Available online at www.sciencedirect.com
SCIENCE DIRECT®
Expert Systems with Applications 31 (2006) 849–858
Expert Systems with Applications
www.elsevier.com/locate/eswa

Enhancing peer-to-peer collaboration using trust

Nathan Griffiths

Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

Abstract

Distributed systems generally require their component parts to interact cooperatively, in order for the system as a whole to function effectively. For any given activity, there are typically several alternative components that have the required capabilities. In decentralised systems, where there is no overarching control, individual components are responsible for selecting other components with which to cooperate. However, the candidate components may be unreliable or dishonest, and are typically locally controlled. Such decentralised systems can be viewed as multi-agent systems, comprising autonomous agents that must cooperate for the system to be effective. Peer-to-peer (P2P) systems are a subclass of decentralised distributed systems, in which not only is there no overarching control, but neither is there any hierarchy of control, power, or responsibly among the system components. Selecting appropriate peers to cooperate with is a challenging problem, since the candidate peers are autonomous and may be unreliable or dishonest. Peers need a mechanism for task delegation that takes the uncertainly of interactions into account. In this paper we present a mechanism, called MDT-R, that enables peers to delegate activities appropriately, using trust and the recommendations of other peers to meet individual preferences, such as minimising risk and maximising quality.

Keywords: Agents; Trust; Reputation; Cooperation; Peer-to-peer

1. Introduction

Complex distributed systems such as cooperative design processes, supply chains, and grid computing can be viewed as multi-agent systems. Each component is considered to be an autonomous agent, and for the system to function effectively these agents must cooperate. The individual computational resources, knowledge bases, and human actors in a system are represented by agents with appropriate goals and preferences (Jennings, 1994; Luck, McBurney, & Preist, 2003). Agents have individual capabilities, knowledge and resources that are made available to others (generally) in return for imposing some cost. Agents are under local autonomous control, making individual decisions to determine their own actions. Furthermore, they have their own individual preferences regarding the nature of cooperation for an activity, in particular with respect to balancing the cost, quality, timeliness etc. of an interaction. Using their knowledge of others' capabilities agents can, according to their preferences, delegate the activities that must be performed in order to achieve their goals.

Agents have varying degrees of reliability, quality and honesty, and activities may fail, produce substandard results, or may cost more and take longer than expected. On delegating an activity, agents enter into an uncertain interaction since a high quality, timely, and on budget outcome is not guaranteed. Furthermore, since agents are autonomous peers they have no control over how others cooperate (d'Inverno & Luck, 1996). Thus, agents determine for themselves when to delegate activities or provide assistance, when to cease cooperating, and how to conduct themselves; they can change the nature of cooperation, or even cease to cooperate, at any time. For example, an agent may choose to delay execution, reduce the quality of execution,

E-mail address: nathan@dcs.warwick.ac.uk
URL: http://go.warwick.ac.uk/nathangriffiths

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.01.017

or simply fail to complete an activity. To function effectively, agents must manage the risk of activity failure (and the risk of reduced performance). Trust provides a mechanism for assessing and managing this risk of failure; trust enables an agent to assess how likely another is to fulfil its commitments. Agents are able to build models of others' trustworthiness, and incorporate the resulting information into their reasoning process, in particular at the point of choosing a cooperative partner.

In this paper we present a mechanism, called *multi-dimensional trust with recommendations* (MDT-R), in which agents use trust and the recommendations of their peers to manage the risks of cooperating. Agents build models of their peers' trustworthiness along several dimensions, based on their experiences. When delegating an activity, a cooperative partner can be selected by combining these trust dimensions with peer recommendations and other decision factors (such as cost). These decision factors are combined in a flexible manner, allowing agents to select cooperative partners according to their current preferences regarding the relative importance of cost, quality, timeliness, and risk etc. The MDT-R mechanism was initially described in Griffiths and Sun (2005). In this paper we expand our discussion of MDT-R, refine the mechanism itself, and introduce our experimental results.

2. Peer-to-peer agents and cooperative design

In this paper, we describe the proposed MDT-R mechanism within the domain of computer supported cooperative work (CSCW) and cooperative design in particular. The last decade has seen a significant increase in the application of agents to CSCW. Initially, developers were attracted to agents due to the intelligence, automation, and communication skills that they exhibit (Guilfoyle, 1998). This early application of agents to CSCW was largely based on the view of agents as an extension to objects, where agents provide the communication, cooperation and intelligence needed to support the human actors in a system. More recently, however, agents have been incorporated into CSCW systems as actors in their own right; autonomous intelligent entities able to perform tasks on behalf of human actors, or other agents (Luck, McBurney, & Preist, 2004; Yiming, 2003). In a cooperative design environment, for example, agents might act as personal assistants performing tasks on behalf of users (Zhang, Ghenniwa, & Shen, 2005), or act as independent entities able to contribute to the overall design task (Chen, Wu, Han, & Xiao, 2005).

A peer-to-peer (P2P) system is one in which two or more peers are able to collaborate in a network of equal peers by using appropriate information and communication systems, without the necessity for central coordination (Schoder & Fischbach, 2003). In many applications of CSCW, including cooperative design, the actors in the system are equal peers, having no explicit hierarchy of control or power. Such CSCW environments can be viewed as P2P systems in which the peers are autonomous agents that can contribute to the task, along with the agents that represent human users. MDT-R is proposed as a general mechanism for agent-based systems where there is no central control. In this paper we describe MDT-R and illustrate its application to a P2P system within the context of cooperative design.

2.1. The cooperative design domain

The workflow of the cooperative design process can be viewed as a directed graph, in which the nodes correspond to activities and the edges show dependencies between these activities. An edge from activity A_i to A_j indicates that A_i is instrumental to A_j being performed. For example, A_j may represent the validation of a design and A_i may correspond to generating and processing simulation results that are required for validation. Furthermore, nodes may represent complex activities, and can be decomposed into corresponding primitive activities. For example, the activity of generating and processing simulation results might be decomposed into configuring and running several iterations of a simulation, followed by an analysis of the results using a particular set of techniques. Fig. 1 illustrates an example fragment of workflow (shown at the bottom of the figure), where the subgraph contained in the circle represents the decomposition of the indicated node into primitive activities. The workflow fragment at the bottom of the figure represents the steps in the design process. In this example, finalising the design (the end of the fragment) is dependent on the design being validated, which in turn is dependent on processing simulation results. The simulation activity is a complex task that can be decomposed into a set of primitive activities $A_1 \cdots A_{19}$, as illustrated by the steps within the circle. Each of the other steps in the workflow

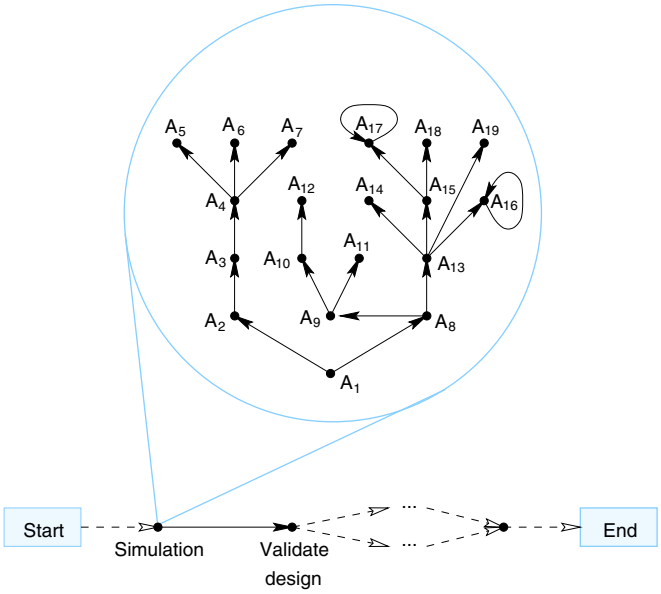

Fig. 1. An example workflow fragment.

fragment, including validating the design and the preceding and proceeding steps, might also be decomposed into a set of primitive tasks.

Individual activities might be specific design tasks or computational tasks, such as running a single iteration of a simulation or processing a set of results using a specific technique. For our purposes, however, we abstract out the low-level details of each task. An activity has certain requirements in terms of the capabilities and resources (e.g. knowledge or network bandwidth) needed for completion. For an agent to successfully perform an activity, these requirements must be met.

For each activity, a corresponding set of agents will meet the capability and resource requirements. The agent responsible for an activity must determine which of the capable agents to delegate the required instrumental activities to. Thus, if an edge exists from A_i to A_j then the agent responsible for A_j should delegate A_i. In the above example, this means that the agent responsible for validating the design must delegate the processing of the simulation. (The task of validating the design will in turn have been delegated by the agent responsible for the task for which it is instrumental.) Where an agent represents an individual human or a design team a human can typically override the agent's choice of delegate if required. The overarching (final) activity is allocated to an agent by the person (or agent) that instigates the design process (or the overall task in the case of tasks decomposed into primitive activities). It is important to note that the edges in the workflow graph only represent dependencies, and *not* control. Agents are *equal peers* and delegation does not imply control over how the delegate performs an activity.

In this paper we focus on segments of workflow where the agents are true independent entities contributing to the design task. As an example, suppose that the decomposition of running a simulation shown in Fig. 1 is such that tasks $A_1 \cdots A_{19}$ are to be executed by autonomous agents. Thus, each of the agents involved is an independent entity contributing to the overall design task. In this context the delegation process involves one autonomous agent selecting another, in such a way as to maximise the likelihood of success and minimise cost etc. Our proposed model is equally applicable to agents that represent humans or design teams, but since these humans are typically able to over-rule the agent's choice of delegate there is an extra element to task delegation that is beyond the scope of this paper.

There is no central or hierarchical control. Agents are equal peers with no control over others, other than the ability to delegate activities. Thus, the cooperative design process is a P2P system with the agents themselves managing the allocation of activities. If substandard, erroneous or late results occur at one stage, these problems can propagate to subsequent activities, leading to errors, lateness, or complete failure. Furthermore, if failure occurs in an activity we assume that the agent responsible for the activity for which it is required re-delegates execution to an alternative agent. Thus, all activities must be executed for the process as a whole to be successful. So, for example, if the agent processing the simulation results fails, then the agent responsible for validating the design will reallocate the simulation task to an alternative peer.

3. Trust

Trust is a well recognised mechanism for assessing the potential risk associated with cooperating with autonomous agents (Castelfranchi & Falcone, 1998; Gambetta, 1988; Marsh, 1994a). Specifically, trust represents an agent's estimate of how likely another is to fulfil its commitments. Previous work on trust can be roughly divided into two categories according to how trust is used. Firstly, trust can be used to enhance security, and secondly it can be used to enhance quality of service (note that these roles are not mutually exclusive). In the context of CSCW and cooperative design we are not concerned with the security aspects of trust, since we assume that all peers are permitted to be members of the system and have been configured with appropriate access privileges. Instead we use trust in a quality of service role, to enable peers to maximise the "quality" of their interactions according to their current preferences regarding cost, timeliness, etc. Previous work focusing on trust in P2P systems, tends to concentrate on addressing the security aspects (Bursell, 2005; Waldman, Cranor, & Rubin, 2001). Our proposed MDT-R mechanism builds on existing work on service-oriented trust in agent-based systems, as discussed later in this section. The proposed mechanism allows trust to be used to address the quality of service issues associated with P2P systems.

Trust itself can be divided into two categories: *experience-based* and *recommendation-based*. In the former, trust is based on individual experience, while in the latter it is based on information provided by others. Experience-based trust is simplest, where agents delegate activities and update their individual models of others' trustworthiness according to the outcome. Recommendation-based trust is more complex, requiring agents to share information (based on their experiences) about the perceived trustworthiness of another. Our approach combines these two categories: agents maintain individual experience-based trust assessments of others, and combine these with information provided by other peers.

An obstacle to using recommendation-based trust is the subjectivity of trust. Agents build trust models based on individual experiences, and such subjective information may not be directly useful when shared with another agent. Thus, an agreed trust semantics is needed to enable recommendation-based trust. Agents must be able to interpret the information provided others. Our solution to this problem is described in Section 5.

3.1. Multi-dimensional trust

Activities are typically more complex than simple succeed or fail interactions. Agents cooperate with an expectation

of successful performance, to a given quality and for some anticipated cost. In addition to possible failure, activities may succeed but be of lower than expected quality or at a higher than expected cost. Building on our previous work, we take a multi-dimensional view of trust, decomposing it into beliefs about the different dimensions of an interaction, such as quality and timeliness (Griffiths, 2005b). Agents can model such characteristics as different *trust dimensions*. MDTR is not restricted to specific prescribed trust dimensions, and agents can model trust along any number of dimensions. However, for the purposes of this paper the trust of an agent α is modelled in the following four dimensions:

- success (denoted T_α^{s}): the likelihood that α will successfully execute the task,
- cost (denoted T_α^{c}): the likelihood that the cost of α executing the task will be no more than expected,
- timeliness (denoted T_α^{t}): the likelihood that α will complete the task no later than expected, and
- quality (denoted T_α^{q}): the likelihood that the quality of results provided by α will meet expectations.

Trust in any given dimension encompasses beliefs about competence, disposition, dependence, and fulfilment (Castelfranchi, 2004). For example, if an agent is trusted in the quality dimension, then it is believed to be capable of performing an activity to a high quality (competence), actually doing so (disposition), being the preferred agent to do it (dependence), and being the means for activity achievement (fulfilment).

3.2. Representing trust

Our representation of trust is based on the theoretical work of Gambetta (1988), the formalism proposed by Marsh (1994a), and our previous work (Griffiths, 2005a; Griffiths, Luck, & d'Inverno, 2003). The trust in agent α along dimension d is defined to be a real number in the interval between 0 and 1: $T_\alpha^{\mathrm{d}} \in [0,1]$. Trust values are merely comparative, and have no strong semantic meaning in themselves. Values approaching 0 represent complete distrust, and those approaching 1 represent complete trust. Trust is inversely related to the perceived risk of interacting: cooperating with a trusted agent has a low perceived risk of failure in the trusted dimensions, while a high risk is associated with distrusted agents. Trust values represent an individual's subjective view, based on experience, and so are not directly comparable across agents. A measure of confidence is associated with each value according to the breadth of experience on which it is based; as agents gain experience this confidence increases.

Trust initially takes a value according to an agent's disposition: optimistic or pessimistic. Optimists ascribe high initial values (implying low perceived risk), and pessimists ascribe low values (implying high perceived risk). An agent's disposition also determines how trust is updated (Marsh, 1994b). After interacting, optimists increase their trust more than pessimists in the dimensions where expectations were met and, conversely, pessimists decrease trust to a greater extent when expectations are not met. An agent's disposition includes: the initial trust T_{initial} ascribed in a trust dimension prior to interacting, and functions for updating trust after successful and unsuccessful interactions, $\mathrm{update}_{\mathrm{success}}$ and $\mathrm{update}_{\mathrm{fail}}$ respectively. These functions are heuristics applying to all trust dimensions, and they have no standard definition. Instead, it is the responsibility of the system designer to choose an appropriate heuristic. In this paper we use the following definitions to update the trust in agent α along dimension d:

$$\mathrm{update}_{\mathrm{success}}(T_\alpha^{\mathrm{d}}) = T_\alpha^{\mathrm{d}} + ((1 - T_\alpha^{\mathrm{d}}) \times (\omega_{\mathrm{s}} \times T_\alpha^{\mathrm{d}})) \quad (1)$$

$$\mathrm{update}_{\mathrm{fail}}(T_\alpha^{\mathrm{d}}) = T_\alpha^{\mathrm{d}} - ((1 - T_\alpha^{\mathrm{d}}) \times (\omega_{\mathrm{f}} \times T_\alpha^{\mathrm{d}})) \quad (2)$$

where ω_{s} and ω_{f} are weighting factors in the interval $[0:1]$ defined by the agent's disposition. The weighting factors determine how optimistic or pessimistic agents are in updating trust after an interaction. An optimistic agent α will typically use weights $\omega_{\mathrm{s}_\alpha}$ and $\omega_{\mathrm{f}_\alpha}$ such that $\omega_{\mathrm{s}_\alpha} > \omega_{\mathrm{f}_\alpha}$. Conversely, a pessimistic agent β will typically use weights $\omega_{\mathrm{s}_\beta}$ and $\omega_{\mathrm{f}_\beta}$ such that $\omega_{\mathrm{s}_\beta} < \omega_{\mathrm{f}_\beta}$. Furthermore, where α is an optimist and β is a pessimist it is typically the case that $\omega_{\mathrm{s}_\alpha} > \omega_{\mathrm{s}_\beta}$ and $\omega_{\mathrm{f}_\alpha} < \omega_{\mathrm{f}_\beta}$.

In addition to the trust value itself, the trust for an agent α in a dimension d has an associated numerical confidence level, C_α^{d}, that is incremented on each application of the trust update function. This allows agents to factor into their decision making the extent of their previous experience with a particular agent in a given dimension. (The extent of experiences varies in different dimensions, e.g. quality is only relevant for successful interactions.) An agent can be more certain of trust values that are based on extensive experience in comparison to those that are based on a small number of interactions.

3.3. Trust decay

Over time trust values may become outdated, if the experiences that gave rise to them are no longer relevant. To address this we apply a decay function to converge trust values to T_{initial} in the lack of subsequent experience. Thus, unless reinforced by recent interactions, the positive effect of expectations being met reduces over time, as does the negative effect of failed expectations. The decay function for the trust in agent α along dimension d is defined as:

$$\mathrm{decay}_{\mathrm{trust}}(T_\alpha^{\mathrm{d}}) = T_\alpha^{\mathrm{d}} - ((T_\alpha^{\mathrm{d}} - T_{\mathrm{initial}}) \times \omega_{\mathrm{td}}) \quad (3)$$

where the trust decay rate $\omega_{\mathrm{td}} \in [0:1]$ is defined by the agent's disposition. Low decay rates mean that an agent's experiences remain relevant for longer (this is analogous to having a large memory window), while a high decay rate leads to experiences rapidly becoming irrelevant (analogous to having a small memory window). As trust values become outdated, the confidence in them also reduces.

Thus, when applying $\text{decay}_{\text{trust}}$ the corresponding confidence level must also be reduced. We define a similar decay function for the confidence level as

$$\text{decay}_{\text{confidence}}(C^{\text{d}}_{\alpha}) = C^{\text{d}}_{\alpha} - (C^{\text{d}}_{\alpha} \times \omega_{\text{cd}}) \quad (4)$$

where the confidence decay rate $\omega_{\text{cd}} \in [0:1]$ is also defined by the disposition.

4. Stratified trust

We represent trust values numerically, however some researchers note that this can introduce ambiguity since the semantics are hard to represent (Abdul-Rahman & Hailes, 2000; Marsh, 1994a). This is problematic when using recommendation-based trust, where agents share trust information. One solution is to divide the trust continuum into labelled strata, and use these to represent trust values. For example, Abdul-Rahman and Hailes (2000) represent trust values using the strata of: "very trustworthy", "trustworthy", "untrustworthy" and "very untrustworthy". However, the resulting semantics remain subjective, with different agents potentially ascribing the same experiences to different strata. Furthermore, this representation reduces sensitivity and accuracy, and comparisons become coarse grained since the trust of agents within a stratum is indistinguishable. For this reason, we concur with Marsh (1994a) in rejecting the use of strata in favour of numerical values. Additionally, updating numerical trust is straightforward, yet stratified approaches often omit details of how strata are related to experience (Abdul-Rahman & Hailes, 2000; Azzedin & Maheswaran, 2002).

However, the use of strata minimises overfitting; numerical values are not considered and so insignificant numerical differences in trust are not misinterpreted as important. An ideal trust model has the sensitivity and accuracy of a numerical approach, combined with the minimal risk of overfitting of a stratified approach. To this end, we draw on our previous work, and use a variable size stratifying of trust *at the time of trust comparisons* (Griffiths, 2005b). Trust values are translated into strata immediately before comparison. The number of strata is not fixed, although typically an agent will use the same number of strata for each trust dimension and in each comparison. Using fewer strata minimises the risk of overfitting but gives the least precise comparison, while using more strata retains precision but at an increased risk of overfitting.

5. Interaction summaries

To enable sharing of trust information, there must be a clear semantics to the information provided. As discussed above, although stratified trust provides a possible solution, it still suffers from subjectivity. Several alternative approaches to recommendation-based trust have been developed, however each of them suffer from potential problems due to the subjectivity of information provided (Huynh, Jennings, & Shadbolt, 2004; Sabater & Sierra, 2002; Yu & Singh, 2002). In MDT-R we address the subjectivity problem by agents providing a summary of their relevant previous interactions, rather than attempting to provide an explicit assessment of trust. Thus, when agent α shares information with β about γ, rather than communicating its trust value in a given dimension, T^{d}_{γ}, it communicates a summary of the experiences that led to T^{d}_{γ} (but not the value itself). As described in Section 3, trust is determined by whether an agent's expectations are met in a given dimension. For example, trust in the quality dimension T^{q}_{γ} is determined by whether previous interactions were of suitable quality. When sharing information, α can communicate to β the number of previous interactions with γ in which quality expectations were met, $I^{\text{q}+}_{\alpha\gamma}$, and the number in which the quality was below that which was expected, $I^{\text{q}-}_{\alpha\gamma}$. The receiving agent β therefore obtains a summary of α's interactions with γ, and an indication of the extent of α's relevant experience (since α has had $I^{\text{q}+}_{\alpha\gamma} + I^{\text{q}-}_{\alpha\gamma}$ relevant interactions). Similar summary information can be communicated in each of the trust dimensions.

The sharing of interaction summaries is still subjective with respect to the nature of the sharing agent's expectations, and whether or not they were met. However, the major problem of subjectivity associated with communicating an explicit numerical trust value is avoided. A numerical trust value is entirely subjective, whereas the subjectivity in summary of interactions arises solely from the sharing agent's assessment of whether its expectations were met and is independent from the sharing agent's disposition. We take this to be an acceptable level of subjectively. Moreover, it is largely unavoidable where information is shared based on another's experiences, since that agent's interpretation of its interactions are necessarily subjective.

5.1. Determining recommendations

The first step in delegating an activity, is for the delegating agent to ask its trusted peers to provide information about each potential delegate. General trust, T_{α}, in an agent α combines all trust dimensions and is based on the notion of general trust proposed by Marsh (1994a). It is beyond the scope of this paper to discuss the alternatives for how general trust might be calculated, but for our purposes it can be considered to be the average value of trust across all trust dimensions. We therefore define general trust in an agent α as $T_{\alpha} = \text{average}(T^{\text{s}}_{\alpha}, T^{\text{c}}_{\alpha}, T^{\text{t}}_{\alpha}, T^{\text{q}}_{\alpha})$. Only trusted peers should be asked for recommendations, to minimise the risk of dishonest or misleading information. Thus, the delegating agent should only ask peers whose general trust is above a minimum threshold, i.e. $T_{\alpha} > T_{\text{min}}$, where T_{min} is a minimum trust threshold defined by the agent's disposition.

On receiving a request for information each trusted peer α will provide $I^{\text{d}+}_{\alpha\gamma}$ and $I^{\text{d}-}_{\alpha\gamma}$, for each potential delegate γ in each of the trust dimensions d. The delegating agent can then combine this information into a single recommendation

value for each trust dimension. For each dimension d the set of responses from the trusted peers about a potential delegate γ are combined by summing the proportions of interactions where expectations are met, weighted by the extent of the peer's experience. Thus, the recommendation for a dimension, R^{d}_{γ}, is defined as:

$$R^{\mathrm{d}}_{\gamma} = \sum_{i=\alpha}^{\xi} \left(\frac{I^{\mathrm{d}+}_{i\gamma}}{I^{\mathrm{d}+}_{i\gamma} + I^{\mathrm{d}-}_{i\gamma}} \times \frac{I^{\mathrm{d}+}_{i\gamma} + I^{\mathrm{d}-}_{i\gamma}}{total\_interactions} \right) = \sum_{i=\alpha}^{\xi} \left(\frac{I^{\mathrm{d}+}_{i\gamma}}{total\_interactions} \right) \quad (5)$$

where $\alpha, \beta, \ldots, \xi$ are the set of trusted peers, and *total_interactions* is the total number of interactions across all trusted peers with agent γ in dimension d, defined as

$$total\_interactions = \sum_{i=\alpha}^{\xi} (I^{\mathrm{d}+}_{i\gamma} + I^{\mathrm{d}-}_{i\gamma}) \quad (6)$$

Applying this approach, the delegating agent can determine recommendations in each of the trust dimensions of success, cost, timeliness and quality, denoted R^{s}_{γ}, R^{c}_{γ}, R^{t}_{γ} and R^{q}_{γ} respectively.

6. Delegation: combining trust dimensions

When delegating an activity the various dimensions of trust, the recommendations from trusted peers, and any other relevant decision factors (such as advertised cost and quality) must be considered. An agent's preferences and its confidence in its trust models determine the emphasis given to each of these factors. For example, one agent may prefer to minimise the risk of failure and achieve the highest quality, while another may prefer to minimise cost at the potential expense of quality and an increased risk of failure. Similarly, if an agent has relatively low confidence in its own trust models, i.e. they are based on a limited number of interactions, then it may place more emphasis on peer recommendations.

To select between agents we use a weighted product model to combine choice factors and give a single performance value for each agent (Bridgeman, 1922; Triantaphyllou, 2000). Each of the relevant decision factors, i.e. advertised cost, quality, trust, and peer recommendations, are incorporated into the calculation of the performance value. Each factor is raised to the power equivalent to its relative weight according to the selecting agent's preferences.[1] For each potential partner a performance value is calculated as:

$$\mathrm{PV}(\alpha) = \prod_{i=1}^{n} (f_{\alpha_i})^{\mu_i} \quad (7)$$

[1] Our previous work described the use of a weighted product model for combining trust values (Griffiths, 2005b), MDT-R extends this by including peer recommendations.

where there are n factors and f_{α_i} is the value for agent α in terms of the ith factor and μ_i is the weighting given to the ith factor in the selecting agent's preferences. The values of the weightings μ_i are defined by the selecting agent's preferences such that: $\sum_{i=1}^{n} \mu_i = 1$.

The best delegate is the agent α whose performance value $\mathrm{PV}(\alpha)$ is greater than that of all other agents. Where several agents have equal performance values, one is selected arbitrarily.

Provided that the μ_i's sum to 1 the individual weightings can take any value in the interval [0:1]. This flexibility is a key strength of MDT-R, since the information maintained by an agent is the same, regardless of its current preferences and factor weightings. Furthermore, agents can use different weightings in different situations. For example, if an agent is relatively inexperienced then more emphasis can be given to others' recommendations, and as experience is gained the emphasis can move toward its own trust models.

Factors such as quality can be used directly in calculating the performance value, provided that they are numerical and should be maximised. Similarly, peer recommendations can be used directly since they are numerical and to be maximised. Factors that should be minimised, such as cost, can be included by using:

$$f_{\alpha_{\mathrm{c}}} = \max(\alpha_{\mathrm{c}} \cdots \xi_{\mathrm{c}}) + 1 - \alpha_{\mathrm{c}} \quad (8)$$

where α_{c} represents the advertised cost from agent α, and $\max(\alpha_{\mathrm{c}} \cdots \xi_{\mathrm{c}})$ is the maximum advertised cost of all potential delegates, also denoted as $\max_{\mathrm{c}}$. (The addition of 1 ensures that for a maximal cost alternative, the factor still has a positive value.)

Trust values must be stratified before inclusion, as discussed in Section 4. The trust range is divided into s equal strata such that each is given a value from 1 to s in order. Trust values are stratified by determining the value of the stratum that they occupy. For a trust value t its stratum is obtained by using:

$$\mathrm{stratify}(t) = \lceil t \times s \rceil \quad (9)$$

For example, using 10 strata, a trust value of 0.35 is given a stratum value of $\lceil 0.35 \times 10 \rceil = 4$.

Peer recommendations are similar to trust in that they are numerical values, based on (necessarily) subjective judgements. Thus, as for trust, there is a risk of overfitting if the numerical values are used directly. To minimise the risk of overfitting we take the approach of also stratifying peer recommendation values before they are included in the calculation of a performance value.

Recall that in this paper we are considering the trust dimensions of success, cost, timeliness, and quality along with the corresponding peer recommendations in these dimensions. When delegating an activity each of these dimensions should be considered, along with the advertised cost and quality of each alternative agent. Thus, using Eq. (7), an agent should calculate a performance value for each potential partner as:

$$\begin{aligned}\mathrm{PV}(\alpha) = {} & (\max_c + 1 - \alpha_c)^{\mu_c} \times (\alpha_q)^{\mu_q} \\ & \times \mathrm{stratify}(T^s_\alpha)^{\mu_{ts}} \times \mathrm{stratify}(T^c_\alpha)^{\mu_{tc}} \\ & \times \mathrm{stratify}(T^t_\alpha)^{\mu_{tt}} \times \mathrm{stratify}(T^q_\alpha)^{\mu_{tq}} \\ & \times \mathrm{stratify}(R^s_\gamma)^{\mu_{rs}} \times \mathrm{stratify}(R^c_\gamma)^{\mu_{rc}} \\ & \times \mathrm{stratify}(R^t_\gamma)^{\mu_{rt}} \times \mathrm{stratify}(R^q_\gamma)^{\mu_{rq}} \end{aligned} \quad (10)$$

where α_c and α_q are α's advertised cost and quality respectively; $\max_c$ is the maximum advertised cost of the agents being considered; the weightings given to advertised cost and quality are denoted as μ_c and μ_q; μ_{ts}, μ_{tc}, μ_{tt}, and μ_{tq} are the weightings for the trust dimensions of success, cost, timeliness, and quality respectively; and μ_{rs}, μ_{rc}, μ_{rt}, and μ_{rq} are the corresponding weightings for recommendations. This approach allows an agent to balance the relevant decision factors when selecting a peer for delegation. For example, an agent with limited experience can emphasise peer recommendations by increasing μ_{rs}, μ_{rc}, μ_{rt} and μ_{rq}; or might emphasise the importance of success (relative to quality, cost, and timeliness) by increasing μ_{ts} and μ_{rs}. Agents are able to change their preferences regarding calculating performance values simply by changing the weighting values, with no changes needed to the underlying trust models or the data that is stored.

7. Example performance value calculation

By way of example, consider an agent selecting between two alternative agents, ϕ and ψ, for the subtask of processing simulation results. Suppose that the delegating agent requests recommendations from three trusted peers, α, β and γ. Furthermore, suppose that these peers give the following information about their interactions with ϕ and ψ, with respect to the trust dimension of success (where i takes the values α, β and γ in turn).

	$I^{s+}_{i\phi}$	$I^{s-}_{i\phi}$	$I^{s+}_{i\psi}$	$I^{s+}_{i\psi}$
α	32	17	12	18
β	65	32	79	13
γ	3	7	48	42

Thus in this example, in agent α's experience, agent ϕ has successfully processed simulation results 32 times and has been unsuccessful 17 times. Similarly, agent ψ has been successful on 12 occasions and failed in 18 instances. Using this information, the delegating agent must calculate recommendation values for both candidate peers. Firstly, the total number of interactions with agent ϕ is calculated, as defined in Eq. (6), which gives a value of 156. The recommendation for ϕ in the success trust dimension, can then be calculated using Eq. (5), as follows:

$$R^s_\phi = 32/156 + 65/156 + 3/156 = 0.64$$

Similarly, for alternative ψ we get $R^s_\psi = 0.66$. Thus, based on the recommendations of α, β and γ agent ψ is more likely to be successful.

Suppose that the information received from the trusted peers regarding the other trust dimensions is such that the delegating agent calculates the recommendation values given below, and that the other decision factors being considered have the following values. (Note that the recommendation values calculated above correspond to a single row in this table, i.e. the recommendations in the success dimension.)

Factor	ϕ	ψ
Advertised cost, units per second (i_c)	10	9
Advertised quality, range 1–10 (i_q)	9	8
Trust: success dimension (T^s_i)	0.73	0.21
Trust: cost dimension (T^c_i)	0.90	0.64
Trust: timeliness dimension (T^t_i)	0.71	0.37
Trust: quality dimension (T^q_i)	0.58	0.42
Recommendation: success dimension (R^s_i)	0.64	0.66
Recommendation: cost dimension (R^c_i)	0.43	0.57
Recommendation: timeliness dimension (R^t_i)	0.77	0.66
Recommendation: quality dimension (R^q_i)	0.51	0.66

Furthermore, suppose that the following factor weightings are used: advertised quality is given more emphasis than advertised cost, trust in all dimensions is given the most emphasis, and recommendations are given the least emphasis.

μ_c	μ_q	μ_{ts}	μ_{tc}	μ_{tt}	μ_{tq}	μ_{rt}	μ_{rq}	μ_{rs}	μ_{rc}
0.08	0.12	0.15	0.15	0.15	0.15	0.05	0.05	0.05	0.05

In this example this means that the delegating agent gives the trustworthiness of ϕ and ψ primary importance, followed by the likelihood that results will be processed to a suitable quality, then followed by the likelihood that this processing will be of expected cost, and finally of least importance are the peer recommendations given by α, β and γ.

The delegating agent must calculate the performance value of each of the alternative partners, ϕ and ψ. Thus, applying Eq. (10) to agent ϕ gives:

$$\begin{aligned}\mathrm{PV}(\phi) = {} & (\max_c + 1 - 10)^{0.08} \times 9^{0.12} \\ & \times \mathrm{stratify}(0.73)^{0.15} \times \mathrm{stratify}(0.90)^{0.15} \\ & \times \mathrm{stratify}(0.71)^{0.15} \times \mathrm{stratify}(0.58)^{0.15} \\ & \times \mathrm{stratify}(0.64)^{0.05} \times \mathrm{stratify}(0.43)^{0.05} \\ & \times \mathrm{stratify}(0.77)^{0.05} \times \mathrm{stratify}(0.51)^{0.05} \\ = {} & 1^{0.08} \times 9^{0.12} \times 8^{0.15} \times 10^{0.15} \times 8^{0.15} \times 6^{0.15} \times 7^{0.05} \\ & \times 5^{0.05} \times 8^{0.05} \times 6^{0.05} \\ = {} & 6.51 \end{aligned}$$

Similarly, for agent ψ we get $\mathrm{PV}(\psi) = 4.92$. Therefore, based on these weightings, agent ϕ is the alternative that best balances the factors considered.

To demonstrate how factor weightings allow agents to balance their preferences, suppose that the selecting agent's preferences change such that the likely quality and cost of processing the simulation results is of higher importance than the other decision factors. In this case the agent may use weightings such as those given below. The quality and cost of results is emphasised (in terms of advertised values, trustworthiness and peer recommendations of the likelihood that potential partners will return the advertised values) with the side effect of reducing the importance of the success and timeliness dimensions.

μ_c	μ_q	μ_{ts}	μ_{tc}	μ_{tt}	μ_{tq}	μ_{rt}	μ_{rq}	μ_{rs}	μ_{rc}
0.15	0.15	0.025	0.15	0.025	0.15	0.025	0.15	0.025	0.15

In this case, calculating the performance values gives $PV(\phi) = 5.25$ and $PV(\psi) = 5.31$. Thus, where greater emphasis is placed on quality and cost, ψ is considered the best alternative. Changing the weightings allows the selecting agent to make an appropriate choice according to its current preferences, without needing to change the underlying trust and recommendation data. In general, since each peer is autonomous we cannot guarantee that an agent makes the absolute optimal choice. However, our proposed approach makes the best expected choice given the agent's (and its trusted peers') experiences so far.

8. Experimental validation

To validate the proposed MDT-R model, we have constructed a simulation prototype. Recall from Section 2 that we are concerned with segments of workflow in which the agents involved are true independent entities contributing to the overall design task. Our example in Section 2 was that of performing a simulation, which might typically involve a large number of steps starting with generating configurations, then running the simulation itself, and finally processing the raw results. These tasks can be delegated to agents, and it is this scenario that the simulations discussed in this section are intended to mirror. We have investigated the performance of MDT-R using several workflow fragments containing 500–3000 primitive activities, each of which has specific capability and resource requirements. Our simulations contain 10–100 peer agents, each with specific individual capabilities. The actual workflow used is not of concern. Rather, our aim is to validate MDT-R in terms of the improvement it can offer when a complex task, such as a simulation in the context of CSCW, is assigned to an autonomous agent that must in turn delegate the associated primitive subtasks. Our experiments are not intended to fine tune the parameter weights in the model (i.e. the ω_i's and μ_i's), but rather to demonstrate the usefulness of the model.

All agents use MDT-R to delegate activities to other peers. The architecture of the individual peers is shown in Fig. 2. Each peer has a set of assigned and working tasks, the former being the tasks that it has delegated to others and the latter being the tasks that have been delegated to it (and any individually executed tasks). Peers also maintain a set of trust models of other peers, and have message buffers for incoming and outgoing messages. There is an external peer directory service with which peers register on joining the system, and can be queried to determine which peers are capable of performing a given task (along with the advertised cost and quality for specific task performance).

A random peer is delegated to the overarching activity (i.e. the final node) of the process graph fragment. This peer then selects other peers, using MDT-R, for the tasks that are instrumental to the achievement of the overarching task, and so forth through the tree. Peers communicate via a message transport using a simple proprietary agent communication language, which allows them to request and provide information, and allocate and manage task execution.

We have performed a number of simulations using a variety of experimental configurations. However, due to space constraints we will concentrate on the four experimental configurations, shown below. The failure, overrun, overcharge, and quality columns indicate the range of the failure, overrun, overcharge and quality deviation rates (%), such that peers are randomly initialised with a value in the specified range. The peers column in the table is the number of peers of each type contained in the simulation.

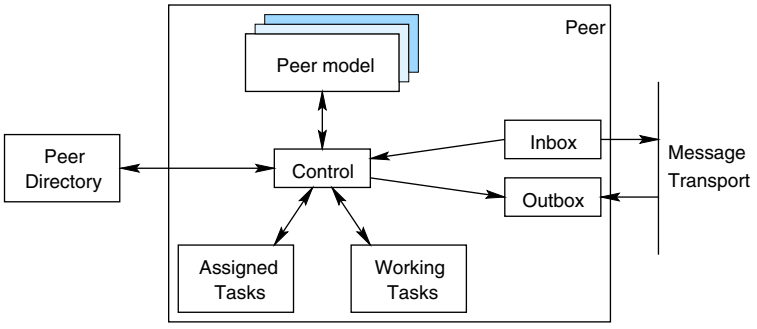

Fig. 2. Peer architecture.

Configuration	Peers	Failure	Overrun	Overcharge	Quality
1	100	0–20	0–30	0–30	10–40
2	45	30–50	30–50	30–50	30–50
	5	0–10	0–10	0–10	0–10
3	10	0–5	0–5	0–5	0–5
	30	10–30	10–30	10–30	10–30
	20	75–100	75–100	75–100	75–100
4	10	0–5	0–50	0–5	0–5
	10	0–5	0–5	0–50	0–5

We performed a number of experiments using these configurations of peers on workflow fragments of various sizes. Each experimental configuration was run for 10 iterations, with the results being averaged. The results shown are for the interactions delegated by a single subject peer, which has failure, overrun, overcharge, and quality deviation rates of 0–20%. The subject peer used 10 strata when using MDT-R, which provides a fair balance between minimising overfitting and maximising precision. (An investigation of alternative strata values is beyond the scope of this paper, but our previous results have shown that 10 strata provides a realistic balance (Griffiths, 2005b).)

One of the key benefits of using MDT-R is the increased number of successful interactions. To illustrate this we used configurations 1–3 and a workflow fragment containing 2000 activities. We ran two versions of the simulation, one using MDT-R for task allocations and the other using a random allocation method (where the choice between peers who have the required capabilities and meet the advertised cost and quality requirements, is made arbitrarily). The success rates achieved are shown in the following table.

Allocation method	Configuration 1	Configuration 2	Configuration 3
MDT-R (%)	58	56	41
Random (%)	54	46	27

In these experiments MDT-R gives an improvement in success rate of 4%, 10% and 14% in configurations 1, 2 and 3 respectively. Thus, MDT-R consistently reduces the number of failures, with a higher advantage occurring when there is a higher proportion of unreliable peers. The greatest improvement is in configuration 3 where there is a small subset of reliable peers in a population of mostly moderate reliability and a small subset of highly unreliable peers. In this case MDT-R enables the unreliable peers to be avoided and the reliable peers selected, where the peer capabilities are such that this is possible.

Since trust is based on experience, MDT-R works best in situations where there are a large number of interactions to allow trust models to mature. To illustrate this we have considered configuration 3 with three different size workflow fragments: 500, 1750, and 3000 activities. The results are given in the following table:

Allocation method	Workflow size		
	500	1750	3000
MDT-R (%)	31	41	67
Random (%)	27	27	28

As trust models mature the success rate increases significantly, more than doubling from 31% with immature trust models (although this is still better than the 27% achieved with random allocation) to 67% once the models have matured. It should be noted that with larger workflows than 3000 activities no further improvement is achieved, since after around 2500 activities the trust models give a fairly accurate representation of peers' trustworthiness.

Finally, to illustrate the effect of the weightings using in calculating the performance value (the μ_i's in Eq. 10) we used three sets of weights. The first, where all factors were give equal emphasis, the second where the cost factors (advertised cost and trust and reputation in the cost dimensions) were emphasised, and finally where timeliness was emphasised. Using these weightings and configuration 4, gave the following results:

Disposition	Success (%)	Timeliness (%)	Cost (%)
Equal emphasis	86	89	65
Emphasise cost	87	87	71
Emphasise timeliness	86	95	63

Thus, using weights that favour cost increased the number of interactions in which cost expectations were met, but also decreased the number in which timeliness expectations were met. Conversely, using weights that emphasised timeliness increased the proportion of interactions in which timeliness was met, but decreased the number in which cost expectations were met. Emphasising cost or timeliness has little impact on the success rate. Similar results have been obtained for the quality dimension.

9. Conclusions

In this paper we have described the notion of multi-dimensional trust and provided a mechanism for peers to share information about their experiences. Our proposed framework allows agents to model the various facets of trust, and combine these with information provided by peers (along with other decision factors) when delegating an activity. The MDT-R model is highly flexible, and system designers have full control of the trust dimensions modelled and the relative weightings given to the decision factors. Our model is generally applicable, but in this paper our inspiration has

been the automated elements of a cooperative design task. In particular we have described a mechanism that can be applied to segments of workflow that are performed and allocated by autonomous agents. We have shown by simulation that MDT-R increases the proportion of successful interactions. Furthermore we have demonstrated that the weighting factors used to combine decision factors can be used successfully to emphasise particular factors according a peer's (or the designer's) current preferences.

Currently, the weightings for decision factors are specified by the system designer. Although the designer may specify different weightings for different situations, agents do not determine appropriate weightings for themselves. Future work involves exploring mechanisms, such as learning using genetic algorithms, to enable agents to tailor the weightings according to their preferences (e.g. maximising quality or minimising failures). Although we have validated MDT-R in a simulation prototype, we are performing ongoing experimentation, and aim incorporate MDT-R into a real-world peer-to-peer system.

References

Abdul-Rahman, A., & Hailes, S. (2000). Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii international conference on system sciences (HICSS-00)* (pp. 1769–1777). ACM Press.

Azzedin, F., & Maheswaran, M. (2002). Integrating trust into Grid resource management systems. In *Proceedings of the international conference on parallel processing (ICPP-02)* (pp. 47–54).

Bridgeman, P. W. (1922). *Dimensional analysis*. Yale University Press.

Bursell, M. (2005). Security and trust in P2P systems. In R. Subramanian & B. D. Goodman (Eds.), *Peer-to-peer computing: The evolution of a disruptive technology* (pp. 145–165). Idea Group Publishing.

Castelfranchi, C. (2004). Trust mediation in knowledge management and sharing. In Jensen, C., Poslad, S., & Dimitrakos, T. (Eds.), *Proceedings of the second international conference on trust management (iTrust 2004)* (pp. 304–318).

Castelfranchi, C., & Falcone, R. (1998). Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In *Proceedings of the third international conference on multi-agent systems (ICMAS-98), Paris, France* (pp. 72–79).

Chen, S., Wu, H., Han, X., & Xiao, L. (2005). Collaborative design environment based on multi-agent. In *Proceedings of the ninth international conference on computer supported cooperative work in design (CSCWD-05)* (pp. 481–485).

d'Inverno, M., & Luck, M. (1996). Understanding autonomous interaction. In W. Wahlster (Ed.), *Proceedings of the twelfth European conference on artificial intelligence (ECAI-96)* (pp. 529–533). John Wiley & Sons.

Gambetta, D. (1988). Can we trust trust? In D. Gambetta (Ed.), *Trust: Making and breaking cooperative relations* (pp. 213–237). Basil Blackwell.

Griffiths, N. (2005a). Cooperative clans. *Kybernetes, 34*(9–10), 1384–1403.

Griffiths, N. (2005b). Task delegation using experience-based multi-dimensional trust. In *Proceedings of the fourth international conference on autonomous agents and multiagent systems (AAMAS-05)* (pp. 489–496). ACM Press.

Griffiths, N., Luck, M., & d'Inverno, M. (2003). Annotating cooperative plans with trusted agents. In R. Falcone, S. Barber, L. Korba, & M. Singh (Eds.), *Trust, reputation, and security: Theory and practise* (pp. 87–107). Springer-Verlag.

Griffiths, N., & Sun, S. (2005). Supporting peer-to-peer collaboration through trust. In *Proceedings of the ninth international conference on computer supported cooperative work in design (CSCWD-05)* (pp. 440–445).

Guilfoyle, C. (1998). Vendors of intelligent agent technologies: A market overview. In N. R. Jennings & M. J. Wooldridge (Eds.), *Agent technology* (pp. 91–104). Springer.

Huynh, T. D., Jennings, N. R., & Shadbolt, N. R. (2004). Developing an integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 7th international workshop on trust in agent societies (TRUST-04)* (pp. 65–74).

Jennings, N. R. (1994). *Cooperation in industrial multi-agent systems. World Scientific series in computer science* (vol. 43). Singapore: World Scientific.

Luck, M., McBurney, P., & Preist, C. (2003). *Agent technology: Enabling next generation computing*. AgentLink II.

Luck, M., McBurney, P., & Preist, C. (2004). A manifesto for agent technology: Towards next generation computing. *Journal of Autonomous Agents and Multi-Agent Systems, 9*(3), 203–252.

Marsh, S. (1994a). Formalising trust as a computational concept. Ph.D. thesis, University of Stirling.

Marsh, S. (1994b). Optimism and pessimism in trust. In Geffner, H. (Ed.), *Proceedings of the Ibero-American conference on artificial intelligence (IBERAMIA-94)* (pp. 286–297).

Sabater, J., & Sierra, C. (2002). REGRET: A reputation model for gregarious societies. In *Proceedings of the first international joint conference on autonomous agents in multi-agent systems (AAMAS-02)* (pp. 475–482).

Schoder, D., & Fischbach, K. (2003). Peer-to-peer prospects. *Communications of the ACM, 46*(2), 27–29.

Triantaphyllou, E. (2000). *Multi-criteria decision making methods: A comparative study*. Kluwer Academic Publishers.

Waldman, M., Cranor, L. F., & Rubin, A. (2001). Trust. In A. Oram (Ed.), *Peer-to-peer: Harnessing the power of disruptive technologies* (pp. 242–270). O'Reilly.

Yiming, Y. (2003). *Agent supported cooperative work*. Kluwer Academic Publishers.

Yu, B., & Singh, M. P. (2002). An evidential model of reputation management. In *Proceedings of the first international joint conference on autonomous agents in multi-agent systems (AAMAS-02)* (pp. 295–300).

Zhang, Y., Ghenniwa, H., & Shen, W. (2005). Enhancing intelligent user assistance in collaborative design environments. In *Proceedings of the ninth international conference on computer supported cooperative work in design (CSCWD-05)* (pp. 107–112).