Gero JS and Brazier FMT (eds) (2002) *Agents in Design 2002*
Key Centre of Design Computing and Cognition, University of Sydney, pp. 71-83.

ROLE OF TRUST IN AUTOMATED DISTRIBUTED DESIGN

FRANCES MT BRAZIER and NIEK JE WIJNGAARDS
Intelligent Interactive Distributed Systems Group
Vrije Universiteit Amsterdam The Netherlands

1. Introduction

Distributed design involves many participants, each with their own expertise and goals. Information acquired from different participants may be valued differently in terms of accuracy and trustworthiness. Human participants in a distributed design setting often know whom they trust, and whose abilities they value. This knowledge is not often made explicit. It does, however, influence distributed design processes (i.e. the way in which members of a design team assess and incorporate each others' designs, objectives, evaluations). These trust relations need to be made explicit to be able to effectively support distributed design.

Trust is a current topic of research in multi-agent systems research (e.g. Falcone et al. 2001; Mass and Shehory 2001; Wong and Sycara 2000). To our knowledge it has not been studied with respect to intelligent agents in design let alone intelligent agents in automated distributed design. It is also possible that trust models underlying automated distributed design, although based on models of human interaction, cannot capture the full scope of trust in human-to-human interaction. Design is, however, a more complex process than most of the processes for which multi-agent systems have been devised. The trust relations involved are more complex and situation dependent. They need to be understood to be able to be modelled if and when necessary. In this paper, the role of trust is discussed in the context of an example setting of an automated design application in a highly dynamic and open environment. Section 2 provides a brief overview of current research on trust in agent systems and design. Section 3 describes an example of a distributed design application, for which trust implications are analysed from the perspective of a single agent. Section 4 discusses the results.

2. Trust

In this section a brief overview is given of research on trust in agent systems and design. Section 2.1 discusses research on trust related to agents and their interactions. Section 2.2 dicusses trust in design by proposing a model of an individual designer in which trust is explicitly modelled.

2.1 TRUST IN AGENTS

Whenever agents (human or automated) co-operate, compete, perform transactions, or engage in other interactions, trust plays a role (Falcone, Singh and Tan 2001) as does security to some extent. Security is necessary to obtain guarantees such as the identity of an agent, identity of the sender of a message, immutability of messages, etc. Trust is more complex, as it involves insecurity, e.g. about the validity of information in a message, or the future behaviour of an agent. Different kinds of trust can be distinguished, among which trust in an agent’s environment, trust in potential partners, trust in information sources and trust in authorities.

Each agent needs to ascertain the trustworthiness of other agents, thereby encountering the “trust dilemma”: the tradeoff between positive results of a successful trust versus the risks of an unsuccessful trust. Another important characteristic of trustworthiness is that it changes over time, warranting an update of beliefs of an agent (Barber and Kim 2001) about other agents and information. In small environments direct interaction-derived reputation based trust mechanisms may suffice (Birk 2000; Witkowski et al. 2001). In open, dynamic environments in which incentives may differ it is questionable whether agents will always be truthful with respect to the information it provides about other agents (Jurca and Faltings 2002; Abdul-Rahman and Hailes 2000). Even given these differences it has also been acknowledged that trust (and distrust) are most often reciprocal by nature, which complicates reasoning about and predicting trustworthiness (Lawson 1997; Falcone and Castelfranchi 2001).

Trust models such as Bell-La-Padula (Bell and La Padula 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 agents. The Bell-La-Padula 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. Another aspect which is of importance is that of situation: the degree of trust in an agent often depends on the characteristics of a specific situation. In some situations one agent may trust another agent, in other situations it may not.

Trust in open multi-agent systems requires solutions without central authorities nor models in which all trustees are known in advance. An approach based on certificates for distributed trust in open multi-agent systems is 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.

2.2 TRUST IN DESIGN

Although not often recognised explicitly (e.g. (Schön 1983; Coates et al. 2000; Chaoet al. 2002; Valkenburg and Dorst 1998; Wang et al. 2002)), trust plays an important role in distributed, collaborative design. Reputation (Baya and Leifer 1996; Lang et al. 2002) is one of the elements involved. Past experience (the result of an agent's own experience or other agents' experiences) with specific agents (e.g. their specific abilities and skills to perform specific types of tasks in specific situations, their attitude, their commitment) influence the way in which the agent functions in a design team. There are, however, more factors involved. Castelfranchi and Falcone (2000) distinguish seven types of beliefs related to trust:

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

These beliefs are related to the different aspects of trust involved in many collaborative situations, in particular design. In our model of an individual designer within collaborative design (Brazier et al. 2001), shown in Figure 1, a number of the above mentioned types of beliefs are described with respect to their role in a distributed design setting.

This model for a co-operative design 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, viz. design. 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.

The aforementioned seven beliefs can be modelled as part of the Own Process Control, as these beliefs govern the behaviour (or attitude) of the agent. These beliefs are based on information acquired by Agent Interaction Management or Cooperation Management. The beliefs are included in the components as follows:

- Competence belief: plays a role in agent interaction management (e.g., choosing the right language to encode information for the other agent) and cooperation management (e.g., when to elicit collaboration from which agent).
- Disposition belief: plays a role in cooperation management (e.g., is it useful to ask a certain agent to do something).
- Dependence belief: play a role in cooperation management (which agents to be dependent on), and own process control (am I too dependent of too many other agents).
- Fulfilment belief: plays a role in cooperation management (e.g., if the other agent is no longer communicative, will it still fufill its agreed task)
- Willingness belief: plays a role in cooperation management (e.g., only collaborate with agents who are willing to do actions) and own process control (e.g., how can I appear as willing to other agents).
- Persistence belief: plays a role in agent interaction management (e.g., the other agent still understands specific languages), cooperation management (e.g., the other agent does not suddenly terminate ongoing collaborative work) and own process control (e.g., how to appear persistent to other agents).
- Self-confidence: plays a role in cooperation management (e.g., only collaborate with agents who know what they can do, instead of relying on side-effects of their actions) and own process control (e.g., do I know what I can do, and how to convince other agents).

This, however, does not suffice. The knowledge of how to reason with these beliefs is of significant importance. An example of reasoning about trust is given in the next section.

3. Automated Distributed Design

Automated distributed design requires autonomy of participants. Although all participants may strive for the same end result, conflicts of interest may arise; malicious intent cannot be excluded. An example distributed design application in which trust plays an important role is described in

Section 3.1. Implications for the role of trust are discussed in Section 3.2. The role of trust from the perspective of an individual location is illustrated in Section 3.3.

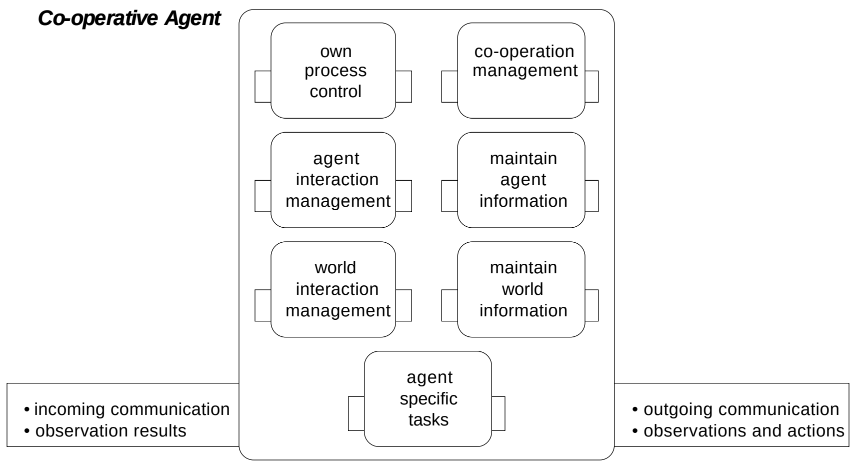

Figure 1. Process composition for a Co-operative Design Agent.

3.1 DESIGN APPLICATION

The example distributed design application described in this section is that of the design (and implementation) of local configurations of a (large-scale) distributed agent operating system. Each location has its own local configuration. The design and implementation of local configurations are the responsibility of a location's management system. An essential element in this design problem is its dynamic and open nature: the environment changes and the artefact (the local configuration) adapts.

Location's management systems need to be able to support migrating agents, communication, and interaction; agent activities which may possibly by harmful, insecure, or illegal. Agents are autonomous and exhibit highly dynamic behaviour: they reside at one location, migrate to other locations, communicate with other agents, and interact with objects and services. Management systems need to decide which agents, from which locations, may migrate, communicate, or interact with objects and services it hosts. It needs to decide which resources will be made available to which agent, for which period of time, and under which conditions. To this end, a management system needs facilities to access and maintain knowledge of the trustworthiness of locations, agents, objects and services, but also to adapt this knowledge on the basis of new information.

AgentScape (Wijngaards et al. 2002) is a world-wide scalable distributed agent platform. Management of AgentScape sites (i.e. locations within AgentScape) can be viewed to be a distributed design problem. Each location management system (human or automated) designs a dynamic artefact (the location's configuration) on the basis of frequently changing requirements (e.g. changing numbers of agents, objects and services, each with different characteristics and requirements). A location management system receives information from human administrators, other agents, and other location management systems (both trusted and untrusted). The collective interest of the location management systems is to facilitate agent migration, agent communication and agent interaction with objects and services to provide continuity at an operational level. A migrating agent has migration requirements and preferences (concerning resource usage, permissions, security levels, trust in specific agents, users and locations) which are, in fact, qualified requirements; destination locations most often need to be (re)configured.

The qualifications of these requirements can be used to (partially) order requirements with respect to their importance. They can play an important role in resolving conflicts among requirements: hard requirements originating from an administrator are more important than a hard requirements from an unknown agent, wishing to arrive at this location (Brazier et al. 1995).

Figure 2 depicts an example of a location configuration. A location consists of different types of resources, including a number of (heterogeneous) hosts, network bandwidth (not shown in figure), and other resources such as operating systems, processors, disks, memory, etc. Agents are always hosted by an agent server. Objects are hosted by object servers. A host can have any number of object servers and/or agent servers. Agent servers have specific properties, e.g. describing their code-base support (e.g., Java agents, Intelx86 binaries, …), access to resources (e.g., disks, libraries, …), and other characteristics.

A location management system places agents on suitable hosts. Trusted agents need to be placed on agent servers which best support their needs. Agents of whom the trust level is unknown or untrusted may be denied access to the location or be placed in a special "playing field": an area where they can function and are given restricted access to resources (sandboxing).

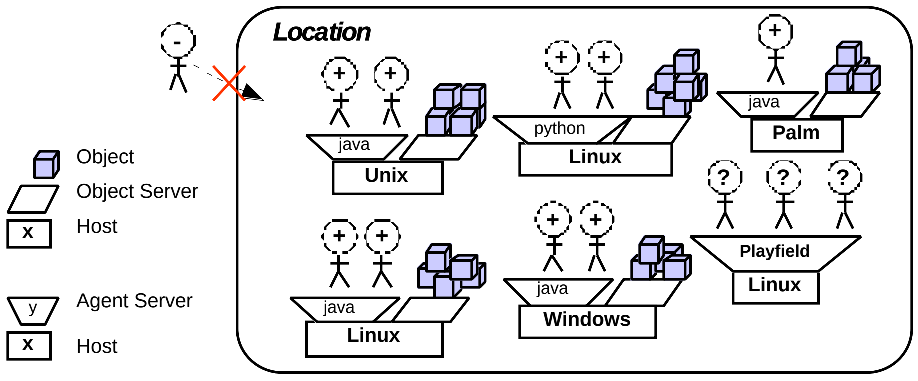

Figure 2. Example of a location configuration, depicting trust of agents: + is trusted, - is untrusted and ? is unknown.

Load balancing, in terms of processor usage, but also in communication overhead, migratory movements, objects access, and agent creation and deletion needs to be managed as well. Agents may dynamically change their resource preferences, warranting a (re)configuration. E.g., an agent currently residing on a Palm device, wishing to use streaming communication, may best be placed on a different host which has better available bandwidth.

3.2 ROLE OF TRUST

Trust plays an important role in distributed agent operating systems. Agents need to know which other agents they can trust (to which degree) and which locations. One pragmatic solution for levels of trust is to define three levels: trusted, untrusted and unknown. In practice unknown often translates to untrusted, but can be more easily "upgraded" on the basis of new information. Location management systems need to know which other locations can be trusted but also need to know which agents they can trust. They may also need to know agents' histories with respect to the locations on which they have been hosted. In all cases the source and authenticity of the information and commands needs to be verified, as well as the reliability.

A location management system acquires some of its information from a human administrator. It should, however, also be capable of automatically adjusting such information on the basis of its own (and others') experiences as the behaviour of agents and locations may change over time, influencing the level of trust assigned. This, however, requires additional knowledge on how to detect “bad” behaviour.

The choice for three-valued trust (trusted, untrusted, unknown) greatly simplifies reasoning about trust. Trust-valuations in continuous domains (e.g., from <-1,1>) may seem to give more control over the 'level of trust', but also make explication of trust relations more difficult.

The current solution for incorporating trust in (automated) distributed design is *explication*; an example of which is presented in section 3.3 below for the role of trust in agent migration from the perspective of an individual location.

3.3. AGENT MIGRATION

Migration of agents requires collaboration between locations: an agent migrates from one location to another. This implies, that both the source and destination location engage in an interaction which may result in a change to their (local) configuration.

Each location is represented by a location manager (an agent), who is not only responsible for the configuration of the location, but also ascertains trust with respect to locations and agents and may deliberate about the 'reputation' it projects to other locations.

The technicalities of agent migration are fairly straightforward. An agent A, that resides at location 1, decides to which location it wants to migrate. As only passive agents can be migrated, agent A needs to make itself passive, e.g. after saving important data. Location 1 engages in interaction with location 2, which results in the transfer of the passive agent A (and personal data of agent A) to location 2. Location 2 is responsible for the activation of agent A.

The migration scenario described above is used in this section to illustrate the role of trust. The three players in the scenario are: agent A, location manager L1, and location manager L2. Each of the three players is discussed below.

The role of trust for agent A
Agent A is assumed to be able to choose a destination location. How agent A makes this decision, and whether it uses models of trust (e.g. based on hearsay) about the trustworthiness of location 2, is not modelled in this paper.

The role of trust for location manager L1
Location manager L1 may have trust about agent A and location 2. However, location manager L1 cannot easily interfere with the autonomy of agent A, which implies that if agent A wishes to migrate to a location 2, which in the opinion of location manager L1 is untrustworthy, location manager L1 cannot prevent agent A from

leaving.

Of course, exceptions can be formulated to this rule; e.g. if the migration service at location L1 was recommending locations to agent A, it could recommend only trusted locations. Alternatively, if location manager L1 is charged with protecting agent A from wandering into untrusted areas, it may also prevent agent A from migrating to location L2.

Location manager L1 can, however, be concerned with its reputation with respect to location manager L2: if it sends too many untrustworthy agents to L2, L2 may no longer accept agents in the future, having decided that L1 is non trustworthy. L1 may wish to avoid this, thereby carefully balancing the needs of agent A and the trustworthiness of agent A, and reasoning from the perspective of location manager L2.

The role of trust for location manager L2

Location manager L2 needs to reason about its trust in agent A and location manger L1, as it is the receiving side. To decide whether agent A is allowed to enter location 2, information is needed on location 1, which may already be known by location manager 2. In addition, location manager 2 requires information on agent A; information that is to be provided by location manager 1:

- certificates specifying the credibility of the owner of agent A,
- certificates specifying the credibility of the code creator of agent A,
- certificates specifying the permissions and capabilities of agent A,
- audit trail of agent A (which contains information on where agent A has been).

For easy of explanation, location manager L2 is assumed to first reason about location 1, then reason about agent A and finally decide on changes to its configuration.

Location manager L2 has a view on location 1 with respect to trusting location manager L1 to not send an agent that has been tampered with. This requires knowledge about location manager L1, how it acted in the past, etc. For example, location manager L2 could employ the following simple heuristics:

- if negative trust in location manager L1, then refuse agent A's migration to me.
- if unknown trust in location manager L1, then actively acquire more information on location manager L1, e.g. by asking for reputation information from other trusted location mangers.
- if positive trust in location manager L1, then acceptance of agent A's migration to L2 depends on L2’s trust in agent A.

Location manger L2 has to decide whether agent A is trustworthy.

This decision may depend on where agent A has been in the past, and the trust in agent A's owner and creator. An agent may have an audit trail, in which changes to the agent's passive state are recorded. This allows for inspection of, on the one hand, unlawful changes to an agent's passive state (e.g., inserting new programming code), and, on the other hand, to provide a travel itinary. The travel itinary may be very useful in determining whether an agent has resided on 'malicious' or untrusted locations. This may provide reasons to refuse an agent's migration. Audit trails may not always be made available to all location managers, as there is a potential conflict with an agent's privacy.

An agent's owner and code creator are important parties as these vouch for the agent: the owner is responsible for an agent’s goals, and the code creator is responsible for the agent’s code.

Location manager L2 needs to reason about trust in both agent A's owner and the agent A's code creator. The table below depicts sample knowledge.

TABLE 1. Knowledge example for deciding on agent's acceptance.

owner ← ***code creator:***	**negative**	**unknown**	**positive**
negative	refuse	refuse	refuse
unknown	refuse	playground	normal
positive	refuse	playground	normal

The rationale for the knowledge in Table 1 is fairly simple: if agent A's owner or code creator are untrusted, then agent A is not accepted by location manager L2. If the code creator is unknown, then agent A is only allowed in, if agent A can run in a 'safe' playground at location L2. Finally, if the code creator is trusted, then agent A can run in a 'normal' environment at location L2.

If agent A is accepted to migrate to location 2, location manager L2 needs to also decide on changes to its configuration, for example, the number and amount of resources allocated to agent A. Location manager L2 may then, for example, provide a time slot for agent A in the (near) future, or refuse agent A.

Agent A has a number of preferences concerning the resources it needs at location L2. Agent A's preferences, permissions and capabilities need to be taken into account when determining changes to the configuration of location 2. In short, these preferences, permissions and capabilities are translated into qualified requirements. Unknown trust in the owner of agent A can, for example, be translated into the qualification 'soft' for the requirements from agent A; while a trusted owner of agent A may yield the qualification 'hard'.

Agent A's impending arrival at location L2 does not only generate a number of qualified requirements, but also other information which is importance in determining the trustworthiness of the parties involved, and for the design task involved (configuration of the location). The agent's certificates are, for example, only trusted to the extent that the certificate authority is trusted (this has not been taken into account in the knowledge described above).

The (re-)configuration process may fail (i.e., no resources available to meet the hard requirements), which results in agent A *not* migrating to location 2. However, the (re-)configuration process may also yield an unacceptable solution for agent A: the migration service at location 1 may then abort the migration process.

Explicitly modelling trust is useful to clarify a number of the issues that play a role in our domain. However, how to acquire and adapt trust, is still an open issue.

4. Discussion

Human designers often rank information received from other designers, based on their experiences with these individuals, and their place in the design project organisation. Agents involved in automated distributed design need not only to be able to deal with trust in the same way, they also need to know how to acquire and adapt trust relationships. As yet it is infeasible to automatically assign trust levels to agents (see Falcone, Singh and Tan, 2001). These relations will, however, need to be understood if automated support is to play a significant role in distributed design.

In this paper the role of trust has been analysed for a specific domain of distributed design: local automated (re-)configuration of a (large-scale) distributed agent operating system. A configuration of a location consists of agents, objects and services with resources (such as agent-servers, disk-space, bandwidth, etc.). Each location is configured by a location manager agent.

This agent continually redesigns the configuration of its location on the basis of (changing and imprecise) qualified requirements and information from other location managers, new agents, objects and service providers. The design task is a configuration task in a dynamic open environment in which the specifications (although constrained) are unknown in advance. The role of trust in agent migration has been used to illustrate how trust influences the behaviour of one single (automated) designer in a distributed design task.

The domain chosen has characteristics which do not match assumptions on trust-models used in literature: open-endedness, unknown participants, and automated (re-)configuration of locations. Although explicitation of trust facilitates understanding the role trust plays in this distributed design setting, a number of issues need to be resolved before trust can be used in automated design. Two of the current research issues are trust acquisition and trust adaptation. Reputation-based schemes may be useful, incentive-based schemes may not be applicable in an open-environments in which not all incentives are comparable.

Acknowledgements

This work was supported by NLnet Foundation, http://www.nlnet.nl/. The authors acknowledge the contributions made by David Mobach, Guido van 't Noordende, Benno Overeinder, Maarten van Steen and Andy Tanenbaum.

References

Abdul-Rahman, A and Hailes, S: 2000, Supporting trust in virtual communities, *Proceedings 33rd Hawaii International Conference on System Sciences(HICSS00)*, IEEE Press, 10 pp.

Barber, KS and Kim, J: 2001, Belief revision process based on trust: Agents evaluating reputation of information sources, *in* R Falcone, MP Singh and Y-H Tan (eds), *Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, workshop on Deception, Fraud, and Trust in Agent Societies, during the Autonomous Agents Conference 2000, Springer, Lecture Notes in Computer Science, **2246**, pp. 73-82.

Baya, V and Leifer, LJ: 1996, Understanding information management in conceptual design, *in* N Cross, H Christiaans and K Drost (eds), *Analysing Design Activity*, John Wiley & Sons, Chichester, pp. 151-168.

Bell, DE and La Padula, LJ: 1973, *Secure Computer Systems: Mathematical Foundations and Model*, The MITRE Corporation, report MTR 2547 v2.

Birk, A: 2000, A Boosting cooperation by evolving trust, *Applied AI* **14**: 769-784

Brazier, FMT, Langen PHG van and Treur J: 1995, Modelling conflict management in design: an explicit approach, *in* IFC Smith (ed.), *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, (AIEDAM)* **9**(4): 353-366.

Brazier, FMT, Moshkina, LVand Wijngaards, NJE: 2001, Knowledge level model of an individual designer agent in collaborative distributed design, *Journal of Artificial Intelligence in Engineering* **15**: 137-152.

Castelfranchi, C and Falcone, R: 2000, Trust is much more than subjective probability: Mental components and sources of trust, *Proceedings 33rd Hawaii International Conference on System Sciences (HICSS00)*, IEEE Press, 10 pp.

Chao, K-., Norman, P, Anana, R and James, A: 2002, An agent-based approach to engineering design, *Computers in Industry* **48**: 17-27.

Coates, G, Duffy, AHB, Hills, W and Whitfield, RI: 2000, A generic coordination approach applied to a manufacturing environment, *Journal of Materials Processing Technology* **107**: 404-411.

Falcone, R and Castelfranchi, C: 2001, The socio-cognitive dynamics of trust: Does trust create trust? *in* R Falcone, MPSingh and Y-H Tan (eds), *Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, Workshop on Deception, Fraud, and Trust in Agent Societies, during the Autonomous Agents Conference 2000, Springer, Lecture Notes in Computer Science, **2246**, pp. 55-72.

Falcone, R, Singh, MP and Tan, Y-H: 2001, Introduction: Bringing together humans and artificial agents in cyber-societies: A new field of trust research, *in* R Falcone, MP Singh and Y-H Tan (eds), *Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, Workshop on Deception, Fraud, and Trust in Agent Societies, during the Autonomous Agents Conference 2000, Springer, Lecture Notes in Computer Science, **2246**, pp. 1-8.

Jurca, R. and Faltings, B. 2002, Towards Incentive-Compatible Reputation Management, *in* R. Falcone, S. Barber, L. Korba and M Singh (eds), *proceedings of the Workshop on Deception, Fraud and Trust in Agent Societies*, ACM Press, 92-100.

Lang, SYT, Dickinson, J and Buchal, RO: 2002, Cognitive factors in distributed design, *Computers in Industry* **48**: 89-98.

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

Mass, Y and Shehory, O: 2001, Distributed trust in open multi-agent systems, *in* R Falcone, MP Singh and Y-H Tan (eds), *Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, Workshop on Deception, Fraud, and Trust in Agent Societies, during the Autonomous Agents Conference 2000, Springer, Lecture Notes in Computer Science, **2246**, pp. 159-173.

Schön, DA: 1983, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, Perseus Books Group.

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

Wang, L, Shen, W, Xie, H, Neelamkavil, J and Pardasani, A: 2002, Collaborative conceptual design - state of the art and future trends, *Computer Aided Design* **34**: 981-996.

Wijngaards, NJE, Overeinder, BJ, Steen, M van and Brazier, FMT: 2002, Supporting internet-scale multi-agent systems, *Data and Knowledge Engineering* **41**(2-3): 229-245.

Witkowski, M, Artikis, A and Pitt, J:2001, Experiments in building experiential trust in a society of objective-trust based agents, *in* R Falcone, MP Singh and Y-H Tan (eds), *Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, Workshop on Deception, Fraud, and Trust in Agent Societies, during the Autonomous Agents Conference 2000, Springer, Lecture Notes in Computer Science, **2246**, pp. 111-132.

Wong, HC and Sycara, KP: 2000, Adding security and trust to multiagent systems, *Applied Artificial Intelligence* **14**(9): 927-941.