designer has provided for the news section of the Website. To this end, the code designer 2 formulates requirements concerning the data items to be stored for a news item, for example, title, abstract, author, section titles, paragraphs, images, movies, links, and so forth. A number of these requirements are violated in the HTML for the news section, provided by HTML designer 2: a conflict occurs. Code designer agent 2 has a number of options to resolve this conflict:

1. The code designer does not change his own requirements, and changes the HTML for the news section accordingly without consulting an HTML designer agent; possibly resulting in suboptimal HTML.
2. The code designer asks HTML designer 2 to change the HTML for the news section; this may take some time but results in suitable HTML.
3. Alternatively, the code designer may ask HTML designer 1 for help; this may take more time, as this HTML designer was unfamiliar with this part of the Website.
4. The code designer works in parallel with an HTML designer; this implies that the code designer quickly fixes the HTML for his own purposes, and later integrates the properly fixed HTML from the HTML designer.
5. The code designer drops his own requirements.

The last option conflicts with the logical design requirements that state the need for elaborate news items, including more possibilities for images, text layout, and external links than supported by the (assumed) design requirements of HTML designer a 2. The fourth option, involving parallelism, was not favored by the code designer, as quickly fixing HTML may involve as much work as doing it properly, and has the drawback that other conflicts may arise.

Code designer 2's beliefs are modeled according to the trust model, represented by code designer 2's agent. The beliefs about HTML designer 2 show the code designer's bias about this HTML designer, who originally designed the HTML for the news section. The code designer's agent also explicitly models beliefs about the code designer himself, as he is a possible candidate for work in this context.

```
format: < Compet, Dispos, Depend, Fulfil,
                    Willing, Persist, Self-conf >

Belief( progger-2, fixNewsHtml,
                         < 1, 2, 2, 2, 2, 2, 1 > )

Belief( htmler-1, fixNewsHtml,
                         < 1, 1, 0, 1, 1, 2, 2 > )

Belief( htmler-2, fixNewsHtml,
                         < 2, 2, 1, 2, 1, 2, 2 > )
```

The code designer agent has a trust threshold based on the code designer's principle that making progress is more important than realizing graphical quality. This results in the following trust threshold, acquired during our knowledge-level analysis, in which willingness and persistence are shown to be more important than other aspects:

```
Threshold_1( fixNewsHtml,
                         < 1, 1, 1, 1, 2, 2, 1 > )
```

The following evaluation of the beliefs about the three parties is obtained:

```
Evaluation( progger-2, fixNewsHtml,
         < 0, +1, +1, +1, 0, 0, 0 > )
                                   = 0, 4, 3

Evaluation( HTMLer_1, fixNewsHtml,
         < 0, 0, −1, 0, −1, 0, +1 > )
                                   = 2, 4, 1

Evaluation( HTMLer_2, fixNewsHtml,
        < +1, +1, 0, +1, −1, 0, +1 > )
                                   = 1, 2, 4
```

In this situation, the models of both HTML designers have at least one violation, and the model of the code designer himself has no violations at all. As a result, the code designer is shown to be the best candidate for fixing the HTML. In other words, the code designer agent 2 decides to choose option 1: to have the code designer fix the HTML himself.

3.3.2. *Second project*

The same situation occurs in the second design project considered: the Website for company, for which look & feel are more important. The same conflict between code designers and HTML designers occurs. In this case the code designer believes the quality of graphics is more important than making progress. His agent is therefore implemented with a trust threshold based on this. This trust threshold, acquired during our knowledge-level analysis, expresses the fact that competence, fulfilment, reliability and self-confidence are more important:

```
Threshold_2( fixNewsHtml,
                         < 2, 1, 1, 2, 1, 2, 2 > )
```

The following evaluation of the beliefs about the three parties is obtained:

```
Evaluation( progger-2, fixNewsHtml,
       < −1, +1, +1, 0, +1, 0, −1 > )
                                    = 2, 2, 3

Evaluation( HTMLer_1, fixNewsHtml,
        < −1, 0, −1, −1, 0, 0, 0 > )
                                    = 3, 4, 0

Evaluation( HTMLer_2, fixNewsHtml,
          < 0, +1, 0, 0, 0, 0, 0 >)
                                    = 0, 6, 1
```

In this situation, the code designer agent and the model of HTML designer 1 have at least two violations, while the model of HTML designer 2 has no violations at all. HTML designer 2 is chosen as the candidate for fixing the HTML. As a result, code designer 2 has time to do other work, before resuming work on implementing code for the HTML of the news section.

3.3.3. *Role of trust*

In both situations described above, the code designer is entrusted with a conflict with respect to the news section. He needs to extend the information stored for news items; the Website design provided by the HTML designer is incomplete. The code designer's approach to resolve this conflict is based on characteristics of the current customer, as shown above. The resulting HTML for the news items is of a better quality in the second case than in the first case (mostly noticeable in details).

The two cases illustrate the role of trust in deliberating delegation of work, and subsequent effects on a design process. The combination of explicitly modeling seven beliefs for a specific agent in a specific context and employing a trust threshold for each of the seven beliefs facilitates the explication of trust in both cases. For this example, the human expert's notions of trust were fairly easily captured in the trust model.

4. DISCUSSION

The role of trust in distributed design is the main topic of this paper. Human designers most often unconsciously assess the trust they have in other members of a design team, judging the value of their input accordingly (Milewski & Lewis, 1997). Automated support of distributed design requires an understanding of these processes. Fully automated distributed design requires computational models of distributed design including explication of trust relations. As yet, it is infeasible to automatically assign trust levels to agents (see Falcone et al., 2001) because of the dynamic nature of trust. These relations will, however, need to be understood if automated support is to play a significant role in distributed design.

The trust model presented in this paper is based on Castelfranchi and Falcone's (2000) model, in which seven beliefs are related to trust. Castelfranchi and Falcone (1998) compute a single value of trust on the basis of seven beliefs; the trust model in this paper uses a more expressive variant involving seven separate values for both trust thresholds and trust evaluation.

The example used to illustrate the potential of the model is that of task delegation within a design team. The reasoning processes involved are similar to the reasoning processes involved in determining the risk in taking a (cooperative) action (Griffiths & Luck, 1999). Griffiths and Luck specify trust related to the notion of general trust (Marsh, 1994), and explicitly omit situational trust for being too computationally expensive. The example in Section 3 illustrates the role of trust in task delegation within a closed environment. Different trust valuations result in different results. This example is based on the experiences of the human code designer and his rationalization of beliefs and trust thresholds. The design results differ in both quality of the Website (albeit in details), different sets of (qualified) requirements, and different usage of time for involved designers.

Current research focuses on a different domain of application: the role of trust in distributed system location management. The simple model of trust reported in Brazier and Wijngaards (2002) is being replaced by the model reported in this paper. AgentScape (Wijngaards et al., 2002) is a worldwide scalable distributed agent platform. Management of AgentScape sites (i.e., locations within AgentScape) is a fully automated distributed configuration problem, an example of automated distributed design (Brazier & Wijngaards, 2002). An essential element in this design problem is its dynamic and open nature: the environment changes and the local configuration adapts.

Open environments place additional requirements on a trust model. The most obvious is the need to determine ways to acquire and adapt the values for the beliefs distinguished. Current experience in the above-mentioned domain is promising. Most beliefs can be identified. Additional factors, however, play a role (e.g., factors related to an agent's environment such as security). TrustBuilder should work well within open environments, as does the approach based on distributed trust in open multiagent systems, based on certificates, described by Mass and Shehory (2001; as an extension of work done by Wong and Sycara, 2000). Mass and Shehory's approach allows agents to establish trust among themselves and update this trust when necessary without necessarily identifying themselves explicitly, basically a reputation-based scheme. Comparing incentives is less easily achieved in open environments as most individual incentives are not easily compared.

Our research on the role of trust in distributed design is just a beginning; more research is clearly needed.

ACKNOWLEDGMENTS

The authors express their gratitude to Cristiano Castelfranchi for his comments on this manuscript, to Us Media (www.usmedia.nl) for providing material for the Website design example and for their permission to use the data on these projects, and to Stichting NLnet (www.nlnet.nl) for their support.

REFERENCES

Abdul-Rahman, A., & Hailes, S. (2000). Supporting trust in virtual communities. *Proc. 33rd Hawaii Int. Conf. System Sciences*, Vol. 6, pp. 6007. Piscataway, NJ: IEEE Press.

Aberer, K., & Despotovic, Z. (2001). Managing trust in a peer-2-peer information system. *Proc. 10th Int. Conf. Information and Knowledge Management*, pp. 310–317. New York: ACM Press.

Adams, R.S., Turns, J. & Atman, F.J. (2003). Educating effective engineering designers: The role of reflective practice. *Design Studies 24(3)*, 275–294.

Anumba, C.J., Ugwu, O.O., Newnham, L. & Thorpe, A. (2002). Collaborative design of structures using intelligent agents. *Automation in Construction 11(1)*, 89–103.

Atman, C.J., Chimka, J.R., Bursic, K.M., & Nachtmann, H.L. (1999). A comparison of freshman and senior engineering design processes. *Design Studies 20(2)*, 131–152.

Barber, K.S., & Kim, J. (2001). Belief revision process based on trust: Agents evaluating reputation of information sources. *Trust in Cyber-Societies, Integrating the Human and Artificial Perspectives* (Falcone, R., Singh, M.P., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 73–82. Berlin: Springer.

Baya, V., & Leifer, L.J. (1996). Understanding information management in conceptual design. In *Analysing Design Activity* (Cross, N., Christiaans, H., & Dorst, K., Eds.), pp. 151–168. Chichester: Wiley.

Bell, D.E., & LaPadula, L.J. (1973). *Secure Computer Systems: Mathematical Foundations and Model* (Report No. MTR 2547 v2). Bedford, MA: MITRE Corporation.

Beth, T., Borcherding, M., & Klein, B. (1994). Valuation of trust in open networks. *Proc. 3rd European Symp. Research in Computer Security*, pp. 3–18.

Birk, A. (2000). A boosting cooperation by evolving trust. *Applied AI 14*, 769–784.

Birk, A. (2001). Learning to trust. *Trust in Cyber-Societies: Integrating the Human and Artificial Perspectives* (R. Falcone, R., Singh, M., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 133–144. Berlin: Springer.

Boujut, J.-F., & Laureillard, P. (2002). A co-operation framework for product–process integration in engineering design. *Design Studies 23(6)*, 497–513.

Brazier, F.M.T., Jonker, C.M., Treur, J., & Wijngaards, N.J.E. (2001*b*). Compositional design of a generic design agent. *Design Studies 22(5)*, 439–471.

Brazier, F.M.T., Moshkina, L.V., & Wijngaards, N.J.E. (2001*a*). Knowledge level model of an individual designer agent in collaborative distributed design. *Journal of Artificial Intelligence in Engineering 15*, 137–152.

Brazier, F.M.T., & Wijngaards, N.J.E. (2002). The role of trust in automated distributed design. *Proc. Workshop on Agents in Design* (Gero, J.S., & Brazier, F.M.T., Eds.), pp. 71–84, University of Sydney.

Busby, J.S. (2001). Error and distributed cognition in design. *Design Studies 22(3)*, 233–254.

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

Castelfranchi, C., & Falcone, R. (1998). Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. *Third Int. Conf. Multi Agent Systems*, pp. 72–80.

Castelfranchi, C., & Falcone, R. (2000). Trust is much more than subjective probability: Mental components and sources of trust. *Proc. 33rd Hawaii Int. Conf. System Sciences*, Vol. 6, p. 6008. New York: IEEE Press.

Chao, K.-M., Norman, P., Anana, R., & James, A. (2002). An agent-based approach to engineering design. *Computers in Industry 48*, 17–27.

Coates, G., Duffy, A.H.B., Hills, W., & Whitfield, R.I. (2000). A Generic coordination approach applied to a manufacturing environment. *Journal of Materials Processing Technology 107*, 404–411.

Craig, D.L., & Zimring, C. (2000). Supporting collaborative design groups as design communities. *Design Studies 21(2)*, 187–204.

Cross, N., Christiaans, H., & Dorst, K. (1996). *Analysing Design Activity*. Chichester: Wiley.

Denton, H.G. (1996). Developing design teamworking capability: Some planning factors emerging from a survey of engineering design courses. *Proc. Int. Conf. Design and Technology Educational Research and Curriculum Development* (Smith, J.S., Ed.), pp. 112–117, IDATER 96, Loughborough University.

Dwarakanath, S., & Blessing, L. (1996). Ingredients of the design process: A comparison between group and individual work. In *Analysing Design Activity* (Cross, N., Christiaans, H., & Dros, K., Eds.), pp. 93–116. Chichester: Wiley.

Falcone, R., & Castelfranchi, C. (2001). The socio-cognitive dynamics of trust: Does trust create trust? In *Trust in Cyber-Societies, Integrating the Human and Artificial Perspectives* (Falcone, R., Singh, M.P., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 55–72. Berlin: Springer.

Falcone, R., Singh, M.P., & Tan, Y.-H. (2001). Introduction: Bringing together humans and artificial agents in cyber-societies: A New field of trust research. In *Trust in Cyber-Societies, Integrating the Human and Artificial Perspectives* (Falcone, R., Singh, M.P., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 1–8. Berlin: Springer.

Gambetta, D. (2000). Can we trust trust. In *Trust: Making and Breaking Cooperative Relations* (Cambetta, D., Ed.), pp. 213–237. Oxford: University of Oxford, Department of Sociology.

Gero, J.S. (1998). Conceptual designing as a sequence of situated acts. In *Artificial Intelligence in Structural Engineering* (Smith, I., Ed.), pp. 165–177. Berlin: Springer.

Gero, J.S., & Kannengiesser, U. (2002). The situated function–behaviour–structure framework. In *Artificial Intelligence in Design '02* (Gero, J.S., Ed.), pp. 89–104. Dordrecht: Kluwer Academic.

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

Griffiths, N., & Luck, M. (1999). Cooperative plan selection through trust. In *Multi-Agent System Engineering—Proc. Ninth European Workshop on Modelling Autonomous Agents in a Multi-Agent World* (Garijo, F.J., & Boman, M., Eds.), Lecture Notes in Computer Science, Vol. 1647, pp. 162–174. Berlin: Springer.

Hertzum, M. (2002). The importance of trust in software engineers' assessment and choice of information sources. *Information and Organization 12*, 1–18.

Hertzum, M., & Mark Pejtersen, A. (2000). The information-seeking practices of engineers: Searching for documents as well as for people. *Information Processing & Management 36(5)*, 761–778.

Hill, A.W., Dong, A., & Agogino, A.M. (2002). Towards computational tools for supporting the reflective team. *Artificial Intelligence in Design '02* (Gero, J.S., Ed.), pp. 305–325. Dordrecht: Kluwer Academic.

Holt, J.E. (1997). The designer's judgement. *Design Studies 18(1)*, 113–123.

Jensen, D., Feland, J., Bowe, M., & Self, B. (2000). A 6-hats based team formation strategy: Development and comparison with an MBTI based approach. In *Proc. American Society for Engineering Education ASEE Annual Conf.* [CD], session 2425, St. Louis, MO.

Jurca, R., & Faltings, B. (2002). Towards incentive-compatible reputation management. *Proc. Workshop "Deception, Fraud and Trust" of the Autonomous Agents Conf.* (Falcone, R., Barber, S., Korba, L., & Singh, M., Eds.), pp. 92–100. New York: ACM Press.

Lang, S.Y.T., Dickinson, J., & Buchal, R.O. (2002). Cognitive factors in distributed design. *Computers in Industry 48(1)*, 89–98.

Lawson, B. (1997). *How Designers Think: The Design Process Demystified*, 3rd ed. Oxford: Architectural Press.

Lees, B., Branki, C. & Aird, I. (2001). A framework for distributed agent-based engineering design support. *Automation in Construction 10(5)*, 631–637.

Lewis, W.P., & Bonollo, E. (2002). An analysis of professional skills in design: Implications for education and research. *Design Studies 23(4)*, 385–406.

Likert, R. (1932). *A Technique for the Measurement of Attitudes*. New York: Archives of Psychology.

Liu, H., & Frazer, J.H. (2002). Supporting evolution in a multi-agent cooperative design environment. *Advances in Engineering Software 33(6)*, 319–328.

Luhmann, N. (2000). Familiarity, confidence, trust: Problems and alternatives. In *Trust: Making and Breaking Cooperative Relations* (Gambetta, D., Ed.), Chap. 6, pp. 94–107. Oxford: University of Oxford, Department of Sociology.

Maher, M.L., Simoff, S., & Cicognani, A (2000). *Understanding Virtual Design Studios*. London: Springer.

Marsh, S.P. (1994). *Formalising Trust as a Computational Concept*. Stirling: University of Stirling, Department of Computing Science and Mathematics.

Mass, Y., & Shehory, O. (2001). Distributed trust in open multi-agent system. In *Trust in Cyber-Societies, Integrating the Human and Artificial Perspectives* (Falcone, R., Singh, M.P., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 111–132. Berlin: Springer.

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. *Artificial Intelligence in Design '98* (Gero, J.S., & Sudweeks, F., Eds.), pp. 537–556. Dordrecht: Kluwer Academic.

McKnight, D.H., & Chervany, N.L. (2001). Trust and distrust definitions: One bite at a time. In *Trust in Cyber-Societies: Integrating the Human and Artificial Perspectives* (Falcone, R., Singh, M.P., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 27–54. Berlin: Springer.

McKnight, D.H., Choudhury, V., & Kacmar, C. (2002). Developing and validating trust measures for e-commerce: An integrative typology. *Information Systems Research 13(3)*, 334–361.

McKnight, D.H., Cummings, L.L., & Chervany, N.L. (1998). Initial trust formation in new organizational relationships. *Academy of Management Review 23(3)*, 473–490.

Milewski, A.E., & Lewis, S.H. (1997). Delegating to software agents. *International Journal of Human–Computer Studies 46(4)*, 485–500.

Mui, L., Mohtashemi, M., & Halberstadt, A. (2002). Notions of reputation in multi-agents systems: A review. *Proc. First Int. Joint Conf. Autonomous Agents and Multiagent Systems*, pp. 280–287. New York: ACM Press.

Pynadath, D.V., & Tambe, M. (2002). Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. *First Autonomous Agents and Multiagent Systems Conf. (AAMAS02)*, pp. 873–880.

Ramchurn, S.D., Sierra, C., Godo, L., & Jennings, N.R. (2003), A computational trust model for multi-agent interactions based on confidence and reputation. *Proc. 6th Int. Workshop of Deception, Fraud and Trust in Agent Societies*, pp. 69–75, Melbourne, Australia.

Schillo, M., Funk, P., & Rovatsos, M. (1999). Who can you trust: Dealing with deception. *Proc. Workshop "Deception, Fraud and Trust" of the Autonomous Agents Conf.* (Castelfranchi, C., Tan, Y., Falcone, R., & Firozabad, B.S., Eds.), pp. 95–160. New York: ACM Press.

Schön, D.A. (1983). *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books (Perseus Books Group).

Simoff, S.J., & Maher, M.L. (2000). Analysing participation in collaborative design environments. *Design Studies 21(2)*, 119–144.

Sonnenwald, D.H. (1996). Communication roles that support collaboration during the design process. *Design Studies 17(3)*, 277–301.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research 7*, 83–124.

Valkenburg, R., & Dorst, K. (1998). The reflective practice of design teams. *Design Studies 19(3)*, 249–271.

Wang, L., Shen, W., Xie, H., Neelamkavil, J., & Pardasani, A. (2002). Collaborative conceptual design—State of the art and future trends. *Computer Aided Design 34(13)*, 981–996.

Wijngaards, N.J.E., Overeinder, B.J., van Steen, M., & Brazier, F.M.T. (2002). Supporting Internet-scale multi-agent systems. *Data and Knowledge Engineering 41(2–3)*, 229–245.

Wilson, J.L., & Shi, C. (1996). Co-ordination mechanisms for cooperative design. *Engineering Applications of Artificial Intelligence 9(4)*, 453–461.

Winslett, M., Yu, T., Seamons, K.E., Hess, A., Jacobson, J. Jarvis, R., Smith, B., & Yu, L. (2002). Negotiating trust on the Web. *IEEE Internet Computing 6(6)*, 30–37.

Witkowski, M., Artikis, A., & Pitt, J. (2001). Experiments in building experiential trust in a society of objective-trust based agents. In *Trust in Cyber-Societies, Integrating the Human and Artificial Perspectives* (Falcone, R., Singh, M.P., & Tan, Y.-H., Eds.), Lecture Notes in Computer Science, Vol. 2246, pp. 111–132. Berlin: Springer.

Wong, H.C., & Sycara, K.P. (2000). Adding security and trust to multiagent systems. *Applied Artificial Intelligence 14(9)*, 927–941.

Wooldridge, M.J., & Jennings, N.R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review 10(2)*, 115–152.

Zha, X.F. (2002). A knowledge intensive multi-agent framework for cooperative/collaborative design modeling and decision support of assemblies. *Knowledge-Based Systems 15(8)*, 493–506.

Frances M.T. Brazier is a full Professor and heads the Intelligent Interactive Distributed Systems (IIDS) Group. Her research focuses on the design and development of support for large-scale intelligent, interactive, distributed systems. This includes middleware (AgentScape), services (an agent factory and directory services), and applications (distributed information retrieval and distributed design), as well as consideration of the analysis of legal implications of agent technology (ALIAS), all in close collaboration with the IIDS group and others.

Niek Wijngaards is currently employed at the DECIS Lab, Delft, The Netherlands. Prior to October 2004, he was an Assistant Professor at the IIDS Group. He received his PhD in 1999, on the topic of self-modifying agent systems using a redesign process. Dr. Wijngaards current research focuses on design processes to support large-scale heterogeneous adaptive multiagent systems and ALIAS. One aspect is automated software configuration, including automated agent adaptation with agent factories and Web-service configuration. Distributed design is studied as an example of a complex multiagent system and as a service for the AgentScape agent operating system.

Hidde Boonstra has been a PhD student at the IIDS Group since 2003. He received his MS in 2001 and has been actively involved in the development of commercial Web applications. His current research interests focus on security and trust in agent operating systems and agent systems. Distributed design is an example of multiagent application to test models and theories.