A Threatmodel for Trust-based Systems Consisting of Open, Heterogeneous and Distributed Agents

Jan Kantert[1], Lukas Klejnowski[1], Sarah Edenhofer[2], Sven Tomforde[2] and Christian Müller-Schloer[1]

[1]*Institute of Systems Engineering, Leibniz University Hanover, Appelstr. 4, 30167, Hanover, Germany*

[2]*Organic Computing Group, Augsburg University, Eichleitnerstr. 30, 86159, Augsburg, Germany*

Keywords: Multi-agent-systems, Trust, Threatmodel, Norms, Desktop-grid System, Agents.

Abstract: Information and communication technology witnesses a raise of open, distributed systems that consist of various heterogeneous elements. Within such an environment, individual elements have to efficiently fulfil their goals, which may require cooperation with others. As a consequence, a variety of threats appears that need to be handled and circumvented in the entity's behaviour. One major technical approach to provide a working environment for such systems is to introduce technical trust. In this paper, we present a basic threat model that comprises the most important challenges in this context – related to the basic system and the trust management, respectively. In order to illustrate the particular hazardous aspects, we discuss a Desktop Computing Grid application as scenario.

1 INTRODUCTION

Open Distributed Systems are comprised of an unknown number of autonomous entities that are heterogeneous with respect to goals, capabilities, preferences and behaviours, see (Centeno and Billhardt, 2011). Additionally, the system is open, hence entities from unknown sources, with code from unknown programmers can enter and leave the system at any time, cf. (Hermoso et al., 2010). Finally, due to their distributed nature and autonomy, there is no form of direct or central control in such a system, see (Centeno et al., 2011).

It is generally agreed that such a system, though beneficial in many ways, introduces a large amount of uncertainty among the participating entities, cf. (Wierzbicki, 2010). Some of them apply adversary strategies to exploit or damage these systems. A commonly used solution to this is to equip such a system with a Trust Management (TM) system: By rating each other's behaviour according to clear defined rules, the participants of such a system can discern cooperative from adversary participants and hence reduce their uncertainty about the system. We refer to the group of cooperative agents that have positive mutual trust-relations, as *Trusted Community*, see (Klejnowski, 2014). We hence relate to these agents when speaking of threats.

In this paper, we present an exemplary threat model for ODS (Open Distributed System), based on the example of a Trusted Desktop Grid. This model can be used to (a) identify which threats can be mitigated by setting up an according TM System among the participants, and (b) identify threats that cannot be tackled by TM, but rather require a security management system on a lower level. A more generic third type of threats, the attacks on TM systems themselves, is then summarised by relating to the according literature.

The remainder of this paper is organised as follows: In Section 2, we present our application scenario. Afterwards, we relate our scenario to a larger class of systems in Section 3. Thereafter, we introduce threats for such systems and classify them according to the STRIDE model in Section 4. Finally, we conclude in Section 5.

2 APPLICATION SCENARIO: THE TRUSTED DESKTOP GRID

As an application scenario, we investigate open grid computing systems which can host numerous distributable workloads, e.g. distributed rendering of films. The system is considered open since there is no central controlling entity, all communication is performed peer-to-peer, and agents are free to join. Worker nodes belong to different administrative do-

Kantert, J., Klejnowski, L., Edenhofer, S., Tomforde, S. and Müller-Schloer, C.
A Threatmodel for Trust-based Systems Consisting of Open, Heterogeneous and Distributed Agents.
DOI: 10.5220/0005696801730180
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 1*, pages 173-180
ISBN: 978-989-758-172-4

mains, thus, benevolent behaviour cannot be assumed. Nodes participate voluntarily to submit work into the system and, thereby, achieve a speedup of their jobs. However, a successful system relies on reciprocity: Agents also have to compute work units for other submitters.

To overcome the inherent problems of an open system where no particular behaviour can be assumed, we introduce a trust metric. Agents receive ratings for all their actions from their particular interaction partners. This allows others to estimate the future behaviour of a certain agent based on its previous actions. To perform this reasoning, a series of ratings for a certain agent can be accumulated to a single reputation value using the trust metric.

Autonomous agents need to become aware of the expected behaviour in the system. Therefore, we influence the desired actions by norms. These norms are valid for an *Action* in a certain *Context* and, thereby, guide the agents. To enforce the behaviour, they impose a *Sanction* if violated or offer an *Incentive* if fulfilled. For details about normative control in Multi-Agent Systems, see (Pitt et al., 2011).

Emergent behaviour as a consequence of self-organised interactions among distributed agents can result in positive and negative effects. Establishing implicit trust communities by increasing cooperation with other well-trusted agents and consequently isolating malicious agents to a certain degree is considered as positive emergent behaviour in this context. In contrast, *negative emergent behaviour* typically impacts the overall system performance and consequently needs to be countered. In the following, we describe two scenarios within the TDG example to illustrate the impact of such effects.

During operation, agents are free to join the Trusted Desktop Grid (TDG). In case of a potentially large group of malicious agents joining the system simultaneously, we can observe a negative emergent effect with respect to the actual trust level of currently participating agents. For instance, a group of colluding *Freeriders* will load the system with additional work packages, while simultaneously rejecting to work for others. Consequently, benevolent agents will also reject work packages issued by the *Freeriders* (following a tit-for-tat concept). As a result, we can observe numerous bad ratings for both, benevolent and uncooperative agents. In sum and on average, the trust levels will drop drastically resulting in a system state where agents do not trust each other any more. We consider such a situation as a disturbed system state; this emergent situation is called “trust breakdown“ (Steghöfer et al., 2010).

To prevent problems such as *Trust Breakdowns* and increase the robustness of the system, we introduced an agent organisation called explicit Trusted Community (eTC), see (Bernard et al., 2011). When a group of agents notices that they mutually trust each other, they can proceed to form an eTC. They elect a leader - called the Trusted Community Manager (TCM). This TCM performs maintenance tasks inside the community. Agents can then reduce security measures such as replication of work units and are able to gain a better speedup. Members of an eTC can easier resist attacks, because they can always just cooperate inside the community and ignore the environment.

3 APPLICABLE SYSTEM CLASS

We first describe the system class that the following threat model can be applied to. This is based on the exemplary description of the Trusted Desktop Grid (TDG), an instance of this class. A typical example of a technical Open Distributed System (ODS) that benefits from the operation of a Trust Management system (TM), is an open Desktop Grid (DG) System. In the following we exactly classify to which instances of this system class this applies.

DG Systems are based on the idea of using shared idle resources (also referred to as “harvesting”) of usual personal computers, in order to allow for fast and parallel computations for suited applications. Desktop Grid Systems are distinguished from (traditional) Grid Systems: The latter operate with dedicated and static, often homogeneous, machines (e.g. clusters) in order to provide computation as a service. Mostly the scheduling of jobs is centralised and machines can be fully controlled. Desktop Grid Systems on the other hand, are a network of rather unreliable machines providing computational power on best-effort basis in often dynamic environments. They are mostly decentralised, and direct control is not possible. In traditional Grid Computing, research is often focused on efficient and fair scheduling, management of the dedicated machines (e.g. fail-over mechanisms, redundancy etc.) and efficient processing of workflows with highly interdependent tasks (e.g. via MPI (Walker and Dongarra, 1996)). Realisations of these systems are mostly based on either the Globus Toolkit (Walker and Dongarra, 1996) or Web-Service-based standards (Foster et al., 2004). In the remainder of this paper, traditional Grid Systems will not be further referred to, mainly for their lack of openness and uncertainty among the participants (no ODS) and thus low motivation for the application of Trust Management.

Despite the clear differences to traditional Grid computing systems, Desktop Grid System realisations are nonetheless rather fragmented. In the following, the exact type of Desktop Grid System used as reference, is classified according to the taxonomy presented in (Choi et al., 2008). This taxonomy is built upon four main perspectives containing several properties to classify DG systems. The main categories are *System*, *Application*, *Resource* and *Scheduler*. The strongest classification is provided by the System perspective, therefore firstly the properties of this perspective are described: The resource provider property discerns two main classes of Desktop Grid Systems: Enterprise and volunteer-based DG Systems. Enterprise (as well as academic) Desktop Grids are networks within a (virtual) organisation, which provide their computational service mainly for members of this organisation. Usually, the connectivity in such systems is rather high, while volatility, machine heterogeneity and distribution of control are low. The most fundamental difference to volunteer-based Desktop Grid Systems is however the user base: Participating clients are mostly from the same administrative domain (sometimes even within a local area network) as the organisation providing and operating this service. Users are thus often known personally and adversary behaviour disturbing the system is seldom an issue. A typical example for an Enterprise DG is a network between research institutions from a domain that depends on computationally intensive experiments (e.g. particle physics). Researchers can benefit from the fact that nowadays computing power is often abundantly present and often not used exhaustedly and consistently. This provides for opportunities to share these resources with other researchers having different experimentation schedules and in turn take advantages of other institutions' resources when experiments are conducted. Realisations of Enterprise Desktop Grid Systems are often based on Condor (Litzkow et al., 1988) or similar frameworks. In contrast, volunteer-based Desktop Grid Systems rely on mostly anonymous users, connected through the Internet, and willing to donate their resources to other users. Volunteers are per se a greater risk than organisation members or even owners of dedicated machines: By volunteering, a user gives no guarantee as to which degree it will provide any service and because of anonymity, adversary behaviour of users can be a serious issue. Additionally, participating clients are heterogeneous in terms of provided computational power, storage capacity and availability. Consider for example users from varying time zones or users connecting only on rare occasions. In summary, the resource provider property discerns Enterprise and volunteer-based Desktop Grid Systems. Enterprise DGs are closed systems (as opposed to Open Distributed Systems) and therefore not suited as hosting system for the application of Trust Communities. From here on, the classification of the Trusted Desktop Grid as a volunteer-based DG is adopted and the further classification according to the properties of the system perspective is applied to this type of systems only.

Further classification of the TDG is based on the *organisation* property: Centralised DGs are based on a client-server-volunteer model while Distributed DGs are managed without servers. In Centralised DGs the servers are mainly responsible for managing volunteers (bootstrapping, identification, exclusion etc.) and scheduling jobs created by the clients on volunteer machines. Most centralised Desktop Grids use the following scheme: Clients generate jobs and contact the servers which then choose tp appropriate volunteer machines and inform them of new tasks to process. Re-scheduling in case of failures (volunteer machines can be unreliable) and result verification follow next, before the clients are requested to fetch the task results. It is important to note, that in those systems volunteer nodes do not submit jobs to the server. Therefore, volunteers need incentives to participate in the systems. A common approach to this is to establish a DG for scientific computations that benefit the greater public good and motivate users connected to the Internet to donate their spare resources for this purpose (Anderson, 2004). In contrast, Distributed DGs transfer the management and scheduling mechanism to the clients, which are then for example responsible for finding suited volunteer machines. Additionally, Distributed Desktop Grids can be designed as Peer-To-Peer (P2P) systems - this not only refers to the connectivity in the system but more importantly to the fact, that each grid node can submit jobs to other nodes, thus the distinction between client and volunteer is not valid any more. This creates an entirely different motivation for volunteers to participate compared to Centralised Desktop Grid Systems: Users are self-interested and participate in the system in order to let other volunteers process tasks from their own computationally intensive applications, like for example the rendering of large animation scenes (Patoli et al., 2009). In turn, they are obliged to donate their own resources to other users. An exemplary implementation of such a system is the Organic Grid (Chakravarti et al., 2005).

In summary, the organisation property discerns between the server-based Centralised DGs and the Distributed Desktop Grid Systems. The management of system participants with a centralised server architecture is a closed system approach: Each new system participant has to contact a server when entering the system and whenever it interacts with other participants (consider for example the centralised scheduling scheme), thus the servers control the participants. In

Table 1: Possible threats class 1 and 2.

	Threat	Target	STRIDE
Class 1: Inside threat, System exploitation	No participation as worker (freeriding)	-	-
	No return of work unit results (hidden freeriding)	-	-
	Submission of false positive trust ratings (collusion)	-	-
	Delegation of accepted work units to other workers, acting as owner (hidden freeriding)	-	-
	Give false information (performance level, work load) to avoid work unit processing requests	-	-
	As TC member exploit trustworthiness and lack of safety means	-	-
	As TC manager exploit trustworthiness, members dependency and lack of safety means	-	-
	Systematically violate norms	Agent	-
	As TC member violate norms on requests from outside agents	Agent	(S)
	Propose norms which would allow (unnoticed) system exploitation by agent	Agent	(S)
Class 2: Inside threat, System damage	DoS-attack via work unit replication	System	(D)
	DoS-attack or slow-down via fake work units	System	(D)
	As TC manager release norms that threaten the operation of the system if followed	System	-
	Spreading of malware via work units (or results)	System or agent	-
	DoS-attack via excessive messaging	System or agent	(D)
	Submission of false negative trust ratings (discrediting attack)	System or agent	-
	Submission of work units with unrealistic processing conditions (leading to legitimation of negative trust ratings)	Agent	-
	Cancelling of accepted work units (slow-down)	Agent	-
	Return of false work unit results	Agent	-
	As TC manager exclude single agents from the TC or dissolve the TC entirely	Agent	-
	Accuse other agents of norm violations	Agent	(D)
	Propose destructive norms	Agent	D

contrast, Distributed DGs distribute the control among all participants, and interactions are executed directly between them. Additionally, new participants can enter the system without following the specifications of servers, and do so anonymously. Therefore, only distributed systems fulfil our requirements of an Open Distributed System as application scenario for Trust Management in general and Trust Communities in particular.

The remaining two properties of the system perspective are *scale* and *platform*. The scale property discerns between internet-based and LAN-based Desktop Grid Systems. LAN-based systems are closed systems controlled within a single administrative domain and thus no ODS. Instead the TDG is designed as internet-based system. The platform property differentiates between Web-based and middleware-based systems in the context of the technical realisation of a client machine. The Trusted Desktop Grid relies on a middleware-based solution, however this property has no influence on the openness of a system and can therefore be neglected. This completes the classification according to the System perspective in the taxonomy presented in (Choi et al., 2008). In summary, in this paper, Desktop Grid Systems are referred to as volunteer-based, distributed, P2P-based, internet-based DGs with client participation over a DG middleware.

Again the taxonomy in (Choi et al., 2008) is used to further classify the TDG according to a selection of relevant properties from the *application* and *scheduling* perspective. The Trusted Desktop Grid is a system where each agent operates its own scheduler (submitter component) to distribute work units (WUs, also referred to as tasks in the literature) generated by an application on the host machine. In the taxonomy,

Table 2: Possible threats class 3 and 4.

Class 3: Outside threat, System exploitation	Manipulate ReputationDB to improve own reputation or deteriorate rival reputation (indirect advantage)	-	T
	Sybil attack: Change own identity to appear as new agent (with no reputation history)	-	S
	Manipulate others processing queues to place (or prioritize) own work units.	-	T
	Manipulate direct experience history of interaction partners to hide misbehaviour and/or appear trustworthy.	-	T
	Manipulate TC Manager to become TC member	-	E,T
	Manipulate messages with negative trust ratings to avoid being detected as misbehaving agent	-	T
	Appear as agent with high reputation, TC membership or TC managing rights (impersonation)	-	S
	Corrupt the election process in order to become TC manager	-	T
	Manipulate messages, work unit results or information of other agents in order to damage their reputation (rival management: relative improvement of own reputation and higher ranking)	-	T
	Manipulate messages from other submitting agents to keep preferred worker agents from having a high workload	-	T
	Manipulate TC manager messages to change norms to own benefit	-	T
	Manipulate existing norms to own benefit	-	T
	Prevent unfavourable norms from spreading	-	T
Class 4: Outside threat, System damage	Manipulate ReputationDB (for example rendering all agents untrustworthy)	System	E,T
	Generate fake rating messages or manipulate existing messages	System or agent	T
	Man in the middle attack to access work unit results or data	Agent	I
	DoS-attack to damage the communication between agents	System or agent	D
	Generate Class 2 agents and control them	System or agent	R
	Introduce contradicting/inconsistent norms to destroy system	System or agent	D

the property application dependency discerns between the type of jobs an application produces: Jobs from Bag-of-tasks (BoT) applications are composed of work units that can be processed (and thus scheduled) independent of each other, whereas all other applications produce jobs with some form of flow or execution dependency among the tasks. For the evaluations presented, BoT-applications were used, as in the majority of research literature. Additionally, the application divisibility property classifies jobs according to their composition flexibility, that is answering the question: Is the division of a job into work units fixed or can it be adapted to the current scheduling situation (for example available resources and resource performance)? Again, the Trusted Desktop Grid is capable of both approaches.

The fine-grained definition of the types of jobs (or applications producing them) in the TDG now allows classifying the scheduling in this system according to the scheduler perspective in the taxonomy. The most important property to distinguish this system from other types is the organisation property. This property describes how scheduling in a Desktop Grid is implemented. As already defined in the system perspective, a distributed scheduling scheme was used: All agents operate their own scheduler and there is no scheduling of foreign work units. This is in contrast to the schemes central and "hierarchic scheduling", and is a distinctive feature of our system, as most DG systems evaluated in the literature consider central or hierarchical scheduling. However, these forms of scheduling are less challenging from the trust context and restrict the openness of an according system. The individual schedulers of the agents further operate in push mode, which means that agents send out work unit processing requests to available workers, which then react with an acknowledgement or a rejection (scheduling mode property). This allows for example

Table 3: STRIDE model.

S	Spoofing	Attackers pretend to be someone (or something) else.
T	Tampering	Attackers change data in transit or at rest.
R	Repudiation	Attackers perform actions that cannot be traced back to them.
I	Information disclosure	Attackers steal data in transit or at rest.
D	Denial of service	Attackers interrupt a system's legitimate operation.
E	Elevation of privilege	Attackers perform actions they are not authorised to perform.

to detect free-riding despite decentralised control. The next relevant scheduling property is the dynamism: (Dynamic) online and (static) offline scheduling are discerned. In an open system like the Trusted Desktop Grid, where host and resource availability are subject to constant change, only online scheduling is possible. The final property needed for the TDG scheduling classification is the scheduling goals property. This property defines which performance metrics are used to evaluate the scheduling subsystem in a Desktop Grid. We apply the speedup metric.

The taxonomy of Desktop Grid Systems followed, further classifies these systems according to some additional properties within the four perspectives. However, a more elaborate systematic classification is not applied, as the Trusted Desktop Grid either does not have strict requirements on the properties or they are not in the focus of the Trust management applicability.

In addition to threats to the class of systems described above, we also model threats to the operation of Trust-based multiagent organisations (Horling and Lesser, 2004; Oussalah and Griffiths, 2005; Klejnowski, 2014) in such a system. We do this exemplarily for the application of explicit Trust Communities (eTC (Klejnowski, 2014)), a Trust-based organisation with management by a dynamically elected TC manager (TCM).

4 THREATS IN OPEN DESKTOP GRID SYSTEMS

The application of Trust Management to technical systems has been pursued with growing popularity in recent decades, especially with the advent of the internet as a world-wide open distributed system. TM can be applied to mitigate many specific threats in ODS, as well as threats in Desktop Grid Systems (Domingues et al., 2007). However, Open Desktop Grid systems based on autonomous agents are a rather new approach and not many TM systems are specifically tied to this system class. As such, we propose a threat model, by picking up threats that can be countered by TM in such a system and threats that cannot be countered by these means. In this we follow the argumentation of, e.g., (Castelfranchi and Falcone, 2010), that trust and security are two conceptually different notions and that a TM system cannot be expected to work if there is no underlying security system. This is also valid if security is understood as facet of trust (Poslad et al., 2003; Steghöfer et al., 2010). We show this with the examination of security threats that can lead to system exploitation or damage despite a working TM system. On the other hand, assuming there is a proper security system, a system can nonetheless be exploited or damaged if there is no, or only improperly configured TM system. We therefore examine threats to the system participants that have to be targeted by such a TM system.

For the modelling of the security threats, we relate to the famous STRIDE approach (Hernan et al.,). In this, security threats are characterised according to the following categories shown in Table 3.

The following threat model is based on the definition of the function scope of the two concepts *trust* and *security*, stating which class of threats is countered by which part of the system:

Firstly, we can group threats to the system by the *capabilities* the attacking entity owns:

- **Inside** threats are executed with capabilities an element of the system (agent) owns due to its autonomy. Attackers are owners of agents representing them and achieving their goals.
- **Outside** threats are executed with capabilities that are beyond those of elements of the system (STRIDE model) and usually require additional software to be performed. Attackers do not necessarily own an agent participating in the system.

Table 4: Threat types.

	System exploitation	System damage
Inside threat	Class 1	Class 2
Outside threat	Class 3	Class 4

Table 5: Responses to threats.

Threat class	Responsible sub-system
Class 1	Trust Management system
Class 2	Trust Management system
Class 3	Security system
Class 4	Security system

Secondly, we can group the attackers according to their *objectives*:

- System exploitation is the objective to gain advantages from the system for the own benefit (performance) without contributing accordingly.
- System damage represents a class of objectives that do not aim at improving the performance, but range from the aim to damage the operation of the system or elements of the system to specific aims like stealing a certain password.

Note that we do not consider threats that appear as attacks although the "attacker" has no respective objective. This is the case when faulty hardware is involved and for example communication affected. We classify the threats with respect to the capabilities and objectives of the attacker in Table 4.

In this model, we focus on threats in a Trusted Desktop Grid System. Additionally, we provided the STRIDE classification for outside threats. In the following, threats comprising the four classes are listed in Table 1 and Table 2. Table 5 lists the sub-systems responsible to counter each threat class. In addition, a Trust Management as such can be the target of security violations, cf. (Marmol and Pérez, 2009) for an elaborate analysis of this generic class of threats.

5 CONCLUSION

This paper introduced a threat model for open, distributed systems that make use of technical trust to counter challenges related to uncertainty of agent behaviour. Based on the Trusted Desktop Computing Grid system as application scenario, we discussed threats that appear in the context of such systems. The basic idea of this work is to specify and categorise these threats to find generalised patterns that counter them.

Future work deals with the question of how this model can be extended towards a unified notion of threats for open, heterogeneous agent collections; this is assumed to provide a basis for the development processes of such systems, as possibly threats are mapped towards strategies to circumvent the negative impacts.

ACKNOWLEDGEMENTS

This research is funded by the research unit "OC-Trust" (FOR 1085) of the German Research Foundation (DFG).

REFERENCES

Anderson, D. P. (2004). BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA. IEEE.

Bernard, Y., Klejnowski, L., Çakar, E., Hähner, J., and Müller-Schloer, C. (2011). Efficiency and Robustness Using Trusted Communities in a Trusted Desktop Grid. In *Proc. of SASO Workshops*, pages 21–26, Michigan, US. IEEE.

Castelfranchi, C. and Falcone, R. (2010). *Trust Theory: A Socio-Cognitive and Computational Model*, volume 18. John Wiley & Sons, Chichester, UK.

Centeno, R. and Billhardt, H. (2011). Using incentive mechanisms for an adaptive regulation of open multi-agent systems. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 139–145. AAAI Press.

Centeno, R., Billhardt, H., and Hermoso, R. (2011). An adaptive sanctioning mechanism for open multi-agent systems regulated by norms. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 523–530. IEEE.

Chakravarti, A. J., Baumgartner, G., and Lauria, M. (2005). The organic grid: self-organizing computation on a peer-to-peer network. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(3):373–384.

Choi, S., Buyya, R., Kim, H., and Byun, E. (2008). A Taxonomy of Desktop Grids and its Mapping to State of the Art Systems. Technical report, Grid Computing and Dist. Sys. Laboratory, The University of Melbourne.

Domingues, P., Sousa, B., and Moura Silva, L. (2007). Sabotage-tolerance and Trustmanagement in Desktop Grid Computing. *Future Generation Computer Systems*, 23(7):904–912.

Foster, I., Jennings, N. R., and Kesselman, C. (2004). Brain meets brawn: Why grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 8–15. IEEE Computer Society.

Hermoso, R., Billhardt, H., and Ossowski, S. (2010). Role evolution in open multi-agent systems as an information source for trust. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 217–224. International Foundation for Autonomous Agents and Multiagent Systems.

Hernan, S., Lambert, S., Ostwald, T., and Shostack, A. Uncover security design flaws using the stride approach (2006). *URL: http://msdn. microsoft. com/en-us/magazine/cc163519. aspx [accessed 2010-01-12][WebCite Cache].*

Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(04):281–316.

Klejnowski, L. (2014). *Trusted Community: A Novel Multiagent Organisation for Open Distributed Systems*. PhD thesis, Leibniz Universität Hannover.

Litzkow, M. J., Livny, M., and Mutka, M. W. (1988). Condor-a hunter of idle workstations. In *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104–111. IEEE.

Marmol, F. G. and Pérez, G. M. (2009). Security threats scenarios in trust and reputation models for distributed systems. *computers & security*, 28(7):545–556.

Oussalah, M. and Griffiths, N. (2005). Cooperative clans. *Kybernetes*, 34(9/10):1384–1403.

Patoli, Z., Gkion, M., Al-Barakati, A., Zhang, W., Newbury, P., and White, M. (2009). How to build an open source render farm based on desktop grid computing. In *Wireless Networks, Information Processing and Systems*, pages 268–278. Springer.

Pitt, J., Schaumeier, J., and Artikis, A. (2011). The Axiomatisation of Socio-Economic Principles for Self-Organising Systems. In *Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on*, pages 138–147, Michigan, US. IEEE.

Poslad, S., Charlton, P., and Calisti, M. (2003). Specifying standard security mechanisms in multi-agent systems. In *Trust, Reputation, and Security: Theories and Practice*, pages 163–176. Springer.

Steghöfer, J.-P., Kiefhaber, R., Leichtenstern, K., Bernard, Y., Klejnowski, L., Reif, W., Ungerer, T., André, E., Hähner, J., and Müller-Schloer, C. (2010). Trustworthy Organic Computing Systems: Challenges and Perspectives. In *Proc. of ATC 2010*, pages 62–76, Boston, MA. Springer.

Walker, D. W. and Dongarra, J. J. (1996). Mpi: a standard message passing interface. *Supercomputer*, 12:56–68.

Wierzbicki, A. (2010). *Trust and fairness in open, distributed systems*, volume 298. Springer.