Trusted Community:

A Novel Multiagent Organisation for Open Distributed Systems

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur (abgekürzt: Dr.-Ing.)
genehmigte Dissertation

von
M.Sc. Lukas Klejnowski

geboren am 21. November 1981
in Piekar

2014

1. Referent: Prof. Dr.-Ing. Christian Müller-Schloer
2. Referent: Prof. Dr. rer. nat. Jörg Hähner
Tag der Promotion: 24.02.2014

Danksagung

Ich möchte zunächst meinem Doktorvater Prof. Christian Müller-Schloer für die ausgezeichnete Betreuung danken. Durch seine konstruktive Begleitung meiner Forschung, gepaart mit Vertrauen in meine Arbeitsweise und dem notwendigen Freiraum hat er meine Entwicklung gefördert und mir die notwendige Sicherheit geboten. Ich danke weiterhin meinem Zweitgutachter Prof. Jörg Hähner für seine fachliche, aber vor allem auch menschliche Unterstützung während meiner Zeit als Doktorand.

Diese Arbeit ist während meiner Tätigkeit am SRA entstanden. In dieser Zeit habe ich sehr viel Zuspruch, fachlichen Austausch und eine wunderbare Arbeitsatmosphäre erfahren. Dafür danke ich Prof. Jürgen Brehm, Monika Lorenz und Lars Maasjost sowie meinen Kollegen Sven Tomforde, Emre Çakar, Aret Duraslan, Michael Nolting, Yvonne Bernard und Ioannis Zgeras. Sie haben mich die meiste Zeit am Institut begleitet, immer ein offenes Ohr für mich gehabt und dafür gesorgt, dass ich mich am Institut wohlgefühlt habe. Ich danke weiterhin den neuen Kollegen Sebastian Niemann, Jan Kantert und Henning Spiegelberg für die gemeinsame Zeit. Ich bin sicher, dass durch sie die familiäre Atmosphäre am SRA bestehen bleibt.

Mein größter Dank gilt meiner Frau Anna Charlotte und unserer Tochter Eva Linea sowie meinen Eltern Beata und Jan Klejnowski. Sie haben mich zu dem gemacht, der ich heute bin, mich auch in schwierigen Zeiten bedingungslos unterstützt und in meinem Vorhaben bestärkt.

Zusammenfassung

Schlagworte: Offene verteilte Systeme, Organic Computing, Multiagentensysteme, Multiagentenorganisationen, Vertrauen und Reputation, Desktop Grid Systeme

Offene verteilte Systeme wie das Internet, Peer-to-Peer-Systeme, Desktop Grid Systeme, E-Commerce-Systeme, drahtlose Sensornetzwerke oder Fahrzeug-Ad-hoc-Netzwerke haben in den letzten Dekaden immer mehr Verbreitung gefunden. Der Entwurf solcher Systeme ist jedoch von Herausforderungen geprägt: Der Betrieb muss stets gewährleistet, oder gar zur Laufzeit optimiert werden und zwar unter Bedingungen wie dynamischen Systemteilnehmern, eigennützigen Systemteilnehmern und unkooperativen oder gar feindlichen Systemteilnehmern. Zusätzlich herrscht in solchen Systeme, bedingt durch die nichtdeterministischen Interaktionen, eine komplexe Dynamik, die zu unerwünschten, emergenten Systemzuständen führen kann. Da die Kontrolle eines solchen Systems dezentral ist und zwischen den Systemteilnehmern verteilt werden muss, erfordern solche Bedingungen von den Systemteilnehmern flexible und komplexe Entscheidungsmechanismen. Diese müssen in der Lage sein,Unsicherheiten bezüglich der Ziele und Motivationen anderer Teilnehmer sowie des Systemzustandes zu bewältigen.

In der Forschungsliteratur wurden viele Ansätze zur multiagentenbasierten Kontrolle solcher offener verteilter Systeme vorgeschlagen. Diese beruhen auf der Delegation von Kontrollmechanismen an autonome Agenten, die diese durch selbstorganisierte und adaptive Prozesse realisieren. In diesem Zusammenhang wird einerseits der Einsatz von (künstlichem) Vertrauen und Organisationen, inspiriert durch die Übertragung soziologischer Konzepte, propagiert. Andererseits entstand vor einigen Jahren das Forschungsfeld *Organic Computing*, das Werkzeuge und Methoden zur Steuerung komplexer Systeme durch naturinspirierte Selbstorganisationsverfahren zum Inhalt hat. In der vorliegenden Arbeit wird zwecks Realisierung von ganzheitlich robusten Kontrollstrukturen, die Verflechtung von Ansätze dieser unterschiedlichen Forschungsfelder vorangetrieben.

Zu diesem Zweck wird in dieser Arbeit die Entwicklung einer vertrauensbasierten Multiagentenorganisation beschrieben. Diese *Trusted Community* genannte Struktur ist für die langfristige Organisation von Agenten mit starken gegenseitigen Vertrauensbeziehungen ausgelegt. Im Kollektiv bilden diese Mitgliederagenten eine vertrauenswürdige geschlossene Umgebung in einem ansonsten unsicheren und offenen Umfeld. Dies erlaubt ihnen, ihre Interaktionen zu optimieren, da der Grad an notwendigen Sicherheitsmechanismen reduziert werden kann. Um weiterhin die Robustheit und fortwährende Optimierung einer solchen Organisation zu gewährleisten, wählen die Mitglieder einen sogenannten *Trusted Community Manager*. Diese Rolle beinhaltet die Repräsentation und Regulierung der Organisation, indem sie die Ausführung von entsprechenden Strategien vorgibt. Diese realisieren beispielsweise die Aufnahme neuer Mitglieder, den Ausschluss von Mitgliedern, die Verteilung weiterer Rollen sowie der Beobachtung und Anpassung dieser Kontrollmechanismen basierend auf der beobachteten Leistungsfähigkeit der Organisation. Zusammengefasst bilden diese Strategien die sogenannten Selbst-X Eigenschaften aus (z.B. Selbstschutz, Selbstheilung, Selbstoptimierung).

Das Trusted Community Konzept basiert auf einem modularen Entwurf: Die Funktionalität dieser Organisationsform wird durch die entkoppelte Komposition von Strategien erbracht. Diese sind konfigurierbar, zur Laufzeit austauschbar und erlauben Anpassungen an die Charakteristiken des Einsatzsystems. Jede dieser Strategien wird formal definiert und für jede wird eine Basisimplementierung vorgestellt. Zusätzlich werden erweiterte sowie szenariospezifische Implementierungsvarianten diskutiert.

In der vorliegenden Arbeit wird zudem ein Systemmodell vorgestellt, das die Anforderungen für den Einsatz von Trusted Communities in technischen Systemen formal spezifiziert und dessen Grenzen diskutiert. Schließlich wird die Anwendbarkeit in einem offenem Desktop Grid System demonstriert. Diese Systemklasse ist gekennzeichnet durch vielfältige technische Herausforderungen, die einen großen negativen Einfluss auf die Leistungsfähigkeit und Robustheit eines solchen Systems haben können. Es wird konsequenterweise in einer ausführlichen Evaluation dargestellt, wie der Einsatz von Trusted Communities zur Lösung dieser Herausforderungen beiträgt und wie sich infolgedessen das System verbessert. Die hierbei unter unterschiedlichen Bedingungen und Störeinflüssen erzielten Ergebnissen werden mit Ergebnissen von verwandten Verfahren aus der Literatur verglichen. Diese Vergleiche belegen, dass der Einsatz von Trusted Communities insbesondere bezüglich der Robustheit des Einsatzsystems signifikante Vorteile bringt.

Abstract

Keywords: Open Distributed Systems, Organic Computing, Multiagent Systems, MAS organisations, Trust and reputation, Desktop Grid Systems

Open Distributed Systems, be it the Internet, peer-to-peer systems, e-commerce systems, Desktop Grid Systems, wireless sensor networks, or vehicular ad-hoc networks, have immensely gained in popularity in the recent decades. The design of such systems is however a challenging domain: The operation must be maintained or even optimised at run time, despite participants entering and leaving the system at will, participants being self-interested and participants being uncooperative or adversary. In addition, the interactions within such systems have complex dynamics and often lead to undesired, emergent system states. As the control of such systems is decentralised and must be distributed among the system participants, these aspects require flexible and complex reasoning mechanism for the participants. These must allow to cope with the uncertainty about motivations and goals of other participants, as well as the uncertainty about the state of the system.

In the research literature on Open Distributed Systems, many approaches have been proposed to realise the control of such a system by the delegation of control to autonomous (software) agents. The control is then realised through self-organising and adaptive processes of the agent society. On the one hand, the realisation of such systems is often aided by the application of computational trust and agent organisation methods, inspired by research in sociology. On the other, hand a relatively recent approach called *Organic Computing* has contributed tools and methods to the control of such complex systems by nature-inspired approaches to self-organisation. It is argued in this thesis that it requires a combination of techniques from these research fields to allow for a truly robust control of technical Open Distributed Systems.

To this end, a trust-based Multiagent organisation approach termed *Trusted Community* is proposed. Trusted Communities are long-term organisations that are formed in a self-organised process by mutually trusted agents. The members establish a highly trusted, closed environment in an otherwise uncertain and open environment. Consequently, the members benefit from optimised interactions due to the reduced degree of required safety means. To ensure the robustness and optimisation of this collective, Trusted Communities designate a members to represent and regulate the organisation. This special role is referred to as *Trusted Community Manager* and allows to execute strategies which decide about the invitation of additional members, the exclusion of members, the assignment of roles and the continuous adaptation of this control based on the performance of the organisation. In sum, these strategies realise the concept of so-called self-X properties (e.g. self-protecting, self-healing, and self-optimising).

The Trusted Community concept presented in this thesis is based on a modular design: The functionality of this organisation is determined by a decoupled composition of strategies. These are highly configurable, exchangeable at runtime and allow to adjust the approach to the characteristics of the applied technical system. Each of these strategies is formally defined, and a basic implementation is provided. In addition, extensions and scenario-specific implementations are discussed.

This thesis provides a system model with a formal specification of the requirements for the application of the Trusted Community approach and a discussion of applicability limitations. The benefits of the application are further demonstrated in a particular instance of a technical Open Distributed System - an open Desktop Grid System. This system class is characterised by many challenging issues that can have a high impact on the performance and robustness of such systems. Consequently, it is shown in an extensive evaluation, which considers many different types of setups and disturbances, that the application of Trusted Communities provides a solution to these issues. The resulting performance and robustness improvements are finally contrasted with the results achieved by related state-of-the-art approaches in the same environment. These comparisons show that Trusted Communities outperform other approaches, esp. with respect to increasing the robustness of the system they are applied in.

Contents

List of Abbreviations

TM	Trust Management
OC	Organic Computing
MAS	Multiagent Systems
AOSE	Agent-Oriented Software Design
TC	Trusted Community
TCM	Trusted Community Manager
TDG	Trusted Desktop Grid
iTC	Implicit Trusted Community
O/C-loop	Observer/Controller-loop
DGS	Distributed Grid System
DG	Desktop Grid
BoT	Bag-of-tasks
WU	Work unit
P2P	Peer-to-Peer
DoS	Denial of Service

List of Figures

List of Symbols

$DT_x^{y,\hat{c}}$	Subjective direct trust value of agent y in context $\hat{c}$, estimated by agent x.	42
$RT_{\mathcal{P}}^{y,\hat{c}}$	Reputation value of agent y in context $\hat{c}$, estimated by opinion providers $\mathcal{P}$.	42
$TC_i(t) \ominus^x a$	Operator denoting the exclusion of an agent a from a $TC_i(t)$ by itself or the TCM.	61
$TC_i(t) \oplus^{tcm_{TC_i}(t)} a$	Operator denoting the inclusion of an agent a in a $TC_i(t)$ by its TCM.	61
$TC_i(t)$	A Trusted Community, being a tuple of members, TCM and roles at time t.	59
$T_x^{y,\hat{c}}$	Subjective trust value of agent y in context $\hat{c}$, estimated by agent x.	42
$U^{TC_i}(t,..)$	Relative TC utility function.	93
$U^x(t)$	Utility function of an agent x.	37
Γ^x	Interaction model of an agent x.	40
Λ_t^i	The implementation i of an agent component type t, e.g. $\Lambda_{TM}^{default}$.	41
$\Psi_{TC_i}(t)$	The set of roles in a Trusted Community $TC_i(t)$ at time t.	59
Θ^x	Observation model of an agent x.	40
$\mathcal{A}(t)$	Agent society of the hosting system at time time t.	45
$\mathcal{B}^{\hat{c}}_{\mathcal{A}(t)}(m)$	A binary fnc. denoting whether there is a global trust breakdown in $\mathcal{H}$	46
$\mathcal{B}^{\hat{c}}_{\mathcal{P}(t)}(m)$	A binary fnc. denoting whether there is a trust breakdown in reference group $\mathcal{P}(t)$	46
$Comp_i^x$	An agent component of an agent x, e.g. $Comp_{TM}^x$.	39
$\mathcal{H}$	The hosting system in which TCs can be applied.	45
$\mathcal{M}^{\mathcal{H}}(t)$	The set of TC members in $\mathcal{H}$ at time t.	60
$\mathcal{M}_{TC_i}(t)$	The set of members of a Trusted Community $TC_i(t)$ at time t.	59
$\mathcal{O}(t) \triangleleft^{\mathcal{F}} TC_i(t)$	Operator denoting the formation of a $TC_i(t)$ by agents $\mathcal{F}$.	61
$\mathcal{O}(t) \triangleright^{tcm_{TC_i}(t)} TC_i(t)$	Operator denoting the dissolution of a $TC_i(t)$ by its TCM.	61
$\mathcal{O}(t)$	The set of operating agent organisations in $\mathcal{H}$ at time t.	59
$\mathcal{U}^{\mathcal{H}}(t)$	The set of unassociated agents in $\mathcal{H}$ at time t.	60
$r \rightsquigarrow^{tcm_{TC_i}(t)} a$	Operator denoting the assignment of a role r to a TC member a by the TCM.	61
$tcm_{TC_i}(t)$	The TC manager of a Trusted Community $TC_i(t)$ at time t.	59

List of publications

[1] S. Tomforde, M. Hoffmann, Y. Bernard and L. Klejnowski, ‘POWEA : A System for Automated Network Protocol Parameter Optimisation Using Evolutionary Algorithms’, in *Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e. V. (GI)*, 2009, pp. 3177–3192.

[2] J.-P. Steghöfer, F. Nafz, W. Reif, Y. Bernard, L. Klejnowski, J. Hähner and C. Müller-Schloer, ‘Formal Specification and Analysis of Trusted Communities’, in *Proceedings of the Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, IEEE, Sep. 2010, pp. 190–195, ISBN: 978-1-4244-8684-7.

[3] J. Steghöfer, R. Kiefhaber, K. Leichtenstern, Y. Bernard, L. Klejnowski, W. Reif, T. Ungerer, E. André, J. Hähner and C. Müller-Schloer, ‘Trustworthy organic computing systems: challenges and perspectives’, in *Autonomic and Trusted Computing*, Springer, 2010, pp. 62–76.

[4] Y. Bernard, L. Klejnowski, J. Hähner and C. Müller-Schloer, ‘Towards trust in desktop grid systems’, in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, IEEE Computer Society, 2010, pp. 637–642.

[5] L. Klejnowski, Y. Bernard, J. Hähner and C. Müller-Schloer, ‘An architecture for trust-adaptive agents’, in *Proceedings of the Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, IEEE, 2010, pp. 178–183.

[6] G. Anders, L. Klejnowski, J.-P. Steghöfer, F. Siefert and W. Reif, ‘Reference Architectures for Trustworthy Energy Management and Desktop Grid Computing Applications’, Universitätsbibliothek der Universität Augsburg, Augsburg, Tech. Rep. March, 2011.

[7] Y. Bernard, L. Klejnowski, R. Becher, M. Thimm, J. Hähner and C. Müller-Schloer, ‘Grid agent cooperation strategies inspired by Game Theory’, in *Proceedings 4. Workshop Grid-Technologie für den Entwurf technischer Systeme - Grid4TS*, Dresden, 2011.

[8] Y. Bernard, L. Klejnowski, E. Çakar, J. Hähner and C. Müller-Schloer, ‘Efficiency and Robustness Using Trusted Communities in a Trusted Desktop Grid’, in *Proceedings of the Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, IEEE, Oct. 2011, pp. 21–26, ISBN: 978-1-4577-2029-1.

[9] Y. Bernard, L. Klejnowski, D. Bluhm, J. Hähner and C. Müller-Schloer, ‘An Evolutionary Approach to Grid Computing Agents’, in *Proceedings of the Italian Workshop on Artificial Life and Evolutionary Computation*, 2012, ISBN: 978-88-903581-2-8.

[10] Y. Bernard, L. Klejnowski, J. Hähner and C. Müller-Schloer, ‘Self-organizing trusted communities of trust-adaptive agents’, *Awareness magazine - Self-Awareness In Autonomic Systems*, pp. 3–5, Apr. 2012.

[11] L. Klejnowski, Y. Bernard, C. Müller-Schloer and J. Hähner, ‘Using Trust to reduce wasteful computation in open Desktop Grid Systems’, in *Proceedings of the 10th Annual International Conference on Privacy, Security and Trust*, IEEE, Jul. 2012, pp. 250–255, ISBN: 978-1-4673-2326-0.

[12] M. Pacher, C. Müller-Schloer, Y. Bernard and L. Klejnowski, ‘Social Awareness in Technical Systems’, in *The Computer After Me*, Imperial College Press / World Scientific Book, 2013.

[13] G. Anders, J.-P. Steghöfer, L. Klejnowski, M. Wissner, S. Hammer, F. Siefert, H. Seebach, Y. Bernard, W. Reif, E. André and C. Müller-Schloer, ‘Reference Architectures for Trustworthy Energy Management, Desktop Grid Computing Applications, and Ubiqitous Display Environments - Technical Report 2013-05,’ Universitätsbibliothek der Universität Augsburg, Universitätsstr. 22, 86159 Augsburg, Augsburg, Tech. Rep. March, 2013.

[14] L. Klejnowski, Y. Bernard, G. Anders, C. Müller-Schloer and W. Reif, ‘Trusted Community - A Trust-Based Multi-Agent Organisation for Open Systems’, in *Proceedings of the Fifth International Conference on Agents and Artificial Intelligence (ICAART)*, Barcelona, Catalonia, Spain, 2013.

[15] Y. Bernard, J. Kantert, L. Klejnowski, N. Schreiber and C. Müller-Schloer, ‘Application of learning to trust-adaptive agents’, in *Proceedings of the Seventh IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, IEEE, 2013.

[16] J. Kantert, Y. Bernard, L. Klejnowski and C. Müller-Schloer, ‘Interactive Graph View of Explicit Trusted Communities in an Open Trusted Desktop Grid System, 2013 Demo Entry’, in *Seventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2013.

[17] Y. Bernard, L. Klejnowski, D. Bluhm, J. Hähner and C. Müller-Schloer, ‘Self-organisation and Evolution for Trust-adaptive Grid Computing Agents’, in *Evolution, Complexity, and Artificial Life*, S. Cagnoni, Ed., Springer, 2014, ISBN: 978-3-642-37576-7.

[18] L. Klejnowski, S. Niemann, Y. Bernard and C. Müller-Schloer, ‘Using Trusted Communities to Improve the Speedup of Agents in a Desktop Grid System’, in *Proceedings of the Seventh International Symposium on Intelligent Distributed Computing (IDC’2013)*, F. Zavoral, J. J. Jung and C. Badica, Eds., vol. 511, Prague, Czech Republic: Springer International Publishing, 2014, ISBN: 978-3-319-01570-5.

[19] J. Kantert, Y. Bernard, L. Klejnowski and C. Müller-Schloer, ‘Estimation of reward and decision making for trust-adaptive agents in normative environments’, in *Architecture of Computing Systems - ARCS 2014*, E. Maehle, K. Römer, W. Karl and E. Tovar, Eds., vol. 8350, ser. Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 49–59, ISBN: 978-3-319-04890-1.

1 | Introduction

1.1 Motivation

In everyday life, we constantly interact with other members of our society. Cultural experience teaches us to be aware that our interaction partners may have motivations and interests about these interactions that are not necessarily compatible with what we expect. To experience negative interaction outcomes - be it deception, disappointment, fraud, exploitation or deceit - have taught us to carefully consider whom to commit to, when our welfare, freedom or integrity are at stake. On the other hand, we are usually not all paranoid and impute only bad intentions to our fellow society members. Neither do we control them all the time, trying to ensure that we really are not exposed to harm. These balanced considerations are based on the socio-cognitive concept we call *trust*. Though highly complex, trust assessment is so natural to us, that we are seldom consciously aware of the exact interpretations and assessments we carry out when deciding to trust someone. At the same time, it guides us convincingly and we never question the very fact that we need trust for a functional society. Research about the notion of trust, conducted in the fields sociology, psychology and economics, has provided us with many answers and theoretical frameworks about why we trust, whom we trust and how we trust (cf. e.g. [20]). Apart from the ongoing analysis of trust, research in this field has also lead to the idea of synthesis, of applying artificial trust systems in technological contexts (as part of *socionics*, cf. e.g. [21], [22]). This has resulted in research of computational trust models and Trust Management systems and their application in the domains of Multiagent Systems (cf. e.g. [23], [24], [25]), as well as Distributed Systems (cf. e.g. [26], [27]).

Especially the domain of Multiagent Systems has received considerable attention from trust research in the last few years. Multiagent systems offer two views on trust application: On the one hand, they allow to test trust theories from the traditional fields of research in controlled environments. Researchers here try and implement the models behind the theories, as accurately and with as much detail as possible, in order to test them in evaluations not feasible in real societies or with real individuals. The results are then fed back into the trust theories and used to extend and detail the according models. On the other hand, Multiagent Systems are a key technology in the design of (technical) open distributed systems (cf. e.g. [23]). These systems are characterised by the distribution of control among autonomous software entities (agents) that represent users in the system, are in general designed by different programmers, and can enter and leave the system arbitrarily. The users let their agents participate in the system with specific goals and their individual success is measured according to formal performance metrics. Examples of these systems can be found in domains such as e-commerce, vehicular networks and Desktop Grid Systems amongst others. In these systems, the application of trust serves a specific purpose: Agents shall be enabled to identify capable, credible, and reliable interaction partners. Obviously, a metaphor is used here, when stating that agents shall be enabled to trust each other. The idea here is to apply concepts that proved essential to the functioning of human societies to improve artificial societies. However, in contrast to the rather vague functional definition of a human society, an artificial agent society for a technical system has a precise and quantifiable performance definition. Accordingly, trust theories from other research fields are taken as inspiration, but not implemented in each detail. Instead, they are simplified and adapted to the specific technical context they are applied in. Research in these applications is then conducted with the aim to improve the performance of agents in such a technical, open distributed system.

In the literature, these two approaches to trust in Multiagent Systems are not clearly separated from each other (cf. e.g. [28]). Many elaborate trust models are for example theoretically valid, but not tested in concrete technical systems (cf. e.g. [29]). On the other hand, oversimplified trust models are applied in technical systems where they allow for good results as long as the system is predictable. If however the system changes too much during runtime, for example if new or different agents enter the system, the Trust Management decreases the performance of the system. This also applies to the problems of over-confidence on the one hand, and too much trust on the other hand: These phenomena make many Trust Management systems slow to react and sub-optimal. Additionally, Trust Management in technical systems has to be always treated as overhead. If no adversary or unreliable agent behaviour is expected, this overhead should be avoided. However, typical Trust Management systems are often not adequately flexible and the systems they are applied in are not optimised for this kind of differentiation.

This is where research from the field of *Organic Computing (OC)* (cf. [30]) can be applied: OC is a recently introduced paradigm related to the notions of biologically-inspired and autonomic computing. The key idea is to design decentralised control structures for complex systems. Here, the term *complex system* characterises a system in which heterogeneous elements show self-motivated, dynamic and stochastic interaction patterns, that can lead to undesired, emergent system states. Control of complex systems is a demanding goal, as these systems tend to change during runtime, express emergent properties and are thus only partially suited to be planned at design time. OC systems can be realised as Multiagent Systems and consequently endowed with trust-aware control and control based on self-X properties like self-organising, self-healing, self-protecting and others. Accordingly, control design by OC-technologies aims at producing mechanisms that adapt to the system state of the complex system they control at runtime. This has for example been demonstrated with a system of decentralised traffic control (cf. [31]) capable of adapting to emergent traffic states like congestion or the failure of control components, and improving the throughput of the traffic system in comparison to traditional control mechanisms. In this thesis, the work on a Multiagent organisation that applies Trust Management and is self-organised using methods from Organic Computing, is presented. By applying Organic Computing techniques in this context, it can account for the dynamics of open distributed systems and counter the problems many Trust Management systems have in this domain.

1.2 Problem Statement and Contribution

In this thesis, a self-organised Multiagent organisation based on Trust Management and Organic Computing techniques for open distributed, technical systems, is presented. The aim of this self-organisation is (1) to allow agents to increase their utility (the technical performance they experience in these systems) and (2) to increase the robustness of these systems against disturbances. This is achieved by enabling the agents to form and operate within the organisation *Trusted Community*.

Open distributed systems based on agents are a challenging domain. Agents, as they are understood in this context, are software elements. As such, agent code originates from unknown sources, agents are autonomous, agents are blackboxes and most-importantly, agents are self-interested. Additionally, the agent society in a system is dynamic and the system composed of these agents can change at runtime. In order to control these complex systems, in terms of allowing for performance, fairness and robustness among the participants, decentralised control mechanisms have many advantages. This is mainly due to the fact, that centralised systems rely on entities that are potential single points of failure (compare for example the argumentation for peer-to-peer systems). The task is thus, to distribute the control of the system among the agents. Here, three main approaches can be identified in the research literature:

- Control by Agent Organisations (cf. e.g. [32], [33], [34])
- Control by Trust Management (cf. e.g. [20], [35], [36], [23])
- Control by Organic Computing (cf. e.g. [30], [37], [38])

Control by Agent Organisations follows the argumentation, that while distributing control among agents often leads to high-performance systems, it is still limited because of the restrictions on agent reliability mentioned above. One way to cope with these challenging conditions is to introduce subsystems composed of groups of agents. Again, a metaphor from human society is used here: These groups of agents are referred to as organisations. Just like in human societies, organisations can have different structures and properties, especially with regard to hierarchy, composition criteria, goals, lifecycle and formation criteria. When specifically designed for a target system, an agent organisation can then allow a different form of control: Enabling member cooperation, aggregating local views to system models, motivating information exchange and other organisational benefits, are strong incentives for agents to become members of this organisation and then coordinate and cooperate. This in turn leads to systems where agents experience a good performance. Additionally the partitioning of the system into organisational subsystems can increase the overall robustness of the system.

On the other hand, the challenges in open distributed systems are often addressed with **control by Trust Management**. By letting agents assess each others trustworthiness, based on interaction experiences and reputation, firstly the uncertainty introduced by autonomous and blackbox-behaviour is reduced. This especially refers to uncertainty with respect to the willingness and competence of potential interaction partners. Secondly, Trust Management can serve as an incentive for agents to change their behaviour to a more cooperative scheme, as usually agents will utilise reciprocity when choosing interaction partners. Overall, to apply Trust Management often leads to increased agent performances and a certain degree of robustness. In fact, as often stated in the literature, Trust Management is seen as fundamental to functioning Multiagent-based open distributed systems (cf. e.g. [20], [25]).

Finally, **control by Organic Computing** is a recent approach to the decentralised control of complex systems, such as open distributed systems. It is focussed on the assumption, that control of complex systems cannot be planned comprehensively at design-time because of emergent processes among the elements, which create unforeseen system states, as well as vast configuration spaces, which do not allow to calibrate these elements efficiently off-line. Control of these systems has therefore, at least in parts, be transferred to the runtime of the system. To allow for this, elements of such a system under observation and control are enabled with the ability to firstly gather and analyse information about themselves, other elements and the system as such, and secondly to adapt their control of the system based on these observations. This allows the control to react to emergent changes in the system, as well as optimise the control of the system based on real data gathered at runtime. It has been shown, that the approach of OC blends well with open distributed Multiagent Systems (cf. e.g. [39]), as these are indeed complex systems with emergent processes, and as agents possess all necessary properties to implement the observation and control model required by OC. This allows for Multiagent Systems based on OC technology and leads to systems that are usually more robust and induce a higher agent performance.

In this thesis, it is argued that **these three approaches can be combined, and that this combination results in an improvement of the performance and robustness of technical, open distributed systems beyond what is currently possible.** On the one hand, agent organisations need some form of member control and counter mechanisms for adversary and uncooperative agent behaviour. This is where Trust Management can improve agent organisations: By incorporating Trust Management, agent organisations are equipped with additional mechanisms for formation, composition optimisation, sanctioning the behaviour of members and classification of non-members. On the other hand, Trust Management can be a serious overhead in a system. Take for example agents that do not stop to evaluate each others' trustworthiness at each interaction, despite having only good experiences with the respective interaction partners (referred to as *too much trust*, cf. [20]). Or agents counting on the slowness of the Trust Management system (referred to as *over-confidence*, cf. [20]) and adapting strategically to their trustworthiness assessment by repeatedly being cooperative only as long as they have not enough reputation, and then show uncooperative behaviour. Additionally, systems with control built solely on the requirement of an operating Trust Management system are vulnerable to system states, where the Trust Management fails or shows unexpected results: Compare, for example, the case of subsequent trust crises (agents loose their previously established trust in another agent , cf. [40]) among agents, which can lead to a breakdown of the Trust Management system in an emergent process. In such a state, agents cannot rely on their trustworthiness assessments any more and the interactions among them are likely to collapse (cf. e.g. [29], [41]). This is where agent organisations in combination with Organic Computing technology can improve systems with Trust Management: Agent organisations can reduce the dependability of Trust Management in open systems, by creating subsystems where the members are not affected by issues in Trust

Management. This is possible where especially trustworthy agents form an organisation and stop utilising Trust Management for normal interactions due to overhead minimisation concerns. Instead, Trust Management is only utilised for aspects regarding the management of the agent organisation, as mentioned above. In such organisations, the members are also less affected by emergent processes regarding the Trust Management in the system. However, such organisations cannot be entirely defined statically at the design-time of the system. Emergent processes and dynamics in the agent society demand not only adaptivity from single agents, but even more so from agent organisation. In order to account for the characteristics of such complex systems, and to sustain successful operation of agent organisations, self-organisation and adaptivity need to be applied. Here, Organic Computing provides methods and design patterns to detect emergent phenomena in these systems and allow for adaptive compensation or where impossible, at least graceful degradation. When applied in the context of trust-aware agent organisations, these techniques provide tools for agents to manage the organisations, such that the operation can continue even in unusual or abnormal system states.

In summary, this thesis introduces a novel agent organisation - Trusted Community - for technical, open distributed systems, that are agent-based and provide a Trust Management system. By combining techniques from research on agent organisations, Trust Management and Organic Computing, this novel organisation form allows for an increased agent performance and an operation that is sustained even in system states that arose from emergent processes, not predicted at design-time. As a result, the robustness of these systems is increased.

1.3 Overview of the Thesis

This thesis is organised as follows: The following Chapter 2 presents work related to the research presented in this thesis. This is introduced with an overview of research on open distributed systems and Multiagent Systems in general and continued with a summary of work on agent organisations, Trust Management and Organic Computing. This chapter is then concluded with a review on approaches similar to the Trusted Community concept.

Chapter 3 is divided in two main parts: In the first part, the system view applied in this thesis is detailed. This is composed of a structural model for systems Trusted Communities can be applied in, as well as models for the requirements on the agents and the Trust Management system in the target system. The second part discusses the application of related work in the target systems, gives a coarse-grained introduction to the Trusted Community concept and analyses the applicability of Trusted Communities in the target system. It concludes with a discussion of the limits of the applicability.

Chapter 4 is the main body of the thesis and presents the novel Multiagent organisation Trusted Community from various views: The chapter starts with a short introduction and a formal representation of the Trusted Community concept. Then the lifecycle and structural, as well as behavioural properties of this organisation are presented, both from the view of a member agent and a managing entity. The chapter then concludes with a discussion of the configuration for this agent organisation.

Chapter 5 presents the evaluation of the Trusted Community concept in an exemplary technical, open distributed system, the Trusted Desktop Grid (TDG). This system is an open MAS realisation of a Desktop Grid System and is perfectly suited to demonstrate the benefits of the Trusted Community approach, as it incorporates complex agent behaviour, selfish motivations and threats to single agents as well as the system as such. The TDG system is first introduced and the applied performance metrics are discussed. In the following, the application of a Trust Management system to improve the Desktop Grid system is presented and the results are discussed. This is concluded by a motivation for the application of Trusted Communities in this scenario and the presentation of the experimental results for the application. These results show that the application of TCs in open distributed systems substantially increases the performance and robustness of these systems under a variety of conditions. Finally, this chapter discusses the motivation to apply this agent organisation in two additional open distributed systems and conceptually defines how this application is to be laid out, in terms of Trusted Community configuration and limitations.

Chapter 6 is the final chapter and concludes this thesis with a summary of the work conducted here and an outlook on further research opportunities based on this work.

2 | Related Work

As motivated in the previous chapter, the approach presented in this thesis has been developed by using technologies from the fields *Multiagent Systems*, as well as *Organic Computing*. In this chapter, these fields are first briefly reviewed. The summary of Multiagent Systems focusses on the application of computational trust and organisations for the control of distributed systems. For both sub-fields, the original application domains in the social and economic sciences are referred, and the development of their application as computational models is highlighted. This is followed by a brief summary of classifications for the research literature in both fields, and concluded with references to the application in technical systems. The summary of Organic Computing first explains the underlying paradigm and then introduces associated techniques and architectures as, for the example, the generic Observer/Controller architecture. In conclusion, applications of OC techniques are referred and an in-depth summary for an exemplary project is provided.

After these summaries, related work that combines control paradigms from these fields, e.g. trust-based multiagent organisations, is referenced, summarised and examined in greater detail. This includes the utilisation of such approaches for the control of Desktop Grid Systems, a system class which is is used for the evaluation in this thesis. The chapter is concluded with a review and coarse classification of related approaches, as well as with a discussion of possible contributions.

2.1 Multiagent-based Open Distributed Systems

This thesis presents an approach to improve the control of (technical) *Open Distributed Systems* such that the participating entities benefit from a higher performance and robustness towards disturbances. In this context, *open* means that the system can be joined and left by entities at any time, and that the exact realisation of these entities is not known to the system designer. *Distributed* on the other hand means that the entities in the system are in general at spatially varying locations and must be connected through a network in order to interact. This *Open Distributed System* model is very generic and applies to many systems that surround us nowadays (cf. e.g. [35]), such as the Internet, most e-commerce systems, the *Internet of Things* (cf. e.g. [42]), Peer-to-Peer Systems, Desktop Grid Systems. In the recent decades, many researchers have taken the point that such systems are complex, dynamic and challenging and that this requires autonomous components that act, and interact, on behalf of the users. It is therefore argued to model such systems as *Multiagent Systems (MAS)* (cf. e.g. [43], [44], [45], [46]). MAS are systems comprised of a population of software entities (*agents*), participating as user representatives and having the ability to self-responsibly achieve goals set by the users (*delegation*, cf. e.g. [47]). The most outstanding characteristics of agents are their *autonomy* (esp. in reasoning about how to achieve these goals) and their *ability to interact with each* *other* according to clearly defined protocols (cf. e.g. [48], [49], [44]). A vast body of research on MAS has been created in the past decades, comprising such diverse fields as for example:

- *Agent Reasoning* and *machine learning* (cf. e.g. *Learning-Classifier Systems*: [50]),
- *Agent Cooperation*, *Collaboration* and *Coordination* (cf. e.g. *contract net protocol*: [51]),
- *Agent Organisations* (cf. e.g. *congregations*: [52]),
- *Agent Communication* (cf. e.g. *knowledge query and manipulation language*: [53]),
- *Agent Models* (cf. e.g. *Belief Desire Intention model*: [54])
- *Agent Norms* (cf. e.g *bottom-up and top-down norms*: [55])
- *Agent-Oriented Software Design (AOSE)* (cf. e.g *The Gaia Methodology*: [56])

The design of a MAS-based Open Distributed Systems is a challenging problem as agents cannot be controlled directly to ensure the desired global behaviour of the system (cf. e.g. [57], [58]). Instead, control relies on self-organised and collective actions of the agents (cf. e.g. [59]). This is further complicated as in open systems agents are self-interested, internal agent states are not accessible, agent code stems from different stakeholders, agents are unknown at design time and the population of agents in the system is dynamic at run time (cf. e.g. [60], [61], [62]). What is more, agents can follow uncooperative or even adversary strategies due to their programming or due to autonomous strategy explorations: Agents can exploit others (cf. e.g. *free-riding*: [63], [64]), or even damage the entire system (esp. by disrupting other participants, cf. e.g. [65] [66]). The control of an Open MAS under these circumstances has been approached in the research community from various directions. The most prominent are the incorporation of *(computational) trust and reputation* into agent reasoning (cf. e.g. [23], [25]), and the design of *organisations* (cf. e.g. [67], [34]) for the enforcement of design goals. In the following, the related work in these two research directions is summarised in more detail.

2.2 Trust in Multiagent Systems

The phenomenon *trust* has been researched in such diverse fields as sociology, biology, psychology, economics and computer science, leading to a great number of definitions, formalisations and models. In fact, in [20] the authors provide an analysis of 72 definitions of the notion *trust* that were published in the literature from 1960 to 1999. However, there is a common understanding that trust is the *subjective* expectation of a *trustor* that a *trustee* will perform a certain *action*. This prediction is based on the assessment of *competence*, *willingness* and *risk*. Also, it is generally agreed that in its consequence, trust is a fundamental prerequisite for cooperation, while strong distrust hinders cooperation (cf. e.g. [68]). Research on the socio-cognitive phenomenon trust finally found its way into computation around the year 1980: As detailed in cf. e.g. [21] and [22], computer scientist on the search for new approaches for distributed problem solving constituted a research field that was later termed *socionics*. In *socionics*, research results on sociological phenomena are adopted for the design of distributed and intelligent systems based on artificial intelligence and software agents. With S. Marsh introducing the first major contribution to a formalisation of trust as a computational concept in his Ph.D. thesis in 1994 (cf. [69]), the incorporation of trust into computation found many followers.

Marsh proposed a model of trust that was implementable, and showed in an experimental evaluation using the Prisoner's Dilemma that agents equipped with such an implementation are able to make decisions based on what we understand as trust. In the following years, these ideas were picked up by many scientists and a vibrant community of computational trust researchers arose. Consider for example [23], [70] and [24] for extensive surveys on the application of computational trust in MAS. Contributions in this field are often classified according to *trust evaluation* vs. *trust-aware decision-making* approaches (cf. e.g. [24]), though *trust-aware decision-making* is sometimes seen as integral part of *trust evaluation* (cf. e.g. [25]).

Trust evaluation has the goal to allow agents to assess each others' trustworthiness in a quantified measure resulting in a time-dependent and subjective trust(worthiness) value. This is achieved by formal *trust models* (also referred to as *trust metrics*, cf. e.g. [23]) that capture the interaction experiences of agents with each other. Such models are most often based on the rating of *direct* observations, *indirect* observations (also referred to as *reputation*), or a combination of both. In general, each new observation influences the trust value of a trustee and consequently allows to identify agents that act uncooperatively towards a trustor agent. The characteristics of trust models can be best illustrated by regarding one of the most cited trust models in the literature: *REGRET* (cf. [28]) is a very elaborate trust and reputation model that aggregates many state-of-the-art properties of trust models (like direct and indirect experiences, credibility, reliability) and extends them by incorporating additional dimensions (social and ontological). In the resulting multi-facet model, the primary source for trust information are direct experiences. For a more complete estimation of a trustees' trustworthiness, This information is extended by the consideration of reputation information. Herein lies the contribution of this approach: Not only is the reputation information processed dependent on the reliability of the reputation providers, but also based on the social group of the agent. Additionally, reputation is seen as linked to a specific context (or aspect). An agent in an e-commerce system can for example have a reputation as a seller of goods, as well as a reputation as a provider of goods with a high product quality. In *REGRET* these aspects are related by ontologies, such that an aggregated reputation value can be estimated. For an extensive and specific overview over trust models, their classification and contributions in the literature, see the surveys presented in [25] and [71]. Besides the characteristics of the trust models themselves, contributions in this field also address questions about the robustness of trust models (cf. e.g. [72]), the dependence on security mechanisms (cf. e.g. [20]), the association to norms (cf. e.g. [73]), or the *alignment* of trust values obtained by different trust models (cf. e.g. [74]).

While the focus of approaches in the *trust-evaluation* field is the modelling and assessment of trustworthiness relationships, approaches in the field of *trust-aware decision-making* aim at the utilisation of this information for the reasoning of agents. Ultimately, these approaches revolve around the decision which interaction partner to choose from a set of trustees (cf. e.g. [24]). This decision is highly relevant for the control of open MAS as described above: The presence of self-interested, exploitative and adversary behaviours in such systems generates a high amount of uncertainty among the agents (cf. e.g. [35]). Agents are put at great risk of decreasing their utility when delegating tasks to such uncooperative agents ([20], [25]). Hence a discrimination of uncooperative agents is needed to allow agents to maximise their utility by choosing to delegate tasks only to interaction partners that provide the negotiated interaction results (cf. e.g. [29]). The choice of interaction partners based on trust criteria can also be seen as an optimisation problem: A trustor agent can exploit the knowledge about highly trustworthy agents or it can explore by interacting with agents that it cannot yet assess with respect to trustworthiness (cf. e.g. [24]). The former *greedy* approach is more intuitive as agents trying to maximise their utility are expected to choose interaction partners that can most reliably provide this utility. This approach is therefore prevalent in the literature (cf. e.g. [69] as example). However, due to the dynamics in such a system, the greedy restriction to the most trustworthy agents can have disadvantages for these agents: The number of interaction requests can become so high that these agents cannot continue to provide a high service quality (cf. e.g. [24]). Besides, the dynamics of the system with new agents entering a system and agents changing their behaviour provide a strong motivation to explore these new agents in order to find agents that are even more trustworthy than the known agents (cf. e.g. [75]). In [76] and [77] for example, the author propose the utilisation of *stereotypes* in order to minimise the risk of exploration in case of new agents in the system. *Stereotypes* allow to generalise experiences with agents to new agents based on typical behaviours. This work is classified as contribution to the *trust evaluation* field discussed above, as the authors do not explicitly evaluate the application of *stereotypes* to actual decision-making. However, they motivate such application and address it as future work. Another example is the work presented in [78]: Here the authors apply a machine-learning approach to the decision-making of agents. The agents use either exploration or exploitation for each interaction decision and the received utility gain from this decision is used to reinforce a Bayesian model. Apart from the value of trust-aware decision-making for the performance of single agents, there is also a strong interest about the regulatory effects of trust management in such systems. For example [66] have examined the isolation of adversary or unreliable agents, also referred to as *soft security* approach. Isolation here means that uncooperative agents are made known to the agent society (esp. via reputation). This allows cooperative agents to avoid interactions with these agents without having to first make unsuccessful direct experiences with them. Also, trust-aware decision-making is shown to enforce the cooperation in a system: If a for example a high reputation is required to allow for certain types of interactions or certain interaction partners, rational agents are expected to cooperate in order to achieve a high reputation. This is where trust and reputation are used as an incentive (cf. e.g. [60], [63], [79], [80], [46]). Furthermore, it is examined how trust management or the dependence on trust can have a negative effect in the MAS. For example in [40] the emergence of *trust earthquakes* is examined, a phenomenon that leads to distrust among agents due to reputation effects. Also, in [29] the *paralysis* of agents dependent on trust is explored: Due to the reliance on trustworthy interaction partners and the temporary lack of these, agents seize to interact with each other, causing more damage than the cooperation with seemingly untrustworthy agents. Additionally, the authors in [24] examine the *reputation damage problem*: Agents that are cooperative develop a high reputation and are increasingly sought as interaction partners as a consequence. This can lead to a decreased capability of the agent to process the requests (for example due to overload). In effect the agent fails to provide positive interaction experiences for the many requesters attracted by its reputation and hence receives many negative ratings, damaging its reputation.

Most of the approaches to computational trust in the literature use theoretic (as opposed to technical) models of agent populations, comprised of agents with specific traits (e.g. defecting, cooperating, random behaviour), as base for their evaluations. Consider for example how the authors in [29] use a pool of agents with randomly assigned characteristics and unspecified interactions to evaluate their trust decision-making approach. Additionally, consider the *ART* approach of a domain-independent testbed for trust and reputation models presented in [81]. Though such theoretic models have a high importance in the development of computation trust approaches, these are really put at test in actual technical domains. As the focus in this thesis is on technical systems, it is especially interesting to see where computational trust has been used to enhance agents in concrete instances of these systems. Examples of this are found in the following examples in the literature:

- *Desktop Grid Systems* (cf. e.g. [82], [83], [84], [85], [86], [87], [88], [89], [11]):
  Here computational trust is mostly used to help agents find reliable workers for the processing of their tasks, as well as identify free-riders and clients that return invalid processing results. The evaluation of the approach presented in this thesis is from this domain, hence the related work is discussed in more detail in the following.

- *Grid Computing Systems* (cf. e.g. [90], [91], [92], [93]):
  More general than Desktop Grid Systems, these type of systems are mainly examined with respect to the benefits of trust and agent application in the security of such systems. Also, approaches aim at improving the establishment and operation of virtual organisations that group resources from various owners.

- *Decentralised Power Grid Systems* (cf. e.g. [94], [95], [96], [97]):
  An only recently established, but highly active field of research in which the possibilities of the representation of energy producers and consumers by agents is explored. Trust is used here to enhance decision-making based on e.g. the reliability of predictions on energy production and usage.

- *Vehicular Ad-Hoc Networks (VANETs)* (cf. e.g. [98], [36]):
  VANETs are an interesting domain gaining increased attention in the recent years. Trust-aware agents are applied here for example to provide reliable information on traffic congestions or routing information. This domain has however some unique aspects that put such traditional approaches at great stress: Due to the high mobility of the vehicles and their high number, trust-relationships are seldom long-term. Instead, approaches must allow for rapid and highly adaptive discrimination of trustworthy agents based on few and short-lived interaction experiences.

- *Wireless sensor networks* (cf. e.g. [99], [100]):
  In sensor networks the autonomous components (the sensor nodes) often produce unreliable data, e.g. due to environmental conditions. Trust and reputation models are used here to classify data providers and adapt to changing data quality. However, this is clearly a challenging application domain for trust-aware agents, as the resources on sensor nodes are far more limited than in other domains.

- *Peer-to-Peer networks in general* (cf. e.g. [101], [102], [103], [104], [105], [106]):
  Peer-to-Peer networks are the most commonly used network infrastructure for technical open distributed systems. Trust and reputation can be applied already on this layer of the systems: Agents need to for example assess the trustworthiness of peer recommenders and service providers, identify colluding agents and peers that tamper with or reject to support the routing functionality of such a system.

- *E-Commerce Systems* (cf. e.g. [107], [108], [109], [110], [111]):
  In e-commerce systems, traditionally the modelling of the trustworthiness of stakeholders like sellers, buyers, service providers etc. is of great interest. With agent technology being increasingly applied for automated interactions such as transfers, the reasoning of these agents was enhanced by the incorporation of computational trust and reputation to improve the agents' performance on the electronic markets.

This brief overview illustrates that the literature on computational trust provides many approaches to the control of Open Distributed Systems. Formalisations of trust in implementable models and especially their application in trust-aware decision-making provides a system designer with tools to incentivise cooperation and isolate exploiting or adversary agents. Additionally, system users being represented by an agent benefit from the agents' ability to increase its utility by incorporating trust-based reasoning. In summary, most authors in the research community agree that computational trust is a key ingredient in the design of MAS (cf. e.g. [46]) and Open Distributed Systems (cf. e.g. [35]). However, only a small number of the referenced contributions is actually designed for the application in technical system (with the exception of the above referenced approaches). Instead most approaches are evaluated in theoretic scenarios. In addition, most of the referenced approaches to trust-aware decision-making assume that trust information is reliable and constantly available. The consideration of trust management breakdowns or emergent effects in trust dynamics is mostly neglected.

2.3 Multiagent Organisations

As pointed out in the introduction, this thesis presents an approach that combines methods from computational trust, MAS organisations and Organic Computing to allow for the control of technical Open Distributed Systems. This section summarises related work in the field of MAS organisations.

Much like the phenomenon of trust, *organisations* are ubiquitous in everyday life - the company we are employed in, the government of the country we live in, and the shop we buy our supplies at are all a form of organisation. Another similarity with the trust notion is the lack of consensus about an all-encompassing definition of *organisations*. A rather generic example of an attempt is made by the author in [112] who defines an organisation as providing "*[..] a framework for activity through the definition of roles, behavioural expectations and authority relationships (e.g. control)*". Here, the basic concepts of division of labour (and specialisation) common to most organisations are paraphrased by their implementation (roles) and control (through authorities and according to known expectations). Another exemplary definition proposed in [113] is: "*An organisation can be defined as an arrangement of relationships between components or individuals which produces a unit, or system, endowed with qualities not apprehended at the level of the components or individuals. The organisation links, in an interrelational manner, diverse elements or events or individuals, which thenceforth become the components of a whole. It ensures a relatively high degree of interdependence and reliability, thus providing the system with the possibility of lasting for a certain length of time, despite chance disruptions.*". Here the motivation for an organisation is the main focus. Due to the "arrangement of relationships" an organisation becomes more than the sum of its parts (the elements belonging to the organisation). An organisation hence allows to overcome the limitations of single elements. Research in the field of economics, psychology, sociology etc. has provided theories of how organisations must be structured, operate and adapt to be efficient. In the recent decades, this research has profited from the rise of a new field, the *computational organisation theory*. In the early survey presented in [114], computational organisation theory is yet described solely as means to gain insight about forms and functions of human organisations by artificial and executable models of organisations. The findings from the research of such models are described as being fed back into economic theories on human organisations. The necessity for this direction is motivated by the complexity of real organisational systems, being subject to nonlinear dynamics, complex interactions and heterogeneity and eluding traditional analysis methods. This motivation has been stated early on for approaches in the entire field of MAS, e.g. by sociologists modelling human interactions with artificial agents and evaluating their behaviours in situations that are not feasible to evaluate in real societies. This has advantages over purely analytical models, especially because it can lead to tractable claims about real systems produced by the execution of these models. However, the reduction of MAS to an analysis tool for the study of real systems has been overcome in the recent decades. Today, it is a generally accepted view that MAS in general, as well as computational organisations in particular, have a great value beyond modelling systems in economics and sociology. Instead, MAS organisation technology is applied as design paradigm for distributed systems, applicable in its own right, and allowing to improve technical systems (cf. e.g. [67]). Here, human organisation theory merely provides templates and patterns for artificial organisations, inspired by, but not restricted to, human forms of organisations[1].

MAS organisations have been extensively examined by the research community. In a survey paper presented by B. Horling and V. Lesser (cf. [32]) these organisation approaches are classified according to common properties such as *persistence (short- vs. long-term)*, *purpose (goal- vs. utility-driven)*, and *hierarchy*. For each organisation type, the authors examine the key characteristics, as well as the formation process, and reference related work. The following summary is a short overview about the most relevant forms of MAS organisations for the work in this thesis:

- *Society*: This type of organisation is the most general. As discussed earlier, a MAS is essentially composed as an *agent society*. Such a form of organisation constitutes an open system and provides rules for agent interactions (e.g. by a formulation of *norms*). Societies are long-term organisations and can have any goal or hierarchy structure, as they allow the agents to form sub-organisations.

- *Hierarchy*: The second general-purpose form of agent organisation is the hierarchy, being an integral part of any other organisation structure. Hierarchies allow for the specification of information and control flows and the implementation of task decomposition and *divide and conquer* solutions. Additionally, *holarchies* can be understood as a specific form of hierarchies. Holarchies describe self-similar *systems of systems* that are structured as levels with characteristics that cannot be attributed to the comprising elements (subsystems) alone.

- *Team*: The origin of distributed problem solving in MAS has inspired the definition of agent teams. A team is a specific unit used to coordinate agents by assigning roles. The members of the team are in general not primarily self-interested, instead the team is formed to reach a common goal.

[1] In the remainder of this thesis, the term MAS organisation always refers to this synthesis approach.

- *Coalition*: This organisation is goal-directed and exists mainly as short-term structure to reach a given goal. Once this goal is reached, a coalition is disbanded. The members of a coalition are organised mostly as peers without hierarchy. Agents join coalitions to increase their benefit or decrease their costs, albeit in conformity to the coalition goal. Coalitions are very common in the literature, especially due to high interest they raised in the game theory community.
- *Congregation*: Unlike coalitions and teams, congregations are formed as long-lived units of self-interested agents. A common goal does not exists, instead the agents join the organisation to increase their personal utility. Congregations have typically a flat hierarchy and attract agents with either homogeneous or complementary capabilities.
- *Federation*: The basic concept behind this organisation is the delegation of control to a single member (higher hierarchy) and the associated partial abdication of autonomy. Such a designated member then acts as representative of the group. This allows other components to interact with the federation as a *holon* by using the representative as interface.

In addition, the authors in [115] provide a more formal analysis of MAS organisations based on three dimensions of the organisation structure: *Power*, *coordination* and *control*. Obviously, MAS organisations are inspired by organisation theory in economics. But what is the benefit of their application in a technical context? Here, especially the research community for AOSE provides answers: For example in [34], the authors refer to the evaluation of drawbacks of traditional MAS (without organisations) and propose the development of an organisation-centric engineering approach for MAS as solution. Specifically, the following statements published by N. Jennings in [116] are used as motivation: The engineering of large agent-based systems is impaired because "*the patterns and interactions are inherently unpredictable*", and because "*predicting the behaviour of the overall system based on its constituent components is extremely difficult (sometimes impossible) because of the strong possibility of emergent behaviour*"[2]. Due to their ability to reduce complexity by decomposition and coupling, MAS organisation as software pattern are then seen as a natural solution to this control problem (cf. e.g. [34]). In [67], the authors follow a similar argumentation: The utilisation of organisations in system design enforces the definition of roles and associated interactions for the agents in a system. Such well-defined interactions are then claimed to be more task-related and lose their property of being unpredictable interdependencies of system components. This is again a reduction in complexity of a system and allows for the easier design of its control. In addition, the authors also point out that the utilisation of MAS organisations can be a benefit in systems where these represent actual human organisations. Such supporting systems are easier to design when the actual work- or control flows of these organisations can be mapped directly via MAS organisations in the supporting system. However, the application of MAS organisations is not limited to the design phase of a MAS covered by these software-engineering argumentations. In fact, a purely static design approach of MAS organisations lacks the flexibility to be applicable in open MAS (cf. e.g. [117], [52]). Consider for example the argumentation in [33], where the authors make a strong case for using approaches of self-organisation in MAS, as opposed to static design time organisation structures. Their argumentation is mainly focussed on the difficulty of anticipating run time situations in which e.g. service providers become bottlenecks, new agents are not explored enough, or agents

[2]This understanding is shared with the Organic Computing community, albeit for technical systems in general, hence not reduced to agent-based systems (as discussed in Sec. 2.4).

suffer from the lack of capability adaptation. Instead, they argue to equip agents with the ability to increase their autonomy such that they can adapt to changes at runtime. A similar argumentation is found in [60]: Here, the authors summarise the literature and find that the enforcement of system goals in Open MAS is often realised by organisation approaches. They note however that these are design time approaches that produce fixed rules. Due to the autonomy of the agents, these rules may be circumvented and additionally, these rules are not flexible enough to allow for appropriate reactions to unforeseen system states. They also propose a more adaptive approach to the organisation. With the increased interest in open MAS, this type of argumentation has also gained in prominence. As a result, a great number of MAS organisation approaches in the literature incorporate some form of self-organisation and adaptivity to account for the dynamics in open MAS (cf. e.g. [118], [117], [52]). These approaches can be further classified: In [119], the authors categorise organisational mechanisms as being either *informative* or *regulative* mechanisms. The former are used to improve the decision-making of agents by providing them with more information than they can locally perceive. Such information can for example be the reputation value of another agent or information about consequences of its own interactions. Informative mechanisms aim at improving a system from the local (agent) level and thus do not require the formulation of a global utility function. In contrast, regulative mechanisms are used to enforce the global behaviour of a system. For this they require on the one hand the formulation of regulatory preferences, e.g. (top-down) *norms* defined by a system designer. On the other hand, they must allow for the actual regulation according to these preferences requires an institutionalised organisation with the capability to change agent behaviours either via incentives or via explicit control actions that alter the capabilities of agents. The authors then claim that all organisation approaches in the literature can be classified according to these dimensions and that the regulative mechanisms are more common. In addition, they contribute to the state of the art by presenting a formal model of MAS organisations that incorporates these mechanism definitions. An example for a combined approach using both mechanisms (on agent and system level) is proposed in [120]: Here organisation structures are adapted on both levels to allow for an increased robustness of a dynamic system towards undesired behaviour. The main focus is on the realisation of a respective decentralised monitoring scheme in the form of an organisation.

Finally, it is generally agreed that although MAS organisations can significantly improve the performance of a system, there is no universal form of organisation that would allow this in any system (cf. e.g. [32]). Instead, a part of the research literature is dedicated to the evaluation of applicability constraints, as for example the existence of group goals (allowing the formation of e.g. coalitions). Consider for example the work on an evaluation of the dependence between organisation structures and their performance regarding different aspects, as presented in [121].

In addition to the concept of a Multiagent organisation reviewed here, relatively recently a similar concept termed *Virtual Organisation (VO)* has been established in parallel by the research community. Again, definitions are numerous and ambiguous. This is further complicated by the fact that the term is used for human forms of organisation (sometimes also referred to as *Virtual Corporations*), as well as independently for MAS organisations. The equivocality of the term *agent*, which can refer to a human agent or a software agent, as well as the fact that sometimes systems with mixed types of agents are examined further adds to the confusion. As for human-centric VOs, the authors in [122] suggest the following definition of VOs: "*A virtual organization is primarily characterized as be-*

ing a network of independent, geographically dispersed organizations with a partial mission overlap. Within the network, all partners provide their own core competencies and the cooperation is based on semi-stable relations. The products and services provided by a virtual organization are dependent on innovation and are strongly customer-based.". The emphasis is here on the aggregation of other human organisations, hence a holistic management structure. This view is shared in e.g. [123], where VOs are defined as being "*composed of a number of individuals, departments or organisations each of which has a range of capabilities and resources at their disposal. These VOs are formed so that resources may be pooled and services combined with a view to exploiting a perceived market niche.*". The authors also give an example for a VO: "*[..] suppose that two relatively small airline companies with complementary routes agree to cooperate and coordinate their services so that they may offer flights, as a coalition, between a wider range of destinations, with a view to becoming more competitive in this market.*". Further definitions of human-centric VOs and their comparison are for example provided in [124]. In MAS, a VO is referred to as an "empty" agent organisations "*[..] that has a fixed purpose (e.g. to provide a set of services) but a potentially transient shape and membership*" because it "*[..] separates form and function [..]*" (cf. [32]). The authors especially emphasize the customer-oriented service purpose of VOs that is shared by most definitions of human-centric VOs. Based on this characteristic, they refer to VOs as resembling MAS coalitions and congregations in a classification attempt regarding other MAS organisations. This view is also adopted in [125], where a VO is defined as "*[..] the aggregation of autonomous and independent organisations connected through a network and brought together to deliver a product or service in response to a customer need*". For a more formal examination of VOs in MAS consider for example [126].

Due to the varying meanings, research literature on Virtual Organisations is found in the economics as well as in computer science. In practice, modern corporations have steadily increased their activities in the Internet since the early 90s. In the course of this progression, many systems have emerged that represent human or organisational stakeholder with autonomous agents and provide a common ground for interactions between humans and software agents (consider for example systems for trade in the stock market). As a consequence, VOs are often used as conceptual, as well as technical basis for such systems. Consider for example the application of VOs in Grid systems as presented in e.g. [127], [128], [129], and [92]).

In summary, computational organisation theory has been developed well beyond its original purpose as modelling tool for the use in economics. Today, MAS organisations are seen as fundamental tool for the development of controllable open MAS. Through the application of design time as well as run time approaches, agent interactions are specified and agents are enabled to overcome the limitations of their local capabilities. This however requires adaptive approaches that allow for self-organisation. In addition, VOs have gained popularity as an interfacing concept between human and software agents that interact in a commonly shared system.

2.4 Organic Computing Systems

So far, the focus in the review of related work has been on Multiagent-based Open Distributed Systems. In these systems, the components of a system are agents and mostly represent users (or groups of users), are autonomous and can interact in a shared environment, the MAS. However, the application of agent technology to control distributed systems is only one particular approach, albeit one with the largest research community. In the recent decades, a new research direction called *Organic Computing (OC)* (cf. e.g. [30], [37], [130]) has emerged. Here, a holistic view on distributed systems is in the focus: Apart from their inherent complexity, such systems are increasingly operated in environments where other complex systems exist. Consider for example the increasing number of smart phones, ambient intelligence devices, navigation systems, intelligent automotive systems etc. that are connected to the internet as well as to local networks. Such ubiquity often leads to interferences, especially where such systems are highly interconnected. Due to the complexity, often emergence occurs and these systems can enter operational states that they have not been designed for. This is seen as a problem that cannot be solved by the improvement of design methodologies alone: It is already challenging to design the control for a complex system, but the difficulties are exponentiated if interconnections to other systems must be anticipated. The same applies to the extrinsic control of such systems at run time. Here, the OC proposes a paradigm shift: Instead of attempting to refine design and control mechanisms to account for more configurations and environmental dynamics, complex systems should be designed such that they autonomously become aware of the need to adapt their control structures and execute this adaptation in an internal self-organised manner. This approach is also referred to as *controlled self-organisation* and shares some common views with the *Autonomic Computing* concept (cf. e.g. [131]) developed by IBM (for a comparison with OC cf. e.g. [30]). As the name Organic Computing suggests, inspirations for the realisation of such self-organising and evolvable systems are drawn from systems observed in nature. Physical, biological, chemical and other systems in nature are mostly complex and self-organising (consider e.g. swarm behaviour of birds or fish, neural networks in the brain, dissipative structures). Undeniably, these systems show an astonishingly degree of robustness towards disturbances. Consider for example the evaluation of universal properties of complex robust systems presented in [132]. In OC, the aim is hence to design systems with organic, life-like properties to endow technical systems with robustness and high performance. These properties are referred to as *Self-X properties* (cf. e.g. [30], [131]) and seen as constituting the term *self-organisation*:

- *self-configuration*: The ability of a system to explore its configuration space in order to adapt its parameters such that it can cope with changes in the environment. Parameter changes
- *self-healing*: The ability of a system to recover from sub-optimal states induced by disturbances without external control.
- *self-explanation*: The ability of a system to provide information about its properties and states.
- *self-protection*: The ability of a system to protect its operation from disturbances in the environment.
- *self-optimisation*: The ability of a system to change its structure and properties if this allows for an improved operation.

The realisation of these properties is facilitated by the increasing computational power of modern systems, as well as the ubiquitous and inexpensive equipment with a multitude of sensors. On the algorithmic and conceptual side, the OC community has developed a set of design methods, tools, architectures and patterns to aid the design of adaptive and self-organising technical systems.

The most prevalent concept is the *Observer/Controller (O/C) design pattern* (cf. e.g. [133], [38], [134]): As depicted in Fig. 2.1, the design pattern is aimed at the creating a configurable regulatory feedback loop for a complex technical system, here termed *production engine* (this is sometimes also referred to as *System under Observation and Control (SuOC)*. By definition, the production en-

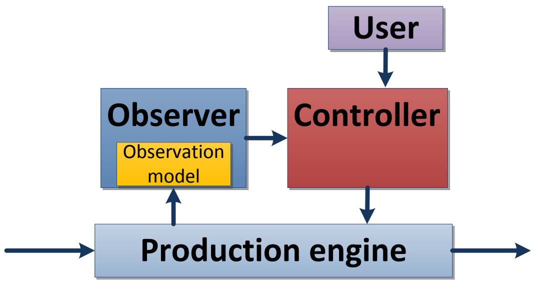


Figure 2.1: The generic Observer/Controller design pattern: Observer and Controller constitute a regulatory loop over a complex technical system (the production engine). The control is dependent on goals prescribed by an user.

gine serves a specific technical purpose, is composed of many decentralised elements (e.g. agents, devices) that interact locally, and is situated in a system environment. This environment, as well as the production engine itself, are monitored by a superimposed *Observer*. The data collected by this component is then pre-processed, analysed and aggregated, resulting in a *situation description*. This is a vector containing values for the system attributes and captures the state of the production engine at the time of the observation. Obviously complex systems cannot always be monitored entirely, instead relevant and accessible system attributes must be selected. This task is performed by the *observation model* included in the Observer. In addition, this model must specify the scope of the data, hence whether a short-term or long-term situation description is required, and whether situation predictions shall be incorporated. The condensed data is then passed on to the *Controller*: This component is responsible for the adaptation of the system control to the current system state according to goals set by a user or designer. If for example, the Controller recognises a deteriorated system performance in a certain system state (compared to previous data), then the control is adapted to this state. Also, if the presence of a disturbance is detected in the situation description, the system control is restructured to allow for a robust handling of this state (cf. e.g. [135]). It is important to note here that such correcting actions are intrinsic and do not require external control mechanisms (e.g. from a system operator). However, they require that the effects of control actions are also monitored. This is where the regulatory loop is closed: By observing the influence of the control on the system and iteratively adapting (and hence optimising) this control, the system becomes self-managed. This loop resembles artefacts known from control theory and related fields. Detailed comparisons and a classification can be found e.g. in [135], [136], and [137].

OC systems can be comprised of several variants of O/C-loops (cf. e.g. [138]): The *central* layout described so far uses one loop for the control of a production engine. However, as the production engine is composed of many elements, each element can be endowed with such an O/C-loop (see e.g. Sec. 3.1.2 of this thesis, where this is applied in MAS). This is referred to as *distributed* approach. Finally, both approaches can be combined resulting in a system that has a central O/C-loop that is superimposed on system elements each with an own O/C-loop (multi-level). In case of the design of a MAS as OC system, with agents as system elements, the multi-levelled approach can be used to realise an organisation structure with an aggregated system view and delegated control over the agents. Finally, it is assumed that due to the complexity and dynamics of the system Observer and Controller components can fail. As the components are superimposed on an independent production engine by definition, it is however guaranteed that such malfunctions do not affect its operation (cf. e.g. [134]).

The OC community has shown interest in a variety of topics related to self-organising, and nature-inspired systems: A recurring theme in the OC community has for example always been the *quantification and control of emergence*. In [139] the authors propose a quantification of emergence in technical systems based on entropy. Special attention is regarded to the fact that the term emergence is ambiguous and its quantification arguable. This quantification approach has been for example incorporated as emergence detector within the Observer component of an O/C-loop in order to suppress negative emergence. Another topic of interest is the *functional verification of self-organising systems*. OC Systems that constantly adapt their structure are hard to predict and can lack trustworthiness. This is especially relevant if the application in safety-critical domains is considered, such as in the control of power plants by means of Organic Computing (cf. e.g. [94]). The work presented in [140] has therefore been conducted with the aim to produce behavioural guarantees for OC systems. This is achieved by a framework for the formal specification of OC systems and the separated verification of their functionality and their self-X properties. The verification is integrated into a theorem prover and has the advantage that it is independent of the actual number of system elements (here: agents) which greatly increases the utilisation for analysis of large systems. The applicability of the framework is then demonstrated with resource-flow system. Another central theme in the research literature about OC systems is the *application of machine learning*: The shifting of the design of control mechanisms from the design time to a self-organised process at run time requires the system to *learn* itself how the control must be structured in diverse situations to yield a high performance. As the authors in [141] analyse, this is a challenging task as the optimisation of the behaviour of system components is in general performed in a so-called self-referential fitness landscapes. This means that the system elements cannot explore optimal behaviour rules in a static fitness landscape but that they change the fitness landscape due to their behaviour. The authors then propose a two-layered O/C-architecture that incorporates an extended learning classifier solution to learn the best control rules for a given observation on-line. In addition, they contribute a new optimisation algorithm that can be used for run-time optimisations of OC systems in noisy environments and self-referential fitness landscapes.

The OC community has demonstrated the applicability of the developed OC methodology in a number of projects from such diverse domains as *Traffic Control* (cf. e.g. [142], [143]), *Robotics* (cf. e.g. [144]), *Middleware Systems* (cf. e.g. [145]), *Network Protocols* (cf. e.g.[146]), *Decentralised Power Grids* (cf. e.g. [94]), *Wireless Sensor Networks* (cf. e.g. [147]) and *Smart Camera Networks* (cf. e.g. [148]) amongst others (cf. e.g. [30]). For the purpose of a more in-depth explanation of OC application, the following description summarise an exemplary project:

In the *Organic Traffic Control (OTC)* project (cf. e.g. [149], [31]), the systems of interest are urban traffic networks. Traffic networks are highly dynamic systems that rely on signalling control mechanisms to allow for the efficient flow of traffic. Yet, especially the control of traffic lights at intersections is still manually designed and considers only few traffic states, such as rush hours or night traffic, if at all. Under normal circumstances, this inflexible control scheme is sufficient. However the flow of traffic is complex, and traffic incidents, sport events, demonstrations, emergent traffic patterns and other seemingly unrelated environmental conditions can have massive effects on it. As the signalling of the traffic networks is not designed for such states, they can hardly be anticipated precisely enough for control consideration, traffic control operates far from the optimum here. This does not only affect the efficiency of the system, but can in the worst case even worsen the situation due to feedback effects of the inflexible control. The key motivation in the OTC project is thus to allow the system to self-organise in a way that allows to constantly optimise the signalling (increase the systems' performance), and to adapt to emergent system states to preserve a high performance (increase the systems' robustness). This is realised by interconnection of these intersections, such that they can exchange data and their equipment with a three-layered O/C-loop (cf. e.g. [136]). The first two layers comprise an O/C-loop as described above. The production engine is a standard traffic light controller, the Observer has access to traffic data measures by standard detectors (e.g. induction loops), and the Controller can change the signalling of the traffic lights. This system is used in conjunction with a Learning Classifier System (cf. e.g. [50]) to individually optimise the control of the signalling at each intersection. This is done at runtime and based on actual traffic flows. In that, the configuration space for the various signalling times is limited by the classification of similar traffic situations. Finally, the third layer comprises an additional superimposed O/C-loop that explores new control rules by simulation and observes their performance in the real system. In the 6-year project period, various refinements to this approach have been developed, such as the cooperation of intersections in the prediction of traffic, the consideration of traffic incidents, the extension to routing service provision and others (cf. e.g. [149]). In addition, numerous evaluations have demonstrated the increase in performance and robustness in urban traffic networks through the application of the described OC technology (cf. e.g. [31], [142], [143], [136]).

Finally, though the approaches in OC refer to general technical systems composed of a great number of interconnected elements, such systems are often (to varying degrees implicitly) interpreted as MAS in OC, albeit MAS operating in a technical environment. The reason for this is the shared interest in self-organisation in MAS (cf. e.g. [59]) and the availability of respective tools and frameworks. Consequently, the application of MAS technology in OC shows in several of the projects described in [30] and [37]. Explicit discussion of the compatibility of the two approaches can be found in e.g. [39], where the implementation of self-x properties by agent-technology is evaluated, and in [150], where an AOSE process for the design of OC systems is in the focus.

2.5 Decentralised Control of Open Distributed Systems

As reflected in the summaries above, the interest in the control of Open Distributed Systems is shared by the research communities for *computational trust*, *MAS organisations* and *Organic Computing*. Based on the varying schools of thought, each of these communities contributes unique approaches and application case studies. It is argued in this thesis that these approaches can be combined, and that such a joint approach can generalise some of the specific benefits provided by approaches from either research field. Consequently, a joint approach termed *Trusted Community*[3] is presented in this thesis. This view is however not exclusive to this thesis, esp. with respect to the combination of MAS trust and organisations, albeit a view not very commonly found in the literature. This section therefore reviews similar joint and approaches from the literature. In addition, the second part of this section is dedicated to the summary of related approaches specifically for the control of Open Desktop Grid Systems, as this is the system class in which the evaluations in this thesis were conducted (see Sec. 5).

2.5.1 Related Joint and Agent-based Approaches

In the following, a selection of related approaches using Multiagent technology are summarised:

Trust-based decision-making with controls: In [29], the authors examine trust-based decision-making for the delegation of generic tasks by self-interested agents in open and highly dynamic systems. They state, in accordance with referenced literature, that though trust and reputation are often used for decision-making approaches, it can be difficult to establish long-term trust relations in system with such challenging environments. According to the authors, this is especially critical when the MAS becomes *paralysed* due to the lack of reliable trust data. As a solution they suggest that *"[..] organisations may make use of controls which permit interaction when trust is low, providing initial evidence from which to bootstrap trust evaluations."*. In the course of the paper, they present an approach that combines trust-based decision-making with the additional controls *explicit incentives*, *monitoring*, and *reputational incentives*. They formalise this approach with the help of a decision tree for trustor agents. Then they successively evaluate their hypotheses that: (1) Agents show a better performance when their reasoning is based on trust, as well as on these controls, and that (2) agents apply the monitoring control preferably when trust information is rare and abandon this control in favour of less costly delegations when trust has been built up. Despite the initial motivation of applying *organisations* for the control of interactions at the beginning of the paper, further references to organisations are limited to the definition of ad-hoc groups of agents in the system. These groups are described as partitioning the system and being a context for information about other agents, such as reputation. The opportunity to extend the incentive and monitoring control approach by e.g. coordination and enforcement through organisation authorities is missed out by the authors. Despite its value on the insights about the necessity to enrich trust-based decision-making with control, the work must be classified as motivating a joint approach, rather than detailing its realisation.

Trust-Based Community Formation in Peer-to-Peer File Sharing Networks: The work presented in [152] discusses the design of a trust-based MAS organisation termed *community*[4] for the application in a peer-to-peer-based research paper sharing community. This MAS organisation is used to group agents that represent researchers with similar interests and lets them recommend relevant papers to each other. The trust relationships between agents in the system thus mainly reflects

[3] Disambiguation: The term *Trusted Community* is also used in [151] for the denomination of a social community approach to prevent cheating in online computer games. This concept is not related in any way to the work presented in this thesis.

[4] At one point in the paper, these are referred to as *trusted communities*, though this term is not used in the definition.

the trust relationships between their owners. In addition, the authors provide a means of evaluating the collective trustworthiness of such a community via a non-normalised trust value, i.e. the size of a community affects the maximal reachable trust value. A *community* of such agents is further defined as a MAS organisation that combines aspects of teams, coalitions and congregations, albeit the classification used is arguable. Community formation is self-organised and agents can join and leave them according to the value of received recommendations. A particularly interesting aspect of this is the ability of communities to fuse if they contain an intersection of mutually trusted members, a process that is decided based on consensus finding of all involved members. In addition, a community organises itself to consists of the *k* most trusted known agents (based on reputation). Agents from that list that are not members are invited to join, while members not included in that list lose their membership. This approach is evaluated in a simulation environment with 50 agents and the results show that its application increases the number of relevant research papers obtained by the users. Finally, the work in this paper features some ideas on the combination of trust and organisations that are also explored in this thesis, especially the self-organisation of a MAS organisation based on the trustworthiness of the members. However, in the opinion of the author, the case study used in the paper is not particularly suited to explore all relevant aspects of such an approach. Especially, the limited autonomy of the agents, along with an implicit benevolence assumption, make the case study appear as less challenging system than other MAS-based peer-to-peer systems.

Isolation of uncooperative agents by groups of trustworthy agents: In [153] an interesting theoretic work is presented on how defecting agents can be isolated by a self-organised process that combines MAS trust and organisations. In that, the behavioural dynamics of the agent population are influenced by game-theoretic considerations: The system is arranged as a spatial grid with neighbours randomly interacting by executing an iterated prisoner dilemma game. The agents can move within the grid and communicate asynchronously. Furthermore, agents choose to either cooperate or defect in the interactions based on the expected utility gain. The outcomes of interactions are remembered by the agents through the utilisation of a trust and reputation model. The authors are then primarily interested in the dynamics of trust among the agents. Especially, the characteristics of that model that agents adapt their behaviour based on the imitation of prevalent behaviours (cooperation/defect) in their neighbourhood, and that feedback loops can emerge are examined. In the evaluations, the authors use 400 agents with varying degree of bias to cooperate. They show that the system converges to an equilibrium state in which clusters of cooperative agents form neighbourhoods (referred to as communities) and isolate defecting agents. In a set of additional evaluations, they show then that this equilibrium is stable, hence that the cooperating communities are robust towards attacks. This work is chosen as a representative of game theoretic trust applications in the literature, esp. focussing on evaluations of (spatial) prisoner's dilemma games (for further examples cf. e.g. [154], [155]). What makes this work particularly relevant for the approach in this thesis, is the examination of the trust dynamics that lead to clustering of mutually trusted agents. Also the authors motivate the application of their model in technical open systems, though this is only sketched.

Group Organisation Algorithms based on Trust and Reputation: The work presented in [156] proposes a discrimination between *individual trust*, *interaction trust* and *group trust* to improve partner selection in open MAS. In that, the group trust is directly linked to MAS organisations and used for the identification of trustworthy collectives, rather than individuals. This is used in two presented algorithms, a centralised and a distributed algorithm. While the centralised algorithm is not particularly suited for the application in Open MAS due to the lack of a scalable interaction information management, the distributed algorithm utilises self-organisation. It allows to identify trusted agents and group them for the selection as interaction partners. Overall, the work presented in the paper features an interesting idea, but is merely a position statement, as it lacks any sort of evaluations.

Trust-based Congregations - Clans: The most relevant piece of related work for this thesis is described by N. Griffiths in his 2005 article called “Cooperative clans” (cf. [118]). In this paper, the author describes a type of MAS organisation called *clan* that is based on the idea of *congregations* (cf. e.g. [52]), as well as on literature on trust management[5]. *Clans* are medium-term compositions of agents that trust each other and have a motivation to enforce their cooperation. This motivation is derived from the agents’ goals and their requirement for cooperation to execute the plans leading to the fulfilment of the goals. More precisely, agents seek to form *clans* whenever they experience a certain amount of *missed opportunities for cooperation*, *lack of scalability*, *lack of information*, or *high failure rate*. This amount is quantified by thresholds. If one of these criteria is met, an agent triggers the *clan* formation and invites other agents to join in. These potential members are chosen based on their subjective trustworthiness, as well as their capability to execute tasks prescribed by the plans of the initiating agent. Additionally, the number of agents invited is restricted by the initiators’ demand for cooperation partners. Conversely, agents invited to join *clan* formation make this decision based on the trustworthiness of the initiator and the expected benefits of the *clan* membership as advertised by the initiator. Once established, each *clan* member can invite other agents to become members which does not require negotiation with the existing members. Additionally, each member leaves a *clan* at any time it finds the *clan* does not provide it any benefits in the execution of its’ plans or that it does not trust the fellow members any more.

Benefits of clan membership are derived mainly from the *kinship motivation*: Motivations are introduced in the reasoning of the agents to reflect their high-level desires and determine situations in which cooperation towards other agents should be executed. In that, the *kinship motivation* is one of possibly many motivations determining the behaviour of an agent associated to a *clan*: When the motivation is high, it provides fellow *clan* members assistance in the execution of their plans, while a low motivation leads to the rejection of cooperation. The motivation value is updated cyclically based on the *beliefs* of an agent and its’ perceived environment. Apart from this *commitment to cooperation* based on the *kinship motivation*, *clan* membership brings the advantage of *information sharing* among members (mainly about the trustworthiness of agents unknown to the information requester), as well as *scalability* (not searching the whole agent society but requesting only fellow *clan* members to cooperate).

Clans are proposed as purely decentralised MAS organisation for self-interested agents that mitigates some of the drawbacks of other MAS organisation, such as the lack of trustworthiness considerations and the focus on short-term relationships. The membership in a clan is then described as being beneficial for an agent as it enforces the motivation to cooperate of fellow agents. However the author does not address issues encountered in an open MAS: The exploitation of clan members by adversary agents, the requirement for the adaptation of formation and maintenance criteria to dy-

[5]In [157], an earlier work of Griffiths et. al. *clans* are yet referred to as being *coalitions*.

namic states of the system, as well as the possibility of emergent system states, esp. the breakdown of the trust management system. In other words, the purely decentralised composition and the lack of meta-reasoning (about the efficiency of the *clan* itself which does affect the benefits for single members) make *clans* vulnerable to exploitation. Consider the following concluding example: Each *clan* member can invite other agents to become *clan* members, merely informing its fellow members about the composition change. Due to the autonomy of the agents, it must be assumed, that membership is sought by an adversary that builds up a high trust value in order to become member and then pull other colluding agents in by providing them with membership invitations. Given that adversary clan members refuse to cooperate with fellow members, this phenomenon will lead to the disruption and finally dissolution of *clans*. In addition, the paper describes the concept of *clans* in an elaborate theoretic approach, however the lack of evaluation and/or application scenario discussions remains a flaw to the estimation of the applicability of this approach.

The shared system model for applicability in open systems with differences in the realisation of a trust and organisation based solution, make the *Clan*-approach a suited benchmark for the evaluation of Trusted Communities.

Trust and Coalitions: Multiagent coalitions have received special interest in the MAS community as they allow for self-organised and adaptive approaches to distributed problem solving. Consequently, joint approaches that utilise computational trust in conjunction with coalitions are more common in the literature than joint approaches for other MAS organisations. As summarised before, coalitions are short-term organisations that are formed due to specific goals and disbanded after their achievement (cf. e.g. [32]). Trust models are then used to initiate the formation of coalitions, make them robust towards the challenges of open environments and to expand them based on trustworthiness criteria. In the following, a few approaches from the literature that exemplify these different aspects are briefly reviewed.

In [158] the examined MAS is used as a service environment, with services being provided only by coalitions of cooperating agents. Agents are described as being unable to provide complex services individually, due to the requirement of complementary agent capabilities. To allow for flexible and adaptive service provision and a high quality of service, the agents are autonomous with respect to the formation of coalitions. To this end, the authors propose an approach with the following components to aid the agents in the self-organisation process:

1. A *trust and reputation model* that produces an aggregated normalised trust value based on weighted product of the single values.

2. A centralised *coalition formation service* that is used in case of insufficient trust information, such as for new agents in the system. This service matches service requests with available agents and suggests the formation of coalitions between them.

3. A *coalition formation process* based on the trust and reputation model. The provided algorithm allows mutually trusted agents to negotiate the formation of a new coalition.

4. An *adaptation heuristic* that allows agents to react to changes in the environment and interaction outcomes. Ultimately, the heuristic implements the decision whether to use the trust and

reputation model to form a coalition or resort to the coalition formation service to be assigned a coalition.

The presented approach is highly relevant for the work presented in this thesis as the self-organised grouping of trusted agents is a shared perspective. However, the produced coalitions are used only as to provide requested services and membership is not long-term. In addition, the coalition formation service provides an interesting approach to the handling of situations with insufficient trust data (which is also the main theme in e.g. [76]). However, due to the centrality of the approach, the applicability in an open MAS is arguable. Finally, the agents are not referred as self-interested, instead they exists only to provide services to request, which represents a fundamentally different system model as the one used in this thesis.

Another work addressing the trust-based formation of coalitions is presented in [159]. Here, the authors focus on the performance of agents in a coalition. They define a two-component trust model that considers the agents' *reputation*, as well as their *competence*. After a coalition of agents has finished its task and is disbanded, the contributions of the agents are assessed with the help of this model. This allows to choose members for future coalitions based on these past performances. The authors then suggest with a very simple simulation model that the application of their model can improve the formation of coalitions, and that it can give more significance to reputation values due to the consideration of the competence of agents.

A stronger focus on the individual utility of agents is pursued in [160]. In the work presented, the agents strictly try to optimise their own utility by joining coalitions. Formally, the model used lets agents rank available coalition options and then choose the most suited to this end. This initiates a negotiation between the potential members and results in either a successful formation or the cancellation of the formation. As members, agents can leave a coalition at any time. This introduces some challenges, as the utility of the other agents is negatively influenced by such an event. The authors therefore propose to utilise a trust model in order to estimate the previous behaviour of agents regarding their willingness to remain in a chosen coalition. This is evaluated in a setup with varying behaviours: Risk-seeking agents leave a coalition by accepting to join the formation of a higher rated coalition, even if the chance of a successful formation is low. On the other end, risk-averse agents make this decision only in case of a high probability for the formation. Finally, neutral agents make a balanced decision regarding expected utility and formation probability. In addition, agents have a *honesty*-level which determines how tempting the utility differences between the current coalition and an inviting coalition are perceived. The authors examine the relation between honesty, risk-orientation, association to coalitions and average utility with in simulation experiments with 20 agents.

The self-interested perspective on coalition members is also regarded in [161] (and the related earlier work presented in [162]). Here the authors examine an electronic market in which *customer agents* form coalitions to benefit individually from greater price discounts of large transactions. In addition, *vendor agents* join coalitions to negotiate such discounts and to increase its sales. The general assumption is here that agents who trust each other can form long-term coalitions and that such coalitions are stable and can reduce the dynamics in the market. This long-term view on coalitions distinguishes the presented approach from the approaches discusses so far. The authors propose mechanisms to undertake coalition formation and operation dependent of trust relationships. In their model, the trust between agents represents commonalities with respect to commercial preferences. A high trust value is then seen as yielding a high chance of successful future transactions (negotiations over a price). In the paper, a coalition reasoning mechanism is proposed that updates these trust values based on the success of transactions. In a next step, the trust relationships are interpreted (in search of the best trust relationship) and based on this it is decided whether to form a new coalition, leave a coalition to become member in a competing coalition, or to maintain the status quo. For the trust interpretations, the authors use agent strategies that prefer either the highest individual trust, or the highest social trust in a coalition (by summing the trust values of its members / counting its members with a positive trust value). In addition, the system is set up such that interactions within a coalition have a higher probability, leaving a coalition induces costs, or neither of the two. Finally, the coalition reasoning mechanism is evaluated in a set of simulation experiments with up to 10000 customer, and up to 1000 vendor agents. The main interest here is in the number of coalitions in relation to the number of interactions in the system. Here, the authors provide support for their initial hypothesis that in most cases the application of the trust-based reasoning can stabilise the dynamics of the system and lead to fewer and longer lasting coalitions. In addition, measurements of the individual gain of the agents are analysed. These show that the agents indeed benefit from such long-term coalition memberships.

In addition to the related work discussed here, coalition formation in conjunction with trust has also been examined in a number of publications focussing on game theoretic scenarios (as referred to in Sec. 2.5.1 where an exemplary paper is discussed).

Trust and Virtual Organisations: As summarised in Sec. 2.3, VOs are often interpreted as MAS organisations resembling coalitions or congregations. Consequently, this paragraph reviews related work that combines VOs with trust-based approaches.

In [163], the authors refer to VOs as open MAS in which a certain type of agent behaviour is prescribed, often by norms, but must be incentivised or enforced as agents can autonomously decide whether to adhere to the prescriptions. This view is compatible with most definitions of open MAS per se, but here the authors already explicitly assume the utilisation of organisation structures for the enforcement of behavioural rules. They further explicate that a trust and reputation model is useful for the decision-making of agents in such environments and that such a system can be improved by incorporating the organisational aspects of a VO. Consequently, the authors propose a trust model based on confidence and reputation and enrich the model by taking advantage of organisation information. For example the choice of good recommenders for the aggregation of a reputation value is based on the similar position (role) in a VO, and roles are assessed with a confidence value. In addition, groups of agents that are organised can be assessed as a unit which again is a holistic approach. Also, as already seen in the reviews of [158] and [29], the authors here see organisational information as a means to make decisions in case of insufficient trust information. However, the work presented in this paper is described as preliminary and lacks an evaluation in an instance of such a VO-based open MAS.

A similar motivation can be found in the Ph.D. thesis of J. Patel (cf. [93]). Here, an elaborate trust and reputation model for VOs is presented. In that, the definition of [45] for VOs is adopted, hence a VO is seen as a short-term coalition with a group goal, albeit the author considers the self-interest of agents in his work. In the development of the trust model presented in this thesis, the author pursues the following research aims: (1) The utilisation of direct trust for the model, (2) the utilisation of recommendation trust, (3) the realisation of robustness towards biased and erroneous recommendations, (4) the exploitation of organisation information to enrich the model, (5) the incorporation of a confidence metric for trust and reputation values, and (6) decentralised design and computations of the model to allow for a scalability and robustness in dynamic environments. Apart from the presentation of a model designed by these principles itself, a particularly interesting aspect of the thesis is the description of the utilisation of this model to influence VO formation, operation, restructuring and dissolution in a Grid environment. Here, the author is influenced by similar work presented in e.g. [45], [127] and [129], albeit he substantially extends it by incorporating aspects of trust into the concept. Finally, the consideration of a Grid system allows an evaluation within a technical context which the author conducts by simulating a respective system and its agent population.

Other work regarding the utilisation of trust and reputation in the isolated aspect of *VO formation* is for example presented in [164] (a reinforcement learning approach) and [165] (focus on attacks and threats in the process).

Trust and Organic Computing So far all related joint approaches have utilised a combination of MAS organisations and computational trust. Here, approaches are reviewed that utilise trust in OC systems. As this field has been established only recently, the literature on joint approaches is yet rather limited. A summary of concepts and motivations for the utilisation of trust in OC systems can be found in [3]. After presenting a literature survey on computational trust and a functional definition of trust as multi-facet concept, the authors discuss research challenges and opportunities of the design of trust-based OC systems. The following are described as most pressing: (1) The requirements of OC systems to represent aspects of both *user* and *device trust* in shared trust models, (2) the possible benefits of trust-based self-organisation on the control of the emergence in OC systems, (3) the influence of trust-based approaches on the verification of OC systems, (4) the consideration of trust in the software-engineering of OC systems, and (5) the influence of user-trust oriented concepts for the design of adaptive user interfaces for OC systems. Exemplary research papers in this field cover the trust-based formation of autonomous virtual power plants (cf. e.g. [94]), a software engineering view on such systems in (cf. [13]), and the application of trust in OC-based robotics systems (cf. e.g. [166]). In addition, the project website for the DFG research unit *OC-Trust* (*FOR 1085*, cf. [167]) contains an overview over the research field and esp. features a list of related publications. Also note that this thesis has been conducted in context of this project and the authors' publications are referred there.

To the best knowledge of the author, the combination of MAS organisations and Organic Computing in joint approaches has not been adopted in the research literature. A single exception being the application of coalitions in formal verification of OC systems, as proposed in [168].

2.5.2 Approaches for Open Desktop Grid Systems

The approach presented in this thesis is generic and can be applied in any Open Distributed System that fulfils the requirements of the system model discussed in the next chapter. However, to demonstrate the benefits of its application, a specific instance of this system class has been chosen, an Open Desktop Grid System. This section is therefore intended to provide a brief description of such systems and to review related work that also explores the opportunities of applying agents in general, and computational trust and/or MAS organisations in particular in such systems.

Desktop Grid (DG) Systems (cf. e.g. [169]) 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 [6]. 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 [171]). Realisations of these systems are mostly based on either the *Globus Toolkit* (cf. [172],[173]) or Web-Service-based standards (cf. [45]). Also, as discussed earlier in this chapter, the utilisation of VOs (sometimes in conjunction with trust models) is quite common in Grid systems. In the remainder of this thesis, traditional Grid Systems will not be further referred to, mainly due to the lack of openness and uncertainty among the participants (no Open Distributed System) and thus low motivation for the application of the approach presented in this thesis. In contrast, Desktop Grid Systems can be realised as open systems, albeit this is not always the case (as for example in Enterprise DG systems). A thorough taxonomy for the different types of DG systems can be found in [80], and an additional taxonomy focussed on evaluations of DG systems is presented in [174]. The former taxonomy is also summarised and used to classify the evaluation scenario for this thesis in Sec. 5.1.1. In the following, only related work considering Open DG systems is referred.

Agent and Trust-based approaches for Open DGs Agent-based open DG systems are particularly challenging as the participants can have unreliable soft- or hardware components, refuse to cooperate with other participants (free-riding), deliver no, invalid, or tardy processing results, and worst of all, be adversary and try to disturb the operation of the system on their own or in collusion (cf. e.g. [80]). This has motivated research that tackles these challenges by the measurement of trust, reliability and reputation and by using these measurements to reduce the uncertainty of agent decision-making in such systems.

The problem of detecting and avoiding invalid processing results in a volunteer-based DG system has been addressed by L. G. Sarmenta in [175], a work that coined some of the terms used in many further publications by other authors. In this paper, the author proposed the following so called *sabotage tolerance mechanisms*: *Spot-checking*, a mechanism that utilises undistinguishable test task that are pre-processed and hence allow to identify false results at very low validation costs and the *Credibility-based fault-tolerance* mechanism. This mechanism is a combination of the classical majority voting, as well as the spot-checking approach. It asserts probability values (the credibility) for correct task results to the system participants, results, result groups and work entries. In particular, the performance of system participants is measured according to the amount of successfully passed spot-checks, hence the utilisation of evidence-based trust. The mechanisms presented in this work have then been extended in a number of contributions from various authors. For example

[6]cf. [170] for an elaborate analysis of the requirements to transform a user application to a DG application.

in [83], sabotage-tolerance mechanisms are seen in a broader context and complemented by further trust-aware decision-making, as well as application check-pointing to reduce the impact of adversary behaviour. Furthermore, in [176] the spot-checking approach is characterised as being impractical due to the requirement that test task must be undistinguishable from other task, whereas it is very hard to realise such tasks. Consequently, the authors propose a generalised version of this mechanism which is to some degree robust towards the detection of such spotter jobs.

In addition to the work on result verification, some contributions have addressed DG participants that behave uncooperatively, referred to as *free-riders*. This phenomenon is best known from file-sharing networks (cf. e.g. [63], [177]), but can also significantly impact the performance of a Desktop Grid system. In [84], the authors therefore propose a reputation-based approach to detecting free-riding and demonstrate its applicability in a system called *OurGrid*. In that, they show that the utilisation of reputation can not only identify free-riders, but also that it serves as an incentive to cooperative behaviour. This is especially interesting as the authors use a setup in which the participants are allowed to change their behaviour based on the expected utility. Note however that this approach does not refer explicitly to agents (as opposed to users), though the MAS perspective seems compatible with the approach. In [178] also a reputation-based approach is chosen to cope with the problem of free-riders in a super-peer structured DG system. Here, the authors show in a set of simulation-based evaluations that the application of their approach improves the makespan and speedup of collaborative agents in a DG.

Another phenomenon of constant interest is *collusion*. Collusion is referred to as the cooperation of adversaries to achieve a common goal, e.g. the disruption of a service. Collusion in DG systems is a highly relevant topic, especially due to the fact that task results can often only be validated by majority voting and hence a colluding majority can have a strong impact on the accuracy of obtained results. While most approaches to get hold of collusion do not consider open decentralised agent-based DG systems, the obtained results can often be generalised for such utilisation. In [179] for example, the authors suggest to assess the behaviour of DG users by utilising a reputation model in conjunction with pre-processed and indistinguishable *quiz* tasks (an approach also referred to as *spot checking* in the literature, cf. e.g. [175]). The approach can easily be incorporated in the decision-making of agents as it is purely decentralised. In addition, their approach is specifically designed to be robust towards collusion and in the paper the authors claim to outperform standard replication-based approaches when collusion is involved. In another work presented in [89], the authors propose a collusion classification approach that allows to identify groups of colluding agents. Although the presented approach does not utilise a trust or reputation model directly, the identification of colluding nodes could be used to improve such a model. However, to allow for this, the approach requires further evaluations in environments with dynamic behaviour (the paper considers only static behaviour). Finally, in [85]) an approach to identify colluding nodes based on the application of the *EigenTrust* model (cf. [106]) is proposed. Such adversaries are then excluded from the processing of further tasks which could also be incorporated in the reasoning of DG agents.

Other work has for example explored the modelling of daily DG user behaviour in combination with evidence-based trust estimations to improve the performance of DG participants (cf. [82]).

Agent and organisation-based approaches for Open DGs The application of MAS organisations within open DG systems is not a very common approach in the research literature. Most of the approaches are rather oriented at traditional grids, such as the VO approaches for these systems discussed earlier in this chapter. Consider for example the work presented in [180]: The authors apply organisational structures to improve a system that is open in terms of nodes being able to join and leave the system at any time. However, the approach targets the application in a strictly hierarchical system, which shows in such organisation roles as *global scheduler* being responsible for e.g. the monitoring of user jobs. An open DG system based on nodes being peers has in part fundamentally different requirements on the control. Similarly, the authors in [181] present a coalition-based approach for the improvement of grid scheduling. Again, a central system is assumed: Tasks arrive at a global point in the system and their processing is assigned to a global utility value. The agents in the system then form coalitions such as to process these tasks in a way that maximises the global utility. The problem is hence one from the domain of distributed problem solving and does not consider self-interested grid participants.

In contrast, the approach presented in [182] explores the application of organisation in an *Organic Grid*[7], a truly decentralised realisation of a peer-based DG system. The authors propose a concept of mobile agents and show how these can self-organise a tree-overlay network to improve the autonomous scheduling in the system. In that the organisations in themselves are rather simple structures that define hierarchies based on e.g. the processing speed of participating nodes. In addition, the authors do not address disturbances in the system, such as adversary strategies. In [183], such disturbances are considered in the form of agents that e.g. leave the system before a computation is finished (requiring re-scheduling), or have a low availability. Here, the authors also explicitly refer to DG systems. Again, an overlay network is created that organises nodes according to their performance as in the approach discusses previously. However, here further criteria are considered, e.g. locality and behaviour. This organisation process is managed by coordinator agents and results in *computation groups*. These are used in an agent-based autonomous scheduling mechanism. The application of organisational techniques is however very limited in this paper and could also be referred to as a clustering approach.

Joint approaches for Open DGs The most complete approach to combine computational trust and MAS organisations in order to tackle the challenges of open DG systems is presented in a Ph.D. thesis by S. Choi (cf. [184]). In his work, Choi first presents his taxonomy on Desktop Grid systems which has already been discussed as [80] and is used later in this thesis to classify the evaluation scenario. The taxonomy is accompanied with an extensive review of DG realisations in the literature. The author then discusses his system model, a centralised and open DG, and formalises a failure model for the system participants. Finally, a very interesting approach based on trust and organisations is proposed, the *group-based adaptive scheduling mechanism*. Here, Choi uses the following three main categories to group volunteered resources into scheduling groups: (1) Volunteer Credibility (trust), (2) Volunteer Service Time (dedication), and Volunteer Availability (volatility or failure). He then describes how the values for each category are determined for the volunteers, and how they are updated whenever the volunteers complete interacts (e.g. finish the processing of a task). In that, he defines the credibility of the volunteers as the probability that they produce a correct result for a task, based on observed evidence in this role. Choi then describes an approach to group volunteers based on the categories above and additional properties such as e.g. their locality (home or enter-

[7]This work is not related to the field of *Organic Computing* as introduced in Sec. 2.4

prise resources). In conclusion, three modes are presented to construct and maintain such resource groups: (1) *task-based* mode initiates the grouping after completed tasks, (2) *time-based* mode does so in regular time intervals, and (3) *count-based* mode groups according to the number of volunteers in the system. Choi then describes how such volunteer groups can be used to improve a centralised DG system: By utilising mobile agents, scheduling and handling of faults is distributed among groups of agents rather than being executed by a central scheduler. In addition, this approach allows to use multiple scheduling algorithms, tied to the properties of a volunteer group, in parallel. For example, groups of resources with low trustworthiness required more safety-oriented scheduling schemes than groups of low rates of failures. Finally, this approach allows to react to the dynamics of the volunteers joining and leaving the system as the utilisation of mobile agents allows for e.g. migrations to other groups. The author then details several aspects of this approach, for example a quantification of replica generation, choice of tasks to replicate, result verification, and re-scheduling in the event of failures. Finally, the thesis presents an evaluation in a simulated system environment with 200 volunteers with varying service time, credibility and availability. In the presentation of the results, the author focusses on the improvement of the throughput in the system, the amount of redundancy and the accuracy. Despite the many very interesting contributions, one could argue that the utilisation of organisation is rather limited and used only for clustering resources. The volunteers organised in the groups do not actively take advantage of their shared membership. In addition, the presented approach is tied specifically to the utilisation in a centralised DG system and lacks the generality of the approach proposed in this thesis.

Further research on joint approaches in the open DG domain is for example presented in [79]. Here, the authors propose an utility-based trust and reputation model for VOs in collaborative computing environments. The model is an interesting approach to the holistic view on a VO, however, the authors do not refer to agents, but rather understand VOs in the sense of human organisations. Furthermore, the paper features an extensive summary of the application of trust in distributed systems.

Another joint approach is presented in [88]. The authors propose a trust and reputation management system termed *H-Trust* that also incorporates the notion of group trust. The application scenario resembles that of Choi, reviewed above: A job distributor can use the model to determine to which group of agents its job should be assigned. In that, the owner of the job can use the model to discriminate trustworthy from untrustworthy groups, which is necessary according to the authors, as in an open system any agents can form such a collective. They then present evaluation results from simulations with up to 500 agents and demonstrate that their mechanism allows to detect malicious nodes based on an accurate reflection of their behaviour in the trust model.

2.6 Summary and Overview

Open, Agent-based Distributed Systems are a challenging domain: The autonomy of the agents, the autonomy of agent owners to use custom code, the self-interested utility definition of agents, and their often complex decision-making and resulting interactions are factors that heavily impact the control of such systems. In this chapter, many approaches to these challenges have been discussed, both for Open MAS in general and open, MAS-based Desktop Grids in particular. To this end, approaches from the research fields of *Organic Computing*, *computational trust*, and *MAS organisations* have been reviewed. A special focus has been laid on approaches that combine methods from these fields, as it is argued in this thesis that such a combination yields a high potential to generalise the benefits approaches from either individual research field. The review of related work has especially revealed the potential of such a combination in the following aspects:

- In the application of computational trust to address undesired agent behaviours, the **Trust Management is most often implicitly seen as stable, and abnormal system states are seldom considered**. Subsequent trust crises and TM breakdowns are neglected and the approaches presented are not prepared for such system states. However, the systems regarded are often composed of many autonomous agents with complex interactions. Such complex systems are described to evolve at runtime and not all trends can be predicted and prevented at design-time of such a system. Hence online reactions to emergent, abnormal system states should be considered in the design of such TM-based approaches.

- The consideration of the trustworthiness of interaction partners in the decision-making of agents allows to improve their robustness towards uncooperative and adversary behaviours. This has been presented in a large body of research literature. However, the overhead of Trust Management, together with an often slow response time to variations in agent behaviour remain challenging issues. Often authors in the literature propose approaches that do not adapt to states in which agents have strong, positive, and mutual trust relationships that render TM and its overhead unnecessary. This can lead to **suboptimal interactions due to TM overhead**.

- Evidence-based Trust Management requires the involved interaction partners to fully comprehend the interaction. However systems composed of self-interested agents often do not allow for comprehensive observations. Instead the agents can only make local observations and hence only rate the trustworthiness accordingly where the outcome of an interaction can be understood with these observations. On the other hand, such systems are described as open towards agents with various types of behaviours aimed at exploiting or damaging the system. As such, behaviours that cannot be observed by single agents, in particular collusion, require agents to cooperate in order to prevent harm to them. Often, MAS organisations are proposed to allow for this cooperation. However, most of the proposed organisations either require group goals (e.g. coalitions) or do not apply trust management despite its benefits in this context (e.g. congregations). **Only few approaches regard trust-based MAS organisations to motivate self-interested agents to make shared and comprehensive observations of their environment**.

- **Most of the approaches in the the domain of computational trust and MAS organisations are not evaluated in technical environments, but according to theoretic models**. While this accounts for various characteristics of the proposed approaches, it is often hard to judge about the applicability in *technical* Open Distributed Systems.

- **The number of research papers addressing truly decentralised and peer-based Desktop Grid Computing systems is still limited and the application of computational trust and MAS organisations in them even more so**. As the computational power of personal devices steadily increases, while ecological and economic reasons advise to harvest these often spare resources, such systems are seen as having a high potential for being adopted more commonly in the future. In addition, these systems are a perfect instance of technical Open Distributed Systems, and very well suited as evaluation scenarios for approaches in the reviewed research fields.

Finally, for a quick overview over the most relevant related approaches reviewed in this chapter, see the following Fig. 2.2.

Approach, authors	Focus of approach	MAS	Organisation	Trust	Self-X properties	Generality	Technical evaluation scenario	Open distributed system	Desktop Grid
H-Trust, H. Zhao et al.	Detection of malicious agents in a grid	✓	◐	✓	◐	◐	✓	✓	✓
Organic Grid, A. Chakravarti et al.	Self-organisatio and mobile agents	✓	◐	✗	◐	◐	✓	✓	✓
Trust in the P2P-based OurGrid, A. Andrade et al.	Incentives for cooperation and detetion of free-riders	✗	✗	✓	✗	◐	✓	✓	✓
Coalition formation based on trust, B. Hoelz et al.	Efficient short-term service provision (group goals)	✓	✓	✓	◐	✓	✗	◐	✗
Communities of trusted agents in spatial PD problems, R. Ghanea-Hercock et al.	Trust dynamics	✓	✓	✓	◐	◐	✗	✓	✗
Trust-based communities, Y. Wang et al.	Organisation composition	✓	✓	✓	◐	✓	◐	◐	✗
Trust decision-making with controls, C. Burnett et al.	Trust-based decision-making	✓	◐	✓	✗	✓	✗	✓	✗
A trust- and reputation model for agent-based VOs, J. Patel	Trust-model	✓	✓	✓	✗	✓	◐	✓	✗
Group-based adaptive scheduling mechanism, S. Choi et al.	DG scheduling mechanism	◐	◐	✓	◐	✗	✓	✓	✓
Clans, N. Griffiths et al.	MAS organisation	✓	✓	✓	✗	✓	✗	◐	✗
Implicit Trusted Communities, L. Klejnowski et al.	Trust-aware scheduling with incentives	✓	✗	✓	✗	✓	✓	◐	✓
Trusted Communities, L. Klejnowski et al.	MAS organisation	✓	✓	✓	✓	✓	✓	✓	✓

Figure 2.2: Overview of related work depicting the most relevant related approaches from the literature, as well as their classification according to key aspects of this thesis. For each related approach, the degree of consideration of the key aspects is depicted.

3 | System Model

This chapter introduces the system model developed and applied in the thesis, by first presenting a system view. Here, the concept of the *hosting system* is defined: This is a model how to apply MAS in general, and Trusted Communities in particular, to technical systems by using Organic Computing techniques. This model is composed of the submodels of *agents*, as elements of the system, the *production engine*, encapsulating the mechanics of the technical system, and a *Trust Management* system the Trusted Communities are built upon. The second part of this chapter presents an analysis of the challenging issues in such a system and introduces the Trusted Community approach to overcome these issues. It describes how Trusted Communities can be applied in a hosting system, and where limitations to the application exists.

3.1 System View

This section specifies for which systems the approach in this thesis has been designed. First, the concept of the hosting system is introduced, followed by a detailed specification of a functional agent model for participants in the hosting system. This is concluded by a specification of the required Trust Management system for such systems. Finally, the composition of a hosting system is discussed and the assumed agent society introduced.

3.1.1 The Hosting System

This thesis introduces the approach to improve open, distributed systems by providing the elements of the system with the ability to form Trusted Communities and thus apply an advanced form of self-organisation. From here on, the system in which Trusted Communities are applied will be referred to as the *hosting system* $\mathcal{H}$. As depicted in Fig. 3.1, the hosting system is an open distributed system without central control. It is composed of a number of elements, the so-called production engines. A production engine is a client software that allows technically to participate in the system, for example a Desktop Grid Client (see Sec. 5.1), a wireless sensor node or a client in an e-commerce system. Additionally, the assumption is made here, that each production engine is under the responsibility of a user or owner, although this is not a requirement for the approach presented in this thesis. It is additionally assumed, that the production engines are independent of each other, such that they belong to different users and/or administrative domains, and that they are heterogeneous with respect to their capabilities and configurations. Owners of production engines use them to interact with other production engines and via a shared environment (usually a network) to reach an individual goal or performance. The interactions rely on a shared protocol defining the exposed functionality, whereas the exact capabilities and configurations of production engines are not visible to others.

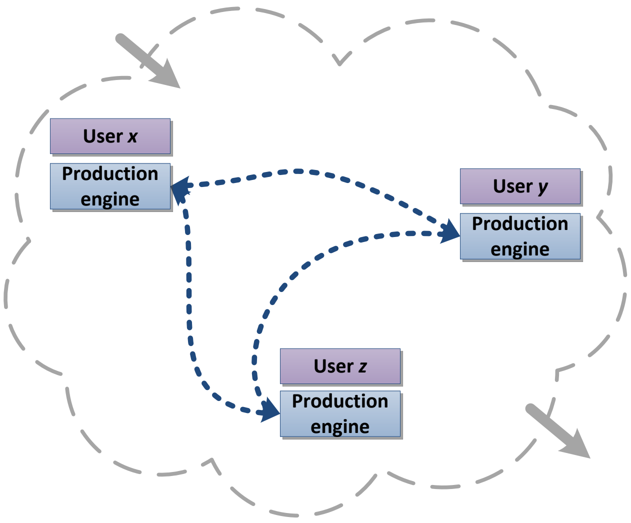


Figure 3.1: System model of the hosting system: Each user owns a production engine, a client software that allows to participate in the hosting system and defines a performance measure.

The hosting system is characterised by its openness, i.e. participants can enter and leave the system arbitrarily. Note that this is a very broad definition of a technical system, comprising most of internet-based, decentralised systems. However this definition of a hosting system is emphasised here in order to distinguish the application scenario of the TC approach from non-technical scenarios like social simulation with agents[1]. In the following, this very generic view is left behind and the system model is refined by the incorporation of agents that represent users in the system and control the production engine. This allows for the self-organisation of the system and thus for adaptivity to system states that were unforeseen at the design time of the production engines.

3.1.2 Agent Model

In this section, an agent model for the application of Trusted Communities is presented. It is not the intention of the author to contribute to the numerous and elaborate agent models described in the literature on MAS (cf. e.g. *BDI* [54] or *MIS* [185]). Instead, the agent model is introduced with the aim to capture a functional representation of the requirements necessary for the TC approach. The work here is based on the Observer/Controller design pattern (see Sec. 2.4) and on agent models in the literature (cf. e.g. [49]).

It is generally agreed in the research community (cf. e.g. [49]), that agents, in the sense of *software agents* in the field of Multiagent Systems, are programs acting on behalf of a user and/or program. Referring to the model of a hosting system from the previous section, this means that the agents control the production engine on behalf of the user (as depicted in Fig. 3.2). In that, agents are characterised by *autonomy* (at least to some extent), i.e. they exhibit reasoning capabilities, as well as *interactions*, i.e. agents have the capability to interact with each other via defined protocols. It is these interactions, that essentially characterise a Multiagent System, a common environment

[1]In e.g. [44] the type of system that is referred to in this thesis is denoted as "Multi-agent decision system", as opposed to "Multi-agent simulation system" and "Agent assistant" system.

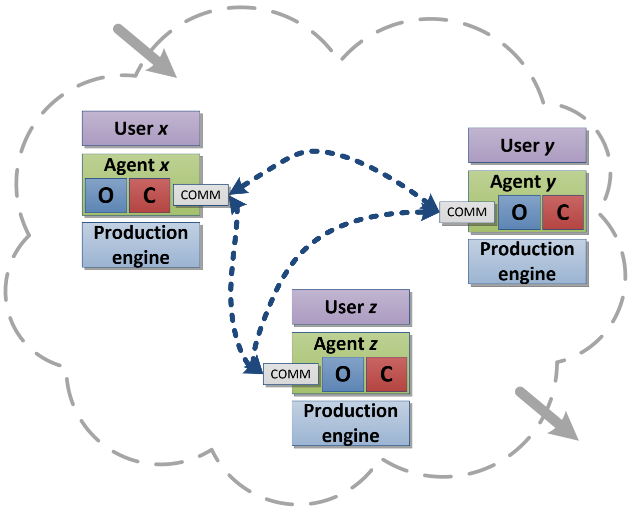


Figure 3.2: System model with OC-based agents: Here the system participants incorporate observer and controller to allow for adaptations based on the utility of the production engine. The communication between the agents is controlled by COMM interfaces encoding an interaction protocol.

shared by a number of agents perceiving each other, and being able to interact. Interactions between agents are of utter importance here: Not only are they often required for agents to reach their user's goals (cooperation), but they are in general one of only three sources[2] of information about other agents. This is due to the autonomy of the agents: Agents have a strict local view, i.e. agents (as well as users) are not able to perceive reasoning states, actual capabilities, strategies, plans etc. of other agents (black box approach, cf. e.g. [186]). Only when agents decide to share information with other agents, is this information disclosed. Agents can always reject to share information with other agents because of strategic considerations. Additionally, in open systems without central control over the implementation of agents, the interactions between agents need to be based on a commonly understandable communication protocol. As depicted in Fig. 3.2, this is abstracted by the use of a communication-interface (COMM) in the specified model.

When applying agents to represent them, users expect these agents to firstly hide the complexity of a system participation, and secondly to perform better in the system than is expected from manual control. The performance of the agents is measured in terms of a **utility function $U^x(t)$**, with x being an agent and the time t being the time of the utility evaluation by the agent. This function is defined by a system designer and reflects the desired agent performance. Consider for example the completion time of jobs in a Desktop Grid System as such performance measure. If an agent is aware of its utility at runtime, it can adapt its behaviour constructively (self-awareness, cf. e.g. [187]). Here technical systems are considered, therefore the definition of a utility function $U^x(t)$ is always derived from the performance of a production engine in the open distributed system. In a Vehicular Ad-Hoc Network (VANET) for example, the utility function of an agent designed to control the according client, will incorporate the aspects *connectivity*, *energy consumption* and *communication economy* among

[2]The other sources being the shared environment (through observation: *stigmergy*) and external entities (like users, providing additional information retrieved from external sources like program analysis).

others. For the agents that are modelled here, the utility function is therefore always technically defined and quantified. What is more, the agents are assumed to be *self-aware* in this respect, i.e. they know and can interpret their own utility value. The reflection about the own utility is a central part of agent decision-making and allows agents to optimise their behaviour.

In the following, it is presented how an agent based on the Observer/Controller design pattern (cf. [30]) is defined. As depicted in Fig. 3.2, an agent is set up by a user to control its production engine. Each agent contains two essential building blocks: The *Observer (O)* and the *Controller (C)*. The Observer monitors the production engine and the environment and aggregates the gained knowledge into a situation description. This situation description classifies the local view of the agent about the system state at the time of observation and is passed on to the Controller. The Controller is the active part of the agent: It adapts the configuration of the production engine based on the information in the situation description. The new configuration of the production engine has then an influence on the performance and the system state, which then again is perceived by the Observer, thus building up a control loop as applied in control theory and similar to the MAPE-cycle (cf. e.g. [131]).

Agent Components

In the following, a more detailed view[3] on the Observer and Controller parts of an agent, as depicted in Fig. 3.3, is presented:

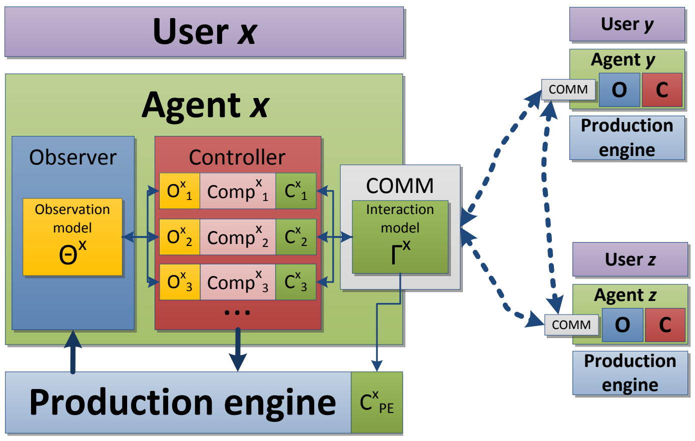


Figure 3.3: Agent model of an agent participating in the hosting system. The observer incorporates an observation model and the controller is composed of single agent components. Communication between agents is based on interactions that the agent components provide.

In this model an agent is defined by encapsulated and coherent function blocks, the so-called *agent components*. These components serve the purpose to be effective on the production engine and represent the reasoning and control capabilities. The Controller of an agent is thus comprised of a set of components, with each agent having a component composition not known to other agents. Each component needs input data to operate and can provide interaction interfaces to other agents.

[3]This view is compatible with the definition of the Observer/Controller design pattern in [30] and extends it with an observation and interaction model. All advantages of the O/C pattern apply, but the presentation is more focussed on agents than the original pattern. Work on this has been conducted in cooperation with Yvonne Bernard.

Consider for example an agent x, with an agent component $Comp_i^x$, that needs to make the decision whether an interaction with another agent y is beneficial to increase its utility and should therefore be accepted. In this case, the agent component needs information about agent y, possibly about its *trustworthiness*, in order to make a good decision about a potential utility gain for x, based on the probability that y will make a trustworthy interaction partner. An agent component $Comp_i^x$ thus needs to define the required input information in the form of a set $O_i^x = \{\sigma_1, .., \sigma_n\}$ of observables with each observable σ_i being the following tuple:

$$\sigma_i = \left\langle \sigma_i^o, \sigma_i^s, \sigma_i^n, \sigma_i^f \right\rangle$$

with:

σ_i^o	the owner of the information
σ_i^s	the scope of the information
σ_i^n	the name of the information
σ_i^f	the observation update frequency

The *owner* of the information is an agent or set of agents in possession of this information. The *scope* of the information is either of the following:

- *Self Knowledge*: Information the agent has about itself, such as the current utility.
- *Private Knowledge*: Private information the agent has about other agents, such as trustworthiness values assigned based on former interaction experiences.
- *Society Knowledge*: Information that is freely obtainable by any agent, such as information broadcasted by agents in the society.

The *name* of the information is an identifier defined by a common communication protocol within the *COMM* interface, while the *observation update frequency* allows to specify periodic information requests. Note here, that due to the autonomy of the agents, an information owner x is free to decide whether to provide the information requested from it by an observable of agent y or not. Finally, an observable σ_i is the specification of an information retrieval requirement. The actual value of this information at the time t is denoted as: $v(\sigma_i, t)$.

Additionally, in order to understand the request of agent y and allow for an interaction with it, a component also needs to define interaction interfaces.

A component $Comp_i^x$ of agent x therefore also comprises a list $C_i^x = \{c_1, .., c_k\}$ of interactions or primitives that this component allows. This does not imply that agents are obliged to provide these interaction possibilities to other agents, but rather that they can be requested of them, with the decision about an interaction being part of the decision making within the component.

The main part of a component encodes the decision-making for the encapsulated functionality of the component. In that, it provides interfaces for other components of the owning agent, an evaluation of the observables, and the control of the functionality based on the respective view on the environment. Finally, a component must also encode adaptable rules for the handling of interaction requests via its specified interaction interfaces.

Observation and Interaction Models

In general, an agent x is composed of k components $\mathcal{C}omp_1^x$,..,$\mathcal{C}omp_k^x$, each encapsulating a different functionality. However, the observables O_i^x defined by each component $\mathcal{C}omp_i^x$ can have similar characteristics: Identical observables, or observables only varying in the update frequency, can be required by two different components. To avoid redundant information retrieval and provide a precise information definition, an Observer aggregates the required observables in an *observation model* Θ^x, defined as:

$$\Theta^x = agg_\Theta(O_1^x, .., O_k^x) \subseteq \bigcup_{i=1}^{k} O_i^x$$

In that the aggregation function $agg_\Theta(O_1^x, .., O_k^x)$ merges the required observables such that all component requirements are met while the redundancy is removed. This results in a set Θ^x of distinct observables. This model is then used by the Observer of agent x to control the information retrieval (also referred to as *observation*) needed by the components. The aim of this observation is to produce a *situation description* $\mathcal{D}(\Theta^x, t)$ at each time t that is as complete as possible[4] with respect to the observation model Θ^x. Formally, a situation description is a set of observable-retrieved-value-pairs, such that:

$$\mathcal{D}(\Theta^x, t) = \langle(\sigma_i, v(\sigma_i, t)), .., (\sigma_n, v(\sigma_n, t))\rangle$$

A situation description is used by the components of x for their decision-making, as well as by the controller as such to allow for adaptations of the components to the present situation (for example the exchange or activation/deactivation of components within the Controller). Note here, that the observation model Θ^x, and consequently the composition of situation descriptions $\mathcal{D}(\Theta^x, t)$, is dynamic: The activation/deactivation or internal reasoning of an agent component $\mathcal{C}omp_i^x$ can inflict changes in the set O_i^x of observables required for the operation of $\mathcal{C}omp_i^x$. This in turn can change the composition of Θ^x, resulting in a new observation model $\widetilde{\Theta}^x$. As a side effect, this introduces a challenging issue[5] for learning-classifier-based adaptation (cf. e.g. [50]) within the Observer/Controller cycle: Two situation descriptions $\mathcal{D}(\Theta^x, t_i)$ and $\mathcal{D}(\widetilde{\Theta}^x, t_j)$ need to be matched in order to compare the rewards achieved by applying different actions in both situations.

Another agent-wide characteristic is the sum of interactions it provides to other agents. This is referred to as the *interaction model* Γ^x of an agent x. It is composed of the union of all interaction interfaces provided by the agent components and the production engine interactions C_{PE}^x, such that:

$$\Gamma^x = \bigcup_i C_i^x \cup C_{PE}^x.$$

In that, the set C_{PE}^x is composed of interactions that the production engine provides even if there is no control by an agent (in which case the user has to manually decide whether to accept interaction requests). In case of agent control, decision-making for the functionality of the production engine is transferred to the agent components. The production engine interactions are then triggered only internally by the components. Both, observation and interaction model, are dependent on the component configuration of an agent at time t, which can be subject to change during the system

[4]This refers to the fact that observable owners can reject to provide the requested information. It is in the responsibility of the function $agg_\Theta(O_1^x, .., O_k^x)$ to account for this and use previously acquired information as needed.

[5]O/C-based adaptations are however not in the focus of this thesis, this is therefore neglected here.

participation of an agent. This is due to the fact that (1) agents can decide to exchange component realisations at runtime (with possibly different input and interaction requirements), and (2) activate new or deactivate existing operating components.

In conclusion, the notation Λ_{type}^{impl} is utilised throughout the thesis to denote the implementation of an agent component type. This $type$ is, for example, a specific component required to control the production engine. The implementation $impl$ is the realisation of a type of component, for example a stereotype *freerider* (see Sec. 5.3.1), being a realisation of a decision-making for this component that rejects all interaction requests from other agents.

3.1.3 Trust Management System

The previous section extended the system model by introducing agents that control the production engines on behalf of the users. The delegation of control to autonomous agents enables the system to self-organise and better adapt to runtime system states. Here, cooperative behaviour of agents can lead to a high performance increase for a user. On the other hand, the exact realisation of behavioural agent strategies is not known to other agents and the system is open, meaning that any agent can enter the system; the only constraint being a functionally compatible production engine and the adherence to a common interaction protocol. This introduces the risk of uncooperative, exploitative or even adversary agents and can, if not accounted for, lead to dramatic performance decrease among the agents. This is where the application of a **Trust Management system** is required.

The Trusted Community concept is based on the notion of (artificial) trust (see Sec. 2.1 for a detailed discussion). As such the Trust Management of a hosting system is of utter importance for the formation and maintenance of Trusted Communities: Trust values of agents are both a signalling system for the agents and a means to evaluate interactions with each other. In the following the requirements on a trust model are specified:

Required Trust Model

The approach presented in this thesis requires a trust model that allows to derive the trustworthiness of agents based on evaluations of direct and indirect past experiences among them. In the literature on trust in MAS, this is the most commonly applied type of trust model (cf. [24]). In such a model, a *direct trust value* $DT_x^{y,\hat{c}}$, which is the subjective estimate of agent x about the trustworthiness of agent y in context $\hat{c}$, is hence the aggregation of the set of interaction ratings $\mathcal{R}_x^{y,\hat{c}}$ that x assigned to y. Most authors in the literature on trust agree (cf. e.g. the analysis in [20] and examples in [24]) that it is not sufficient to state that an agent x trusts another agent y (in everything y does). Rather, this statement can only be made for a specific context $\hat{c}$ in which these agents interacted, resulting in a set of ratings $\mathcal{R}_x^{y,\hat{c}}$. As an interaction of the type c_i, between an agent x with an agent y is always executed via an interaction interface $C_i^y = \{c_1, .., c_k\}$ provided by a component $Comp_i^y$ of y, the context is derived from this component, such that $\hat{c} = i$. In other words, the trustworthiness estimations of x about y always refer to the trustworthiness of a certain component of y, with y having components that are potentially heterogeneous in terms of trustworthiness. An agent y can for example have a set of default components, and additionally some components with modified code for adversary behaviour.

In cases where an agent x has made only few interactions with another agent y, it is often desirable to not only rely on this little evidence about y, but also consider other agents' testimonies about y. From the view of x, these are referred to as indirect trust values about y. The result of an aggregation of indirect trust values to a single value about y is further referred to as the *reputation value* $RT_{\mathcal{P}}^{y,\hat{c}}$ of y. In general[6], an indirect trust value can be defined as the direct trust value $DT_{z}^{y,\hat{c}}$ of an agent z about y provided to x. The reputation is hence composed of a number of such values $DT_{p_i}^{y,\hat{c}}$ provided by a group $\mathcal{P}$ of recommenders or opinion providers p_i.

Now to benefit from a broader picture about an agent y (cf. e.g. [188]), the direct trust value $DT_{x}^{y,\hat{c}}$ of an agent x about y can be combined with the reputation value $RT_{\mathcal{P}}^{y,\hat{c}}$ of y to receive a single value, referred to as the *trust value* $T_{x}^{y,\hat{c}}$.

Formally, the composition of the three values, as required for the approach in this thesis, is then defined as:

Direct trust value of y in context $\hat{c}$, as estimated by x:

$$DT_{x}^{y,\hat{c}} = agg_{DT}(\mathcal{R}_{x}^{y,\hat{c}}) \tag{3.1}$$

with ratings and direct trust value aggregation function:

$$\forall r \in \mathcal{R}_{x}^{y,\hat{c}} : r \in [-1,1] \text{ and } agg_{DT} : \mathcal{R}_{x}^{y,\hat{c}} \rightarrow [-1,1]$$

Reputation value of y estimated by a group of opinion providers $\mathcal{P}$:

$$RT_{\mathcal{P}}^{y,\hat{c}} = agg_{RT}(DT_{p_1}^{y,\hat{c}}, .., DT_{p_k}^{y,\hat{c}}) \tag{3.2}$$

with the set of opinion providers:

$$\mathcal{P} = \{p_1, .., p_k\}$$

and the reputation value aggregation function:

$$agg_{RT} : \left\{DT_{p_1}^{y,\hat{c}}, .., DT_{p_k}^{y,\hat{c}}\right\} \rightarrow [-1,1]$$

Trust value of y, estimated by x and including the opinions of providing agents $\mathcal{P}$:

$$T_{x}^{y,\hat{c}} = agg_{T}(DT_{x}^{y,\hat{c}}, RT_{\mathcal{P}}^{y,\hat{c}}) \tag{3.3}$$

with the trust value aggregation function:

$$agg_{T} : [-1,1]^2 \rightarrow [-1,1]$$

These trust values are defined for a range between -1 (total distrust) and 1 (blind trust). For the Trusted Community approach, the aggregation functions (agg_{DT}, agg_{RT} and agg_{T}) must be provided for the described ranges. It is however not further specified how the functions perform the aggregations. Example functions, such as an agg_{T}-function utilising a weighted average between direct trust and reputation for the aggregation of the trust value, can be found in the literature for different trust models (cf. e.g. [24]). Additionally, example definitions for these functions are provided in the specification of the application scenario for this thesis, the Trusted Desktop Grid (see Sec. 5.1.4).

Finally, the direct trust value has been introduced as being based on ratings about the interac-

[6]Some authors (cf. e.g. [74]) state that in Open MAS, agents should also be assumed to have heterogeneous trust models which need to be "aligned" in order to understand each others decisions and opinions. The focus in this thesis is not primarily on trust models, hence, this is neglected here.

tions with an agent component (the context $\hat{c}$). To be applicable, the trust model applied in such a system must hence define how such a rating is retrieved. For this, a mapping of possible observable outcomes $\left\{o^1_{c_i}, .., o^k_{c_i}\right\}$ for each interaction $c_i \in \Gamma^x$ needs to be provided. Additionally, for each of these outcomes, a rating value r^j_i must be implemented that rates the usefulness of this outcome. The retrieval of a rating value by an agent x for such an interaction-outcome pair is formalised with the following function:

$$rate(c_i, o^j_{c_i}) = r^j_i \tag{3.4}$$

with:

$$rate : \Gamma^x \times \left\{o^1_{c_i}, .., o^k_{c_i}\right\} \rightarrow [-1, 1]$$

Consider the following basic example: An agent component $Comp^x_1$ provides an interface to the interaction c_1, encoding the request of a private information of this agent x. From the view of an agent y such an interaction with x has two outcomes: Either x cooperates and provides this information (outcome $o^+_{c_1}$) or it rejects the request and no information is delivered to y (outcome $o^-_{c_1}$). The agent y defines the ratings $r^+_1 = 1$ and $r^-_1 = -1$ for these outcomes, such that:

$$rate(c_1, o^+_{c_1}) = r^+_1 = 1 \quad \text{and} \quad rate(c_1, o^-_{c_1}) = r^-_1 = -1$$

This is obviously only possible if x always provides correct information when it cooperates, or if y cannot validate this. In case y could discern valid from invalid information provided by x, the outcome $o^+_{c_1}$ could be split into two outcomes, one for each case (provided y does not equal the value of an invalid answer with the value of no answer at all). Note here, that interactions are always executed between a pair of agents. In general both involved parties can rate each others' behaviour. This is compatible with this model, as the agents both perform the interaction via specific components allowing each of them to derive a rating context for the other one.

To conclude the specification of the required trust model, an additional term used in this thesis is formally defined. This term is the **strong mutual trust relation**, mainly used within the context of decision-making for the formation of Trusted Communities. A trust relation of two agents x and y is (positively) strong and mutual if the following condition holds:

$$DT^{y,\hat{c}}_x > thres^{DT}_m \land DT^{x,\hat{c}}_y > thres^{DT}_m \tag{3.5}$$

In this, the quality *strong* is quantified by the threshold $thres^{DT}_m \in [0, 1]$. Note here, that this is an exclusively subjective evaluation of a trust relationship of two agents, not affected by the opinions of other agents. This is why this definition utilises only the direct trust values (as opposed to additional consideration of the reputation). Additionally, the value $DT^{y,\hat{c}}_x$ of agent x for agent y (and vice versa) is strictly private information. In order for two agents to realise that they have such a strong mutual trust relation, they need to exchange this private data. Also, this definition refers to mutual trust in a certain context $\hat{c}$, a definition that can be expanded for specific systems where there is need for a definitive statement about mutual trust, independent of the context. This is reasonable in systems where agents do not execute interactions with each other in the same context, but two different contexts, nevertheless building up substantial mutual trust within these different contexts.

Trust Management Agent Component

So far, the required trust model has been specified. In this specification actions from agents have been referenced, such as the provision of direct trust values $DT_i^{y,\hat{c}}$ to allow other agents to aggregate a reputation value $RT_{\mathcal{P}}^{y,\hat{c}}$ for an agent y. Such actions, as well as the general reasoning based on the trust model, need to be incorporated in the agent model. This is achieved by a dedicated agent component that needs to be provided within an agent: All operations related to trust management, such as the aggregation of trust values, are placed in the Trust Management (TM) component (depicted in Fig. 3.4) $Comp_{TM}^x$.

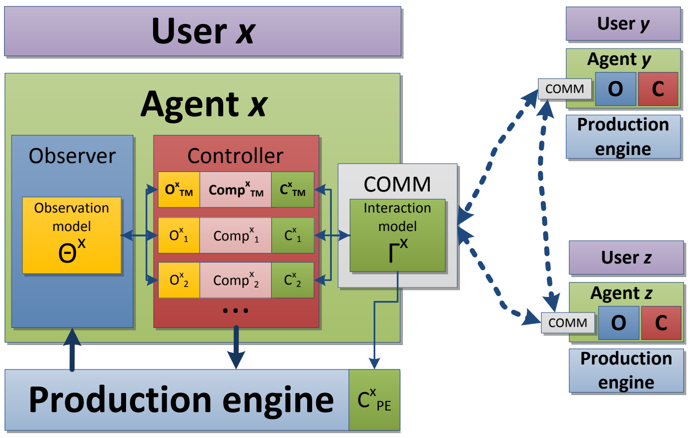


Figure 3.4: The refined agent model including the Trust Management agent component $Comp_{TM}^x$ along with its specification of a set of observables O_{TM}^x and interactions C_{TM}^x.

The most important function of the TM component $Comp_{TM}^x$ is to enable the rating of interaction outcomes with other agents. A component $Comp_a^x$ of an agent x that initiated an interaction c_i with a component $Comp_b^y$ of agent y evaluates the outcome of the interaction when completed. This pair $(c_i, o_{c_i}^j)$ is submitted to the TM component, where it is used to generate an according rating for y's behaviour and consecutively to update the trustworthiness estimation $DT_x^{y,b}$ about y with this rating. On the other hand, a component $Comp_c^x$ that executes some form of trust-based decision making needs to retrieve the values $T_x^{z,d}$ before it can decide whether to interact with the component $Comp_d^z$ of an agent z or not. Again, these values are requested from the TM component.

Besides implementing the trust model and providing other components access to it, the TM component provides a specification of observables O_{TM}^x and interactions C_{TM}^x adding to the observation and interaction models like any other component. However, the specification of observables is demand-driven: The TM component provides information requested by other components. Hence, only exact specifications of other components allow to derive a set O_{TM}^x of TM observables. On the other hand, the TM component needs to provide an interaction that allows other agents to retrieve indirect trust values for an agent from it. C_{TM}^x hence contains the interaction c_{ret_DT} specified as:

$$c_{ret\_DT} = retrieveDT(y, \hat{c}) \tag{3.6}$$

This function then allows any agent to retrieve the direct trust value $DT_x^{y,\hat{c}}$ an agent x has about an agent y in the context $\hat{c}$. This function has two possible outcomes: Either x provides the information or rejects to provide it. The according rating values should however be provided dependent on the ratings for the other interactions from the interaction model Γ^x.

Finally, the utilisation of such a default TM component allows the agents to have a common specification for rating values for the interaction models of a set of default agent components applied in a specific hosting system. Without such a component, the described implementation of the trust model requires the agents to assign a rating value for each interaction outcome of each known agent.

3.1.4 Composition and States of the Hosting System

In this chapter, the hosting system $\mathcal{H}$ has been introduced as open, distributed system without central control, embracing entities that participate in the system by connecting to it via an according client (production engine). These entities have been further specified as agents with an O/C- and component-based approach that encapsulates their functionality, as well as a communication component that allows them to interact based on according protocols. These agent components have been described as controlling the production engine and hence realising the actual functionality of the hosting system. Therefore, all clients that want to participate in the system need to be composed of these components, with the freedom of utilising additional components. The exact realisation of components (the decision-making within) is however not visible among the clients. In summary, the hosting system is specified as the following tuple:

$$\mathcal{H}(t) := \langle ProductionEngine, SystemComponents, AgentSociety \rangle$$

In that definition, the *system components* are a set of components that are required to participate in the system. It is assumed that the specification of a hosting system includes default implementations for each system component, such that an interested potential user can obtain these along with the production engine. The default set of system components is hence composed as:

$$SystemComponents = \left(\Lambda_1^{default}, .., \Lambda_k^{default} \right)$$

with $1..k$ denoting the required system component types. It is assumed, that these system components represent the prevalent behavioural strategies exhibited in the system. Due to the openness of this system, it however depends on the intention of the agent owner whether default component implementations are used. Just as technically versatile users can apply customised components, adapted to their individual preferences, can adversary users introduce agents with component implementations aimed at exploiting or damaging the hosting system. Such components produce interactions that are not desired by the majority of other users. It is therefore in the responsibility of the system designer to provide a Trust Management system that, based on a threat model, identifies adversary behaviour types for the hosting system. The detection of these behaviours has to be realised by the interpretation of the Observer data within the specialised components. In that, the Trust Management component has a special role. The hosting system needs to provide a default implementation $\Lambda_{TM}^{default} \in SystemComponents$ of that component. This implementation encodes the trust model and specifies rating types for behaviours, such that agents can estimate the trustworthiness of other agents.

The hosting system is also defined by a time-dependent *agent society* $\mathcal{A}(t) = \left\{ a_1, .., a_{|\mathcal{A}|} \right\}$ (cf. e.g. [32]) which represents all agents a_i that participate in the system at time t. The agent society comprises agents that incorporate the required system components and hence are available as interactions partners in the context of the hosting system. Agents that enter the system are assumed to know a subset of $\mathcal{A}(t)$. Note here, that for the sake of comprehensibility, this is implicitly assumed throughout this thesis, such that the term $\mathcal{A}(t)$ is used instead of a local subset of $\mathcal{A}(t)$ known by the agent. In addition, the symbol $\mathcal{A}$ denotes the set of all agents that have been members of the society at some point in time t.

Additionally, the hosting system is in a certain state at each time step t. The states are constituted by the agents that are in the system at that time, the agent society $\mathcal{A}(t)$, as well as their relations. A particularly interesting state of the hosting system is that of a **trust breakdown**. Such a trust breakdown is a condition in which a **substantial amount of agents is perceived as not trustworthy within a reference group of agents**. This is formalised by the following function denoting whether there is trust breakdown in the hosting system for a reference group $\mathcal{P}(t)$ at the time t:

$$\mathcal{B}^{\hat{c}}_{\mathcal{P}(t)}(m) := \begin{cases} true, & \text{if } \left|\{y \in \mathcal{P}(t)\} \ : \ RT^{y,\hat{c}}_{\mathcal{P}} \leq 0\right| \ \geq \ m \cdot |\mathcal{P}(t)| \\ false, & \text{else} \end{cases} \tag{3.7}$$

Note here that the aggregation function $agg_{RT}(DT^{y,\hat{c}}_{p_1}, .., DT^{y,\hat{c}}_{p_k})$ (see Eg. 3.2) influences this function, that this lack of trust refers to a trust context $\hat{c}$, and that the definition of a majority of agents is flexible, controlled by the parameter m, with a usual setting of $m \in [\frac{1}{2}, 1]$. The most interesting reference group is that of the entire agent society $\mathcal{P}(t) = \mathcal{A}(t)$: **If the majority of agents in the agent society** $\mathcal{A}(t)$ **of the hosting system is not trustworthy**, this is referred to as **global trust breakdown** $\mathcal{B}^{\hat{c}}_{\mathcal{A}(t)}(m)$.
This definition of a trust breakdown is inspired by the examination of feedback effects of trust-crises in [40]: Here a trust crisis of an agent x towards an agent y can be paraphrased as the loss of trust towards an agent, such that there is a transition from $DT^{y,\hat{c}}_{x} > 0$ to $DT^{y,\hat{c}}_{x} \leq 0$. Such trust crises are described to initiate dynamics in the relationships of agents and lead to *trust earthquakes* due to feedback effects. As is shown in the evaluation of this thesis (see Sec. 5.3) trust crises and global trust breakdowns can occur in open dynamic systems due to attacks of adversary agents. Additionally, the literature on *Organic Computing* teaches us that *emergent system states*, generated by local dynamics and unpredicted at design time, must always be expected in systems that are sufficiently complex. The global trust breakdown, as a result of the complex trust-based interactions within the agent society, is such an emergent phenomenon. If the existence of this phenomenon is expected, and its characteristics are defined (as in Eq. 3.7), it can be observed and countered by the system.

Finally, it is assumed in the remainder of this thesis that agents are only interested in their own performance, whereas aspects like average performance in the hosting system, fairness or robustness are aims of the designer of such a system, but not of particular agents. To reach such system-level goals is a design challenge, because central elements in the system must be avoided. Instead incentive mechanisms have to be applied in order to motivate the self-interested agents in the agent society to participate in the pursuit of these goals. In the following section, it is analysed what makes this design goal particularly challenging and how work presented in this thesis can contribute to such design.

3.2 Trusted Communities for Open, Technical MAS

In Sec. 2.6, work related to this thesis has been summarised and challenging issues have been identified. In the following, it is discussed how these issues can manifest in hosting systems as defined in the previous section. This is followed by the introduction of the MAS organisation *Trusted Community* and its realisation as part of the presented system model. Subsequently, a motivation is provided why this approach can help to increase the performance and robustness of a hosting system despite the discussed issues, and where this is not expected to work.

3.2.1 Challenging Issues in the Hosting System

The hosting system, as defined in the previous section, is a system open to any client with a production engine, agent components to control it and the Trust Management component. Users that join the system provide a utility definition for their agent, derived from the production engine. In addition, agents are self-interested and try and maximise their utility, based on decision-making encoded in their components.

In such a setup, the utility of agents is often defined such that the agents must cooperate in order to maximise it. However, due to the openness of the system, the interactions with other agents involve risks. The decision-making of an agent x hence resorts to trustworthiness estimations $T_x^{y,\hat{c}}$ about potential interaction partners y, often applying trustworthiness thresholds or choosing interaction partners with the highest trust value. The trust values $T_x^{y,\hat{c}}$ are based on ratings of direct and indirect experiences made with these agents. In the system model, this has been formalised: A rating $r_j^i \in \mathcal{R}_x^{y,\hat{c}}$ of agent x about agent y has been described as mapping of the usefulness of a particular outcome $o_{c_i}^j$ for an interaction c_i with an agent component $\hat{c}$.

In the description of the system model, as well as in the related literature, often assumptions are implicitly made that have consequences for the control of such systems:

1. Trust Management in Open Distributed Systems is used to reduce the uncertainty among participants and allow for cooperation. Decision-making based on trust is mostly static (cf. e.g. [24]) with interaction partners being chosen based on high trust values. In that, the dependence of an agent x to generate much evidence about the trustworthiness of another agent y before it will delegate high-risk tasks to y, can be perceived as overhead and lead to missed cooperation opportunities (cf. e.g. [20], [118]). **In the context of a technical system, such as the hosting system, interactions between agents are often affected by the overhead of the TM system and are hence sub-optimal**. Consider for example the utilisation of additional safety means in agent interactions. Additionally, the assignment of high trust values in an evidence-based TM system can also lead to *over-confidence*. This means that an agent's trustworthiness is estimated higher than is justified. Consider the example of agents that build up a high reputation and start to defect once this is reached (cf. e.g. [24]). A high number of positive evidence of their trustworthiness must be compared to a small number of recent negative experiences. While trust models vary in their ability to cope with such cases, the decision-making in agents often is slow to react to such changes when the decisions are made based on the trust values. **In consequence, trust-aware decision-making in a technical system should not rely entirely on trust management to avoid problems with too-much trust and over-confidence.**

2. An agent x makes the decision $D(x)$ to cooperate with agent y, because it expects this interaction i to increase its utility $U^x(t)$. It then initiates the interaction and requests agent y to cooperate with it. Due to the autonomy, the passive agent y can either accept or reject this interaction request, referred to as decision $D(y)$. The most influencing factor on this decision is, that **the passive agent y is assumed to have no *direct* benefit from this interaction**. This means that the interaction does not increase the utility $U^y(t)$ of y. On contrary, the interaction usually involves y to perform a task for x and hence block some of its resources, which can even result in a lower utility of y. Because the agent y is self-interested, this means that there must be some incentive for it to cooperate. The reasoning within $D(Y)$ must then determine whether this incentive is sufficient, for example because of the prospect of a *future benefit* ($U^x(t+a) > U^x(t)$) due to this interaction i. It is commonly agreed that a Trust Management system can be applied as an incentive mechanism in such a system of self-interested agents. The fundamental idea behind this is: If cooperation increases the reputation of passive agents and active agents require to have a high reputation to find willing interaction partners, then it pays off to be cooperative. However, such **a self-reinforcing Trust Management system is complex and negative emergent phenomena, such as a breakdown of this system, must be expected**. Consider for example the common assumption in such TM Systems that agents with a high reputation are preferred over agents with a low reputation. Consequently, these agents receive more interaction requests. But as the capacity to perform these requests is limited, the agents start to reject interactions. In succession, this leads to a reputation loss until the agents are not attractive as interaction partners any more. The collective and self-interested behaviour of agents initiating interactions hence leads to oscillating reputation values, referred to as *reputation damage problem* (cf. e.g. [24]). Another case analysed in the literature is that of a *paralysed* agent society (cf. e.g. [29]): Here the volatility of agents in an open system leads to short-lived trust relationships and low overall reputation of agents, blocking high-risk interactions because of high trust requirements. Such effects can be even reinforced by adversary agents deliberately enforcing such system states and in consequence even lead to chain reactions reducing the expressiveness and significance of reputation values. Despite these effects, only few authors assume that the underlying TM system can fail entirely, and most approaches to decision-making with trust do not account for these system states.

3. The capability of an agent y to perform a task delegated to it by another agent x does not necessarily imply that y can determine what the consequences are. Due to the autonomy of the agents, a passive agent can for example not differentiate whether x really requires the result for this task or just intends to block y's resources. Also x does not know whether it has been the only agent requested to perform this task, or if y has delegated the task to a group of other agents. As such, the interaction outcome cannot be rated with the Trust Management system without the cooperation with other agents. But **as a passive agent is self-interested and has no direct utility gain from such an interaction, it will in general not invest an effort to determine how useful this interaction is for x, or how detrimental it is for other agents or the system as such**. If left unattended, this can lead to exploitation or damage to the whole system from adversary agents and reduce the performance of participating agents. This especially refers to *collusion* among adversary agents. To prevent these system states, agents need

to consider these effects in their decision-making and restrain from pure utility-based considerations. Only if agents coordinate and share their local observations, such undesired interactions can be detected and prevented. Again, incentives must be provided for self-interested agents to cooperate in this matter. This is also highly relevant for cases where only a group performs such regulatory cooperation, while the other system participants benefit from it without contribution (referred to as *second-order* freeriding in the literature, cf. e.g. [189]).

These are challenging issues in the design of decentralised control mechanisms for such systems. In the following, it is described, how these challenges are addressed in this thesis.

3.2.2 Trusted Community - An Introduction

So far, approaches to the control of Open Distributed Systems have been discussed in this thesis and their general assumptions and challenges have been described. In the following, a novel approach is introduced, inspired by system control with MAS organisations, Trust Management and Organic Computing. This is concluded by a discussion about the contribution of this approach with respect to the summarised challenging issues.

The control structure proposed in this thesis, the **Trusted Community (TC)**, comes in the form of a MAS organisation. This organisation is formed in a self-organised process among agents from the agent society $\mathcal{A}$ of a hosting system. *Self-organised* here refers to the fact that no TCs exist in a hosting system when it is deployed, and that agents form such TCs, without the involvement of any central entity, when they determine that it is beneficial for them. This self-interested decision is part of a larger decision-making process of the agents. The Trust Community organisation is then based on the following design concepts:

The TM in a system is a good instrument to determine suited interaction partners. But once enough evidence of trust has been received, the TM implies overhead, as no further evidence is needed. Hence, instead of making risk-aware decision-making, agents should optimise their interactions by resigning safety measures introduced because of the uncertainty with respect to interaction partners in an open system. In that, it is assumed that such interactions are always preferable over risk-aware interactions in the context of technical hosting systems, because they increase the performance of an agent. Consider for example *task replication* in Open Desktop Grid Systems: An agent replicates tasks to compensate for unreliable interaction partners. However, task replication involves the utilisation of additional resources and increases the workload in a system, prolonging the response time for further delegated tasks. By resigning replication, an agent here is able to increase its performance. Obviously, the abandonment of safety means based on a high trust values of interaction partners involves the risk of behavioural change and exploitation. The aim is hence to create an environment between agents in which there are incentives to cooperate that go beyond reputation gain. The most convincing incentive for such a case is reciprocity-based. Only if agents do not defect when performing interactions without safety means are they allowed to perform such interactions themselves. **In other words, only if agents cooperate to provide an optimal performance to other agents have they the opportunity to receive an optimal performance themselves**. To construct such an environment in an open system, agents have to form an organisation and agree on common rules for it. In this thesis, the following assumptions are made:

1. The ultimate goal of agents in the hosting system is to increase their utility. If an organisation provides the possibility to increase it by means of optimised interactions, then a rational agent must consider joining it. Consequently, agents that find that the organisation is not suited to increase their performance will not join at all or leave it when being members already.

2. The motivation of agents to form such an organisation is based on their self-interest. As such the organisation does not postulate any group goals, such as e.g. a coalition, but provides an environment for agents to achieve their own goals, such as e.g. a congregation.

3. Only agents that have proved their willingness to cooperate are prospective members. Such experiences are made among groups of agents with the reputation incentive reinforcing relationships and partitioning the system into clusters of heavily interacting agents. An organisation is therefore expected to apply subjective trust as formation membership criterion and agents are expected to cluster into several groups of mutually trusting agents.

4. This membership criterion cannot be enforced if agents are allowed to become members at their own will. Rather do the members of a TC need to decide whether an unassociated agent is allowed to join.

5. Member agents do what best suits their performance, hence they adhere to interacting with non-member agents in case this is beneficial.

6. The agent society in the hosting system is composed of heterogeneous agents with varying competence and willingness to cooperate with other agents. It is hence not to expect that all agents in the society are suited as members of such an organisation.

7. The hosting system is open and can comprise thousands of entities. The number and composition of such an organisation is not constrained in any way by the hosting system. Especially the case of a single organisation is not assumed. Such an organisation would render the hosting system to a centralised system, along with all disadvantages such as poor scalability.

The organisation characterised by these assumptions is referred to as *Trusted Community* from here on. A schematic overview is depicted in Fig. 3.5. The Trusted Community is formed by agents with strong mutual trust relationships (see Sec. 3.1.3) and provides the benefit of interactions with less overhead. Trust Management is in general not utilised between members. Firstly, this protects members of the Trusted Community from the failure of the TM System: The decision-making of member agents requires trust values only for non-members. Inbound interactions between members are still executed despite such an abnormal system state. Secondly, the incentive to cooperate between members is based on the benefit of the membership as such: If agents defect, this affects not only their reputation as is usual in the hosting system, but more importantly, they risk to lose the TC membership and the benefits of it. This allows the decision-making of passive agents to directly link their decision to cooperate with a utility gain in future interactions as active agent: With a high probability it will find interaction partners among the other TC members. In sum, this approach is a means to cope with the phenomena of over-confidence and too-much trust discussed in the previous section and provides robustness towards TM breakdowns for TC member agents. In addition, it creates an environment in which agents can perform optimised interactions.

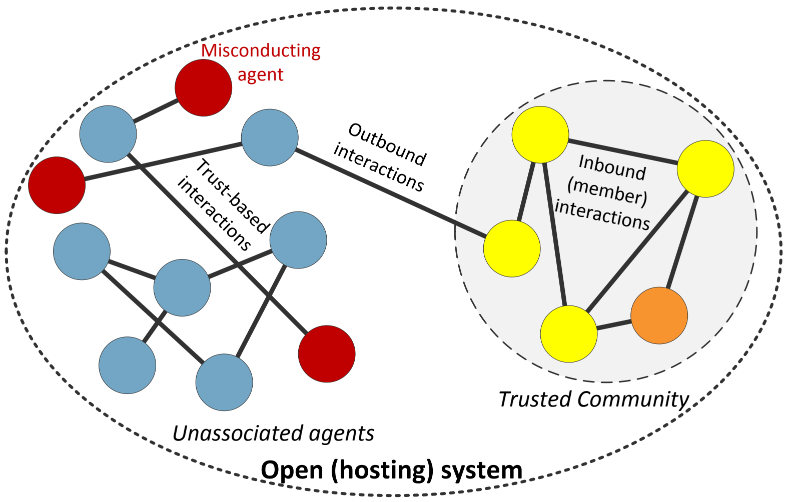


Figure 3.5: System view on a hosting with a single TC. Unassociated agents, cooperative (blue) as well as adversary (red) are not part of the TC. TC members (yellow) and TC manager (orange) perform *inbound* interactions among themselves and *outbound* interactions with unassociated agents.

A Trusted Community, with the definition so far, could however be exploited: As discussed in the previous section, passive agents in an interaction cannot always perceive what the effect of the interaction is. Instead, they need to cooperate in order to detect such behaviours as collusion. The incentive has been discussed: It is the benefit of remaining a member of the TC. However, there is a requirement for coordination to allow for the goal-oriented cooperation of self-interested member agents in this matter. Most importantly, interactions that endanger the successful operation of a TC, such as emerging abnormal system states, must be regarded here. As this regulatory responsibility is restricted to a limited section of the hosting system, the members of a TC, this control is more feasible than the control of the whole system as such in this respect. The realisation of this self-organised control is based on roles that each member of the TC can be assigned. The assignment of roles to agents, along with the requirement to control the access to the TC, is realised with a second layer of hierarchy: One of the member agents of a TC is empowered to be a TC Manager (TCM). This agent is responsible for the coordination of the members with respect to the self-management of the TC. In that, it has the regulatory aim to maintain the operation of the Trusted Community. However, such agent is still self-interested and the execution of management tasks must hence be distributed among the members with the TCM merely coordinating the effort. Note here, that each member of a TC can become its TCM.

The application of TCs in a hosting system improves the interaction efficiency among the members, as well as their robustness towards emergent abnormal system states, such as the breakdown of the Trust Management system. As these agents are part of the agent society, and system-wide performance and robustness metrics are often defined over the aggregated performance of single agents, TC application can hence increase the performance and robustness of the hosting system as such. This is especially true for agent societies in which the majority of agents self-organise into multiple TCs operating in parallel. These claims are evaluated in the thesis with the help of an exemplary hosting system from the domain of Open Desktop Grid Systems.

Finally, the application of Trusted Communities in a hosting system allows for the following interpretation: Agents are allowed to form a single TC, with each agent in the society either being a member or not. Alternatively, agents are not restricted in TC formation which can result in the independent formation of multiple TCs as depicted in Fig. 3.6. In this thesis, the latter assumption is made

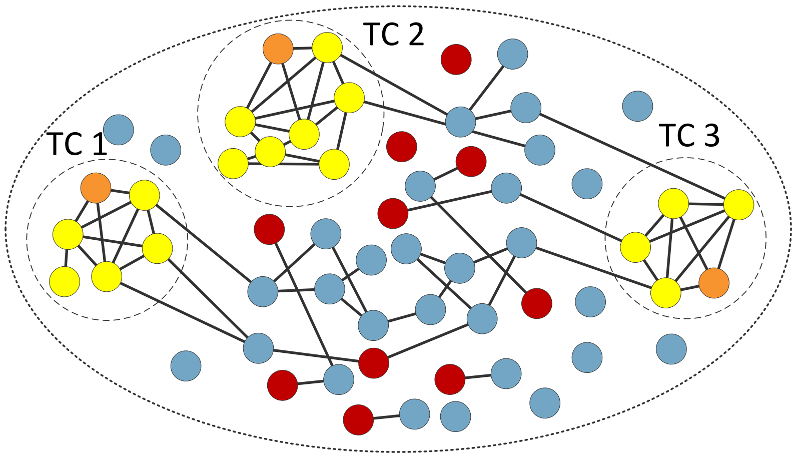


Figure 3.6: System view on a hosting system in which multiple distinct TCs have formed and operate independent of each other (yellow colour denotes members, orange denotes the respective TC managers). Unassociated, cooperative (blue), as well as adversary agents (red) are not part of any TC.

for the following reasons: The formation of multiple TCs in a system adopts the idea of modularity - the failure of a single Trusted Community is contained within this community and the effect on the whole system is limited. This is consistent with the argumentation of robustness (cf. e.g. [132]). Besides, multiple TCs are a means to enable scalability - a single TC can only support a limited number of members before the overhead renders the operation inefficient. In open systems, where the number of participants cannot be estimated at design time, this is an important issue. Additionally, agents in large hosting systems interact only with a comparably small subset of other agents. They thus have only locally the opportunity to develop strong mutual trust relationships, which are the main criterion for forming and joining a TC. Finally, there is no substantial reason to limit the self-organisation of agents into TCs in an open system by introducing a fixed number of allowed TCs. When speaking of the application of Trusted Communities in a hosting system in this thesis, it is hence always assumed that multiple TCs are involved.

In summary, the proposed approach in this thesis is a novel MAS organisation called *Trusted Community*. A TC is characterised as long-enduring organisation between mutually trusting member agents and a Trusted Community Manager responsible for the coordination of regulatory means. Trust Management is applied to form a TC, however interactions enabled by TC membership do not require TM to be performed. In the following, the application of the TC approach in a hosting system is defined. The focus is here on the presentation as part of the system model defined in this chapter. The actual design of a Trusted Community is then detailed in the following chapter.

3.2.3 The Application of Trusted Communities in a Hosting System

The Trusted Community has been introduced as a result of the self-organisation of agents within a hosting system. Besides, the influence of TC membership on the reasoning of agents about more efficient interactions has been introduced. The abilities to form, maintain, manage, and participate in TCs, as member, as well as TCM, must hence be grounded in agent capabilities. In conformance with the system model described in the previous section, these capabilities are modelled as dedicated agent component. This additional agent component, as depicted in Fig. 3.7, is referred to as *TC-Organisation agent component* $Comp^x_{TCO}$.

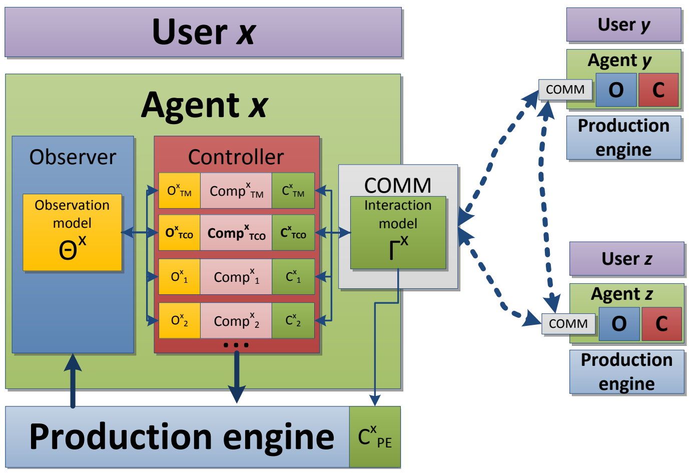


Figure 3.7: Refined agent model including TM and TC organisation agent components. The TC organisation component specifies a set of observables O^x_{TCO} and a set of interactions C^x_{TCO} required for the decision-making of the TC approach.

Similar to the Trust Management agent component, this component is provided in a configurable default implementation, such that each agent in the hosting system can form or join a TC. The default implementation of this component is then referred to as $\Lambda^{default}_{TCO} \in SystemComponents$, being a system component like the default implementation of the TM component $\Lambda^{default}_{TM}$. The observables O^x_{TCO} of this component are information required for the TC-aware decision-making of an agent x, for example the set of agents with a strong mutual trust relation with x. The set of interactions C^x_{TCO} provided by this component is composed of optimised interactions interfacing interactions provided by the other components, as well as interactions required for the communication of a TC member or manager. Such a set contains for example the interaction to assign a TC role to an agent, or to request an inbound interaction with it.

The TC organisation component is, like any other agent component, part of the Observer/Controller loop. An agent can hence adapt the parameterisation of this component at runtime, based on the evaluation of situation descriptions. Note here that the TCM extends this system of observation and control by introducing a second O/C layer (cf. e.g. [134]) by coordinating information retrieval of its member agents and processing this information to regulate the operation of a TC.

The detailed configuration, decision-making and composition of the observables and interactions set are provided in Chapter 4. In the following, the applicability limitations of TCs are discussed.

3.2.4 Applicability Limitations of TCs

Trusted Communities are designed for the application in a hosting system, being an open technical system populated by an agent society. However, this is a broad class of systems and some of these system instances hold conditions under which TCs cannot be formed or maintained. In these conditions, the extension of agents with the TC agent component will thus only introduce additional overhead without improving the performance or robustness of the respective system.

In the following, system conditions are discussed under which the Trusted Community approach, realised via the default TC agent component $\Lambda_{TM}^{default}$, is not expected to work as described in the remainder of the thesis.

Decision-making not considering trust The utilisation of Trust Management for decision-making about interactions is only reasonable where there is risk involved (cf. e.g. [20]). It is hence assumed that the components of agents in the system consider trust, and that this involves overhead. The interactions in a hosting system are hence assumed to be sub-optimal in comparison to those executable in closed systems. If this does not hold, Trusted Communities cannot grant a benefit to members and hence the actions of the TC organisation component must be considered overhead.

Substantial exploitation of openness The previous section has defined agents as being composed of agent components that encapsulate their functionality. It has been assumed, that due to the openness of the system, users are allowed to modify the composition of their agents. This has been somewhat constrained by the assumption that the designer of such a hosting system provides default implementations of these required system components and that the majority of users adheres to the utilisation of these components. The degree to which the openness of the system is exploited, for example by users participating with adversary agents, is hence assumed to be low enough that the Trust Management system can handle this in general. This is not to say, that there are no system states in which this does not hold (as described in the challenges in Sec. 3.2.1), in fact one of the key contributions of the TC approach is its ability to cope with these system states. As a consequence, the application of Trusted Communities in not feasible, if the system is largely populated by agents composed of detrimentally modified agent components. This is however hard to quantify as it depends on the modifications and the exact type of production engines and components. Also, as some authors in the literature have stated (cf. e.g. [185]), the openness of the system can be qualified by the difficulty of deploying such modified agents. This difficulty is hence correlated with the probability of successful TC application in a hosting system.

Additionally, security violations from users can exploit or damage a hosting system in ways that compromise agents and hence prohibit the successful application of TCs. There is a general agreement in the literature that such security threats cannot be compensated by Trust Management (cf. e.g. [20]) and that indeed Trust Management and security are different concepts. It is hence assumed that the hosting system utilises a security subsystem preventing such violations.

Lack of enduring trust relationships The formation of Trusted Communities depends on strong mutual trust relationships. However, not all systems or system states provide for such relationships. Consider for example the case of a hosting system with a high degree of volatility within the agent society: Instead of developing strong trust relationships with few agents, agents are forced to interact with a great number of different agents, the relationships often remaining shallow (cf. e.g. [76]). Also, a high frequency or intensity of disturbances to the system (e.g. due to the exploitation of openness), can prevent the emergence of strong trust relationships. Though the TC is suited to work in system states, where such issues occur, the initial formation of TCs requires system states without such issues.

Asymmetrical trust relationships In hosting systems or system states where trust relationships are seldom symmetrical (bilateral), Trusted Communities cannot be formed, either. Such asymmetrical relationships between an agent x and an agent y are characterised by conditions where $DT_x^{y,\hat{c}} \gg DT_y^{x,\hat{c}}$. Such constellations occur in systems where the heterogeneity in agent capabilities is very high. Consider for example a system where agents have highly varying resources at their disposal. When the number of agents with high amounts of resources is low, those agents are constantly requested to interact and building up reputation very quickly, due to the self-reinforcement effect of reputation (cf. e.g. [24]). On the other hand, these agents have difficulties in establishing strong relationships to other agents, as these have substantially lower resources and it is harder to choose between them. These constellations allow for the formation of TCs only, if the asymmetry can be broken, for example through cooperation among over-provisioned resource owners.

Inappropriate Trust model The trust model encoded in the respective agent component has been described as integral part of the hosting system. The previous section has detailed the requirements on this model, introducing the trust aggregation functions agg_{DT}, agg_{RT} and agg_T. Formally, these aggregation functions allow for the specification of binary trust values, i.e. it is only discerned between trustworthy and not trustworthy. This is particularly undesired in this context, as it leads to a partitioning of the agent society into only two groups. In consequence, the group of trustworthy agents immediately qualifies as Trusted Community without further differentiation. This becomes especially challenging when new agents enter the system: These newcomers have to be either accepted as members of a TC instantly, violating the core idea behind the concept, or be marked as not trustworthy. In the latter case, these agents will have hardly any opportunities to gain in reputation as the interactions of TC members will be mainly among themselves.

Inappropriate TC agent component configuration As described above, the agent capability to interact as TC member or manager is installed as agent component $Comp_{TCO}^x$. While the details of the architecture and composition of this component have not yet been presented (this is done in the following chapter), it has been stated that this agent component is configurable and that a default implementation $\Lambda_{TCO}^{default}$ is expected to be deployed along with the other required system components. Without advancing too much into the properties of such an implementation, it is adequate to note here that such an implementation can be inappropriately configured as well, just like the TM component. Such an improper configuration can affect the dynamics of TC formation and maintenance, and make it impossible for agents in the society to self-organise into Trusted Communities.

3.3 Summary

This chapter has introduced the class of open, distributed, technical systems in which agents make decisions on behalf of their users in that they control the software to participate in such a system.

The model for such a *hosting system*, as well as for single participants has been described: The system is constituted by a time-dependent composition of participants that belong to a user, operate in the system with a production engine, strive for increasing their respective utility, and apply agent technology for the decision-making in this environment. The chapter has continued by defining how an agent is understood in this thesis by presenting an agent model. The main building blocks of this agent model have been presented: An *Observer* for information gathering and interpretation, a *Controller* for the adaptive decision-making based on this information, and a communication interface for the specified interactions among the agents. The *Controller* has been further elaborated, and a model based on *agent components* constituting the *Controller* has been laid out. Here, the focus has been especially on the specification of a set of *observables*, as well as *interactions* for each agent component. Additionally, the aggregation of all these sets provided by any agent component applied has been defined as the *observation*- and respectively *interaction model*. These two models have been described as determining the entire information an agent requires for its decision-making, as well as the entire interaction opportunities it allows other agents in the system. Subsequently, a *Trust Management* system has been defined. It allows the agents to reduce their uncertainty about interaction partners by analysing, rating and comparing their behaviour. The requirements on such a TM system have been formally specified, and the encapsulation in a dedicated *TM agent component* detailed. The first part of the chapter has then been concluded by remarks about assumptions regarding the composition of the *hosting system* with respect to the applied *agent components* and the specification of an *agent society* for the system.

The second part of this chapter, has been started with the analysis of challenging issues in such *hosting systems*. This analysis has been based on observations about approaches in the related literature on control of such systems. The main issues identified have been

- the sub-optimality of interactions due to overhead of TM and associated safety means,
- the unconditional reliance on a working TM system and hence vulnerability towards emergent phenomena (such as a trust breakdown),
- the lack of incentives for agents to interact passively when they have no utility gain from the interaction,
- the lack of incentives for agents to observe the quality of their interactions and their possibly detrimental consequences for the operation of the system.

This thesis has continued by introducing the *Trusted Community* approach in order to account for these issues. The main guideline for the operation of an according agent organisation has been identified as the self-organised establishment of a closed-system environment for the members. This closed system, guaranteed by access control based on trustworthiness, is then used to allow for optimised interactions among the members. These interactions are optimised, because of the abandonment of safety means. The approach has been detailed by the presentation of fundamental assumptions that further characterised Trusted Communities. In the following, the embedding of TC-reasoning in a dedicated *agent component*, as defined in the first part of the chapter, has been described. The chapter has then concluded by the examination of the applicability limitations, considering such issues as *asymmetrical trust relations*, and *inappropriate trust models*.

The following chapter presents the design of the Trusted Community approach in far more detail than the introduction provided above, and elaborates on the contribution of this approach as a solution to the examined issues in systems as defined in this chapter.

4 | Trusted Community - A Novel MAS Organisation

In the previous chapter, the system model for the application of the Multiagent organisation *Trusted Community (TC)* in a hosting system has been presented and all relevant submodels have been elaborated. This chapter is dedicated to the Trusted Communities as such. After an introduction, the Trusted Community organisation is formally defined. Then, the delegation of control to a Trusted Community Manager (TCM) is explained and its responsibilities are laid out. In the following, the life cycle of the organisation is presented which includes the designation of the most important agent decisions as strategies. These strategies are then detailed, including descriptions of basic and advanced realisations, as well as a default configuration. This configuration is independent of the application scenario and allows agents to form Trusted Communities that express the described properties. The chapter continues with an explanation and generic classification of organisation benefit strategies and concludes with the discussion of the dynamic management of Trusted Communities.

4.1 Introduction

When realising technical systems based on an open Multiagent System model, challenges occur through agents that join and leave the system arbitrarily, and show various types of behaviours ranging from cooperative to selfish or even adversary. In the previous chapter, it has been discussed that trust management can be applied to model the relationships between agents and that these trust relations can be used to improve the performance and robustness (towards misconducting agents) of these systems. However, the application of Trust Management (TM) alone has been found lacking, especially in cases of abnormal system states and sub-optimal agent interactions. To address these issues, a novel approach has been proposed to capitalise on enduring and strong mutual trust relationships between agents. By means of a self-organised process a higher form of organisation between trustworthy agents is sought. This organisation is referred to as *Trusted Community* and is characterised by a decentralised, yet hierarchically managed, operation that provides performance benefits to its members by improving interaction efficiency, information sharing and cooperation. The management of a TC allows for the optimisation of the composition, as well as regulatory actions to preserve the stability of the organisation. This is essential, as composition and stability of an organisation consisting of self-interested members can easily become fragile.

Trusted Communities have been described as an organisation partitioning the hosting system and establishing subsystems in which the members can avail of optimised interactions. The capability of the agents to form, maintain and interact in such TCs has been realised as agent component in conformance with the system model (see Sec. 3.2.3). This component is responsible for the decision-making with respect to Trusted Community self-organisation. Central aspects of this decision-making concern the decisions whether to form, join or leave a Trusted Community from the member view, as well as how to maintain the operation of a TC despite adversary actions, changing behaviours and abnormal system states. In this chapter of the thesis, this decision-making and the overall design of the TC agent component are elaborated and the resulting dynamics are analysed. This is started with a formalisation of the key terms and concepts in the following section.

4.2 Formal Definition

Trusted Communities form among agents from the agent society $\mathcal{A}$ and are persistent in the hosting system $\mathcal{H}$ until their dissolution. The definition of the hosting system, as presented in Sec. 3.1.4, is hence extended such that the hosting system is understood as the following tuple from here on:

$$\boldsymbol{\mathcal{H}(t) := \langle ProductionEngine, SystemComponents, \mathcal{A}(t), \mathcal{O}(t)\rangle} \tag{4.1}$$

The new set $\mathcal{O}(t)$ contains all agent organisations that operate (are already formed and not yet resolved) in the hosting system at time t. In general, this set can be composed of organisations from several types $\mathcal{T}$ (*coalitions*, *clans*, *TCs*), such that:

$$\mathcal{O}(t) = \bigcup_{type \in \mathcal{T}} \mathcal{O}^{type}(t)$$

In the remainder of this thesis, it is however implicitly assumed that organisations are applied exclusively and that $\mathcal{O}(t)$ denotes the set $\mathcal{O}^{TC}(t)$ of TCs in the hosting system[1].

A Trusted Community $TC_i(t) \in \mathcal{O}(t)$ is further defined as the following tuple at time t:

$$\boldsymbol{TC_i(t) := \langle \mathcal{M}_{TC_i}(t), tcm_{TC_i}(t), \Psi_{TC_i}(t)\rangle} \tag{4.2}$$

in which:

- $\boldsymbol{\mathcal{M}_{TC_i}(t)} \subseteq \mathcal{A}$ denotes the **members of a TC**,
- $\boldsymbol{tcm_{TC_i}(t)}$ denotes the **TC manager**, and
- $\boldsymbol{\Psi_{TC_i}(t)}$ denotes a set of **roles** that can be assigned to members in this TC.

In that, the members of a TC are from the agent society $\mathcal{A}$, but not necessarily online in the hosting system at the time t. Also, the assignment of roles to members is in the responsibility of the TC manager.

Additionally, at each time t, the agent society $\mathcal{A}$ of the hosting system $\mathcal{H}$ is partitioned into the two sets of:

- **Unassociated agents** $\boldsymbol{\mathcal{U}^{\mathcal{H}}(t)}$: agents that are not member in any TC, and
- **TC members** $\boldsymbol{\mathcal{M}^{\mathcal{H}}(t)}$: agents that are a TC member.

[1]The comparison with the *clan* approach in the evaluation introduces a set $\mathcal{O}^{Clan}(t)$ of clans being applied exclusively in the hosting system, see Sec. 5.3.1

Furthermore, the Trusted Community approach requires the following properties to hold:

Each agent in the agent society is part of either set $\mathcal{U}^{\mathcal{H}}(t)$ or $\mathcal{M}^{\mathcal{H}}(t)$:

$$\mathcal{U}^{\mathcal{H}}(t) \cup \mathcal{M}^{\mathcal{H}}(t) := \mathcal{A}\ ,\ \mathcal{U}^{\mathcal{H}}(t) \cap \mathcal{M}^{\mathcal{H}}(t) := \emptyset,$$

the set of TC members is composed of all agents that are a member in any of the operating TCs:

$$\mathcal{M}^{\mathcal{H}}(t) := \bigcup_{TC_i(t) \in \mathcal{O}(t)} \mathcal{M}_{TC_i}(t),$$

the initial ($t = 0$) composition of the hosting system does not contain any operating TCs:

$$\mathcal{U}^{\mathcal{H}}(0) := \mathcal{A}(0)\ ,\ \mathcal{M}^{\mathcal{H}}(0) := \emptyset,$$

each operating TC is composed of at least two members at any time t:

$$\forall TC_i(t) \in \mathcal{O}(t) : \left|\mathcal{M}_{TC_i}(t)\right| > 1,$$

each agent is always either the member of a single TC or unassociated (exclusive *OR* $\veebar$):

$$\forall a \in \mathcal{A}(t) : \left(\exists! TC_i(t) \in \mathcal{O}(t), a \in \mathcal{M}_{TC_i}(t)\right) \veebar \left(a \in \mathcal{U}^{\mathcal{H}}(t)\right),$$

a TC has always either no TCM, or one of the members is the TCM:

$$\forall TC_i(t) \in \mathcal{O}(t) : tcm_{TC_i}(t) = \emptyset \vee tcm_{TC_i}(t) \in \mathcal{M}_{TC_i}(t).$$

Dynamics and operations

Additionally, the following operations define the dynamics in the composition of these sets and tuples:

The **formation of a new Trusted Community** $TC_i(t)$ by a group of unassociated, founding agents $\mathcal{F} \subseteq \mathcal{U}^{\mathcal{H}}(t)$ is denoted by:

$$\boldsymbol{\mathcal{O}(t) \lhd^{\mathcal{F}} TC_i(t)} \tag{4.3}$$

with:

$$TC_i(t) \notin \mathcal{O}(t) \wedge TC_i(t+1) \in \mathcal{O}(t+1) \wedge \mathcal{M}_{TC_i}(t+1) = \mathcal{F}$$

$$tcm_{TC_i}(t+1) = \emptyset$$

$$\mathcal{U}^{\mathcal{H}}(t+1) = \mathcal{U}^{\mathcal{H}}(t) \setminus \mathcal{F} \wedge \mathcal{M}^{\mathcal{H}}(t+1) = \mathcal{M}^{\mathcal{H}}(t) \cup \mathcal{F}$$

These founding agents are then the initial members of the TC, and the TC does not have a TCM. On the other hand, the **dissolution of an operating Trusted Community** $TC_i(t)$ by its manager $tcm_{TC_i}(t)$ is denoted by:

$$\boldsymbol{\mathcal{O}(t) \rhd^{tcm_{TC_i}(t)} TC_i(t)} \tag{4.4}$$

with:

$$\mathcal{U}^{\mathcal{H}}(t+1) = \mathcal{U}^{\mathcal{H}}(t) \cup \mathcal{M}_{TC_i}(t) \wedge \mathcal{M}^{\mathcal{H}}(t+1) = \mathcal{M}^{\mathcal{H}}(t) \setminus \mathcal{M}_{TC_i}(t)$$

Apart from the description of global dynamics, the dynamics within TCs are needed: The $\oplus^{tcm_{TC_i}(t)}$-operator is applied to denote the **inclusion of an agent** $a \in \mathcal{A}(t)$ in $TC_i(t)$. The membership is granted by $tcm_{TC_i}(t)$, the TCM of the TC:

$$\boldsymbol{TC_i(t) \oplus^{tcm_{TC_i}(t)} a} \tag{4.5}$$

with:

$$a \notin \mathcal{M}_{TC_i}(t) \wedge a \in \mathcal{M}_{TC_i}(t+1)$$

$$\mathcal{U}^{\mathcal{H}}(t+1) = \mathcal{U}^{\mathcal{H}}(t) \setminus \{a\} \wedge \mathcal{M}^{\mathcal{H}}(t+1) = \mathcal{M}^{\mathcal{H}}(t) \cup \{a\}$$

The opposite operation, the **exclusion of an agent** a from a Trusted Community $TC_i(t)$, is denoted with the operator $\ominus^x$, such that:

$$\boldsymbol{TC_i(t) \ominus^x a} \tag{4.6}$$

with:

$$a \in \mathcal{M}_{TC_i}(t) \wedge a \notin \mathcal{M}_{TC_i}(t+1)$$

$$x = a \veebar x = tcm_{TC_i}(t)$$

$$\mathcal{U}^{\mathcal{H}}(t+1) = \mathcal{U}^{\mathcal{H}}(t) \cup \{a\} \wedge \mathcal{M}^{\mathcal{H}}(t+1) = \mathcal{M}^{\mathcal{H}}(t) \setminus \{a\}$$

The exclusion from a TC can hence be either performed by a member agent that wants to become unassociated again, or as regulatory action by the TCM.

Finally, the **assignment of a role** $r \in \Psi_{TC_i}(t)$ to an agent a by the TCM of the shared Trusted Community $TC_i(t)$, is denoted with the operator $\leadsto^{tcm_{TC_i}(t)}$, such that:

$$\boldsymbol{r \leadsto^{tcm_{TC_i}(t)} a} \tag{4.7}$$

with:

$$a \in \mathcal{M}_{TC_i}(t), r \in \Psi_{TC_i}(t)$$

4.3 Organisation Benefit Strategies

The Trusted Community approach has been designed around the key concept of providing an environment for self-interested agents in which they can execute interactions among themselves that resemble those in a *closed* system, hence without applying safety means. Additionally, an organisation composed of mutually trusting agents provides opportunities to share information and cooperate among members. In the remainder of this thesis, agent strategies employing these benefits to increase the utility of the executing agent are referred to as *Organisation Benefit Strategies*. The agent utility $U^x(t)$ of an agent in the society of a hosting system is dependent on the utilised production engine (e.g. a Desktop-Grid Client or VANET client etc.). Therefore, the exact realisation of organisation benefit strategies aiming to improve this utility is likewise dependent. However, it is in general possible to group organisation benefit strategies according to the following classes:

Interaction efficiency: These strategies regard interactions that are optimised versions of system component interactions executable among each agent in the hosting system. Optimisation here refers to subadditive costs or superadditive outcome of these interactions, such as the abandonment of safety means. Consider the set of optimised interactions $\left\{c_1^{opt}, .., c_k^{opt}\right\}$ for a hosting system modelled in compliance with the system view defined in Sec. 3.1: Each optimised interaction c_i^{opt} is derived from a non-optimised interaction $c_j \in C_k^x$ provided by the default implementation of a system component $\mathcal{C}omp_k^x$ of the hosting system $\mathcal{H}$, such that $\Lambda_k^{default} \in SystemComponents$. These interactions are executable only between TC members, which is to be guaranteed by the decision-making of the Trusted Community organisation component $\mathcal{C}omp_{TCO}^x$. To allow for this control, optimised interactions are provided by this component, such that $\left\{c_1^{opt}, .., c_k^{opt}\right\} \subset C_{TCO}^x$. As such, these interactions extend the interaction model of the agent with $C_{TCO}^x \subset \Gamma^x$. Agents can then utilise optimised versions of system component interactions in inbound interactions as TC members and non-optimised versions of the same interactions when performing outbound interactions with unassociated agents. Consider for example the case that highly trusted agents do not apply full task replication when interacting with each other in a Desktop Grid System (see also Sec. 5.2.1).

Cooperation: Efficient interactions, exclusive to TC members, represent an incentive mechanism for cooperation. This willingness to cooperate among members can be further exploited: Interactions that have not been considered by system components because of their high risk in open environments, their scalability, or their demand for coordination can be realised within a Trusted Community. Consider for example the cooperation of agents to detect and avoid colluding agents, or to collectively observe the hosting system in order to perceive abnormal system states.

Cooperation strategies for TC members are realised by the provision of a set of additional interactions $\left\{c_1^{coop}, .., c_k^{coop}\right\}$ by the TC organisation component $\mathcal{C}omp_{TCO}^x$,such that $\left\{c_1^{coop}, .., c_k^{coop}\right\} \subset C_{TCO}^x$. Unlike the formally defined optimised interactions, these interactions are not derived from existing interactions. They have however in common that they are performed exclusively among members, despite their inclusion and visibility in the interaction model Γ^x of an agent. The decision-making whether and how to participate in such cooperative interactions, realised with cooperation strategies, is under the control of the TC organisation component. Note here that in general, such strategies can only be formulated for a specific hosting system $\mathcal{H}$.

Information sharing: The capability and willingness of agents to share information with each other is an essential design principle of Multiagent Systems in general, and the concept of the hosting system in particular. Many decision-making strategies within agent components require the processing of information about potential interaction partners, be it opinions about the trustworthiness of other agents, or information needed to derive the competence of an agent with respect to the delegation of a task. On the other hand, the autonomy of agents, here their exact implementation of agent components, prescribes the privacy of information and the local control of their transfer. This is due to the fact that agents providing private information about their internal state to other agents, do not know whether this information is used to exploit them. Additionally, self-interested agents do not gain in utility by providing information to agents actively requesting it. In consequence, the control of information transfer in a hosting system is expected to be rather conservative and sub-optimal for the local decision-making of agents.

This is where a closed environment like the Trusted Community can further exploit the incentive of optimised agent interactions and strengthen the exchange of private information among member agents. Interactions of this class hence provide TC members the opportunity to request additional information from members. Information providers on the other hand can assume that they are not being exploited if such exploitative actions are perceivable and lead to the exclusion of the TC. An example of sensitive information, with access restricted to members, are personal observations (*local world model*) which could be abused by untrustworthy agents. Additionally, information sharing can be an auxiliary enabler for the other classes of organisation benefit strategies discussed above. Consider for example cooperation strategies relying on additional information in order to coordinate the workflow of the cooperating agents.

In sum, the class of information sharing strategies implements a set $\left\{c_1^{inf}, .., c_k^{inf}\right\} \subset C_{TCO}^x$ of additional interactions as part of the interaction set of the TC component. These interactions are then included in the interaction model Γ^x of an agent. Again, the exact realisation is dependent on the production engine definition of the hosting system. Furthermore, the control of information transfer is part of the decision-making of the agent component $\mathcal{C}omp_{TCO}^x$.

The specification of adequate organisation benefit strategies that belong to either class and provide additional interactions $c_i^{opt}, c_j^{coop}, c_k^{inf}$ is an essential requirement for the application of TCs. Only if agents have this incentive to become TC members can the self-organised formation and operation of TCs improve the performance and robustness of the hosting system. In this, the interaction efficiency strategies, providing optimised interactions c_i^{opt} are the most important strategies. As discussed in Sec. 3.2.4, it is thus assumed in the remainder of this thesis that these organisation benefit strategies are realised in the default implementation of the TC organisation component ($\Lambda_{TCO}^{default}$) and include the according interactions in $C_{TCO}^x \subset \Gamma^x$.

Agents incorporating such a default TC organisation component $\Lambda_{TCO}^{default}$ can perceive these additional interaction interfaces. The decision-making within such a default component is assumed to be based on the rational approach to become a TC member in order to capitalise the provided benefits with the objective to increase the agent utility. It is equally valid to assume that this agent rationality will induce a member to leave a TC when its utility has not been improved by the association.

4.4 Trusted Community Lifecycle and Management

Until now, the description of the self-organisation process to form a TC has been reduced to a description of the rational pursuit motivated by the expected benefit and cooperation of other members. Also, the presence of organisation benefit strategies has been described as incentive mechanism without further consideration about defecting members and consequences. This view has been taken for the purpose of comprehensibility. This section extends this description by the analysis of the dynamics involved in the Trusted Community approach and the decision-making of the TC organisation agent component.

Agent organisations in the literature are often defined with a lifecycle model (cf. e.g. [126],[93]) with the definition of phases or stages in which an organisation is at each step in time. This is applied accordingly here to describe the global dynamics in a hosting system:

In the initial state of the hosting system, there are no operating Trusted Communities and each agent from the agent society is unassociated. Potential TCs are in the **Pre-Organisation Phase**. In this phase, agents have to apply sub-optimal interactions because of their uncertainty about other agents' behaviours. These interactions are rated according to the trust model, constantly reducing the uncertainty. With time passing, groups of trustworthy agents develop strong mutual trust relationships (as defined in Eq. 3.5) with each other. Agents execute decision-making here to determine whether it would be beneficial, in terms of utility, for them to form a Trusted Community within this group. This self-organised process can eventually lead to a critical number of agents deciding to initiate the formation of a TC which starts the next phase in the lifecycle.

The **Formation Phase** is characterised by negotiations of the initiating agents about the subset of these agents that actually constitute the TC members. Such negotiations are necessary as not all agents have had the same interaction partners or have made negative interactions with each other. Finally, these negotiations can lead to the formation of a TC with the negotiated initial members which enters the next phase.

At the start of the main phase of the lifecycle, the **Operation Phase**, the forming agents become regular TC members, elect a TCM and can finally execute the organisation benefit strategies to increase their utility. Unassociated agents can still interact with TC members in outbound interactions but cannot engage in optimised interactions. Members formed or joined the TC out of self-interest because they expected to increase their utility by being member. If agents find that this does not hold, they will eventually leave the TC and become unassociated again. If too many members leave, the operation cannot be maintained. A TC is dissolved in this case and the TC is transferred back to the pre-organisation phase[2].

Each of these phases requires the involved agents to make decisions about their exact approach. Consider for example the decision whether to form a new or join an inviting TC, the decision which agent to elect as TCM, or the criteria to leave a TC. The importance of this reasoning becomes even more clear when the challenges introduced by the openness of the system are considered. Each phase involves particular challenges that require the agents to execute decisive actions to deal with them:

- In the **Pre-Organisation Phase** agents must decide whether to join a TC, and with which other agents. However, these agents can leave the system during this process. Additionally, the decision to accept an invitation to join a particular TC may be outdated when the TC dissolves before the agent can complete the decision-making.

- In the **Formation Phase**, agents that are trustworthy from the view of the formation initiator are invited to form the TC. Despite their mutual trust relationship to the initiator, these agents can have had negative trust relationships with each other, in which case the formation can fail.

[2]In discord to some approaches in the literature (cf. e.g. [126]), the dissolution is not assigned a dedicated phase for TCs. This is because of the lack of guarantees in open systems that the dissolution can be executed as an ordered process. Here, it is assumed that this must be rather assumed as a spontaneous and unplanned event without providing opportunities for coordinated actions

Additionally, agents can leave the system despite accepting the invitation or before they can answer it.

- The **Operation Phase** is the most challenging phase. Here, the TC is operating with members executing optimised interactions without safety means. The incentive mechanisms to enforce inbound cooperation through the membership motivation works if agents behave rationally. However, the system is open and members can change behaviours, and start to act irrationally or adversarily. The behaviour of such agents must not influence the successful operation of a TC, else other members will leave it. Such agents must hence be sanctioned and ultimately excluded from the TC. Additionally, the agent society evolves and agents that have not been seen as worthy TC members might have proven otherwise in the meantime. These agents should hence be invited to become TC members if they improve the TC composition. Finally, as TC members remain self-interested, the incentive of TC membership must always be provided to prevent members from leaving the TC.

The reasoning and decision-making within the TC organisation component must comprise these different areas. In that, it must remain flexible enough to be adaptable at runtime to allow for control reactions to observations, such as an abnormal system state. In the following, this design of the TC organisation agent component is examined.

As depicted in Fig. 4.1, the TC organisation component is comprised of dedicated strategies that cover all aspects of the decision-making necessary for the TC approach. These strategies encapsulate decision-making for each phase of the lifecycle. They are applied by each agent with the exact implementation of a strategy being open and hence possibly heterogeneous among the agents. Before the operation of a TC, the following strategies are required:

- *Potential Member Search* strategy: Used by an agent to determine which other agents are the most trustworthy ones from its view.

- *Association Evaluation* strategy: This type of strategy is used to determine whether to form a new TC, join the formation of a TC, join an inviting TC or remain unassociated.

- *TC Initiation* strategy: This strategy is needed to define criteria for a successful TC formation with respect to the composition of the group of initiating agents.

Apart from these strategies executed by every agent, the operation phase requires special strategies that are executed exclusively by TC members including the Trusted Community Manager. TC Members need the following strategies:

- *Membership Evaluation* strategy: Here, a TC member determines whether its utility has improved because of TC membership. It hence makes the decision whether to remain TC member or leave the TC.

- *Distributed Leader Election* strategy: This strategy is required to elect a TCM. This involves criteria for TCM qualification as well as distributed leader election algorithms to perform the election as such.

After being elected, a TCM requires the following strategies:

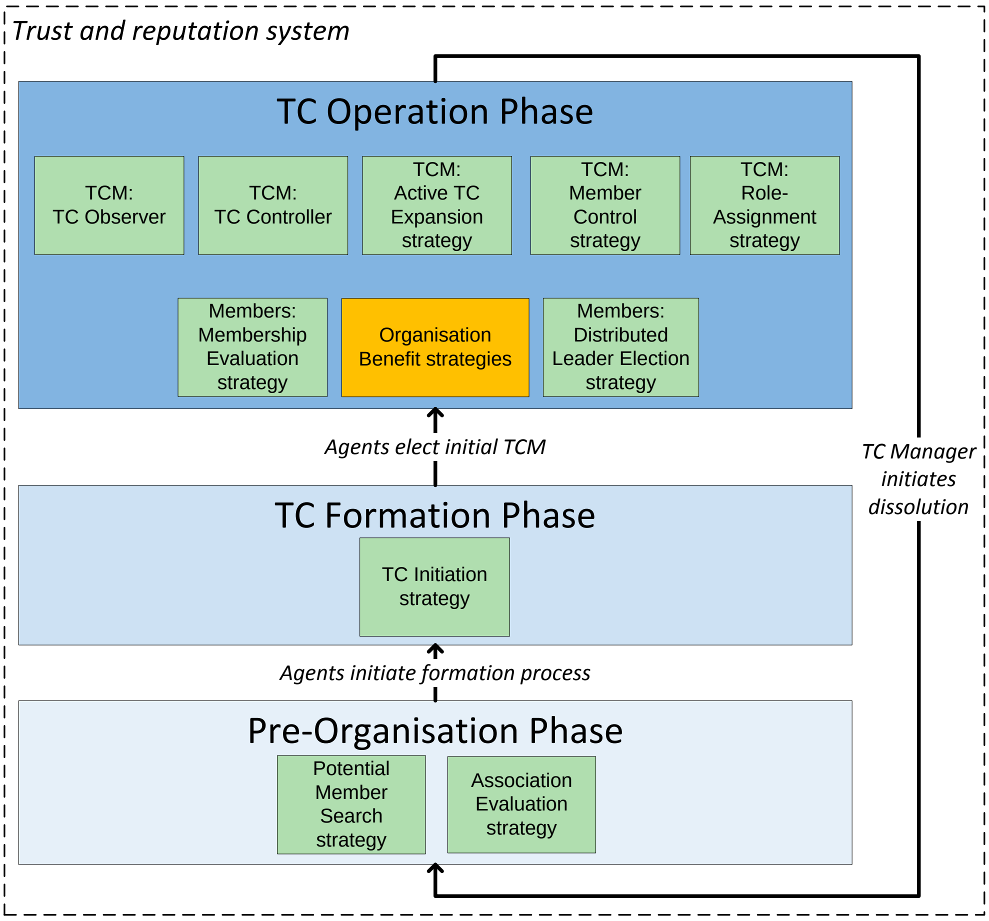

Trust and reputation system
TC Operation Phase
TCM: TC Observer
TCM: TC Controller
TCM: Active TC Expansion strategy
TCM: Member Control strategy
TCM: Role-Assignment strategy
Members: Membership Evaluation strategy
Organisation Benefit strategies
Members: Distributed Leader Election strategy
Agents elect initial TCM
TC Manager initiates dissolution
TC Formation Phase
TC Initiation strategy
Agents initiate formation process
Pre-Organisation Phase
Potential Member Search strategy
Association Evaluation strategy

Figure 4.1: Trusted Community lifecycle and composition. The depicted strategies constitute the configurable mechanics of the TC approach and are relevant only in certain phases of the TC lifecycle. The main phase is the TC operation phase. It further differentiates between strategies required by TC members and strategies required by a TCM for the management of the TC (prefix *TCM*).

- *TC Observer*: This strategy constitutes the Observer part required for a regulatory O/C-loop of a TCM. Here, observation criteria and approaches are encapsulated.
- *TC Controller* Being the complement of the Observer, the Controller is used to regulate the operation of a TC based on the received observations.
- *Active TC Expansion* strategy: Here, strategies to extend the composition of the TC at the time of the TC operation are utilised. This allows to adapt the TC to changing environmental conditions.
- *Member Control* strategy: This strategy allows to control members of a TC. Such control involves to provide feedback to member actions, punish members and exclude them.
- *Role Assignment* strategy: This strategy allows the TCM to distribute its management tasks among the members by assigning roles to them.

These strategies serve the purpose to realise the self-management of the TC that deals with the challenges of the TC operation. Without such strategies, the openness of the hosting system and the autonomy of the agents could lead to exploitation and damage of member agents and forbid the application of TCs. In other words, these strategies provide a means to manage and protect the operation of the organisation benefit strategies in order to maintain the incentive of TC membership. Central to this is the election of a Trusted Community Manager that coordinates the execution of special TCM strategies. The power to execute these strategies, for example allowing the TCM to invite new agents to the TC, or exclude members from it, is delegated from the members to the TCM upon election.

To ensure the robustness of a TC towards influences from adversary agent behaviours and system states, the design of this strategy composition has been inspired by two approaches:

(1) In **Systems Theory**, the maintenance of the robustness of a system is sometimes (cf. e.g. [132]) described to be constituted by the following generic mechanisms: *System control* designates a mechanism that utilises positive and negative feedback to regulate the operation of a system. This mechanism is realised by the incorporation of the TCM with its ability to provide feedback to its members based on the influence of their behaviour to the TC. *A fail-safe, redundant mechanism* that allows to continue the operation of a system despite the loss of single components. In the presented strategic composition, this mechanism is realised by the ability of members to elect a new TCM when the previous TCM changed its behaviour or left the system. On the other hand, the TCM is able to compensate for the loss of single members by inviting new agents to the TCM, and by reassigning roles among the members. The next mechanism is *modularity*, in computer science also referred to as *separation of concerns*. The contribution of modularity to robustness is the feature that disturbances and failures in the operation of single modules are contained locally and do not affect the whole system. In the TC design, this is considered by the logical encapsulation of decision-making procedures in dedicated strategies and the separation of member and TCM responsibilities. Finally, robust systems often incorporate some form of *decoupling*: This designates the principle of isolation of disturbances and variations in low-level components from high-level functionalities. In the design of TCs, this can be understood as the robustness of the TC as such to adversary behaviour of single agents. In this the TCM has a special part with the TCM-strategies.

(2) **Organic Computing** (see Sec. 2.4) aims at increasing the controllability of technical systems which consist of a large number of distributed and highly interconnected subsystems. In such highly dynamic systems, it is impossible to foresee and verify all possible system configurations at design time. Therefore, OC systems are designed such that they self-organise and incorporate so-called *self-X properties* to allow for the efficiency, robustness and online adaptation of these systems. As summarised in Sec. 2.4, self-organisation is seen as being constituted by the following self-X properties:

- *self-configuration*
- *self-healing*
- *self-explanation*
- *self-protection*
- *self-optimisation*

The design of the strategies incorporated in the TC approach is mainly based on these properties. This is detailed in the discussion of the respective strategies in Sec. 4.6.

Before the realisation of the lifecycle and the incorporated strategies are described, the TCM, being the central entity in the design, is examined in more detail.

4.5 The Trusted Community Manager

In the presentation of the TC approach, the Trusted Community Manager has been introduced as necessary entity of the TC design. This has however not been justified so far and is accounted for here.

The hosting system has been described as distributed system without central control. As such, the delegation of control from the members of a TC to a central TCM is a design decision that requires explanation. The key idea is the general agreement that many systems with a purely decentralised design often lack the effectiveness of hierarchical systems. The generic concept of task decomposition and responsibilities is found in almost each system of a certain size, be it in nature with the architecture of organisms, in economy with the division of companies, or in technical systems like the Internet with the routing hierarchy. In the context of the TC design, the avoidance of hierarchy would require the members to constantly negotiate about all management actions and strategies. This distributed decision-making does not only require far more communication, but makes it also necessary to account for lack of consensus, agents leaving the system during the negotiation process etc. The TC approach hence utilises the TCM for the control of recurring tasks and the regulation of the TC. This is not to say that the installation of such a hierarchical entity solves all control issues without costs. Consider for example the case of an adversary TCM and the damage it can inflict on the TC members. However, the regulatory mechanism of TC members, namely their free decision to leave a TC, the ability to re-elect a TCM, and finally the ability to form a new TC with control over the composition (esp. excluding agents as members that showed such behaviour), allows for a robust resolution of such issues.

The TCM-strategies introduced above represent the power that the TCM has been granted by the TC members: The *Member Control* and the *Active TC Expansion* strategies encapsulate the responsibility of the TCM to regulate the access to the organisation. The members hence delegate the power over the TC operations **inclusion** and **exclusion of an agent** a from a Trusted Community $TC_i(t)$, as defined in Eq. 4.5 and 4.6 by the operators $\oplus^{tcm_{TC_i}(t)}$ and $\ominus^{tcm_{TC_i}(t)}$, to the TCM. The design decision to regulate the access to the organisation is without alternative for the TC approach for the following reasons: An *open* organisation, such that each agent can declare itself a member (cf. e.g. *congregations* in [52]), does not account enough for the fact that in open systems known adversaries can also become members. An open TC would hence require the continued application of Trust Management among TC members to realise a selective partner determination for the execution of interactions based on the organisation benefit strategies. In effect, this removes the differentiation between inbound and outbound TC interactions and violates the core concept of the TC.

On the other hand, the openness of a TC could be understood less strict: Agents are not allowed to declare themselves members, but they are allowed to actively declare their interest in TC membership. This case is an alternative to the process of *Active TC Expansion* strategy execution by the TCM. However, this does not involve any substantial difference in the process as the TCM still needs to determine whether an applying agent is suited or not.

The design decision to delegate access control to a central TCM, as opposed to negotiation among the members about this matter, is then a decision according to the above mentioned reasons.

Additionally, the TCM is granted the power over *Role Assignment* strategies. These strategies control the utilisation of the assignment of TC management roles to TC members via the $\rightsquigarrow^{tcm_{TC_i}(t)}$-operator as defined in Eq. 4.7. The **assignment of roles to members**, associated with tasks to perform in each role, is a concept to distribute the overhead of TC management among the TC. This is necessary because the TCM is self-interested just like any other agent and has become a TC member to increase its own utility. If TCM management tasks involve too much overhead, the utility is affected and the incentive to remain in the TC is lost. In this case, rational agents would always decide to leave a TC once elected as TCM. This in turn would prevent the operation of a TC. The aim of the role assignment strategies is hence to distribute as much of TC management tasks as possible to the members.

Finally, the *Observer* and *Controller* abilities allow the TCM an **adaptive regulation of the TC**. As presented in the agent model in Sec. 3.1.2, each agent in the hosting system is composed of an Observer and Controller, allowing it to adapt its behaviour to situations observed in the system. It has been further discussed in Sec. 3.2.1 that, with autonomous agents involved, such observations must always remain incomplete. On the one hand, agents can only perceive information that others have disclosed. On the other hand, passive agents being requested to interact with another agent, have often no incentive to investigate the consequences of these interactions, as they themselves have no direct utility gain from such interactions. With incomplete observations, control based on these observations is necessarily lacking. A Trusted Community allows to improve the local O/C-loops of member agents by firstly providing an incentive to share more information (as discussed in Sec. 4.3), and secondly by aggregating local views in organisation-wide views. This is realised by the equipment of the TCM with a superimposed O/C loop, which constitutes a TC-wide multilevel O/C architecture (cf. e.g. [134]). The aggregation of the member observations to a TCM observation are then utilised to detect system states that are not detectable by single agents, such as abnormal system states or a collusion of agents. These observations allow for more accurate control and are utilised to regulate the adaptation of the TC as such to these system states by the TCM.

4.6 Trusted Community Strategies

The behaviour of TC agents is determined by a trust-based decision-making process. The most important of these decisions are encapsulated in the strategies depicted in Fig. 4.1. In the following, these strategies are further analysed and exemplary realisations are presented. Additional information on the incorporation of these strategies in workflows can be found in appendix B where the design of the TC Organisation agent component is detailed.

The TC strategies are based on the behavioural design pattern *Strategy Pattern* and allow for varying realisations (algorithms) of the reasoning process they encapsulate, by being (a) exchangeable at runtime, and (b) heterogeneous among the agents (clients). This accounts for the requirement to implement agent behaviour that is adaptive (at runtime) and protected by the autonomy guarantee for the agents.

The only imperative for the realisation of these strategies is that a clearly defined output (decision) is needed (this is required for the incorporation in workflows within the agent component, see appendix B). On the other hand, there are no explicit requirements on the input data used to produce the desired output. This allows for alternative realisations of strategical decisions, based on varying input data (for example by applying various degrees of agent awareness), and the actual reasoning process on this data.

In the following, this flexible realisation of the agent decision processes is presented in detail: For all strategies, the encapsulated decision process (including output) is described and formalised, a basic realisation is presented, and implementation requirements and strategy refinements are discussed. In addition, the self-x properties of the TC management strategies, constituting the self-organised TC regulation, as discussed in Sec. 4.4, are elaborated. The section is concluded by the declaration of a basic strategic configuration that offers a complete TC agent behaviour scheme based on the basic realisations of the strategies. This strategic configuration is generally applicable without incorporating any scenario-specific knowledge.[3]

4.6.1 Potential Member Search Strategies

Trusted Communities are formed between agents with mutual trust relationships. The purpose of this type of strategy is **to determine, from the view of a single agent, which agents are trustworthy enough to form a TC with**. The problem to search good interaction partners can be understood as optimisation problem. The search for trusted agents *among agents with many interaction evaluations* can be described as an *exploitation* approach to the optimisation. This is opposed to *exploration* approaches that search for good interaction partners among agents with no or few interaction experiences.

Formally, the execution of such a strategy by an unassociated agent x at time t determines a set $\mathcal{P}(t)$ of agents that fulfil the criteria of fellow TC members. The most general form of this strategy is hence described with the following function:

$$f^{search}(\mathcal{A}(t)) := \mathcal{P}(t) \tag{4.8}$$

[3]Advanced strategies based on scenario-specific knowledge are presented in the evaluation section for the Trusted Desktop Grid, see Sec. 5.2.2.

with:

$$\mathcal{P}(t) \subseteq \mathcal{A}(t)$$

The implementation of such a strategy $f^{search}(\mathcal{A}(t))$ is then concerned with the filtering of the set $\mathcal{A}(t)$ according to these criteria. The resulting set $\mathcal{P}(t)$ is further used as input for the association evaluation strategies where its composition is checked (see 4.6.2 for details). In case the set fulfils the TC formation criteria implemented in this strategy, the formation of a TC is initiated. The determination of this set hence involves the responsibility to choose suited agents, as the executing agent must rely on them as TC members eventually.

Basic strategy implementation

The basic implementation, applicable in each hosting system, interprets the filtering criteria as follows: Other agents are expected to form a TC only with trusted agents themselves. Agents that are trusted by agent x, but do not trust x (asymmetrical trust relationship) should hence be filtered to avoid the overhead of contacting these agents. Additionally, agents that are already a TC member in a different TC should not be included in these considerations, as TC members are not expected to leave their TC without a guarantee that their potential new TC actually forms successfully. Finally, only agents that are online at the time of execution should be considered. These criteria result in the following filter for the set of potential member agents:

$$\mathcal{P}(t) = \left\{ p \in \mathcal{U}^{\mathcal{H}}(t) \setminus \{x\} \;\middle|\; DT_x^{p,\hat{c}} > thres_m^{DT} \wedge DT_p^{x,\hat{c}} > thres_m^{DT} \right\}$$

According to this definition, $\mathcal{P}(t)$ contains all agents p that have a strong mutual trust relationship (as defined in Eq. 3.5) with x. This requires a specification of the context $\hat{c}$ and the threshold $thres_m$. Additionally, the executing agent must retrieve a set of opinions $\left\{ DT_{p_1}^{x,\hat{c}}, .., DT_{p_{|\mathcal{P}(t)|}}^{x,\hat{c}} \right\}$ of the other agents about itself. This information needs to be provided as input for the function, such that:

$$f^{search}\left(\mathcal{U}^{\mathcal{H}}(t), thres_m^{DT}, \hat{c}, DT_{p_1}^{x,\hat{c}}, .., DT_{p_{|\mathcal{P}(t)|}}^{x,\hat{c}}\right) \tag{4.9}$$
$$= \mathcal{P}(t) = \left\{ p \in \mathcal{U}^{\mathcal{H}}(t) \setminus \{x\} \;\middle|\; DT_x^{p,\hat{c}} > thres_m^{DT} \wedge DT_p^{x,\hat{c}} > thres_m^{DT} \right\}$$

Implementation requirements and refinement

The implementation of this strategy has a major influence on the application of TCs in a hosting system. Consider for example an implementation $f^{search}(\mathcal{A}(t)) = \mathcal{A}(t)$ of this strategy: This allows to form a TC with any agent in the society that is online at the time of execution. In case of a wide-spread use of such a non-discriminating strategy among the agents, the choice of suited TC members must be compensated with an according *TC initiation* strategy or will lead to the formation of TCs with not suited members. A TC composed of such unreliable agents is unstable as exploitation within a TC decreases the utility of members and makes them leave the TC. The worst case for an implementation is hence a strategy that does not discriminate enough between trustworthy and not trustworthy agents and subsequently leads to unstable TC compositions in which the executing agent has no utility gain. On the other hand, an optimal implementation composes the set $\mathcal{P}(t)$ such that the decision to form a new TC with these agents as members is always successful. This avoids overhead (i.e. in terms of communication) of the TC formation.

Finally, more elaborate strategies can further refine the search criteria, for example by including agents that have no high trustworthiness yet, but are expected to develop it (e.g. because of planned interactions), and to exclude agents that are expected to lose their trustworthiness (e.g. because of adversary behaviour in the recent past). These phenomena have also been referred to as *too-much trust*, and *over-confidence* (cf. e.g. [20]) and discussed in Sec. 3.2.1. Strategies to allow for such filtering can apply *trust development prediction*, which is for example examined in [76], [77], and [190].

4.6.2 Association Evaluation Strategies

The *Association Evaluation* strategies are a key element in the self-organisation of the TC approach: With such a strategy **agents, autonomous with respect to their organisation association status, decide whether to form or join a Trusted Community based on the prospect of increasing their utility**. This decision-making can be broken down into two separate decisions:

(1) An agent x must determine whether to join (the formation of) a certain Trusted Community $TC_i(t)$. For a rational agent, this decision is based on the expected benefit (the gain in utility $U^x(t+a)$) of membership. The evaluation of the expected benefit of TC membership in a given TC is related to the estimation of a *coalition value* (cf. e.g. [191]): In the latter, agents try to determine how well a potential coalition is suited to achieve the goal it is planned for, based on the estimation of the agent capabilities that consider the formation. However, in TCs there is no group goal and hence each potential member must consider for itself, whether the membership in a forming or inviting TC will increase its utility. For this, the other members (potential members $\mathcal{P}(t)$, or actual members $\mathcal{M}_{TC_i}(t)$) need to be analysed: The organisation benefit strategies utilised within a TC allow members to execute optimised interactions. The *willingness* of the members to cooperate in these interactions is based on the incentive of maintaining the TC membership and the enforcement by regulatory observation and control of the TCM. However, the given *willingness* of other members makes no statements about their *competence*. Due to the heterogeneity of agent capabilities, an optimised interaction performed with a TC member may still be less beneficial to an agent than an interaction with a non-member with strong capabilities. On the other hand, the contract to remain a TC member involves the commitment to cooperate with fellow members. As discussed in Sec. 3.2.1, an interaction as passive agent does not involve any direct benefit, but can be perceived as overhead (e.g. blocking of resources). In summary, a rational agent x reasoning about membership in a $TC_i(t)$ must hence analyse the composition of the TC, especially the capabilities of its members ($\mathcal{P}(t)$, or $\mathcal{M}_{TC_i}(t)$), and additionally the estimated overhead of the demanded cooperation towards these members.

(2) In general, a hosting system $\mathcal{H}$ provides opportunities for several Trusted Communities to form and operate independently of each other (such that $\left|\mathcal{O}^{TC}(t)\right| > 1$). This is opposed by the definition in Sec. 4.2 that each agent is allowed to be the member of a single TC only. The second decision expected from implementations of the *Association Evaluation* strategies is hence whether to:

- remain unassociated, or
- follow the invitation to join an inviting $TC_a(t)$, or
- initiate the formation of a new TC with a set of initiating agents $\mathcal{P}(t)$, or
- follow the invitation to join the formation of a new $TC_b(t)$, along with initiating agents $\mathcal{P}(t)$.

This decision is the more complex the more options an agent has: Not only must an agent choose between those state changes, but also among the varying options for each state change. Consider for example a trustworthy agent x that has been invited by several operating TCs to join them, and additionally has strong mutual trust relationships to a large group of agents $\mathcal{P}(t)$ allowing it to initiate a TC formation, while possibly having been invited to join a TC formation already. Clearly, this requires some form of comparison of the options and a prioritisation between them.

Basic strategy implementation

The basic implementation of the *Association Evaluation* strategies is an *opportunistic algorithm with the constraint to reduce overhead* where possible. The key idea is to use a prioritisation of the status changes, with the option of joining an inviting TC having top priority. The basic implementation

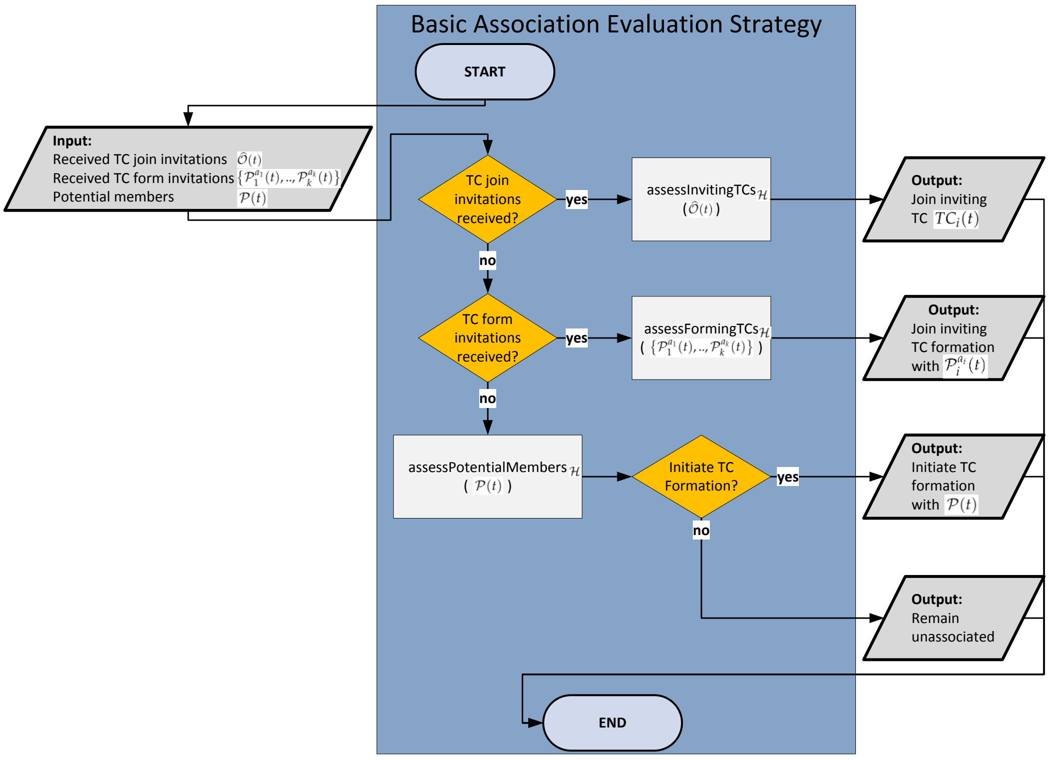

Basic Association Evaluation Strategy
START
Input:
Received TC join invitations $\tilde{\mathcal{O}}(t)$
Received TC form invitations $\{\mathcal{P}_1^{a_1}(t),..,\mathcal{P}_k^{a_k}(t)\}$
Potential members $\mathcal{P}(t)$
TC join invitations received?
yes
no
assessInvitingTCs$_{\mathcal{H}}$ $(\tilde{\mathcal{O}}(t))$
Output: Join inviting TC $TC_i(t)$
TC form invitations received?
assessFormingTCs$_{\mathcal{H}}$ $(\{\mathcal{P}_1^{a_1}(t),..,\mathcal{P}_k^{a_k}(t)\})$
Output: Join inviting TC formation with $\mathcal{P}_i^{a_i}(t)$
assessPotentialMembers$_{\mathcal{H}}$ $(\mathcal{P}(t))$
Initiate TC Formation?
Output: Initiate TC formation with $\mathcal{P}(t)$
Output: Remain unassociated
END

Figure 4.2: Decision flow of the Basic Association Evaluation Strategy: Based on received invitations and potential TC members, a prioritisation process for an association status change is executed. Joining an inviting TC has top priority, then joining a TC formation, initiating a TC formation and finally the maintenance of the unassociated status.

depicted in 4.2 represents the following decision-making process: If invited, an agent will always join a TC. This decision is opportunistic for the following reasons: The choice to join can always be withdrawn by leaving the TC any time thereafter. This means that an agent does not need to perform any elaborate assessment of an inviting TC with respect to the TC membership benefits. If the benefit is not provided, the agent simply leaves the TC again without having any loss. This is further motivated by the fact that the agent does not need to change any of its interaction decisions with unassociated (possibly favourite) interaction partners, as outbound interactions are still allowed for TC members. This behaviour is also opportunistic because it is not optimal for the management of a TC, as leaving members cause management overhead (composition updates, role-reassignment etc.) and should be avoided. It is hence preferential for a TC to accept agents as members that do not withdraw this decision.

The depicted implementation accounts for the special case of invitations from different TCs for an agent at the same time. This case can for example occur when the TC managers of the different TCs use the same (parameterisation for their) *Active TC Expansion* strategies making the invited agent seem a suited member based on the same criteria. Additionally, an advanced *Association Evaluation* strategy that requires a long time for the decision-making also raises the chance of multiple invitations. Where multiple TCs are inviting the agent, further assessment is necessary to make the decision which of the different TCs to join. This involves a comparison of the TCs by analysis of the member capabilities. Such an analysis is however scenario-dependent and hence designated with a generic function here: Let $\widehat{\mathcal{O}}(t) \subseteq \mathcal{O}(t)$ be the set of TCs inviting an agent x at the time of the association evaluation t, then the following function chooses of which Trusted Community $TC_i(t)$ to accept the invitation:

$$assessInvitingTCs_{\mathcal{H}} : \widehat{\mathcal{O}}(t) \rightarrow TC_i(t)$$

In case, an agent is not invited to join a TC, but has received invitations to join the formation of new TCs by an initiator $a_i \in \mathcal{A}$ with its respective set of forming agents $\mathcal{P}_i^{a_i}(t)$, the decision must be made, which of the formations to join. Again, this involves a scenario-dependent assessment of the agents and is encoded by a function:

$$assessFormingTCs_{\mathcal{H}} : \{\mathcal{P}_1^{a_1}(t), .., \mathcal{P}_k^{a_k}(t)\} \rightarrow \mathcal{P}_i^{a_i}(t)$$

Note here that the basic implementation prefers the joining of a running formation over the initiation of a formation. This is because of the opportunistic nature of the implementation and the consideration that the initiation involves more communication overhead, and the parallel formation of different TCs reduces the chances for formation success (as invited agents can choose based on different criteria).

Finally, if no invitations were received, an agent must decide whether to initiate the formation of a Trusted Community. As discussed, this decision is based on the presence of trusted agents that are suited as potential members. The set $\mathcal{P}(t)$ is determined by the *Potential Member Search* strategies (see Sec. 4.6.1). Here, the decision is made, whether this set promises a successful formation. If not for the reduction of overhead, the most basic approach is to always initiate the formation as long as the set contains agents. However, the repeated initiation of a TC formation is costly for the initiator, and can lead to agents ignoring the invitations if they are issued too frequently. As such, the basic implementation must analyse the set of potential members based on the factors of a successful TC composition, as well as the last attempt of a formation. This behaviour is again dependent on the realisation of the hosting system and hence generalised with the following function:

$$assessPotentialMembers_{\mathcal{H}} : \mathcal{P}(t) \rightarrow \{true, false\}$$

If the function returns $true$, the formation is attempted as initiator (with the success determined by the *TC Initiation* strategy, see Sec. 4.6.3), else the agent remains unassociated.

Implementation requirements and refinement

Implementations of the basic strategy must provide realisations of discussed scenario-dependent assessment-functions. However, an opportunistic approach as presented here may not always lead to the desired dynamics. Consider for example agents that use the possibility of a "trial" membership to exploit it and then leave. This can in turn lead to TCMs inviting less agents to join a TC and block the self-organisation process. As such, advanced approaches can include a more thorough assessment of the given options, and for example prioritise the formation of a TC over other options if overhead is no issue. Here, approaches from related work on the estimation of an organisation value, as discussed for the *coalition value* (cf. e.g. [191]), can be applied.

4.6.3 TC Initiation Strategies

The *TC Initiation* strategies are required when an agent decides to initiate the formation of a new TC with a group of potential members $\mathcal{P}(t)$: Agents from the group are invited to join the formation and react to this invitation based on the decision-making within their *Association Evaluation* strategies (see Sec. 4.6.2). The agents that accept the invitation constitute the set $\mathcal{F}$ of potential initial members of the forming TC. The purpose of the initiation strategies is now to **determine whether the set of accepting potential TC members allows for a successful TC formation**. Consider for example the trivial case that all agents reject the invitations, hence $\mathcal{F} = \emptyset$. This does obviously not allow for a successful TC formation. The strategy must hence implement a function of the following form:

$$f^{init}(\mathcal{F}) = \begin{cases} true, & \text{if } |\mathcal{F}| \geq 1 \\ false, & \text{else} \end{cases} \tag{4.10}$$

The only constraint in this generic form is that the set of agents forming the TC must include at least one more agent, such that the formed TC will be composed of two agents (including the initiating agent) and hence adhere to the minimum size requirement formulated in Sec. 4.2. However, this very broad constraint only serves the purpose of guaranteeing a correct TC formation, disregarding how beneficial such a TC would be for its members. The regulatory TCM-strategies for example involve some overhead for TC members. This overhead is not well invested in a very small TC. Additionally, a small TC will in general generate a lower utility gain for its few members, as the set of interaction partners for optimised interactions is very small. The following basic implementation of this strategy hence accounts for this by defining a minimum size based on overhead considerations.

Basic strategy implementation

The overhead of TC membership, generated by regulatory strategies and induced on TC members, is dependent on the strategic configuration of the TC manager. Additionally, the quantification of an acceptable amount of overhead for a single agent is defined by its implementation of the *Membership Evaluation* strategy (see Sec. 4.6.4). An exact *and* generic measure for an overhead-based minimum TC-size is hence not possible. The approach for the basic implementation is thus to derive such a measure from known values: As discussed in Sec. 4.5, the TCM is a regular self-interested agent that does not accept to sacrifice its utility for the regulation of the TC. It is hence equipped with the instrument of assigning management tasks to its TC members by means of roles. The composition of the set of roles $\Psi_{TC_i}(t)$ included in the tuple that constitutes a TC can thus be further analysed and used as a criterion. Note that this set is again dependent on the implementations of the TCM-strategies, and an initiator does not know which of the agents will become TCM in the TC, thus how exactly this set will be composed. To allow for any type of qualification, this is hence applied as a best-effort estimate by referring to the (known) *default* implementation of the TC organisation component $\Lambda_{TCO}^{default}$ with the assumption that the majority of members will apply such configuration when elected as TCM. An additional assumption here is that the realisation of each role $r \in \Psi_{TC_i}(t)$ is such that it does not imply an overhead that outgrows the benefit of the TC membership, when assigned to an agent. This is based on the consideration that a validation of this assumption would mean that the assignment of a role to an agent would force the agent to give up its membership due to the lack of benefit. On the other hand, it can be generically stated that the (re-assignment) of roles involves overhead, albeit the exact quantification again depends on the implementation of the *Role Assignment* strategy of the TCM (see Sec. 4.6.8). The assumption here is hence that despite limiting the overhead induced in a single role, a TCM also limits the lower bound for effort in roles (and thus their number), in order to reduce its overhead for the assignment of roles.

In summary, the following assumptions are made to allow for a best-effort estimate of a minimum TC size: The future TCM of the forming TC will utilise a default implementation of regulatory strategies, hence the set of roles $\Psi_{TC_i}(t)$ will be known to the decision-making agent. This set will be composed of roles that do not focus more overhead on them that would marginalise the benefit of TC membership, yet the number of roles is limited due to substantial realisation of roles. As a consequence, the minimum number of TC members for a reasonable TC formation is equalled to the number of roles needed for the management of the TC. This means that the TC will have enough members to provide for a stable operation, thus without agents leaving because of lack of benefit. On the other hand, the constraint for a minimum size is not so hard that it would not allow for the formation of small TCs with just enough members to preserve the operation. The basic implementation of the *TC initiation* strategy is then characterised by the following function:

$$f^{init}(\mathcal{F}, \Psi_{TC_i}(t)) = \begin{cases} true, & \text{if } |\mathcal{F}| \geq |\Psi_{TC_i}(t)| \\ false, & \text{else} \end{cases} \tag{4.11}$$

Implementation requirements and refinement

The implementation of advanced *TC Initiation* strategies can be adapted to specific requirements of a hosting system $\mathcal{H}$. Especially, the balance between the overhead for TC management and the received benefit by the application of the *Organisation Benefit* strategies (see Sec. 4.3) can be used to determine the potential of a successful and stable TC operation. Additionally, an approach to identify key agents from $\mathcal{P}(t)$, based on their application-specific capabilities, can be applied to restrict the formation of TCs with a satisfying number of members, but low performance. Such an approach would hence allow for the TC formation only if the key agents accept the invitation, such that they are in $\mathcal{F}$. Finally, the worst case for the implementation is that the formation constraints are formulated too strictly, such that TCs never form in a hosting system. On the other hand, the best case for an implementation leads to the formation of TCs only, where the resulting TCs can maintain a stable long-term operation.

This completes the strategies required for the formation of a TC. The following strategies are all applied either by TC members or the TC manager during the operation phase of a TC.

4.6.4 Membership Evaluation Strategies

A self-interested agent x joins a Trusted Community with the expectancy to experience a higher utility $U^x(t)$ as TC member. The *Association Evaluation* strategies have been introduced in Sec. 4.6.2 as instrument to determine whether a utility gain can be expected from a TC membership. The *Membership Evaluation* strategies are the complementary estimate. Here however, an agent has more information: It has already measured the actual costs (execution of tasks for assigned roles, cooperation with TC members) and benefits (optimised interactions) of TC membership and can compare these measures with its performance as unassociated agents. **Periodic execution of this strategy allows a rational agent to track the development of the TC membership benefits and eventually leave the TC in case of no utility gain**. The most general form of a membership evaluation strategy makes the decision whether to remain a TC member according to a function of the following form:

$$f^{eval}(U^x(t), U^x(t_a)) = \begin{cases} true, & \text{if } U^x(t) > U^x(t_a), t > t_a \\ false, & \text{else} \end{cases}$$

Basic strategy implementation

An agent utility $U^x(t)$ has been described as being derived from the specification of a production engine that is controlled by the agent. Also, the owner of the agent can modify the utility function to closer match its goals. Consequently, the development of the utility value over time is dependent on this usage scenario: In most cases, the evaluation of an interaction outcome influencing the utility can only be performed deferred. As such, the membership duration must be considered when making this comparison. Only after a specified time interval, can an agent expect to realistically estimate its benefit. An improved implementation of such a strategy is hence of the following form:

$$f^{eval}(U^x(t), U^x(t_a), thres_a) = \begin{cases} true, & \text{if } U^x(t) > U^x(t_a) \vee (t - t_a) < thres_a \\ false, & \text{else} \end{cases} \tag{4.12}$$

with $thres_a$ being a scenario-dependent threshold that determines a minimal membership duration correlated to the deferral of the utility function evaluation.

Implementation requirements and refinement

Advanced strategies to be applied here can focus on the comparison of the expected utility of being unassociated for the *following* time interval (as opposed to the time interval before the TC membership). This kind of estimate accounts for the fact that the environment of an agent could have changed considerably since it became a TC member, a possibility that grows with the membership duration and the observation of system anomalies. Hence, such a utility prediction for the unassociated case can involve the processing of cooperatively aggregated TCM situation descriptions, an analysis of other currently unassociated agents $\mathcal{U}^{\mathcal{H}}(t)$, and finally the prediction of the utility as TC member for the following time interval. Such utility predictions can be for instance realised by the utilisation of game-theoretic approaches for the analysis of other agents' behaviour. An optimal strategy implementation will make an agent leave a TC only in case it will actually improve its utility function $U^x(t)$ for a subsequent time interval. On the other hand, the worst case for a strategy realisation is that an agent leaves a TC resulting in a utility degradation, making it strive for a renewed membership. The case of wide-spread utilisation of such sub-optimal strategies threatens the stability of the involved TCs and can degrade the utility of other member agents.

4.6.5 Distributed Leader Election Strategies

In most distributed systems, some tasks (like making a fast decision) benefit from centralised control. In scenarios where centralisation is needed, exactly one of the nodes in the system has to be given the right to execute this control and all other nodes have to know which node this is. In literature this node is called the *leader*, and the process of letting nodes designate one of them as the leader and propagate this information, is called *distributed leader election*. In the context of the TC approach, **the task is to let the members of a TC elect one of them as Trusted Community Manager**. The election of a TCM is necessary whenever a TC has no TCM, i.e. right after TC formation or leaving of a TCM (see appendix B for details). The *Distributed Leader Election* strategy is formalised by the following function:

$$f^{elect}(\mathcal{M}_{TC_i}(t)) := tcm_{TC_i}(t) \tag{4.13}$$

Implementations of this strategy then realise an algorithm for the distributed execution by all members that finds a consensus in finite time. Formally, all electing agents are in one of the three states at each time of the execution: *Undecided*, *Decided(leader)*, *Decided(not leader)*.

The requirement of a distributed leader election algorithm is that at the finite end of the algorithm each node is in a decided state and that exactly one of these nodes is in the leader state. Published distributed leader election algorithms mostly focus on the technical process of election and leader information propagation, regarding properties like connectivity (incomplete topology (e.g. ring) vs. fully connected graph), single node identification (anonymous vs. nodes with unique id) and group identification (electing nodes known to algorithm vs. uniform). Success criteria here are low time and message complexities (cf. e.g. [192]). Reasoning about the leader decision is mostly reduced to having the highest unique id (or another low form of discrimination in anonymous networks). For the application of TCM election, it is however of greater interest by which criteria members should be elected. This is due to the power the TCM is delegated, granting it partial control over the other agents in the TC (e.g. their exclusion from the TC). In the following, both views, the procedure and criteria of the election, are respected and respective strategies outlined.

From the distributed systems point of view, the TCM election is classified as incorporating:

- *A fully connected graph*: All TC members know each other. This is also a prerequisite for collective TC formation, esp. for the determination of its success as defined by the *TC Initiation* strategies (see Sec. 4.6.3).
- *Non-uniform*: All members know that every other member is also electing the manager. This does not mean that the set of electing agents remains constant, however new agents are not allowed to join a TC during an election (new members are only accepted by the TCM). An algorithm has therefore only to foresee that members may go offline during an election.
- *Non-anonymous*: All members have unique ids. This is also a prerequisite for the application of a trust- and reputation system, in that it allows agents to distinguish each other.

Basic strategy implementation

A distributed leader election problem with these characteristics is trivial to solve (without communication) if the election is based on the highest id of the electing members. If this problem is however approached from the view of a TCM election, only having the highest id is obviously not an optimal criterion for electing an agent to be the manager of a TC. As described in detail in Sec. 4.5, the manager of a TC is granted the right to both exclude single members from the organisation and dissolve the entire organisation. Resulting from the nature of the open hosting system, the decision which agent to turn into a TCM should be based on the approach to choose the most trustworthy agent. Additionally, the election process should disclose as few private information of the members as possible to protect them from agents faking trustworthiness. Performance-wise, the algorithms used should rather minimise the time complexity than the message complexity as a Trusted Community cannot be maintained without a manager. The election should hence be as fast as possible. In the following, the basic election strategy is presented. It is based on trust between the electing member agents in order to reflect the importance of the TCM election.

The basic implementation of this strategy is termed *Highest TC Reputation Election* strategy. The key idea is to choose the member with the highest average direct trust value among all members to be the manager. This value is the *TC reputation* obtained by requesting all members as opinion providers about all other agents, hence: $max\ RT^{m,\hat{c}}_{\mathcal{M}_{TC_i}(t)\setminus\{m\}}$ for all $m \in \mathcal{M}_{TC_i}(t)$. In that, the aggregation of the opinions is computed with the arithmetic mean of the direct trust values. By choosing the most trusted agent, it becomes more difficult for malevolent agents to become TC managers as they have to reach a higher trustworthiness than each other member agent just in time for an election. As the direct trust (opinion) $DT^{m,\hat{c}}_x$ of each member x towards each other member m is private information, it should be protected during the election. For this, the strategy builds a ring of responsibility such that each agent is responsible for collecting the direct trust values for exactly one other agent. After computing the average, this aggregated value is broadcasted to the other agents, making it possible for each agent to locally compute whether it has the highest average trust value, yet hiding how single members contributed to the aggregate. Messages between members are denoted as $msg^{rnd}_r(c)$ with rnd denoting the round a message belongs to (as determined by the sender), $r \in \mathcal{M}_{TC_i}(t)$ being the recipient of the message, and c being the content of the message. The complete presentation and step-by-step explanation of the algorithm can be found in appendix A.1.

Finally, the algorithm requires the current round as input for the iterative execution, hence the basic strategy is formalised by the following extended function:

$$f^{elect}(\mathcal{M}_{TC_i}(t), round) := tcm_{TC_i}(t) \tag{4.14}$$

This algorithm implies that at the end of the execution each member knows whether it has been elected as a TCM, and - if not - it can identify the TCM without further communication.

Implementation requirements and refinement

Advanced strategies can further aim to increase the robustness of the procedure by for example explicitly targeting the presence of adversary actions during the election process (cf. e.g. [65]), or implicitly discouraging the adaptation to an election procedure by using randomisation (cf. e.g. [192]). Additional strategies can be derived from the literature: The body of literature on distributed leader election is vast, as the designation of single elements in a system is a ubiquitous requirement in technical systems. The worst case in TCM election is a strategy implementation that does not terminate or is not correct. Given these two requirements, the worst case is then a strategy that can be easily manipulated by adversary agents, such as to maximise their chances of being elected without investing considerable effort. On the other hand, the best case for an implementation is a procedure that finds the consensus in a very short time, and prohibits the adaptation to the rules of the election procedure (is *robust*).

This completes the strategies applied by TC members during the operation of TC. The following strategies are applied only by the TCM of a TC, a single agent.

4.6.6 TCM: Active TC Expansion Strategies

The hosting system is characterised by constant dynamics: New agents can join the system, agents can go offline or leave the system permanently, change their behaviours (consider for example the Observer/Controller-based adaptation), or provide additional interaction possibilities by exchanging component implementations. Additionally, the management of a Trusted Community introduces additional dynamics: Agents can lose their membership status because of sanctioning actions of the TCM (see Sec. 4.6.7), or choose to leave a TC because of lack of benefit (see Sec. 4.6.4). These dynamics provide both challenges and opportunities for the composition of a Trusted Community:

(1) On the one hand, the initial composition of a Trusted Community is fixed with the set $\mathcal{F}$ of forming agents. This set is composed of agents that have been estimated as trustworthy by the formation initiator (see *Potential Member Search* strategies in Sec. 4.6.1), and then accepted the invitation to join (see *Association Evaluation* strategies in Sec. 4.6.2) at the time of formation. If agents from this set are excluded from the TC because of the dynamics, the stability of the TC is in danger as the management overhead has to be distributed among fewer members. Additionally, these agents are not available as preferred interaction partners anymore, decreasing the benefits for the other members. Consider for example the discussion of a minimal TC size as constraint for the formation of a TC in Sec. 4.6.3. This challenge makes it necessary to **compensate the exclusion of TC members with the inclusion of new members in order to *preserve* the stability of a TC and the benefits for its members**.

(2) On the other hand, the dynamics can reveal trustworthy agents in the society that were not considered as members when the TC was formed, because they were either not in the system, or because the initiating agent did not trust them at this time. As TC members adhere to interactions with non-members (outbound interactions), the trust relationships with non-members can improve over time and non-members can be perceived as suited for TC membership. In general, the initial members of a TC can benefit from new members, as the overhead is distributed among more agents, and the opportunities to execute optimised interactions increase. The TCM hence needs an instrument to **grant membership to newly discovered trustworthy agents after the formation, in order to improve the composition of a Trusted Community, i.e. *raise* its stability and the benefit for its members**.

This TC management instrument is encapsulated in the *Active TC Expansion* strategies. The functionality can be formalised by a function to find a set $\mathcal{N}(t)$ of potential new member agents in the set of agents that are not yet members of the Trusted Community $TC_i(t)$:

$$f^{expand}(\mathcal{A}(t) \setminus \mathcal{M}_{TC_i}(t)) := \mathcal{N}(t) \tag{4.15}$$

such that the manager agent tcm_{TC_i} performs the inclusion operations following the execution of the strategy:

$$\forall n \in \mathcal{N}(t) : TC_i(t) \oplus^{tcm_{TC_i}(t)} n$$

resulting in the extended composition $\mathcal{M}_{TC_i}(t+1)) = \mathcal{M}_{TC_i}(t)) \cup \mathcal{N}(t)$ of the TC.

The aim of an implementation of such a strategy is then to find (possibly by active observation and testing) potential members with the goal to optimise the composition of the Trusted Community. The strategy is however executed by the TCM alone. It can, but does not need to include the members of its TC in the decision-making. The TCM has the power to gather new member agents without the need to ask other member agents about their consent. This is grounded in the fact that by electing a member as TC manager, the agents trust this agent to make decisions for the organisation. In any case, the members have the option to leave the TC based on evaluations of the TC composition in the *Membership Evaluation* strategies. As the aim of the TCM is to improve the composition of its TC, implementations of this strategy should try and find potential agents to accept as members that are acceptable for all other members.

The application of the *Active TC Expansion* strategies improves the robustness of the Trusted Community by making it **self-healing** (compensation for leaving members), and **self-protecting** (application of TC access control as opposed to open TCs that can be joined by any agent without constraints). Finally, to search for unassociated agents that are suited as TC members can also be understood as *exploration* approach to the problem of finding suited interaction partners for TC members (as opposed to exploitation approach, see Sec. 4.6.1).

Basic strategy implementation

The basic implementation of the *Active TC Expansion* strategy is based on the following considerations:

(1) The set of agents to recruit new members from should be limited to unassociated agents, hence agents that do not already belong to another TC. Though it is formally allowed to invite the members $\mathcal{M}_{TC_j}(t))$ of a $TC_j(t)$ to the own $TC_i(t)$, such competitive behaviour can create a heavy restructuring dynamic among the TCs and affect their stability. This restructuring of TC compositions may however be desired to implement mechanics that regard the search for optimal TC compositions in the hosting system as a decentralised approach to a set partitioning problem (cf. e.g. [96]). This is nonetheless neglected here as the aim of a basic strategy implementation is its universal applicability. Such an approach however has manifold implications on the use of other strategies to support it. Consider for example the description of the basic *Association Evaluation* strategy in Sec. 4.6.2 and the basic *Membership Evaluation* strategy in Sec. 4.6.4: These strategies would have to be massively extended to account for the switch between membership in different TCs, making the already complex decision-making less predictable.

(2) The task to estimate which unassociated agents are suited as TC members can be reduced to the initial problem of finding suited agents to initially form a TC. Agents already have a *Potential Member Search* strategy at their disposal that can be reused as TCM. If performed by the TCM alone,

this approach would however limit the search for new members to agents that the TCM has interacted with, because the said strategy evaluates the direct trust relationships only. It is hence desirable to include the TC members in this task by instructing them to execute their *Potential Member Search* strategy and providing the TCM with the resulting set of trusted agents. The TCM then aggregates the resulting sets by union.

Additionally, former TC members that have been excluded from the TC by the TCM should be excluded from the resulting set $\mathcal{N}(t)$: This is to further protect a TC from agents that defect only when being TC member, in order to exploit the lack of safety means. This additional protection is advisable as the monitoring of TC members involves overhead and is to be minimised by the TCM (see *Member Control* strategies in Sec. 4.6.7). The disregard of such excluded agents requires the synchronisation with the *Member Control* strategies and is best realised by a blacklisting approach. This blacklist must however account for the fact of a dynamic, open system, such in which agent behaviours can change due to learning or adaptation by the user. In result, the TCM should apply *forgiveness* (cf. e.g. [193]) here, such that agents are removed from the blacklist after a time interval.

Let then the set of potential member agents that a TC member $m \in \mathcal{M}_{TC_i}(t))$ determines by executing the *Potential Member Search* strategy be the set $\mathcal{P}^m(t)$. Let additionally $\mathcal{L}(t)$ denote the blacklist generated and maintained by the *Member Control* strategy implementation. The resulting basic strategy then determines the resulting set $\mathcal{N}(t)$ as follows:

$$\mathcal{N}(t) := \left\{ n \in \mathcal{U}^{\mathcal{H}}(t) \;\middle|\; n \in \bigcup_{m \in \mathcal{M}_{TC_i}(t)} \mathcal{P}^m(t), n \notin \mathcal{L}(t) \right\} \tag{4.16}$$

Implementation requirements and refinement

This formalisation of the basic strategy leaves it open how, in terms of strategy implementation, each TC member m estimates its set $\mathcal{P}^m(t)$. This is to respect the autonomy of the members and allow for a greater exploration of the agent society. The TCM then invites each agent $n \in \mathcal{N}(t)$ that has been identified as potential members by any of its current members. This is based on the assumption that the TC members aim for the improvement of the TC composition, too. Advanced implementations of this strategies can further validate this by first testing the agents before accepting them as new members. This approach is based on the assignment of tasks to potential members in order to test them[4]. The exact realisation of such an approach is scenario-specific. In Desktop Grid Systems for example, it is referred to as *spot-checking* (cf. e.g. [175], [176], and [83]), and involves the comparison of pre-computed work unit results with the results delivered by a tested node. If the application of such test is openly advertised, it can also be used to communicate which requirements a TC has. This can further improve the robustness of a TC by also making it **self-explaining**, which means here that it allows non-members to understand the rules for a TC membership grant and to adapt their behaviour accordingly. This resembles the approach for the *Member Control* strategies described in Sec. 4.6.7.

The approach of testing agents despite their trustworthiness ensures that a TC is protected from adversary agents during its operation even more than at the time of its formation. However, the implementation of an *Active TC Expansion* strategy can also take the directly complementary view:

[4]This is referred to as *invitation with conditions* in the workflow depicted in appendix B.

Instead of applying additional security means despite a good trustworthiness value, a strategy can be based on the trust-prediction in order to identify a potential member earlier. This is also a highly valid approach, as the search for the best unassociated agents is a competition among the operating TCs. The TC with the earliest invitation to an agent has then the greatest chance of winning this competition. Approaches for the prediction of a trust value development can for example be found in [76], [77], and [190].

Optimally, these two approaches are combined, such that only agents that have not a sufficient trust value, but are predicted to develop it in the following time interval are checked with test tasks.

In the previous specifications of the *Active TC Expansion* strategies, it has been implicitly assumed that the number of TC members should be constantly increased. While this is certainly a valid approach to optimise small TCs, the question arises whether there is an upper bound to the TC size, such that the inclusion of new members beyond this boundary will decrease the stability or provided benefits of the TC. There is no generic answer to this question: In order to determine a generic boundary, describing the relationship between TC size and TC management overhead is needed. The form of this function is however determined by the specific configuration of the TCM management strategies and the scenario-specific capabilities of the agents. Consider for example a TCM applying a very thorough and costly *Member Control* strategy to provide high protection against adversary members. Here, each additional member increases the overhead substantially. It is then dependent on how well the TCM is able to break this additional overhead down into single tasks that can be assigned to members within the *Role-Assignment* strategies. Additionally, it is important to consider whether the capabilities of the new member increase the average benefit of the other members in the TC more than the additional overhead introduced by the newly assigned control tasks decreases it. For a possible quantification of the overhead linked to a TC role, consider the argumentation for a minimal TC size in the description of the *TC Initiation* strategies in Sec. 4.6.3. Also the scalability of the interaction decision-making of the agents is of essence here. Consider for example the realisation of a *Distributed Leader Election* strategy here. In this line of argumentation, e.g. [194] propose to restrict the size of coalitions to a maximum member size in order to provide for the feasibility of the member decision-making. Finally, the existence of a maximum TC size is directly related to the question whether there exists an optimal TC size. Implementations of this strategy can be used to balance the composition of a TC by regarding upper bounds or an optimal size as restriction to stop searching for new members. This however requires the synchronisation with other strategies, such as the *Member Control* strategies that exclude members, in order to provide the required results. This is further discussed in Sec. 4.6.9 in context of the *TC Observer*- and *TC Controller* strategies.

Finally, the worst case in the implementation of this strategy is that adversary agents are granted membership. This reduces the stability of a TC and the benefit provided to its members and must be countered by excluding these agents again, which generates a considerable overhead. No strategy can guarantee that it does not include adversary agents, as agents can adapt their behaviour strategically to behave trustworthy in order to be granted TC membership, and exploitative afterwards. However, an implementation can make the costs for the initial membership high enough to prevent such a strategy, at least for rational agents.

On the other hand, the best case for an implementation is a strategy that accepts trustworthy and capable agents reliably and ahead of other TCs. In the end, a strategy extending a TC composition with new members will always be subject to a trade-off between risk and benefit.

4.6.7 TCM: Member Control Strategies

The key concept of the Trusted Community approach has been described as the provision of a closed environment for the execution of optimised interactions between TC members. As described in Sec. 4.3, these interactions have been orchestrated in the *Organisation Benefit* strategies and belong to the types of *interaction efficiency*, *information sharing* and *cooperation*. These types of strategies provide benefits to TC members, however they also require them to invest some effort to provide these benefits to others. The successful operation of a TC depends on this investment of the effort:

- *Interaction Efficiency* is achieved by the abandonment of safety means. Though the exact nature of such an interaction is scenario-specific, it can be stated that it is in general far worse if a passive TC member defects in such an interaction, than in a non-optimised interaction with safety means. This is because it must be assumed that the agent starting the interaction does not know whether it has been deceived, which is due to the lack of safety means (see Sec. 5.2.1 for a concrete example). Additionally, TC members initiating inbound interactions can be guaranteed that their fellow members will accept the interaction request. If this is not provided for, hence if agents in a TC reject interactions with fellow members, the benefits of TC membership are substantially lowered.

- *Information sharing* is an opportunity for agents to enrich their decision-making with additional input data observed by other members. On the other hand, this decision-making relies on the provision of this data and TC members are required to answer according requests. Consider for example an agent m that does not answer a request of its TCM to provide a set $\mathcal{P}^m(t)$ of potential members for the execution of the basic *Active TC Expansion* strategy (see Sec. 4.6.6). This does not only reduce the ability of the TCM to manage the TC, but can indirectly reduce the benefits for other TC members.

- *Cooperation* strategies are exclusive to TC members and provide them with additional interaction possibilities. While again increasing the TC membership for agents, these strategies require TC members to actually execute the requested interactions. The TCM for example relies on the execution of tasks linked to a role $r \in \Psi_{TC_i}(t)$. If the TCM assigns such a role to a TC member as a result of the execution of its *role assignment strategy* (see Sec. 4.6.8), the agent is expected to execute the tasks. If this is not the case, again the management of a TC is damaged and the benefit of other members is negatively affected.

But why should TC members not cooperate and perform the required operations? The acceptance of these agents has been based on their trustworthy behaviour and rational agents should not be interested in decreasing the benefit of a TC they are part of. However, these considerations must be neglected due to the openness of the system, constituted by self-interested agents and being highly dynamic, a combination that can lead to various forms of misconduct. Consider for example a strategic agent that tries to gain membership in order to exploit the other members because of the lack of safety means in inbound interactions. Such an agent will behave trustworthy in order to be granted membership and starts to defect only then. But no matter what exact motivations for uncooperative behaviour towards fellow TC members exist, a TCM must ensure that such behaviours are detected and sanctioned to allow for the successful operation of its TC. This is where the *Member Control* strategies are applied. In its most generic form, such a strategy decides which agents $\mathcal{E}(t) \subseteq \mathcal{M}_{TC_i}(t)$ to exclude from a $TC_i(t)$ (being the ultimate sanction) to protect its operation. It can hence be formalised as:

$$f^{control}(\mathcal{M}_{TC_i}(t), t) := \mathcal{E}(t) \tag{4.17}$$

such that the manager agent tcm_{TC_i} performs the exclusion operations following the execution of the strategy:

$$\forall e \in \mathcal{E}(t) : TC_i(t) \ominus^{tcm_{TC_i}(t)} e$$

The implementation of this function must specify **how a TCM can observe undesired TC member behaviour and how to sanction the agents executing it**. Such an implementation increases the robustness of a TC: Even if the *active TC expansion strategy* fails to detect agents with adversary behaviour and accepts them as TC members, these agents can be excluded again. This realises the property of **self-healing**. Additionally, such a strategy realises on the property of **self-protection** by protecting its members from the effects of uncooperative agents. In the following, a basic realisation of such a strategy is proposed.

Basic strategy implementation

The implementation of the basic *member control strategy* is based on the following considerations:

- The abandonment of safety means in the *interaction efficiency* strategies must be compensated by collective safety means that are centrally coordinated by the TCM. These collective observations allow for the detection of misconducting members that is not possible by a single agent.
- These collective safety means must be applied only limited and situation-aware in order to prevent that the associated overhead prevents the membership benefit. Situation-aware means here that members should only be observed in case of suspicion, such as complaints of other members about their behaviours.
- Additionally, the limited application of sporadic observations can increase the incentive to co-operate as TC member, if these observations are transparent to the members, such that they always have to expect to be tested.
- The observations should not be (exclusively) limited to the development of the reputation of a TC member: The intuitive approach to the estimation of member behaviour is here to use the Trust Management already available to the agents to judge their behaviour. However, this introduces a dependency to the operation of the TM which was intended to be avoided for the TC approach in order to provide for robustness towards anomalies like the trust breakdown (see Sec. 3.2.2 for the according discussion). Additionally, the outcomes of interaction without safety means by the agents are not assumed to be interpretable by the initiator of such interactions. This prohibits the rating of the interaction partner and hence does not influence its reputation.
- The exclusion of members from the TC can lead to decreased benefits for the other members or even to a dissolution if the TC size becomes too small. Members leaving the TC are in general

degrading the efficiency of the TC, as the number of available member interaction partners for the remaining members is reduced. Besides, management tasks delegated by the TCM have to be distributed among fewer agents, increasing the overhead of single agents. This instrument should hence be handled with great care by the TCM. Especially agents that do not continuously exploit other members, but have been uncooperative only due to the exploration of their behaviour range, need not necessarily be excluded. Additionally, the sanctioning should be based on *forgiveness* (cf. e.g. [193]), to prevent such single cooperation short-comings to stick to members indefinitely and further motivate members to restrain from such actions. In sum, the sanctioning should be gradual and forgiving, with the possibly harmful exclusion of agents only as ultimate sanction.

- The exclusion of agents should by remembered in the form of a blacklist $\mathcal{L}(t)$ in order to avoid making these uncooperative agents TC members again (see the discussion of *Active TC Expansion* strategies in Sec. 4.6.6). This should however be a temporary measure, such that eventually the agents are removed from the blacklist. This application of *forgiveness* (cf. e.g. [193]) allows to account for changing agent behaviours due to learning and adaptation.

- The rules of sanctioning should be communicated to the TC members, unlike the rules for the observation. While a transparent observation mechanism prevents adaptations in agent behaviour to evade these observations, behaviour adaptations towards sanctions are desired. If agents know what behaviour imposes which sanction, they can avoid these behaviours. This introduces the additional property of **self-explanation** to the TC and helps to increase its robustness.

The exact realisation of the observations is based on scenario-specific realisations and cannot be provided in generic form here. If the observation mechanism is provided, it can however be generally assumed that each observation $o_b^{x,t}$ of the execution of uncooperative behaviour of type b at time t by a member, can be assigned a sanction $s_b^{x,t}$. This assignment is then formalised by the following function:

$$sanction(o_b^{x,t}) := s_b^{x,t} \in [0,1]$$

This function resembles the assignment function of interaction outcomes to trust ratings as described in Sec. 3.1.3, with the difference that it is restricted to negative TC interactions. Just as with a *trust value*, the TCM can then determine a related *member score* which aggregates the sanctions and can serve as decision criterion for the application of a final sanction, the TC exclusion. The score for an agent x at a time t is then formalised by the following function:

$$score(x, t, thres_f) := \sum_{t-thres_f}^{t} s_b^{x,t} \tag{4.18}$$

with $thres_f$ denoting a *forgiveness threshold*, being an interval beyond which sanctions are forgiven (ignored). Such a score can then be utilised to decide whether to exclude an agent x from a TC by the application of a maximum score $score_{max} \in [0,1]$, such that the basic *Member Control* strategy encodes the following function:

$$f^{control}(\mathcal{M}_{TC_i}(t), thres_f, score_{max}) := \mathcal{E}(t) \tag{4.19}$$
$$= \left\{ e \in \mathcal{M}_{TC_i}(t) \setminus \{tcm_{TC_i}(t)\} \;\middle|\; score(e, t, thres_f) \geq score_{max} \right\}$$

All agents (except the TCM itself[5]) that have a higher score than the maximum score, based on sanctions within the forgiveness interval, are identified as agents to exclude. Additionally, the basic implementation lets the TCM advertise the function $sanction(o_b^{x,t})$ to its members, and allows the members to request information about their current score $score(x, t, thres_f)$ from the TCM. This forms an implicit contract between TCM and members, and can incentivise cooperative behaviour if the members adapt their behaviour based on this information.

Finally, each excluded agent $e \in \mathcal{E}(t)$ is added to the blacklist $\mathcal{L}(t)$ (along with the time t of the exclusion), while agents that have been blacklisted for a duration longer than a *forgiveness interval* $t_forgive$ are removed from $\mathcal{L}(t)$.

Implementation requirements and refinement

Implementations of *Member Control* strategies must provide some means of member behaviour observations in order to allow for the sanctioning of members. For the case of Desktop Grid systems, this can be again achieved by applying situation aware *spot-checking* (cf. e.g. [175]), see Sec. 4.6.6 for a discussion of the application for *Active TC Expansion* strategies. In general, the worst case of an implementation leads to extensive observation overhead that devours the benefit of TC membership by exchanging the initially abandoned safety means with a different type of costly safety means. This can lead to an exodus of TC members based on the execution of their *Membership Evaluation* strategies. On the other hand, a too weak control of members can likewise damage the benefit and stability of a TC, as TC members can be exploited by fellow members, in the worst case without even being aware of this. Again, the implementation of such strategies is therefore dependent on a trade-off between risk and control overhead.

4.6.8 TCM: Role-Assignment Strategies

The management of a Trusted Community is delegated to the TC manager. For this, the agent is equipped with TCM-strategies that require data for the decision-making, and specify tasks to retrieve and process this data. These tasks have to be executed to ensure the operation and optimisation of the TC. However, as discussed in Sec. 4.5, the TCM is a self-interested agent like any member. As such, the TCM is interested to distribute the responsibility for the execution of these tasks to its members and reduce its own effort to the processing of the task results for the decision-making. The key principle of the TC approach is hence to **divide the management tasks of the TCM to roles, and assign these roles to TC members**. TC members are obliged to execute the associated tasks. This is controlled through the execution of the *Member Control* strategies (see Sec. 4.6.7) and motivated with the incentive to remain a TC member.

[5]A rational and self-interested agent will never exclude itself from a TC if it provides an increased utility to it. If the benefits are not provided, the agent leaves the TC anyway, based on the execution of its *Membership Evaluation* strategy. There is hence no reason for a TCM to consider its own exclusion.

The organisation of agent collaboration by the division of responsibilities to roles and assignment of these roles to agents is one of the most active research fields in Multiagent Systems (cf. e.g. [195]). The application of roles is reported to allow for an engineering perspective to MAS (specification of functionalities without the need to specify responsible executors), allow for the development of specialisation, and reduced competition for tasks. For the purpose of TC management, the engineering approach is clearly in the focus: Management roles specify the functionality required to maintain the operation of the TC without directly determining which members are responsible for which functionality. This is especially useful as the dynamics in the system (TC members being included and excluded) require to frequently reassign the responsibilities. This view on roles, adhered to in the remainder of this thesis, classifies roles as *functional* (as opposed to *social*), *explicit* (as opposed to *implicit*), and *individual* (as opposed to *collective* requiring coordination among the executors) (cf. e.g. [195]). Furthermore, the meta role specifying the assignment of roles is taken solely by the TCM (as opposed to collective role assignment with consensus approaches). Finally, this allows to formalise the *Role-Assignment* strategies as follows: The TCM assigns each role from the set $\Psi_{TC_i}(t)$ to its TC members $\mathcal{M}_{TC_i}(t)$, such that:

$$f^{roles}(\mathcal{M}_{TC_i}(t), \Psi_{TC_i}(t)) := \left\{ (m, \Psi_m(t)) \;\middle|\; m \in \mathcal{M}_{TC_i}(t), \Psi_m(t) \subseteq \Psi_{TC_i}(t) \right\} \tag{4.20}$$

with $\Psi_m(t)$ being the subset of roles assigned to a member m at time t, such that all roles are assigned to the agents:

$$\bigcup^{m} \Psi_m(t) = \Psi_{TC_i}(t),$$

but no role is assigned to more than one agent:

$$\bigcap^{m} \Psi_m(t) = \emptyset$$

The TC manager $tcm_{TC_i}(t)$ then performs the role assignment operation for all pairs $(m, \Psi_m(t))$ generated by the execution of the strategy, to inform all members about the new role allocation, such that:

$$\forall\, (m, \Psi_m(t))\;\; \forall r \in \Psi_m(t) : r \rightsquigarrow^{tcm_{TC_i}(t)} m$$

The division of TC management tasks to roles and the execution of *Role-Assignment* strategies to distribute these to TC members increases the robustness of the TC approach: The reduction of the regulatory overhead for a TCM is **self-protecting**. This is due to the fact that a TCM is a self-interested agent that joined a TC to gain some benefit (also see the discussion in Sec. 4.5). An exclusive occupation of a TCM agent with management tasks would substantially lower this benefit and consequently render the TCM position a highly undesired one. Agents would have a strong incentive to not be elected as TCM, and once elected, to immediately leave a TC and try to join another TC as regular member. This would decrease the stability of a TC. Additionally, these strategies also make a TC **self-optimising** as an effective management division and role assignment can lower the overall overhead of managing the TC. This overhead reduction as superior goal of the *Role-Assignment* strategy execution is then guiding the following basic implementation of this strategy.

Basic strategy implementation

The basic implementation of this strategy is based on the following considerations:

The set of roles specified for a Trusted Community represents the total overhead of TC membership. The assignment of these roles to the members is then a distribution of this overhead. The members of a TC are expected to possess the *willingness* to execute assigned roles (other cases must be accounted for by the *Member Control* strategies as described in Sec. 4.6.7). The TCM does hence not need to reason about the reliability of the role execution. However, it is assumed that the members of a TC also possess a *competence* to execute assigned roles: Each agent m_i has a scenario- and TC-configuration-specific capability to execute tasks associated to a role r_a, and this allows for a quantification of the execution costs c_{m_i,r_a} for this agent. Additionally, it is assumed that the *competence* is heterogeneous among the agents, such that the costs incurred by the execution of a role r_a can differ for two agents, with $c_{m_i,r_a} \neq c_{m_j,r_a}$. It is further assumed that the TCM knows these costs for its members. The basic implementation of the *Role Assignment* strategy is then aimed at allocating the roles $\Psi_{TC_i}(t)$ to the members in a way that minimises the total costs and thus the overhead for TC membership.

Given these assumptions, the above defined function is then interpreted as *(linear) assignment problem*. This allows to apply verified algorithms from the literature. Details about these algorithms, as well as the discussion of their application for the assignment of roles to TC members can be found in appendix A.2. Summarising this approach, it can be stated that if the scenario-specific nature of the roles allows for the formulation of a *cost matrix*, such that the costs for the execution of a role by an agent can be quantified, then the implementation of the *Role Assignment* strategy can be the algorithm to solve an assignment problem. The exact choice of that algorithm then depends on the number of roles in relation to the number of agents.

Implementation requirements and refinement

So far, the description of the *Role-Assignment* strategies has assumed an available set of roles $\Psi_{TC_i}(t)$ that encapsulate the required TC management tasks and need to be assigned to the TC members. The process of dividing the required TC management into these roles has however not been defined. In fact, the division of a system's functionality to appropriate roles is still one of the open issues in the literature on (explicit) role allocation (cf. e.g. [195]). A generic procedure has thus far not been proposed, but the modular design of the TC approach allows for at least coarse guidelines as how to divide the functionality into roles: The total of distributable management tasks is derived from the decision-making of the TCM-strategies. Consider the following examples:

- *Basic Active TC Expansion* strategy: As described in Sec. 4.6.6, the TCM can actively search the unassociated agents for potential TC members by determining which agents it estimates as trustworthy. However, this limits the search to agents that had interactions with the TCM and thus does not cover a large set of potential agents. The basic implementation hence introduced the processing of sets of member candidates $\mathcal{P}^m(t)$ proposed by single TC members m, to explore more unassociated agents. Depending on the number n of such opinions, the TCM can utilise a set of *exploration roles* $\{r_1, .., r_n\}$: An agent m assigned such a role is then responsible for the tasks of (periodically) generating the set $\mathcal{P}^m(t)$ by executing its *Potential Member Search* strategy, and of sending this data to the TCM.

- *Basic Member Control* strategy: As described in Sec. 4.6.7, this strategy realises a sanctioning scheme for undesired TC member behaviours as observed by the TC members. One source of these observations are the members themselves that have an instrument to complain about each other in case of negative interaction outcomes. The second source described by the basic strategy are the *collective observations* $o_b^{x,t}$ of members. Here, the TCM can generate observation tasks that are associated to *observation roles*. These tasks can for example incorporate the mediation of inbound interactions, or the explicit testing of members (e.g. by spot-checking in Desktop Grid Systems). These observations are then passed to the TCM that can process them and decide about sanctions.

- *Basic Member Control* strategy: This strategy allows for additional division of roles. As discussed in Sec. 4.6.7, the TCM applies a scoring system to track the sanctions of its members and judge on their exclusion from the TC. The score $score(x, t, thres_f)$ for each agent x is then an aggregation of these sanctions. However, the storage of these scores at a central position, the TCM, is a single point of failure: If the TCM leaves the system or the TC, the member scores are lost and the next agent elected as TCM must work with a whitewashed view on the TC members. It is hence advisable to generate *storage roles* that are constituted by tasks to store a set of scores for other members and provide access to it to the TCM (cf. *distributed hash tables*). If a new TCM is elected, it only needs to determine from which agents to retrieve which data, but the data itself is present and can be used by the TCM. This further increases the robustness of the approach.

Further roles are then derived from specific implementations of the TCM-strategies and can incorporate scenario-specific tasks. Note here that the execution of these roles is perceived as overhead (of TC membership) by the agents. This is because they require actions that would usually not be performed by the agents: The roles have absolute costs (e.g. communication or processing costs), as well as opportunity costs (the resources required for the execution of roles are blocked by the execution and cannot be used for other actions that can raise the agents' utility) associated to them. This also allows to specify the costs of the execution: An agent that represents a client with low amounts of energy, disk space, communication bandwidth etc. will have higher costs for the execution of these roles than an agent with an abundance of these resources.

Given the existence of such management roles, the assignment of these roles has been formulated as *assignment problem* in the *Basic Role-Assignment* strategy. The implementation then utilised the (modified) *Hungarian Method* to find an assignment that minimises the summed costs for the execution of these roles. This assignment goal is however not without alternatives:

- *Fairness goal*: The minimisation of the total costs does not enforce *fair* role assignments. Members with low costs (due to high amount of resources) are assigned more roles than agents with low resources. Also, the basic implementation is not iterative, thus it does not account for past role allocations. This will result in capable agents being constantly occupied with roles. Dependent on the realisation of their *Association Evaluation* strategy, this can influence their decision to remain a member. An alternative implementation might hence have the goal to enforce fair role assignments based on even long-term distribution among all members. Higher total costs are then accepted. The even distribution of roles can also help to increase the robustness against the impact of leaving members. Consider for example an agent with many important roles leaving the TC.

- *Preservation of TC composition goal*: As discussed e.g. in Sec. 4.6.7, each leaving member of a TC can decrease the utility for the other members as it reduces their choice of partners for optimised interactions. Agents leave a TC when the benefit of TC membership is not strong enough. This is negatively influenced by assigned roles, because of their execution costs. An alternative implementation can hence be primarily aimed at preserving the composition of a TC by relieving agents that are close to leaving a TC from their roles. This requires firstly the information about the members disposition to leave a TC, and secondly involves the potential for conflicts and exploitation by members. The decision-making must hence allow for a fine-grained balancing of the various interests.

Additionally, a changed TC composition or set of roles do not necessarily require to reassign all roles. Instead, an iterative approach could be chosen to reassign only those roles that have been unassigned by the exclusion of the formerly responsible agent. Such an implementation however only allows for the reaction to such change events and not for the active avoidance of them. It has therefore not been applied in the basic implementation.

Apart from these TC specific ideas, the body of literature on MAS *role allocation* is rather large and allows for various additional approaches to be used in this context. Finally, all implementations must avoid the case of affecting the TC stability by assigning roles to single members such that they are driven away because of the associated overhead. In the worst case, this generates feedback effects and leads to the dissolution of a TC. On the other hand, the best case is to assign roles such that the costs of TC management are minimised while the stability of the TC is preserved through fair role assignments.

4.6.9 TCM: TC Observer and TC Controller

The TC Manager is responsible for the management of its Trusted Community. So far, strategies for the inclusion of new members, the exclusion of uncooperative or adversary members, as well as strategies for the assignment of roles have been presented. These represent the main activities required to allow for a robust TC operation, by realising the properties *self-protection*, *self-healing*, *self-explanation*, and *self-optimisation*. For each of these TCM-strategies a basic implementation, as well as directions for refined implementations have been proposed. These implementations contain many parameters and require a fine-tuning towards the application in the hosting system. Here, the question arises, which implementations of these strategies should be applied by the TCM for the management of a certain TC, and how these implementations should be parameterised. This is especially challenging when the characteristics of the hosting system are considered: As discussed on many occasions in this thesis, the hosting system is a complex system with high dynamics. Emergent states, such as a *trust breakdown*, can occur due to agent interactions. Additionally, agents, even TC members, can strategically adapt to their environment, interacting with each other, and expressing behaviours unforeseen at design time. While the TCM has been given instruments to cope with such threats, the exact utilisation of these instruments cannot always be specified in advance. The TCM rather requires tools to analyse its management, based on the received performance, and to adapt it when necessary. This is beneficial for both, the prevention of inappropriate management in abnormal situations (e.g. utilisation of trust-based strategy implementations despite trust breakdown), and optimisation of the management to the current situation (e.g. adaptation of thresholds to account for changes in agent behaviour).

This requirement for self-aware observation and control at runtime is the focal point of the *Organic Computing* initiative. Research in this field has resulted in the specification of tools for such environment. In the following, the *TC Observer and TC Controller*, realising the *OC*-tool *Observer/Controller design pattern* (cf. e.g. [134], [38]), are presented as final TCM-decision-making strategy.

Observer and *Controller* have already been introduced in Sec. 3.1.2 as central part of the agent model. Their defining design pattern allows for the implementation of a regulatory loop to optimise and adapt the control over a *production engine*. In the agent model, the production engine has been directly linked to a *client* for the participation in the hosting system. The observation and control of this client, especially its performance expressed in terms of a utility $U^x(t)$, then allows agents for a situation-aware adaptation of the client control. Agents can for example exchange their TC strategies at runtime, and thereby their decision-making and control, based on observations of the hosting system or their own state. While *TC Observer* and TC *Controller* adhere to the same principle of a regulatory loop, this loop is located on a higher level of hierarchy (cf. e.g. [134]). Here, the term *system under observation and control* (SuOC) fits the description better than the term *production engine*: This SuOC is the *Trusted Community* managed by the TCM. The *TC Observer* and *TC Controller* are hence utilised for the **observation of the TC environment, and adaptation of the TC management based on these observations**.

For a system designer, it is possible to analytically evaluate the performance of TC application for its members a posteriori. However, in order to adapt the management of a Trusted Community during its operation, the TC Manager needs to assess the Trusted Community **at runtime**. A metric is hence required that allows for the assessment of this *SuOC*, equivalent to the utility function $U^x(t)$ for the assessment within the agent's O/C-loop. The realisation of this metric via a generic function $U^{TC_i}(t)$ is based on the following considerations:

- The aim of a Trusted Community is to provide an environment in which agents increase their utility (compared to the utility when being unassociated). As described in Sec. 4.6.2, this is the reason why self-interested agents join a TC, and as described in Sec. 4.6.4 the lack of such increase is the reason why agents leave a TC. The goal of the TCM must then be to execute its TC management such that it increases the utility of its members. This can be expressed with a *relative TC utility* quantifying the utility benefit for all TC members.

- The TCM can request private information (such as the current utility) from its members, and the members have an incentive to provide this information as they are sanctioned if they reject to do so.

- The members also have an incentive to provide *valid* private information as this information is used to improve the benefits of TC membership in general, and their own benefit in particular.

Given these considerations, the *relative TC utility* U^{TC_i} is formalised by the following function:

$$U^{TC_i}\left(U^{m_1}(t),..,U^{m_{|\mathcal{M}_{TC_i}(t)|}}(t),U^{m_1}(t_a),..,U^{m_{|\mathcal{M}_{TC_i}(t)|}}(t_a)\right) \tag{4.21}$$

$$:= \frac{1}{|\mathcal{M}_{TC_i}(t)|} \cdot \sum_{k=1}^{|\mathcal{M}_{TC_i}(t)|} U^{m_k}(t) - U^{m_k}(t_a)$$

which requires the *current* utility values $U^{m_k}(t)$ of each member m_k, as well as the utility values $U^{m_k}(t_a)$ for the time t_a of the TC association (when the agent was granted membership). The sum of the utility differences is divided by the number of members a TC has in order to decouple the utility value from the TC size and allow for comparisons between TCs.

Let the utility function $U^{m_k}(t)$ of an agent be assigned values from the interval $[0,1]$, then the relative utility $U^{m_k}(t) - U^{m_k}(t_a)$ of a single agent takes values from the interval $[-1,1]$. Consequently, the relative TC utility function takes values from the interval $[-1,1]$. When the function approaches 0 though, this means that the TC does not, in average, generate benefit for its members.

Now that the metric for the adaptation of TC management is defined, the information to observe and the control to adapt by the TCM can be examined: The management of a Trusted Community is executed via implementations of the TCM strategies *Active TC Expansion*, *Member Control*, and *Role Assignment*. Additionally, the *Distributed Leader Election* strategy is counted as TC management strategy, because it is executed at the operation phase of a TC and indirectly determines the realisation of the other TC management strategies by electing an executing TCM. These are the instruments that are at disposal for the Controller. These strategies must be executed such that they result in a high *relative TC utility* U^{TC_i}. The outcomes of the execution are however dependent on the environmental state: The hosting system, as well as the TC, are subject to dynamics that influence this outcome. The TC Observer must hence add a description of this environment to its observation of the outcomes. In sum, the purpose of this O/C-loop is to **determine the right sets of TC strategy implementation parameters for a given situation, based on the resulting *relative TC utility***.

Consider the following illustrating example: The *Basic Member Control* strategy, described in Sec. 4.6.7, defines how TC members should be sanctioned, based on the specification of a *forgiveness threshold* $thres_f$ and a *maximum score* $score_{max}$. The setting of these parameters determines how TC members are excluded from the TC. If for example, *forgiveness* and *max score* are set very high, TC members are allowed for many uncooperative actions before finally they are getting excluded. On the other hand, a very strict interpretation that applied close to no *forgiveness* and a low *max score* punishes TC members for uncooperative actions directly and ultimately. But how should these parameters be set? This is dependent on the application scenario and can be only approximately be defined at design time. Rather must the environment be observed to allow for the best setting: Do TC members constantly exploit high *max scores* by being frequently uncooperative? Or do uncooperative member agents come in bursts and agents express a high degree of cooperation before? This may be hinting at limited periods of overload that result in the rejection of further commitment. In order to analyse such interrelations, the observer must here for example monitor the patterns of uncooperative member behaviour, their degree of commitment for other members etc. Then the controller can adapt these parameters and the TCM can evaluate whether these adapted parameter settings lead to a better TC management in terms of a higher *relative TC utility*.

The ability to set the parameters of the applied TCM-strategies is a *low-level adaptation* of the TC regulation. Parameters can take many different values and the exploration of the result space is a time-intensive optimisation problem. Apart from these adaptations, the O/C-loop however allows for further *high-level adaptations*: The result of the TCM-strategies is obviously determined mostly by their implementation. In this thesis, basic implementations of these strategies have been defined, and scenarios discussed in which refined implementations can be applied to improve the execution results. The high-level adaptation focusses on the adaptive application of the most suited strategy implementation for a given situation. This optimisation problem is then much more limited than the previously defined one: **Given a set of strategy implementations, the O/C-loop must determine which implementation should be chosen to achieve the highest *relative TC utility* for a given situation**. Obviously, the difficulty of this problem is then related to the number of available implementation alternatives (as depicted in Fig. 4.3).

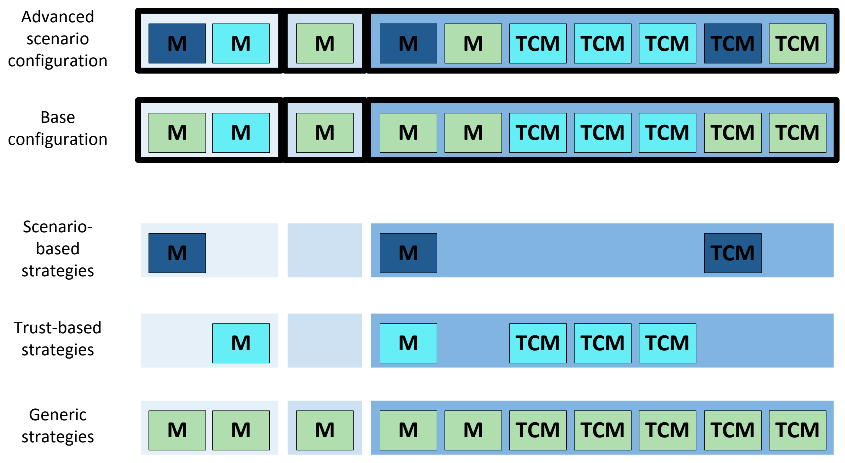


Figure 4.3: TC strategy implementations (for (M)ember or (TCM)-strategies) to choose from for a TC configuration. The choice of the best implementation for a given situation is part of the TCM responsibility and realises as search problem via an O/C-loop

In the realisation of such a TC O/C-loop, the following assumptions are made: (1) The TCM does not need to determine an initial set of control parameters or strategies. Rather is it equipped with a default implementation of the TCM-strategies (as part of $\Lambda_{TCO}^{default}$, see Sec. 3.2.3) that also presets the required parameter values. The aim of the TCM is hence to optimise these settings, but not to find an initial working configuration. This is equivalent to the assumption of training data for the case of machine-learning. (2) Just as the availability of parameter settings for a default system state is provided for, the TCM can also refer to strategy implementations for known disturbance states. This allows the TCM to utilise these special strategies in case it detects such a state. Apart from scenario-specific disturbance states (e.g. overload in Desktop Grid Systems), a particularly relevant state is the *trust breakdown* referred to frequently in this thesis. In such a state, the TCM must assume that trustworthiness data is incorrect, and hence requires strategies that refer to other information for the decision-making. The TCM hence knows beforehand that the application of trust-based TCM-strategies will have a low utility when applied in a trust-breakdown scenario, and that the application of trust-independent strategies will increase this utility. (3) The execution of *TC Observer* and *TC Controller* must **at least determine whether a TC should be dissolved** (for further detail see appendix B). This minimal requirement is motivated by the following considerations: The processing of a TC utility function by the O/C-loop allows to determine whether a TC should be dissolved, because it indicates that the TC provides no benefits to its members ($U^{TC_i} \leq 0$, see the discussion for Eq. 4.21). Although the TC members can evaluate the lack of their own benefit by execution of their *Membership Evaluation* strategies, which also applies to the TCM, it is generally more beneficial to dissolve a non-proficient TC than to try and preserve it. This is especially true because cooperative members can form new TCs.

Finally, the realisation of the O/C-loop, such that it allows for a performance-driven TC management adaptation (*low-level* and/or *high-level*), enforces the self-x property **self-optimisation** of the TC approach. Additionally, the situation-aware choice of strategy implementation increases the TC robustness towards abnormal system states by making the TC **self-protecting**. Overall, both types of adaptations are additionally forms of **self-configuration**.

In the following, a basic O/C-loop implementation is presented and additional implementations discussed.

Basic strategy implementation

The basic implementation of the O/C-loop is focused on an aspect of system operation mostly neglected in the literature: The maintenance of the successful operation of a MAS organisation, managed with trust-based decision-making, in the event of a trust breakdown (as defined in Sec. 3.1.4).

This realisation of an O/C-loop is hence targeted at the *high-level* adaptation of the TC management: After detecting a trust breakdown, a TCM must adapt its TCM-strategies such that it chooses implementations for execution that do not require a valid TM operation. This is based on the consideration that the application of strategy implementations with such a requirement in a trust breakdown state have no effect on the TC management (in the best case) or even a detrimental effect on it. Consider the following two examples:

1. The application of a trust-based implementation of the *Distributed Leader Election* strategy (as presented in Sec. 4.6.5): In a trust breakdown scenario, most TC members can be expected to have a low or negative reputation (see Eq. 3.7). The choice of a TCM based on its reputation in the TC, as proposed by the basic implementation, hence allows no statement about its willingness to manage the TC. Here, the application of this strategy implementation can have a negative influence on the utility of a TC, if an adversary agent whose negative behaviour is hidden by the equality of low trust values is elected as TCM. A trust breakdown is however hardly predictable, the strategic adaptation of an adversary agent to become TCM only in case of a trust breakdown is therefore a possible, if unlikely threat. Thus, in most cases it can be expected that a TCM chosen based on the rather random reputation value is cooperative. Nonetheless, the *random election* of a TCM is less costly (in terms of duration and message complexity, see Sec. 4.6.5), and revealing less private information. Its application should hence be preferred in this scenario.

2. The application of a trust-based implementation of the *Active TC Expansion* strategy (as presented in Sec. 4.6.6): The aim of this strategy is to compensate leaving TC members and to op-

timise the TC composition by finding new potential TC members in the set of unassociated agents. The proposed basic implementation of this strategy utilises the opinions of members about unassociated agents to make this selection. More precisely, the TCM requests the output of their *Potential Member Search* strategies. The basic implementation of this strategy is based on a trust threshold as main criterion for the suitability of unassociated agents. In case of a trust breakdown, this threshold is never met due to the low or even negative trust values of agents in the system. The respective TCM *Active TC Expansion* strategy will hence receive only empty sets as input and consequently never determine any agents to invite. The strategy can thus not fulfil its task of preserving and optimising the composition of the TC. While the latter task can be neglected in case of a satisfying *relative TC utility*, the failure to compensate leaving TC members will eventually lead to the dissolution of the TC. By the utilisation of a trust-independent *Active TC Expansion* strategy, such as a *test-task*-based approach (see discussion in Sec. 4.6.6), the fulfilment of these tasks can be maintained and negative influences on the TC operation avoided.

Above, the assumption has been made that the TCM is aware of alternative non-trust-based implementations for its management strategies, such that it can decide to utilise these instead of the trust-based implementations. Let Λ^{trust}_{TCO} denote a *trust-based* TC organisation component (as defined in Sec 3.2.3), i.e. an implementation of this component that uses trust-based TCM-strategy implementations. Let then Λ^{ntrust}_{TCO} be an implementation of the TC organisation component that is composed of TCM-strategy implementations that do not require trust (the above mentioned alternatives). Let additionally the binary variable $\mathcal{I}(\Lambda^{trust}_{TCO})$ denote whether the *trust-based* implementation is used or not. Let further the function $switch(\Lambda_{TCO})$ denote the exchange of the TC organisation component implementation by the TCM. In reference to the definition of a *trust breakdown* presented in Sec. 3.1.4, the basic implementation of the O/C-loop can then be formulated as the realisation of the following function:

$$f_1^{O/C}\left(\mathcal{B}^{\hat{c}}_{\mathcal{G}(t)}(m)\ ,\ \mathcal{I}(\Lambda^{trust}_{TCO})\right) := \begin{cases} true, & \text{if } \mathcal{B}^{\hat{c}}_{\mathcal{G}(t)}(m) \wedge \mathcal{I}(\Lambda^{trust}_{TCO}) \\ true, & \text{if } \neg\mathcal{B}^{\hat{c}}_{\mathcal{G}(t)}(m) \wedge \mathcal{I}(\Lambda^{ntrust}_{TCO})\,) \\ false, & \text{else} \end{cases} \tag{4.22}$$

This function encodes the decision whether the $switch(\Lambda_{TCO})$ operation should be applied by the TCM: In case of a trust breakdown, the TCM arranges for the utilisation of trust-independent implementations, and vice-versa. The remaining question is how the TCM determines $\mathcal{B}^{\hat{c}}_{\mathcal{G}(t)}(m)$, hence whether there is a trust-breakdown in the group $\mathcal{G}(t)$. This is approached based on the following consideration: The trust breakdown is defined over the aggregation of direct trust values within a reference group $\mathcal{G}(t)$ of agents, also referred to as the *reputation* of agents within this group. In order to evaluate the presence of a trust breakdown, the TCM must firstly define this reference group $\mathcal{G}(t)$, and secondly acquire[6] the reputation values for this group of agents. A relevant agent group $\mathcal{G}(t)$ is determined as such: All TCM-strategies that consider a set of agents and require positive trust relationships within this group, provide subsets of this group. In the basic implementations, these are the group of unassociated agents $U^{\mathcal{H}}(t)$ (for the *Active TC Expansion* strategy implementation),

[6]Note that in case of the availability of a global reputation value $RT^{y,\hat{c}}_{\mathcal{A}(t)}$, the global trust breakdown $\mathcal{B}^{\hat{c}}_{\mathcal{A}(t)}(m)$ is trivially observable.

and the group of TC members $\mathcal{M}_{TC_i}(t)$ (for the *Distributed Leader Election* strategy implementation). The composition of this group is hence:

$$\mathcal{G}(t) := U^{\mathcal{H}}(t) \cup \mathcal{M}_{TC_i}(t)$$

The TCM then divides this set of agents, and distributes the task of collecting of according reputation values as roles among its members (see Sec. 4.6.8 for the discussion of the understanding of roles).

Additionally, the basic implementation of this O/C-loop is used to determine whether a TC should be dissolved (for further detail see appendix B). The basic approach to this is formalised by the following decision function:

$$f_2^{O/C}\left(\left|\mathcal{M}_{TC_i(t)}\right|, U^{TC_i}\left(t, U^{m_1}(t), .., U^{m_{\left|\mathcal{M}_{TC_i}(t)\right|}}(t), U^{m_1}(t_a), .., U^{m_{\left|\mathcal{M}_{TC_i}(t)\right|}}(t_a)\right)\right) \tag{4.23}$$

$$:= \begin{cases} true, & \text{if } \left|\mathcal{M}_{TC_i(t)}\right| = 1 \\ true, & \text{if } U^{TC_i}\left(t, U^{m_1}(t), .., U^{m_{\left|\mathcal{M}_{TC_i}(t)\right|}}(t), U^{m_1}(t_a), .., U^{m_{\left|\mathcal{M}_{TC_i}(t)\right|}}(t_a)\right) \leq 0 \\ false, & \text{else} \end{cases}$$

This implementation causes the TCM to dissolve its TC when it is its only member (see requirement definition in Sec. 4.2), or if the *relative TC utility* is below 0, such that the TC provides no benefits to its members (see the motivation for this in the above description of this utility function).

Implementation requirements and refinement

The application of an O/C-loop for the situation-aware adaptation of the TC management by the TCM is a potent tool. In the basic implementation of this loop, only the *high-level* adaptations have been exploited. A refining implementation can complement this approach by the utilisation of *low-level* adaptations, i.e. the adaptation of strategy *parameters* instead of strategy *implementations*. In this, the literature on Organic Computing provides many case studies that can serve as blueprints for such a realisation. Consider for example the realisation of an O/C-loop with a Learning Classifier Approach for the control of a Traffic Light System (cf. e.g. [31]). The basic implementation presented here would benefit from such an approach in the following way: The trust breakdown observation is based on the utilisation of a parameter m, for the quantification of the percentage of agents with a low reputation in the reference group (see Eq. 3.7). Consequently, the basic implementation of the O/C-loop presented defines such a parameter m to specify whether the observed reputation distribution in a group $\mathcal{G}(t)$ should be interpreted as trust breakdown (see Eq. 4.22). Now if machine-learning-based *low-level* adaptations of this strategy parameter are utilised, the proactive detection of trust breakdowns is enabled: A trust breakdown does not occur instantly, rather does it follow a monotone development leading to low reputation values of a majority of agents ($m \cdot |\mathcal{G}(t)|$). An adaptation of m to a lower value results in a faster detection of a trust breakdown. However, not in each such situation will a trust breakdown actually develop. The distribution of low reputation values can also be a temporal state in the system from which the system recovers. The O/C-loop has hence the responsibility to analyse the correlation of the system states with respect to the indication of a later trust breakdown. Finally, the fast detection of this emergent state allows the TCM to increase the robustness of its TC, by switching to non-trust-based strategies earlier.

A different approach that can be realised by the utilisation of a refined O/C-loop is the evaluation of the synchronisation of the TCM-strategies. Consider the following example: The *Active TC Expansion* strategy is responsible for the inclusion of new members in the TC. Complementary, the *TC Member Control* strategies exclude members from the TC. Both operations influence the *relative TC utility*, which can be observed by the TCM. An adaptation approach here could allow to analyse the agents invited to the TC with respect to their influence on the TC utility. As long as this strategy implementation would invite agents that increase the utility, it would be left unaltered. If however the utility would steadily decrease after the inclusion of new agents, the invitation process should be stopped. If respectively implemented for the exclusion of agents, such adaptation could be utilised to determine the composition of a TC that yields the highest *relative TC utility*.

Finally, the worst case implementation of the O/C-loop would decrease the stability of a TC by the misconfiguration of the TCM-strategies, such that they would in turn provide only worst case behaviour. The best case of the implementation allows for a full situation-aware *low-level* and *high-level* adaptation that chooses and configures the TCM strategies such that they operate in their best case. This would in turn result in a high *relative TC utility*.

This concludes the description of TC strategies and their implementations. In the following, the configuration of a TC as a whole is briefly discussed, and the chapter summarised.

4.6.10 Strategy Configuration

The strategies included in the TC approach, and their implementations, capture the behaviour of this approach. In this, the basic implementations of the strategies provided in this chapter are generic and can be utilised in all instances of the hosting system class. Additionally, the hosting system may allow for improved implementations that utilise system-specific mechanisms. While the configuration of the TC approach with the choice of suited implementations at design time may not provide for optimal results in all system states, the approach is self-optimising by design: The adaptation of the TC functionality by the utilisation of *TC Observer and Controller* strategy implementations can be exchanged at runtime such that they better match the required functionalities. This is detailed in appendix B where the design of the TC organisation agent component $Comp^{x}_{TCO}$ is presented.

4.7 Summary

After the previous chapter ended with a motivation for the application of Trusted Communities in a hosting system, this chapter was dedicated to the presentation of the TC design. At the beginning, a formalisation for the TC structure and its dynamics has been defined. The main focus here has been on the specification of *TC operations*: The *formation* of a new TC, the *dissolution* of a TC, the *inclusion* of a new TC member, the *exclusion* of a TC member, and finally the assignment of a role to an agent. The chapter continued with a classification of TC *Organisation Benefit* strategies. These strategies capture the core motivation of becoming a TC member for an agent, as they allow more and optimised interactions among TC members, and thus can increase the utility of an agent. These strategies are scenario-specific, but can always be assigned to one of the three classes (1) *Interaction Efficiency*, (2) *Cooperation*, or (3) *Information Sharing*.

The following section has then presented the complete *lifecycle of a Trusted Community*, introducing the three phases in which a TC can be: (1) *Pre-Organisation Phase*, (2) *Formation Phase*, and (3) *Operation Phase*. Decision-making that needs to be executed in each of the phases has then been introduced by the discussion of *TC strategies*, which are encapsulations of parts of this decision-making. Additionally, the robustness of a TC has been brought up, and the inspiration for its realisation has been discussed: The *system theoretical* view on robustness states that in order to allow for robustness, a system must have the generic properties *System control*, *fail-safe redundant mechanism*, *modularity*, and *decoupling*. Additionally, the *Organic Computing* initiative advertises the requirement for *life-like* properties in such systems. These constituting properties of *self-organisation* are: *self-configuration*, *self-healing*, *self-protection*, *self-optimisation*, and *self-explanation*. Here, the realisation of the TC approach as system with such properties has been discussed.

The following section has presented a rationale for the utilisation of a hierarchy in a TC, by examining the head of this hierarchy, the *TC Manager*. Here, the motivation for the delegation of control by the TC members to such a *TCM* has been discussed. This control refers to the regulation of the TC and is especially concerned with the decision-making for the execution of the above mentioned *TC operations*.

The following section of the chapter has then presented each of the TC strategies in detail: First, the general encapsulated decision has been described, followed by a formalisation of this decision, and a basic implementation of the strategy. Finally, the requirements for the implementations have been analysed and refinements or alternatives to the basic implementation have been discussed. Additionally, the *self-X* properties of the TCM-strategies have been explained. In conclusion, the configuration of the TC organisation agent component has been discussed, especially with respect to its contribution to the observation model.

5 | Evaluation

The preceding chapter covered the detailed presentation of the Trusted Community concept. In this chapter, the evaluation of this concept in an exemplary application scenario is presented.

This exemplary instance of a hosting system is the *Trusted Desktop Grid*, an open Desktop Grid System based on agents that control the grid client software (production engine). After an introduction to this system, the system model is referenced by providing the applied Trust Management system, as well as the agent model for this application scenario. In addition, evaluation results achieved without the application of Trusted Communities are discussed and the use of the TC approach is motivated. This is followed by the main part, the description of the application of Trusted Communities in this system. Here, configurations composed of TC member and TCM strategies, and organisation benefit strategies adapted for this scenario are provided. This is concluded by the presentation and discussion of the achieved evaluation results in the Trusted Desktop Grid, esp. in comparison to state of the art approaches.

5.1 The Trusted Desktop Grid

The Trusted Desktop Grid (TDG)[1] is an open distributed Desktop Grid System based on MAS- and Trust Management technology which satisfies all requirements of a hosting system for the application of Trusted Communities. In the following, the TDG is first defined and classified. Then the challenges in this system, and partial solutions to these challenges based on MAS and Trust Management, are discussed. The application of TCs to improve the performance and robustness of the TDG is motivated in conclusion. This TC application is then detailed, followed by a discussion of how the previously defined challenges can be tackled by TC application. Finally, the results of the evaluations (based on metrics discussed in appendix D) are presented and discussed.

5.1.1 System Classification

In this thesis, evaluations of the performance of the Trusted Community concept focus on the application in an open Desktop Grid System - the *Trusted Desktop Grid (TDG)*. Certain types of Desktop Grid Systems are instances of Open Distributed Systems and thus a valid choice for a hosting system. In the following, a taxonomy of DG Systems from the literature is used to classify the TDG.

Despite the clear differences to traditional Grid computing systems (see Sec. 2.5.2), Desktop Grid System realisations are nonetheless rather fragmented. In the following, the exact type of the DG

[1]Work on the *Trusted Desktop Grid* has been in part conducted in cooperation with Yvonne Bernard and Jan Kantert in the context of the DFG research unit *OC-Trust (FOR 1085)*.

system used as evaluation scenario is classified according to the taxonomy presented in [80]. This taxonomy is built upon four main categories called *perspectives*, containing several *properties* to classify DG systems. As depicted in Fig. 5.1, these perspectives are *System*, *Application*, *Resource* and *Scheduler*. The strongest classification is provided by the System perspective, therefore firstly

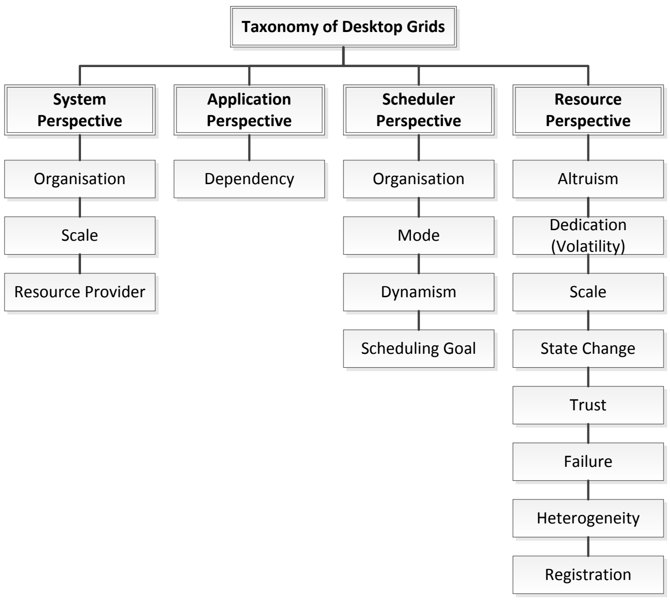


Figure 5.1: Excerpt from the Desktop Grid Taxonomy of [80]. The taxonomy classifies DG systems according to four perspectives and their associated properties. Here, the set of relevant properties for the TDG classification is depicted.

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). Here, researchers can benefit from the fact that nowadays computing power is often abundantly present and often not used exhaustively 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 (cf. [196]) 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 Trusted 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 appropriate volunteer machines and inform them of new tasks to process. Rescheduling 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[2]. 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 DGs 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 DG systems: Here, 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 (cf. e.g. [198]). In turn, they are obliged to donate their own resources to other users. An exemplary implementation of such a system is the *Organic Grid*[3] (cf. [182]).

In summary, the organisation property discerns between the server-based Centralised DG and the Distributed DG 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 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

[2]The most prominent example is the *Berkley Open Infrastructure for Network Computing (BOINC)* (cf. [197]) enabling research institutions to establish projects and set up clients that submit tasks to volunteer machines via the BOINC-servers. An exemplary project is *Docking@Home*, a collaboration between the University of Delaware, the Scripps Research Institute and the university of Berkley. The aim of this project is to find effective drugs for diseases like AIDS and cancer by finding promising molecule binding combinations via simulation (called *docking*).

[3]This work is not related to the field of *Organic Computing* as introduced in Sec. 2.4

do so anonymously. Therefore, only distributed systems fulfil the requirements of an Open Distributed System as application scenario for Trusted Communities.

The remaining two properties of the system perspective are *scale* and *platform*. The scale property discerns between Internet-based and LAN-based DG 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 TDG relies on a middleware-based solution, however this property has no influence on the openness of a system and can therefore be neglected here.

This completes the classification according to the System perspective in the taxonomy presented in [80]. In summary, in this thesis DG Systems are referred to as *volunteer-based, distributed, P2P-based, internet-based DGs with client participation over a DG middleware*. The main classification is depicted in Fig. 5.2.

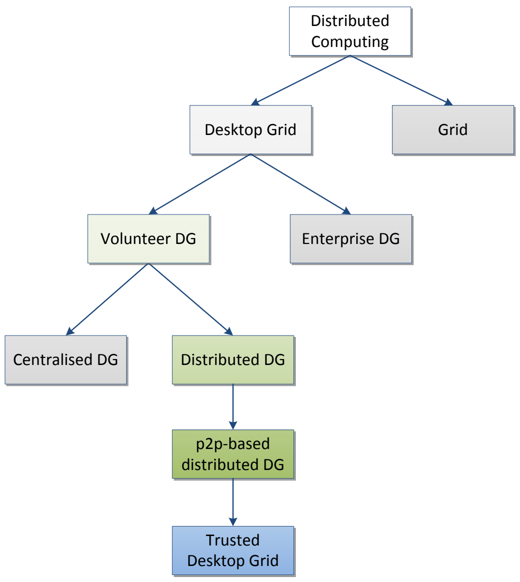


Figure 5.2: Hierarchy of distributed computing systems and classification of the *Trusted Desktop Grid*. The main discriminating characteristics are the volunteer-based approach and the decentralisation.

In the following, the taxonomy is used to classify the TDG according to a selection of relevant properties from the *application*, *scheduling* and *resource* perspective.

The Trusted Desktop Grid is a system where each client 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, 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 here, BoT-applications were used, as in the majority of research literature[4]. The metrics are hence chosen for the context of independent jobs and work units. Additionally, the application divisibility property classifies jobs according to their composition flexibility, i.e. 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 in general capable of both approaches here. As the main focus of this work is not on the distributed scheduling in a DG System, the evaluations were limited to the basic form of fixed divisions of jobs into work units.

The fine-grained definition of the types of jobs (or applications producing them) in the TDG now allows to classify 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 DG is implemented. As already defined in the system perspective, a distributed scheduling scheme was used here: All clients 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 this system, as most DG systems evaluated in the literature use 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 clients further operate in push mode. This means that clients send out WU processing requests to available workers, which then react with an acknowledgement or a rejection (scheduling mode property). For instance, this allows to detect free-riding despite decentralised control. The next relevant scheduling property is dynamism, where (dynamic) online and (static) offline scheduling are discerned. In an open system like the TDG, 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 sub-system in a DG. In the TDG, the metrics *speedup*, *waste*, and *accuracy* are used. For a definition, extensive analysis and rationale consider the performance metrics section in appendix D.

The *resource* perspective of the taxonomy specifies properties of single DG participants, the resource owners. In the TDG, participants are connected through the Internet (scale property) and donate their resources voluntarily. The latter implies that on the one hand, their resources are not dedicated to the DG, but users only donate spare resources with the availability of these resources being highly dynamic (dedication and state change properties). On the other hand, it means that altruism cannot be expected: Users can participate in the grid for the sole motivation of exploiting the resources provided by other participants without providing their own resources to the system (referred to as *free-riding*). Also, due to the openness of the system, participants are free to join and leave at will (registration property). With only low requirements on the joining participants, their machines are heterogeneous (heterogeneity property) and can be composed of unreliable hardware that produces errors (faulty property). Most importantly, the openness of the system, along with the autonomy of the participants, involves the risk of adversaries in the system (trust property).

This completes the classification of the TDG in this thesis. However, the taxonomy of DG systems followed here (cf. [80]) further classifies these systems according to many additional properties within the perspectives (see Fig. 5.1). In this thesis, a full systematic classification is not applied, as the

[4]However, the TDG is in general also suited for flow and execution-dependent applications, provided an according submitter component implementation.

TDG either does not have strict requirements on the remaining properties, or they are not in the focus of the TC applicability.

5.1.2 System Formalisation

In the following, the characteristics of the TDG, as classified according to the taxonomy, are described in more detail, and the underlying assumptions regarding the processing of work units are formalised.

The Trusted Desktop Grid is spanned between the machines of a group of hosts over a network. The owners of these hosts execute applications that produce DG jobs consisting of atomic work units. A user wants all generated jobs to be completed. For this, all contained work units must be processed until a valid result is obtained. In this, the host can request resources from other hosts in the system: By letting other hosts process its work units, several work units can be processed in parallel and hence the containing job can be completed much faster. In the remainder of this thesis, the following terms are used: A host that schedules the processing of own work units to other hosts or itself, is referred to as *submitter*. A host that processes work units, is referred to as *worker*. In the TDG, all hosts act as submitters as well as workers, though the participation as a worker cannot be taken as given. This is formalised as follows:

The TDG consists, at each time t, of m machines $M_1, .. \ ,M_m$. Throughout this thesis, the TDG is understood as working in a time-discrete way (in the evaluation, a simulation of the TDG is used), such that $t \in \mathbb{N}$. A time step is referred to as *tick*. A machine M_j is modelled according to the argumentation presented in [199] and others, i.e. by defining a binary host availability (host is on- or offline) and a resource availability $availableResources_j(t)$ as fraction of all potentially available resources for this host. The latter aggregates CPU-availability and task execution availability (as in [199]), hence it is assumed that, as long as a fraction of resources is available to the host, it can actually process a work unit[5]. As the available resources are varying for a host (non-dedicated machines, see above), also the available resources are dependent of a time t. Additionally, a machine M_j is characterised by a constant performance level PL_j that abstracts from its actual hardware performance (cf. e.g. [200]). Furthermore, computational jobs generated by DG applications are modelled as follows: A job ϕ_i^a belongs to an owner $a \in \mathcal{A}(t)$ and is composed of a finite number (denoted as $|\phi_i^a|$) of work units $\left\{\tau_1^a, \tau_2^a, ..., \tau_{|\phi_i^a|}^a\right\}$. Each job has a *release time* $r_{\phi_i^a}$ that denotes the time it was generated. A work unit τ_i^a is characterised[6] by the constant and abstract size $costs_\tau$ that provides an estimate for its computational intensity. Jobs are released when the division in work units is complete, hence work units share the release time of the containing job. The formalisations so far are depicted in Fig. 5.3.

This allows to define the processing model: As depicted in Fig. 5.4, each work unit τ_i^a from the containing job ϕ_i^a of owner a traverses a life-cycle: A work unit τ is released at release time t_τ^{rel} which begins its life-cycle. The life-cycle ends when there is a valid result for τ. This point in time is denoted with t_τ^{compl}. In between, the life-cycle is divided in rounds that can be interrupted at any of the

[5]This is because of the agent autonomy in the TDG and the assumption that an agent would not accept a processing request with good intentions if it was not able to process it.

[6]The abbreviated form τ is used wherever the owner of the work unit and its index inside the set of work units of a job are not important for the context.

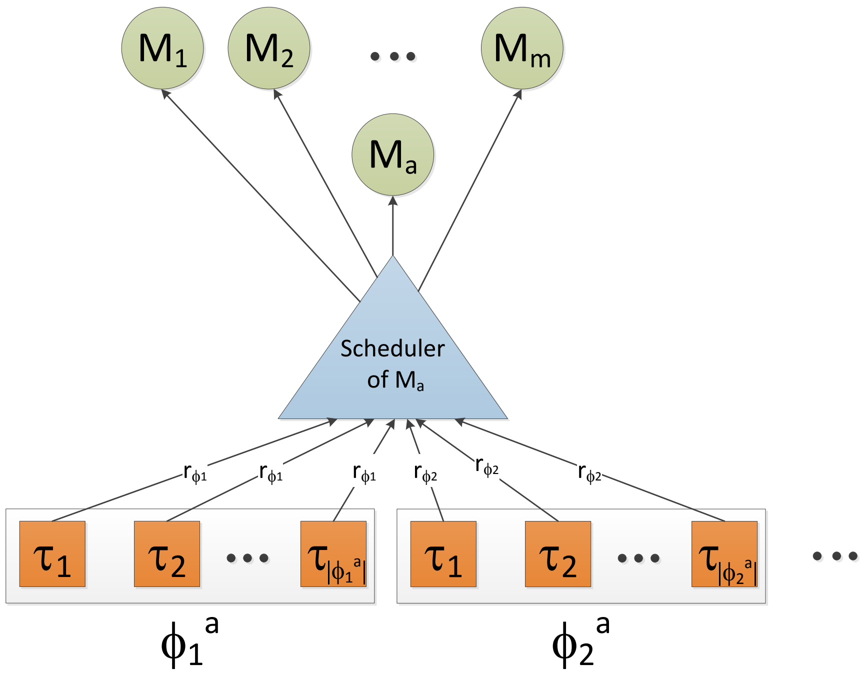


Figure 5.3: Overview of the job scheduling formalisation of a host in the TDG. Each machine M_a utilises its own scheduler to delegate the processing of work units τ_i from jobs Φ_j^a to other machines.

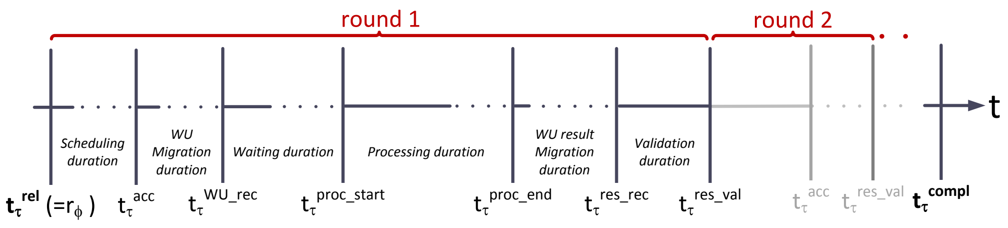


Figure 5.4: Life-cycle of a work unit τ from job ϕ in the Trusted Desktop Grid. The lifecycle begins with the release of the according job of the work unit and ends after at least one round of processing with the determination of the completeness of the processing. The time in between is divided into several periods, with the processing duration normally being the longest.

delimited time points t_τ. The major interrupting events of a round r are the rejection of a processing request by a potential worker (at time t_τ^{acc}) and the validation of a received WU result $t_\tau^{res\_val}$, which either completes the life-cycle or restarts a new round. Additionally, as the system is open, a worker can leave the system at any time during a round, also interrupting it. In the following, the different phases in the life-cycle are discussed:

At release time t_τ^{rel}, a WU is scheduled for processing. In the according *scheduling duration*[7], suited workers are requested to process the work unit (this process is described in detail in Sec. 5.1.4). In case of a positive response, at time t_τ^{acc}, the WU is transferred to the worker and put in a waiting queue. It remains there until the worker starts to process it (usually when the worker has completed the processing of the other WUs in the queue), this is denoted as the *waiting duration*. In the following, the WU is processed for a certain amount of time (*processing duration*, see Eq. 5.4) which generates a WU result (outcome). This result is either valid, denoted as o_τ^+, or invalid, denoted as o_τ^-, where a valid result can only be produced if the processing is fully completed. Subsequently, the WU result is transferred back to the owner of the work unit, where its accuracy is validated (*validation duration*), if the respective application supports this[8]. In case the validation detects an invalid result or the round has been interrupted (worker or submitter cancelling), the whole process has to be repeated in subsequent rounds, until finally a valid result is returned and the work unit is completed (time t_τ^{compl}). The time it took to completely process a work unit τ, i.e. including all necessary rounds to obtain a valid result, is denoted as completion duration and defined as:

$$completionduration_\tau = t_\tau^{compl} - t_\tau^{rel} \tag{5.1}$$

A job ϕ_i^a composed of n WUs is completed, when the owner a is in possession of valid results for each of the n work units τ_i^a. Hence, the completion time of the job is defined as:

$$completiontime_{\phi_i^a} = max(t_{\tau_1}^{compl}, t_{\tau_2}^{compl}, .., t_{\tau_n}^{compl}) \tag{5.2}$$

This accounts for the fact that these WUs can in general be processed in parallel (by different workers). The main (and in general also longest) phase in the life-cycle of a work unit is the *processing duration* a WU τ has when processed by a certain machine M_j. Note here that firstly M_j can also be the owner of τ, and secondly that processing is always dedicated to one WU at a time (no parallelism). This duration is of key importance here, since it is usually the longest time interval (from several minutes to days) in the life-cycle of a WU.

The processing duration can be defined for static, as well as dynamic environments. In case of constant values for available resources during processing, the duration is defined (cf. [200]) as:

$$processingduration(\tau, M_j) = \frac{costs_\tau}{PL_j \cdot availableResources_j} \tag{5.3}$$

However, the available resources are usually volatile in a DG environment, hence a more comprehensive definition is needed. The processing duration for dynamic resources, dependent on the

[7]Also referred to as task allocation phase in the literature (e.g. cf. [80]).

[8]The accuracy can either be validated by an according application (validation duration is only a fraction of processing duration) or the application does not support validation (validation time equals processing time). In this case, the *validation time* is neglected. In the TDG, both types of applications are evaluated.

starting time of the processing, is then defined as:

$$\sum_{t=t_\tau^{proc\_start}}^{t_\tau^{proc\_start}+processingduration(\tau,M_j,t_\tau^{proc\_start})} \frac{PL_j \cdot availableResources_j(t)}{costs_\tau} = 100\% \tag{5.4}$$

Here, the idea is to define the processing duration of a WU over a progress measure (0-100% processed): The processing starts at $t_\tau^{proc\_start}$ with a progress of 0%. At each next point in time t, a progress increment is added to the overall progress, based on the available resources of j at this time unit. When the progress reaches 100%, the WU processing is finished and it can be determined how many time units were needed to complete the processing. Here, a discretisation with t is used, i.e. the available resources always remain constant for a least a time unit.

Note that in theory the problem of an indefinite processing duration can be encountered here when the available resources of the worker reach 0 and remain at this level. In the literature, this case is often eased by using checkpointing (cf. e.g. [201]). In the TDG, checkpointing is not used, however this case is countered by practical considerations: The owner of a work unit would not wait for an indefinite time but negotiate a deadline with the worker and decide to reassign the WU to another worker in case of a missed deadline (cf. [11] for details).

This processing duration is defined as a posteriori measure. To be usable as a worker selection criterion, this duration has to be predicted. It is assumed that the worker starts the processing of a WU τ at $t_\tau^{proc\_start}$ on its machine M_j as in the original measure. However, the available resources of the worker j are not known to the WU owner and hence have to be predicted for the duration of the processing. The resulting expected processing duration is then defined as:

$$\sum_{t=t_\tau^{proc\_start}}^{t_\tau^{proc\_start}+processingduration_{exp}(\tau,PL_j,t_\tau^{proc\_start},availableResources_{j,pred}(t))} \frac{PL_j \cdot availableResources_{j,pred}(t)}{costs_\tau} = 100\% \tag{5.5}$$

5.1.3 Open Desktop Grids - A challenging environment

The system classified and formalised in the previous sections is an instance of a technical, Open Distributed Systems, as users are connected to the system via the Internet, as they can enter and leave the system arbitrarily and central control cannot be applied (cf. e.g. [202]). Open Distributed Systems are comprised of an unknown number of autonomous entities that are in general heterogeneous with respect to goals, capabilities, preferences and behaviours (cf. e.g. [60]). 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. e.g. [61]). Finally, due to their distributed nature and autonomy, there is in general no form of direct or central control in such a system (cf. e.g. [58]). It is generally agreed that such a system, though beneficial in many ways, introduces a large amount of uncertainty among the participating entities (cf. e.g. [35]), especially because there are always participants involved that apply adversary strategies to exploit or damage these systems.

In the following, a discussion is conducted about why this kind of DG system should be realised as an Open **Multiagent** System and how the TDG utilises Multiagent technology to represent users.

In general, participants of DG Systems will seek a high degree of automation: The scheduling of WUs should transparently choose the best workers without manual control, while the decision to process WUs for other participants would usually be based on a policy set (and seldom changed) by the user (cf. e.g. the *BOINC* client, [197]). It is however not a trivial task to develop an online scheduler at design time that can cope with the requirements of such a dynamic environment (cf. e.g. [203]). Besides, the resource donation via fixed policies is rigid and requires the user to take regular manual control in order to provide an optimal participation as worker. In conclusion, it is worthwhile to delegate the control of a DG client to an adaptive agent that interacts in the dynamic environment and applies reasoning approaches to reach the goals set by the user (cf. e.g. [44]). Goals in this context are mainly to schedule the WUs of the user optimally while often simultaneously demanding not to dedicate the users' machine entirely for the processing of other users' WUs. The fitness of such an agent is hence defined by its ability to reach these goals. Obviously, this approach intensifies the self-interested nature of the DG participant: While there can be assumed from the user a general willingness to donate resources to the DG, an agent controlling the client could reason that the best strategy would be to (partially) free-ride and not accept any processing requests. This adds the additional challenge to control the system as such, when agents do not commit to a common goal but focus on the self-interested fulfilment of local goals.

Here, many researchers in the ODS community argue that the control of DG Systems has a lot in common with the control of Multiagent Systems, especially where Open Systems are concerned. It is stressed that essentially, the approaches in both fields provide solutions to shared problems from different perspectives: According to[9] [45] and [202], DG system research in this context aims at building systems with robust infrastructures and services, while MAS research focuses on autonomous problem solving entities, both for dynamic environments. It is then a general consent that these approaches can complement each other and that Desktop Grid Systems are an interesting field for the application of MAS technology. This becomes especially obvious when the challenging issues in Open Desktop Grid Systems, as in the taxonomy in [80], are considered:

- *Volatility*: Nodes in DG systems are non-dedicated, and resource provision is voluntary, thus the quality of service cannot be guaranteed.
- *Dynamicity*: Open DG Systems are complex systems that change during the runtime - host and resource availability, node behaviour, resource demand, workload and many other properties are subject to constant changes.
- *Lack of trust*: Anonymous nodes donate resources and adversary behaviour can always occur in this environment, decreasing the performance of other nodes. Nodes cannot be assumed as being trustworthy.
- *Failure*: Nodes in the DG are prone to network and hardware failures. This can decrease the performance of these systems.
- *Heterogeneity*: Nodes are controlled by many users with different behaviours, demands and a variety of computing resources.
- *Scalability*: Centralised approaches to scheduling in DGs are not scalable, therefore decentralised and sub-optimal approaches have to be applied to guarantee scalability.

[9]More examples for the application of MAS technology in the domain of DG systems are discussed in Sec. 2.5.2.

- *Voluntary participation*: DGs must provide incentives and benefits in order to convince volunteers to participate in these systems.

The key motivation for a client to participate in a such a system is that the client can take advantage of the resources of other clients by delegating the processing of own work units to these workers. A submitter can hence decrease the completion time for its jobs. However, the issues described above can have a negative influence on this. More precisely, the workers involved in an interaction in such a system can exhibit the following behaviours:

A worker can:
Refuse to process WUs for other clients.
Not return the results for processed/accepted WUs.
Cancel the processing of accepted WUs.
Return invalid WU results.

Table 5.1: Worker behaviours that pose threats to submitters in the Trusted Desktop.

A submitter must try and avoid interactions with workers exhibiting such behaviours in order to have a good performance. Here, the *autonomous problem solving* capabilities of agent technology can be applied: By observations of interaction partners, its environment, and its state, an agent deployed as submitter can realise strategies to avoid such threats. This is especially feasible, if the agent can reason about interaction partners based on the notion of *trust*. As discussed in Sec. 2.5.2, a commonly used solution is therefore to equip such a system with a Trust Management (TM) system: A submitter is at risk of being deceived by a worker to whom it delegates a work unit. The submitter is hence the *trustor*, the worker being the *trustee* (cf. e.g. [24]). A trustor that decides to interact with a trustee then evaluates the outcome of the completed interaction. In the TDG, the submitter can hence rate the worker based on the exhibited behaviour of the worker. This can be either the successful processing of the delegated WU, or either one of the adversary trustee behaviours listed in Tab. 5.1. By successively rating each other's behaviour according to clearly defined rules, the participating agents in such a system can discern cooperative from adversary behaviours and hence reduce their uncertainty about other agents in the system. Additionally, TM generates incentives for agents to adhere to cooperative behaviours.

The Trusted Desktop Grid has been designed as a system where the control over Desktop Grid Clients has been delegated to agents and a Trust Management system has been installed to guide these agents. The composition of these TDG agents is presented in the next section.

5.1.4 The Trusted Desktop Grid: Trust-Aware Agents

In Sec. 3.1.2, a generic agent model has been presented that allows Trusted Communities to be applied in a hosting system. This agent model has been instantiated for the application in the Trusted Desktop Grid. In the following, the term *agent* in the context of the TDG hence refers to an agent as defined by the following instantiation of the agent model depicted in Fig. 5.5.

The lowest layer, denoted with the term *production engine* in the generic agent model, is here the

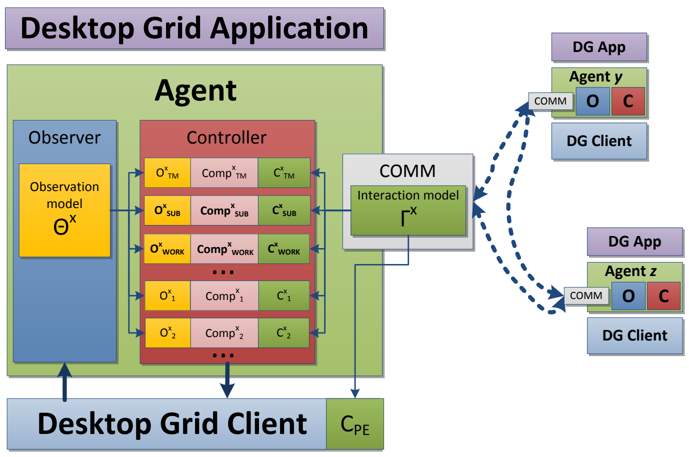


Figure 5.5: Refined agent model of a TDG agent including *Trust Management (TM)* component, *submitter (SUB)* component and *worker (WORK)* component. The latter components are responsible for the choice of delegation partners, and the complementary decision whether to accept a delegation request.

Desktop Grid Client. It enables the user to participate in the TDG by providing all protocols necessary for bootstrapping, the communication and transfer of work units between agents, etc. (provided as interactions in C^x_{PE}). This DG client is the workhorse, controlled by the core layer of this model, the agent layer. On top of the agent is the DG application layer. In the generic model this was denoted with the term *user*. In this layer, the users' applications that produce grid jobs are interfaced with the agent layer, i.e. the agent is informed about new jobs, passes the results for a completely processed job on to the respective application etc.

The focus in the design of the TDG has been the agent layer. The main responsibility of the agent is to make decisions regarding the distribution of own work units, as well as the acceptance of processing requests for other users' work units. As described for the generic model, the agent is based on the Observer/Controller architecture with the observer being responsible to collect private, as well as environment data that is needed by the controller. The controller is further divided into components, functionally encapsulated sub-controllers. The controller of a TDG agent $x \in \mathcal{A}(t)$ is composed of at least three components:

- The *Submitter (SUB)* component $Comp^x_{SUB}$ controls the submission of WUs to the DG and is active whenever user applications have unprocessed jobs. Delegation strategies are applied that allow for decisions such as whether to schedule the processing of a WU to a group of certain workers or to process it on the owners machine.

- The *Worker (WORK)* component $Comp^x_{WORK}$is the complement of the SUB component: It encapsulates all logic that is concerned with the decision whether to accept to process a WU when contacted by the SUB component of another agent or not. Strategies are applied here, making decisions such as whether to accept the processing request of an agent that is not seen as suited worker for own WUs by the SUB component etc. The main interactions provided by this component are the delegation of WUs for processing by the agent, as well as information requests (current work load, available resources etc.).

- The *Trust Management (TM)* component $\mathcal{C}omp^{x}_{TM}$ is a required component for the application of Trusted Communities. The TM component is used to manage the trust relations to other agents as specified in Sec. 3.1.3. In the TDG, the SUB as well as the WORK component produce interactions with other agents. These interactions are rated according to their outcome. In case of positive outcomes, the interaction partner receives a positive rating. In case of negative outcomes, as for example listed in Tab. 5.1, a negative rating is registered. Additionally, the other components have access to the aggregated form of these ratings, the subjective trust value.

Concluding the architectural view on a TDG agent, Fig. 5.6 depicts a successful round of the work unit lifecycle, as displayed in Fig. 5.4, as a traversal through the agent architecture.

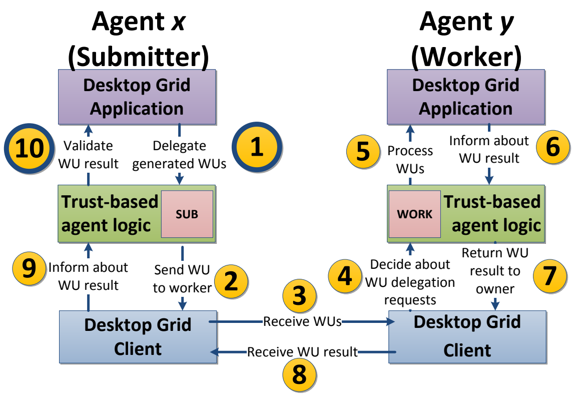


Figure 5.6: Traversal through the TDG agent architecture depicting the relationship between application, agent, and DG client layers with respect to the WU lifecycle.

In the system model presented in Sec. 3.1, the hosting system has been defined as a tuple representing the system. Accordingly, the TDG is defined as the following tuple:

$$\mathcal{H}_{TDG}(t) := \left\langle TDG\ Client, \left(\Lambda^{i}_{SUB}, \Lambda^{j}_{WORK}, \Lambda^{k}_{TM}\right), \mathcal{A}(t)\right\rangle$$

with *TDG Client* being a DG client software and $\left(\Lambda^{i}_{SUB}, \Lambda^{j}_{WORK}, \Lambda^{k}_{TM}\right)$ representing the system components required for TDG participation, namely submitter, worker and Trust Management component. In the following, the internals of the default implementations $\left(\Lambda^{default}_{SUB}, \Lambda^{default}_{WORK}, \Lambda^{default}_{TM}\right)$ of these TDG agent components are described in detail.

The TDG trust model:

As presented in the generic agent model, the TM component encodes a trust model. The following definitions describe the trust model incorporated in the TM component $\mathcal{C}omp^{x}_{TM}$ of a TDG agent $x \in \mathcal{A}(t)$. To define such a trust model, aggregation functions for the various trust values are provided.

For the aggregation of a set of interaction outcome ratings $\mathcal{R}^{y,\hat{c}}_{x}$ from an agent x with an agent y, to a direct trust value $DT^{y,\hat{c}}_{x}$, the following approach is applied: Only recent ratings, hence ratings no older than t_{max}, are considered. This allows for a faster detection of changes in the agents' behaviour.

Otherwise, the ratings are aggregated by the arithmetic mean formalised in the function agg_{DT}:

$$DT_x^{y,\hat{c}} = agg_{DT}(\mathcal{R}_x^{y,\hat{c}}) := \frac{1}{\left|\mathcal{R}_x^{y,\hat{c}}\right|} \cdot \sum_{r_i \in \mathcal{R}_x^{y,\hat{c}}} r_i \tag{5.6}$$

The next aggregation function to define is agg_{RT}, used to aggregate a set of indirect trust values into a single reputation value $RT_{\mathcal{P}}^{y,\hat{c}}$. In the TDG, the aggregation is done via an arithmetic mean (thus not preferring any opinion provider over any other). The choice of suited opinion providers $P \supseteq \mathcal{A}(t)$ is left to the implementation of the respective submitter and worker components, with the most basic approach being to request opinions from the entire agent society. The aggregation function is formally defined as:

$$RT_{\mathcal{P}}^{y,\hat{c}} = agg_{RT}(DT_{p_1}^{y,\hat{c}}, .., DT_{p_k}^{y,\hat{c}}) := \frac{1}{|\mathcal{P}|} \cdot \sum_{p_i \in \mathcal{P}} DT_{p_i}^{y,\hat{c}} \tag{5.7}$$

In accordance with most authors in the literature (cf. [24]), the aggregation function of the trust value $T_x^{y,\hat{c}}$ of y by x utilised in the TDG is defined as weighted average between the direct trust value $DT_x^{y,\hat{c}}$ of x and the reputation $RT_{\mathcal{P}}^{y,\hat{c}}$ of y, such that:

$$T_x^{y,\hat{c}} = agg_T(DT_x^{y,\hat{c}}, RT_{\mathcal{P}}^{y,\hat{c}}) := \gamma \cdot DT_x^{y,\hat{c}} + (1-\gamma) \cdot RT_{\mathcal{P}}^{y,\hat{c}} \tag{5.8}$$

In the model applied in the TDG, the weight γ of the direct trust value is dependent on the number of ratings in $\mathcal{R}_x^{y,\hat{c}}$ that make up this value $DT_x^{y,\hat{c}}$. Such an approach is referred to as *confidence*- or *certainty*-based in the literature (cf. e.g. [188]). The following linear function represents the approach here:

$$\gamma := 1 - \left|\mathcal{R}_x^{y,\hat{c}}\right| \cdot \frac{1-\gamma_{min}}{thres_{\mathcal{R}_x^{y,\hat{c}}}}$$

with:

$$\gamma_{min} \in [0,1] \text{ and } thres_{\mathcal{R}_x^{y,\hat{c}}} > 0$$

Here, γ_{min} is the minimal value γ is allowed to have. This represents the approach to always include opinions from other agents despite having many direct experiences, helping to identify changes in y's behaviour faster. Additionally, the threshold $thres_{\mathcal{R}_x^{y,\hat{c}}}$ specifies the number of required ratings in $DT_x^{y,\hat{c}}$ to be fully confident about y's behaviour and thus weigh y's reputation only with γ_{min}. Note here that this threshold also relies on the past rating interval t_{max} used for the composition of $DT_x^{y,\hat{c}}$, it represents a sliding window approach to the counting horizon.

The evaluations described in this thesis use the following parameterisations of the trust model:

$t_{max} := 1000$	Lookback interval for ratings [tick]
$w^+ := \frac{1}{3}$	weight of positive ratings
$w^- := \frac{2}{3}$	weight of negative ratings
$\gamma_{min} := \frac{1}{10}$	min weight of reputation
$thres_{\mathcal{R}_x^{y,\hat{c}}} := 50$	rating confidence threshold

The definition of the agent model in Sec. 3.1.3 has described the requirement to define the contexts in which the model is applied. These are formulated based on the agent components involved in interactions between agents, such that the trust model always refers to the trustworthiness of a particular agent component. To conclude the definition of the TDG Trust Management system, it is necessary to specify the interaction model Γ^x of a typical TDG agent x, along with possible outcomes and according ratings for each included interaction c_i.

As described above, the most important interaction in such a Desktop Grid System is the delegation of a work unit from a submitter $x \in \mathcal{A}(t)$ to a worker $y \in \mathcal{A}(t)$. In order to allow for this, the Worker component $Comp^x_{WORK}$ of an agent provides the interaction $c_{del}(\tau^x_i)$. From the view of the submitter, this can either result in a positive outcome (a valid WU result from a successful worker processing) or a negative outcome due to any of the worker behaviour deviations listed in Tab. 5.1. In the TDG, single ratings composing the set of ratings $\mathcal{R}^{y,WORK}_x$ are hence drawn from the following specification:

	Outcomes and ratings for interaction $c_{del}(\tau^x_i)$:	
$o^1_{c_{del}(\tau^x_i)}$	Return valid result for τ	$r^1_{del} := 1.0$
$o^2_{c_{del}(\tau^x_i)}$	Refuse to process τ	$r^2_{del} := -0.05$
$o^3_{c_{del}(\tau^x_i)}$	Accept, but cancel processing of τ (inform x)	$r^3_{del} := dur(\tau, y)$
$o^4_{c_{del}(\tau^x_i)}$	Accept, but do not return result for τ (do not inform x)	$r^4_{del} := -1.0$
$o^5_{c_{del}(\tau^x_i)}$	Accept, but return invalid result for τ	$r^5_{del} := -1.0$

Table 5.2: Outcomes of worker behaviour and according rating values for the delegation interaction.

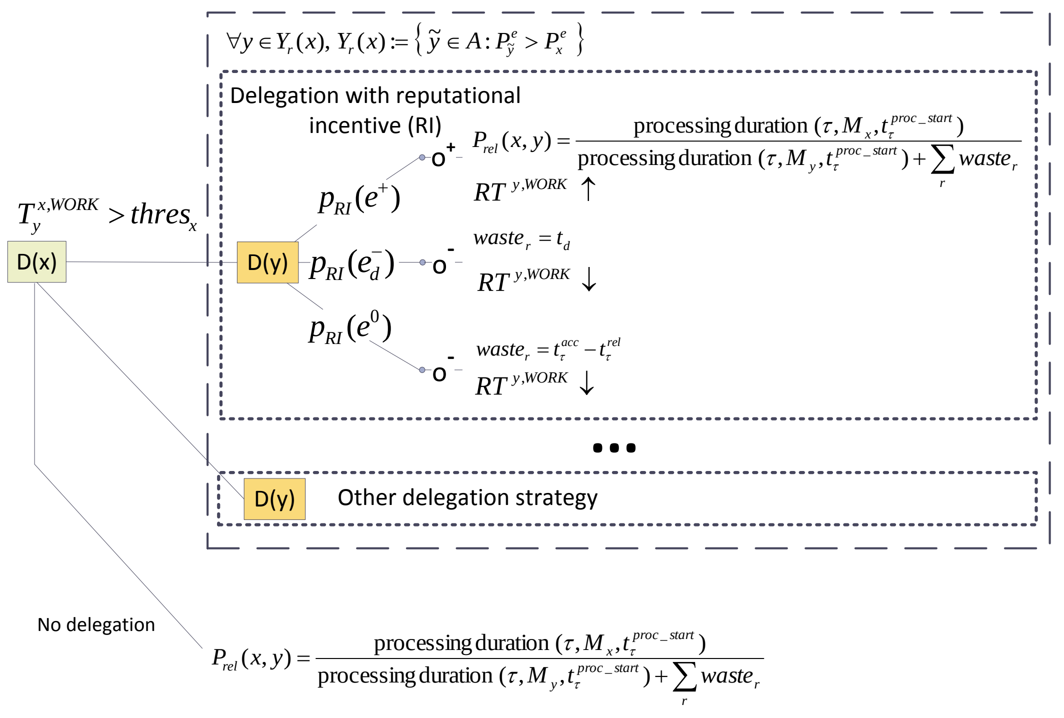
The rating values are defined and ordered according to the impact on the performance the submitter x receives from the delegation of τ to the worker y. This is also why the rating r^3_{del} is specified as function $dur(\tau, y)$: The negative impact on the submitter performance is greater the later the worker cancels the work unit, as more time is wasted, although the act of informing the submitter about the cancellation is a desired action.

Now that the composition of a TDG agent, as well as the underlying Trust Management System have been defined, the control of the DG client via trust-based decision making within the Submitter and Worker is detailed.

5.1.5 Agent Interactions in the Trusted Desktop Grid

In the following, the decision making of the default SUB and WORK components in the TDG are presented in detail and the required situation description, composed of data to be observed and generated by the agent Observer, are specified.

Consider an agent $x \in \mathcal{A}(t)$ that has recently generated a grid job ϕ^x_i. The job is passed to the SUB component of the agent, and x is hence referred to as acting in the role of a *submitter*. To complete the job, the SUB component has to decide for each WU τ contained in ϕ^x_i where to process it, which is mainly dependent on the trust relationships of the submitter. After the current work unit τ of a job has been either scheduled for owner processing or successfully delegated to another worker agent $y \in \mathcal{A}(t)$, information required for the decision making is updated. Then the delegation decision process is repeated for the next WU contained in ϕ_i^x, until all WUs from this job have been scheduled[10]. Additionally, this process is triggered whenever the processing of τ has been interrupted. In this case, this process has to be executed again, albeit with updated information. The iterative process to schedule a single WU τ for processing is formalised in the following *submitter decision tree*, depicted in Fig. 5.7. This methodology is based on the generic decision tree for trustor agents as presented in [29][11].



$\forall y \in Y_r(x),\ Y_r(x) := \{ \tilde{y} \in A : P^e_{\tilde{y}} > P^e_x \}$
Delegation with reputational incentive (RI)
$P_{rel}(x,y) = \frac{\text{processing duration }(\tau, M_x, t_\tau^{proc\_start})}{\text{processing duration }(\tau, M_y, t_\tau^{proc\_start}) + \sum_r waste_r}$
$RT^{y,WORK} \uparrow$
$p_{RI}(e^+)$
$T_y^{x,WORK} > thres_x$
D(x)
D(y)
$p_{RI}(e_d^-)$
$waste_r = t_d$
$RT^{y,WORK} \downarrow$
$p_{RI}(e^0)$
$waste_r = t_\tau^{acc} - t_\tau^{rel}$
$RT^{y,WORK} \downarrow$
D(y)
Other delegation strategy
No delegation
$P_{rel}(x,y) = \frac{\text{processing duration }(\tau, M_x, t_\tau^{proc\_start})}{\text{processing duration }(\tau, M_y, t_\tau^{proc\_start}) + \sum_r waste_r}$

Figure 5.7: Decision tree for submitter agents in the TDG. The submitter decision whether, and to whom, the delegation of a WU should be executed is denoted with $D(x)$, and the according worker decision is denoted with $D(y)$. Delegations are carried out according to delegation strategies.

This decision tree is built as follows: **The submitter x makes a decision $D(x)$ whether to process the work unit τ on its own or to delegate it with one of several delegation strategies**. This decision requires information about the available potential workers. As the agents in the system are autonomous, this information must be explicitly requested from these workers, and these requests can be rejected. When a suited worker y has been chosen for the processing of τ, the submitter sends a processing request to this worker. In turn, the worker makes a decision $D(y)$ whether to process τ or not. This is based on reasoning whether the submitter is trustworthy and whether the fulfilment of the request brings strategic advantages for the submission of own work units. The result

[10] In the TDG, agents are allowed to schedule one WU per time unit to provide fairness among the agents regarding the availability of workers with a high expected performance.
[11] In this thesis, this work is extended by applying it to the DG domain, as well as proposing an additional delegation strategy based on Trusted Communities.

of this decision is connected with an effort that is necessary for the outcome. This is marked with probabilities, as the decision tree is interpreted from the view of the submitter. At the right hand side of each effort-outcome pair, the consequence of this decision chain is shown, both for the submitter x as well as the worker y. The decision tree formalises the following process:

The choice of suited workers: As previously discussed, the aim of an agent x is to process all work units τ of a job ϕ_i^x as fast as possible. This means to minimise the function $completiontime_{\phi_i^x}$, the completion time of the job. In a Desktop Grid, the agent usually does not need to process each work unit τ contained in ϕ by itself, but can take advantage of resources donated by other agents. This allows to process the WUs of a job in parallel which reduces the completion time of ϕ_i^x. However, the agents in the DG system control machines with heterogeneous computational power. Therefore, the **submitter must first determine which workers are able to process τ faster than it could do by itself, given its current workload**. The resulting processing duration of such a delegation, as defined in Eq. 5.5 and discussed in Sec. 5.1.2, is the major contribution of the completion duration of a work unit. This actual effect of a successful delegation of τ to a worker y is depicted on the right hand side of the decision tree in Fig. 5.7. To also quantify the benefit of the grid participation for the owner x of τ (this is detailed in Sec. D), this duration is compared with the hypothetical processing duration of x. This performance definition is formalised in the following *relative processing duration* function:

$$\begin{aligned} P_{rel}(x,y) &= \frac{\text{owner processing duration}}{\text{delegation processing duration}} \\ &= \frac{processingduration(\tau, M_x, t_\tau^{proc\_start})}{processingduration(\tau, M_y, t_\tau^{proc\_start}) + \sum\limits_r waste_r} \end{aligned} \tag{5.9}$$

In the term for a worker y, the term $waste_r$ is included. This is the duration that τ spent in round r (see WU life-cycle in Sec. 5.1.2) without an obtained valid result[12]. In the TDG, the positive outcome o^+ of the delegation is a state, where a valid result for τ exists. This state can only be reached if y processes τ completely. The corresponding effort is denoted as e^+. Each other effort level produces a negative outcome o^-, meaning no or no valid result for τ[13]. Two cases are discerned: τ has been straight rejected for processing by a worker y, denoted as effort e^0. The waste in this case equals the *scheduling duration*, hence $t_\tau^{acc} - t_\tau^{rel}$. The second case is that τ has been processed only to a certain degree d, which is denoted with the effort e_d^-. Here, also no valid result was generated, but the amount of waste depends on the time t_d spent with this effort. The worst case scenario is a complete processing of τ that produces an invalid result, the waste for this case equals the duration $t_d = t_\tau^{res\_val} - t_\tau^{rel}$. In the following, the focus is on the resulting performance of wasteful delegations, not why negative outcomes and thus waste are experienced. This is discussed in the subsequent paragraph.

The work units of the submitter y are delegated iteratively. When a delegation of τ, to the worker y, in round r, has lead to a negative outcome o^-, it generates the waste $waste_r$. The work unit is not completed and hence needs to be re-scheduled. This is done in a subsequent round $r+1$, as defined in Sec. 5.1.2. For this, the submitter chooses the next best worker $z \in Y_r(x)$ ($Y_r(x)$ is ordered ac-

[12]This is true for applications that do not allow checkpointing.
[13]Note here that this is an aggregation of the possible outcomes presented in Tab. 5.2, for the sake of readability.

cording to performance) as a delegation partner or processes τ itself if there is no agent left in $Y_r(x)$ that has not been asked. This is done until the outcome o^+ is finally achieved. However, waste from preceding rounds is summed up and adds to the actual processing duration that can be achieved by delegation of τ to other workers from $Y_r(x)$. The submitter will therefore check at each round for τ if it can reach a satisfying relative processing duration at all[14]. If not, the submitter will finally process the work unit on its own, with the assumption that x always produces o^+ when processing an own τ, thus always terminating. This relative processing duration is depicted on the right hand side of the decision tree in Fig. 5.7 for each of the scheduling options for τ.

Congruously, **a worker y should be included in the set of suited workers $Y_r(x)$ for a round r only if the delegation of τ to this worker would lead to a processing duration that is shorter than the owner's processing duration in this round**. Until here, the implicit assumption has been made that this delegation performance is known to the submitter. However, this measure can only be made a posteriori which means that the submitter does not know the exact performance of a worker before it has actually delegated τ to it. In order to build the set $Y_r(x)$, the submitter hence needs to *estimate* the actual performance of each known and available worker y. This expected performance of y is denoted as P_y^e and refers to the expected processing duration as defined in Eq. 5.5. The function requires the following input:

- The work unit τ. The costs of the work unit are required to estimate the processing duration. This information is available to the submitter as it is the owner of the WU.
- The performance level PL_y of the worker. The performance level is a static value (see Sec. 5.1.2) and has to be retrieved only once. For this purpose, it is assumed that the agents in the society are aware of the performance level of their fellow agents.
- The time $t_\tau^{proc\_start}$ at which the worker will start the processing of τ when delegated to it. This time depends mainly on the workload of the worker and is private information. The submitter has to request this information of each worker y in order to calculate the estimate. If a worker does not answer this request, it is excluded from the set of potential workers.
- A prediction function $availableResources_{j,pred}(t)$ of the available resources of y for the time interval of the processing. The exact course of the available resources is information that is not available to the submitter nor the worker, because it is an external value controlled by the user of the agent y. The submitter can here either request a prediction by the worker (worker monitors its available resources) or generate it based on past interactions and the currently available resources of the worker (private information requested from the worker).

Information needed to be retrieved in this process is hence specified as observables of the submitter component. On the other hand, the worker component must allow for the interaction of such an information request. When the submitter has requested and processed all necessary information about the available workers, it can build the set $Y_r(x)$ for a round r, containing all workers y that are expected to be suited for a delegation of τ. This is based on their expected performance P_y^e in comparison with the expected submitter performance P_x^e. The set $Y_r(x)$ is then ordered according to

[14]This can be also realised by including x in the set $Y_r(x)$. As $Y_r(x)$ is sorted according to the expected performance, x will finally be the top agent in the list, and hence process τ by itself.

the highest expected performance, such that the best available workers are requested a processing of τ first. The set $Y_r(x)$ for a round r is then formally composed as follows:

$$Y_r(x) := \left\{ \widetilde{y} \in \mathcal{A}(t) : P^e_{\widetilde{y}} > P^e_x \right\} \tag{5.10}$$

Note here that $Y_r(x) \subseteq \mathcal{A}(t)$, hence only workers y that are in the agent society at time t (online) can be in this set.

The submitter decision: As discussed, the requirement for the inclusion of a worker y in this set is the *competence* as a delegation partner for x regarding τ. The submitter can hence estimate what to expect from y performance-wise. However, the submitter does not know if the worker is actually inclined to provide the effort e^+.

The worker can also invest less effort: Consider for example an adversary agent that wants to invoke damage on the submitter, leading to an unsuccessful delegation and degrading the performance for the submitter. The probability $p(e^+)$ that y is willing to invest the effort e^+ is then referred to as the workers' *willingness* to cooperate. Just like the estimate of a worker performance, **the submitter needs to estimate this willingness of a worker, in order to prevent a performance degradation**.

Additionally, from the view of a system designer, the willingness of agents to cooperate cannot be enforced, due to their autonomy and open realisation. It is thus a design aim to provide delegation strategies that set **incentives to raise the willingness of the agents to cooperate**. In the following, a trust-based delegation strategy for the TDG is presented. This represents a desired behavioural pattern and reference implementation for the TDG submitter and worker component as defined in Sec. 5.1.4. This delegation strategy is contained in a default implementation of the agent software for the TDG and hence can always be chosen in the submitter decision $D(x)$. The assumption here is that this is the prevalent behaviour. However, the TDG is not limited to submitters and workers with this delegation strategy, as discussed in appendix C, where a threat model is presented, mainly referring to worker behaviours that deviate from this desired behaviour.

In the following, the major TDG delegation strategy, the *reputation incentive* strategy, is described. In the TDG, Trust Management is used to derive the willingness of workers based on their past interaction performances, as well as to provide an incentive for cooperation. As described in Sec. 3.1.3, this is applied via the TM component of a TDG agent. This component encapsulates the generic Trust Management system required by the agent model for the application of TCs. Interactions between agents (here submitters x and workers y) are rated according to the specified trust model. These ratings have an influence on the personal trustworthiness estimates of x regarding y (denoted as $DT_x^{y,WORK}$), as well as on y's reputation (denoted as $RT_{\mathcal{P}}^{y,WORK}$) if x is requested to provide an opinion about y's performance as a worker (hence $x \in \mathcal{P}$). Reputation gain and loss due to WU processing outcomes are depicted in the decision tree (see Fig. 5.7) with $RT^{y,WORK} \uparrow$, and $RT^{y,WORK} \downarrow$ respectively.

Trust is applied as follows by the agents in the TDG: The aggregated trust values $T_x^{y,WORK}$ of a submitter x towards all competent workers y in the set $Y_r(x)$ are used to filter the set: Only those workers that have proven[15] their willingness to cooperate with x by positive interaction outcomes are

[15]Note that this is a simplification for the sake of explanation. If applied as described, this approach would else lead to a problem with the newcomers in the system, also referred to as *Initial Trust Problem* in the literature.

considered as delegation partners. This filter is realised by a subjective trustworthiness threshold $thres_x$, such that workers y are removed from $Y_r(x)$ where $T_x^{y,WORK} \leq thres_x$ holds. After applying the filter, the set $Y_r(x)$ contains all workers that are expected to be *competent*, as well as *willing*. Further interactions consolidate their qualifications, as the outcomes are tracked and update the trustworthiness value. Also, the threshold $thres_x$ is adapted by the agents, based on observations about their environment. If for example a submitter realises that no worker remains in $Y_r(x)$ after it has applied the filter, the threshold can be lowered. Here, machine learning techniques can be applied to determine situation-optimal value pairs for this threshold, based on observations defined in the model of observation by the submitter component (see. Sec. 3.1.2). However, this is not in the focus of this thesis and is thus neglected in the following.

The worker decision: Agents that successively produce positive outcomes as workers are perceived as trustworthy and increase their reputation over time. Given their competence, these agents are then frequently requested to process further work units to the benefit of the submitting agents. But how does this high effort pay off for the workers? Or to put it the other way round: How can agents be persuaded to regularly invest this effort and cooperate with other agents, such that the probability for a successful outcome $P_{RI}(e^+)$ is higher than the probabilities for the rejected cooperation $P_{RI}(e^0)$ or at least higher than the probability for the failed interaction $P_{RI}(e_d^-)$. In the following, the realisation of the default worker decision $D(y)$ in the TDG is presented. This is a binary **decision to either accept or reject a processing request by a submitter x for a work unit τ**. Again, this is only the preferred reasoning strategy, in the evaluations agents with diverse strategies, e.g. freerider agents, are examined.

In order to persuade agents to completely process WUs for other agents a **reciprocity-based incentive mechanism** is applied: A worker y rejects a processing request of a submitter x, if the submitter has shown only a low willingness to invest an effort as worker towards y. This relies on past experiences of y as submitter with x being requested to process work units of y. These interactions built up the trust value $T_y^{x,WORK}$ which is applied as follows: If $T_y^{x,WORK} \leq thres_y$, the processing request is rejected. This trustworthiness threshold $thres_y$ for the worker decision $D(y)$ is usually set to the value of the threshold $thres_x$ for the submitter decision $D(x)$. This encodes the following additional considerations of a worker: Only if the submitter agent x would be accepted as a (willing) worker for a WU delegation of the agent y, i.e. only if y expects to rely on x for future own interactions, is the processing request accepted. This supports the self-interested nature of the agents in the grid. Additional considerations in the TDG worker component relate to the limitation of the amount of accepted work units (work load): Here, the worker strategies need to account for the fact that a high work load reduces the submitter performance of an agent as it cannot process own work units without costs. These costs result from either waiting for the processing of foreign WUs before starting to process an own WU (increase of completion time), or from cancelling the processing of foreign WUs to prefer own WUs, an operation that results in trustworthiness decreases and hence reduces the agents' chance of successfully submitting further WUs.

The reciprocity incentive mechanism can be summarised as follows: **Agents that invest high efforts as workers for others have a higher probability to successfully delegate their own WUs to competent and willing workers**. On the other hand, agents that invest low efforts struggle to find workers that accept their processing requests. This is because the worker performance is registered by other agents directly, as well as indirectly through the reputation mechanism. Consequently, a low reputation results in the *isolation* of low performing agents, which refers to the fact that they are forced to process own work units by themselves, despite their participation in the DG system. Hence, there is an incentive for TDG agents to invest high efforts.

Work unit validation: The process has been described for the case of validatable work unit results, thus with the assumption that the submitter is able to discern the outcomes o^- and o^+. This assumption often does not hold in Desktop Grid Systems. Rather do according **non-validating applications** produce work units τ with results that cannot be programmatically validated. A submitter that has delegated the processing of a WU, must then decide whether to blindly accept the result returned by the worker, or **apply safety measures**: A submitter can **replicate the work unit, delegate the processing of these replicas to a number of different workers, and finally compare the results** from the different workers. The usual approach here is to apply a *majority voting* validation approach (cf. e.g. [204], [205]), hence to test for result consensus. The number of replica results from different workers required to judge on the accuracy of the results, is referred to as *quorum* (cf. e.g. [206]) and must be at least 3 in order to allow for majority statements. Workers that produce a result that diverts from the result obtained by the majority are then assumed to have defected, while those producing the seemingly right result are assumed to have cooperated. This type of validation is obviously not fail-safe, as colluding workers that aim at reaching a majority may fool a submitter into thinking to have received a valid result where this is not the case (cf. e.g. [85], [89]).

The delegation process described in this section is executable for such non-validating DG applications as well. The only variations are that the number of work units to distribute through the submission process is multiplied (dependent on a set *quorum*), the set $Y_r(x)$ is composed such that a single worker is never requested to process multiple replicas of τ (this would bias the majority voting), and that the evaluation of the outcome for a single WU processing is deferred until enough results are returned to allow for the validation.

Summary: In summary, this section has described how the Trusted Desktop Grid is realised via the system model defined in Sec. 3. The production engine incorporates the Desktop Grid Client and the user layer is represented with applications that produce DG jobs. The agent, being the most important layer, is at least composed of the Submitter, Worker and Trust Management components. The submitter and worker components control the Desktop Grid Client by applying autonomous decision making. The TM component supports this by monitoring the behaviours of agents and applying the trust model defined here to derive estimations of their willingness. The interactions between the TDG components have been detailed in this section, focussing on the delegation of work units from a submitter to a worker. In that, the reciprocity-based incentive mechanism has been described.

So far, the challenges of an open system design for such a system have been mapped to the threats that a WU delegation to a worker involves. This is a sound approach, as successful WU delegations are the main motivation for a participation in the TDG. However, these types of worker behaviour are not the only threats that influence the benefit of the participation in such a system. To allow for a more comprehensive analysis of the TDG characteristics, especially in consideration of the application of Trusted Communities, see the presentation of a threat model in appendix C. In the following, it is examined how the described agent interactions, and the threats introduced by the openness of the system, affect the *reputation incentive* approach in the TDG.

5.1.6 Discussion

The application of a Trust Management system, as described in Sec. 5.1.4, to cope with the challenging issues in the TDG has been evaluated[16] and the results published in e.g. [4], [5], [8], and [11]. These evaluations have mainly demonstrated that the TDG *reputation incentive* strategy (introduced in Sec. 5.1.5) allows for the *isolation* of agents that express some of the forms of class 1 and 2 behaviours listed in Tab. C.3. *Isolation* here means that these agents found no workers for the delegation of own work units, and thus had to process all WUs on their own. This in turn imposes a long processing duration (due to sequentiality), low utility values, and consequently, no benefit of TDG participation. The avoidance of such an *isolation* is hence an incentive to cooperate for agents in the TDG. Cooperative agents, on the other hand, were shown to profit from their good conduct, by developing strong trust relationships with each other that opened for them many opportunities to delegate own work units to other workers, due to the *reputation incentive* delegation strategy. Consequently, it was demonstrated that these agents had lower processing durations, a higher utility and hence a benefit from TDG participation. From the organisation point of view, the relationships of these cooperative agents created a loose coupling without explicit membership notion: Each of the agents had a subjective view on various trustworthy interaction partners, and the agents did not differentiate among interaction partners apart from the division into trustworthy and not trustworthy agents based on a subjective threshold. For the discrimination (and later result comparison) with the TC approach, a TDG with this *reputation incentive* strategy approach alone, without the application of any type of explicit MAS organisation, is referred to as the **implicit Trusted Community (iTC)** approach. This term accounts for the strong trust relationships among cooperative agents that are established due to this approach (and resemble the composition of a Trusted Community), and the fact that no form of organisation is communicated, negotiated, or maintained among the agents in such an *iTC* (making it implicit). Before the application of Trusted Communities in the TDG is motivated, the *iTC* approach is discussed with respect to its shortcomings.

In Sec. 3.2.1 of this thesis, generic challenging issues in the class of the *hosting system* have been examined. The TDG is an instance of this class, and the scenario-specific impact of these generic issues is hence encountered in the TDG:

TM exploitation As described in Sec. 5.1.5, the *reputation incentive* strategy involves the application of a submitters' trust threshold $thres_x$ within the worker decision, such that agents are required to build up reputation as workers in order to be able to delegate own WUs as submitters. Given that agents know this threshold, this incentive mechanism can easily be exploited: Agents must ensure that their reputation is higher than that threshold, but beyond it, they have no incentive to cooperate. This leads to oscillating commitment of agents (and hence reputation), in its effect similar to the *reputation damage problem* (cf. [24]), but due to strategic considerations instead of emergence. Such an incentive malfunction is not to overcome by the increase of the threshold, as too high a threshold will

[16]Work on these results has been conducted in cooperation with Yvonne Bernard in the context of the DFG research unit *OC-Trust (FOR 1085)*.

hinder initial cooperation and intensify the *newcomer problem* (cf. e.g. [76], [61]) in the system. In sum, **the *reputation incentive* can be exploited by strategic agents, reducing their commitment and decreasing the performance of the *hosting system***.

Over-confidence The trustworthiness of workers in the TDG is rated by the submitter according to the received outcome (see Tab. 5.2). The Trust Management then aggregates this set $\mathcal{R}_x^{y,WORK}$ of experience ratings to a single trust value $DT_x^{y,WORK}$. Consider a worker that cooperates by returning only valid WU results over a longer period of time, and then suddenly starts to defect by accepting a WU without processing it. This illustrates the problem of *over-confidence* (cf. e.g. [20]): As long as the submitter does not detect the defect, the worker maintains a high trust value and is seen as trusted worker. As soon as the submitter becomes aware of the actual behaviour, it rates the worker with a negative rating r_{del}^4. This single rating will however not have a great influence on the workers' reputation, due to its extended period of cooperation. Even successive negative ratings will reduce the trust value only bit-by-bit. In the meantime, many submitters estimate the worker as trustworthy and delegate WUs to it. Only after finally reaching the threshold *willingness* $thres_y$, will the trust value denote the current behaviour of the worker. Although the extent of the *over-confidence* built up is dependent on the design of the *rating aggregation function* $agg_{DT}(\mathcal{R}_x^{y,WORK})$, the problem itself is universal for sudden changes of behaviour. In sum, the application of worker trustworthiness estimations for **the submitter delegation decision is susceptible to the development of over-confidence**, because the TM is slow to react. In summary, sudden changes of worker behaviour to defection are **perceived by submitters only in a deferred way, decreasing the received performance and hence the performance of the TDG**.

TM reliance In the TDG, *worker D(y)* and *submitter D(x) decisions* are based on the evaluation of the trustworthiness of the counterpart: A submitter won't delegate a WU to an untrusted worker with a low trust value to avoid the risk of increasing its processing duration. On the other hand, a worker won't accept a processing request from an untrusted submitter to maintain the incentive mechanism and avoid the blockade of its resources for an agent that it does not want to delegate a WU to (see Sec. 5.1.5). Consider the event of a *global trust breakdown* $\mathcal{B}_{\mathcal{A}(t)}^{\hat{c}}(m)$ as defined in Eq. 3.7: Only $(1-m)\cdot|\mathcal{A}(t)|$ agents have a positive reputation value, and in general even less are expected to have trust values above the thresholds $thres_x$ and $thres_y$. Hence, only a small minority of agents will be considered as delegation partners. Additionally, these agents will firstly accept processing requests only for agents with equal trustworthiness, and secondly, will be overloaded due to being the only cooperative agents in the system. Such a global trust breakdown cannot be neglected: The interactions of submitters and agents in the TDG are dependent on the heterogeneous and dynamic *competence* and *willingness* of the agents, the presence of adversary agents, the changes of agent behaviours due to autonomy and self-interest etc. In sum, the rating activity of submitters and workers in the TDG is highly complex, and trust breakdowns must be reckoned with in the agent society. Consider for example the case of a prolonged overload situation, where all agents reject to process WUs (or are not estimated competent enough due to high workload, see Sec. 5.1.5) and lose their good reputation as a consequence. Even if the overload situation is ended then the damage is done, and the agents have lost their trust in the *willingness* of each other, and will hence not delegate the WU processing but process WUs themselves. In the literature, such a MAS state, i.e. a state in which agents do not execute any interactions with each other due to their risk assessment, is referred to as *paralysed* (cf. e.g. [29]). In sum, **the TDG *reputation incentive* relies on the undisturbed operation of the TM**, such that there is no trust breakdown. Yet, on the other hand, **the emergence of a trust breakdown cannot be precluded** - quite on the contrary, it can be shown to develop in certain system states (see robustness evaluation in Sec. 5.3.3). **In result, the TDG is not robust towards abnormal system states as it can become *paralysed***.

Sub-optimality due to safety means The TDG delegation strategy has been described as applicable for *validating*, as well as **non-validating** applications (see Sec. 5.1.5). In the latter case, a submitter cannot directly validate the result of a WU processing. In order to reduce the probability of accepting an invalid result as valid, the application of safety means has been discussed. In the TDG, these safety means are based on the state of the art concept of "WU replication" and following "majority voting" validation on a *quorum* of WU results. The application of WU replication is a means to counter the presence of *malicious volunteers* (as stated in e.g. [175]) returning invalid results on purpose, or workers returning invalid results due to hardware errors. **These safety means introduce redundancy, which materialises as additional workload of system participants and necessarily leads to a slowdown of the processing speed** if there are more WUs than available workers (cf. e.g. [175]). The application of such safety means is hence a risk reduction mechanism with performance costs, due to overhead, assigned to it. The TDG already uses a Trust Management mechanism for the evaluation of agent behaviour. This allows for a modification of the submitter decision, such that the required number of WU replicas is derived from the trustworthiness of the worker, an approach also described in the literature (cf. e.g. [80], [175]). While this can reduce the costs of redundancy, it introduces an additional reliance on the operation of the TM, and a susceptibility to over-confidence (as described above). In sum, the TDG uses a static WU replication scheme that increases the workload in the system and hence reduces the performance. While alternative, **adaptive approaches exist, these are still susceptible to the issues described above**.

Sub-optimality due to undetectable submitter behaviour The description of the TDG agent interactions has focused on the probability of uncooperative *worker* behaviours. This is a sound approach, as such behaviours mainly determine the performance of agents in the TDG and are to be avoided. Besides, these behaviours are detectable by single agents and reflected due to the validation of the interaction outcome by a submitter (either directly or through *majority voting*, as described above). However, apart from such worker behaviour, also **undesired submitter behaviour exists in the TDG. These behaviours are much harder to detect, as workers are passive interaction partners and cannot directly determine the consequence of these interactions. Nor do incentives for the enforcement of worker cooperation exist in the TDG**. Such incentives could motivate workers to collaborate in order to detect undesired submitter behaviours. In the following, this problem is detailed with the help of an additional submitter delegation strategy:

The submitter decision $D(x)$ for the delegation of τ in round r has been so far described as the selection between two options, as depicted in Fig. 5.7. This decision means either to delegate τ to a worker $y \in Y_r(x)$ that is believed to be competent and willing to successfully process τ, or not to delegate τ but process it by itself. The latter is chosen when there is no such worker left that has not already rejected the processing of τ. As discussed, the openness of the TDG allows for several other delegation strategies. Here, another delegation strategy, the *delegation with reputational incentive and WU replication* is discussed. This strategy is particularly relevant, because on the one hand it does not include adversary submitter behaviour (as opposed to the behaviours listed in Tab. C.3), but on the other hand is undesired nonetheless, because it leads to overhead and can deteriorate the performance for other submitters (similar to the WU replication overhead described above).

This delegation strategy with *WU replication* also applies the reputation incentive, however, submitters try to decrease the probability for a high WU completion duration to the disadvantage of the system: Instead of waiting until the result for τ is produced by a worker y, the submitter x generates copies τ_i of τ and tries to delegate these to other workers, in order to minimise the costs of a wrong processing duration estimate for the worker y (cf. e.g. [207], [208]). As soon as a valid result is returned by any of the workers processing τ or its copies, the processing of the remaining replicas is not required any more. This is an appropriate strategy from the cost perspective of a submitter, as replication generates hardly any additional overhead. However the workload in the system is raised by the replication factor until o^+ is reached: This not only reduces the probabilities $p_{RI,rep}(e^+)$ and $p_{RI}(e^+)$, as $D(y)$ depends on y's work load, for other submitting agents, but also for x itself, as work units τ come in bursts (jobs). In the long term the speedup of x can therefore even decrease. **On the worker side, wasteful processing of replicas blocks the worker. This reduces the opportunities to work for agents that could reciprocate and thus counters the effects of the replication incentive**.

In sum, these drawbacks of the application of the *iTC* approach in the TDG reduce the robustness of the TDG, as well as the performance that participants experience. The literature review in Sec. 2 of this thesis has discussed approaches from several research directions that are in principle suited to counter these aspects. However, none of the discussed approaches is appropriate to cover all of the TDG issues simultaneously. Consider for example the application of more robust trust metrics, such as discussed in e.g. [72], [25], or [24]: While the susceptibility to *over-confidence* can be reduced by elaborate metrics from the literature, the reliance on the operation of the TM with these metrics still persists. In effect, a trust breakdown or other emergent system state can hence not be countered, and the TDG can still become *paralysed*. On the other hand, an adaptive scheduling scheme not (entirely) based on trust, such as the combined credibility/spot-checking approach presented in [204], could be applied by submitters. This would increase the robustness towards a trust breakdown by providing fall-back mechanisms that allow for submitter decision-making despite a lack of valid trust values. However, such an approach is still susceptible against TM exploitation by strategic adaptation of the agents. Additionally, it does not include any incentive for the cooperation of workers with respect to submitter misbehaviours. This concludes the discussion of TDG drawbacks. In the following, the application of the TC approach in the TDG is described.

5.2 Application of Trusted Communities in the TDG

In Sec. 3.2.2, Trusted Communities have been introduced as approach to address the generic challenging issues of the *hosting system*. In the following, it is examined how the TC approach is realised to address the specific drawbacks of the *iTC* approach in the TDG, and hence how it can improve the performance and robustness of the TDG.

To allow for a self-organised formation of a TC in a hosting system, it is necessary to define organisation benefit strategies, a default TC strategy configuration, potentially containing scenario-specific strategy implementations, as well as an agent utility function $U^x(t)$. The remaining mechanics (life-cycle, maintenance etc.) of the TC approach are generically encapsulated in the *TC Organisation* agent component. For the TDG, the application of TCs is hence dependent on the extended agent model depicted in Fig. 5.8. This model specifies a *Controller* that is constituted by the *Trust Management* agent component, the *TC organisation* agent component, as well as the TDG-specific *Submitter* and *Worker* agent components. In the following, it is assumed that a TDG agent is composed as defined by this model. This allows to consider the realisation of organisation benefit strategies for the TDG.

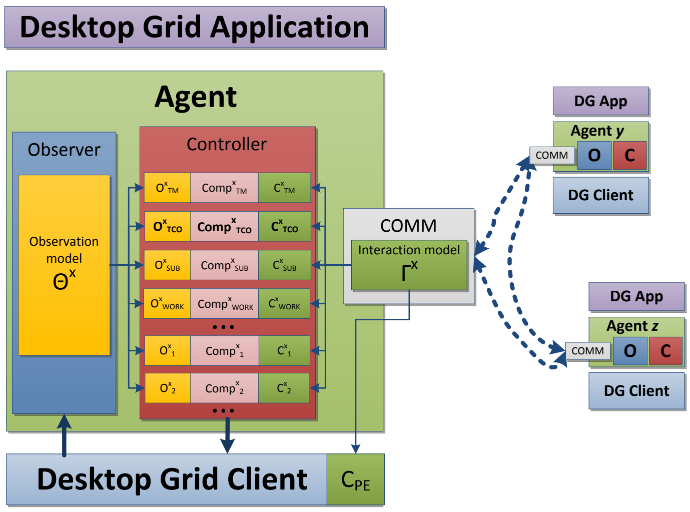


Figure 5.8: Complete model of a TC-forming TDG agent. Includes Trusted Community Organisation component (TCO), Trust Management component (TM), submitter component (SUB) and worker component (WORK).

5.2.1 Organisation Benefit Strategies

Trusted Communities have been proposed as approach to let agents self-organise in a closed environment. At the core of this TC environment are benefits that must be provided to agents in order for them to request and maintain TC membership. The presence of such benefits hence generates a *TC membership incentive*. TC benefits have been generically classified as strategies in Sec. 4.3. In the TDG, the following specific organisation benefit strategies are realised:

Worker Guarantee Incentive In the TDG, interactions between agents are mainly restricted to the processing of each others' work units, and to negotiations about the respective terms and the exchange of information necessary for the identification of suited partners (esp. reputation). In that, the most critical interaction is the delegation of the WU processing by a submitter to a worker (see Sec. 5.1.5): Workers are autonomous in their decision to accept or reject such a processing request. The reasons for a worker to reject a request (branch with effort e^0 in Fig. 5.9) reach from a too low submitter trust value (in fulfilment of the *reputation incentive strategy*), and the reservation of resources for own WUs (as discussed in Sec. 5.1.5), to the avoidance of worker overhead due to self-interested or adversary Worker component implementations (e.g. *freeriders*).

Additionally, as described above, the *reputation incentive strategy* is susceptible to exploitation as agents can safely reject processing requests as long as their reputation is above the threshold $thres_x$. For a submitter, a rejected request means that it has to search and contact additional workers (which are less competent, as the set $Y_r(x)$ is ordered according to the expected performance), or even process the WU on its own in case it has not found a single worker that accepts the request. This not only extends the *scheduling duration* as described by the WU life-cycle (see Fig. 5.4), but also increases the message overhead.

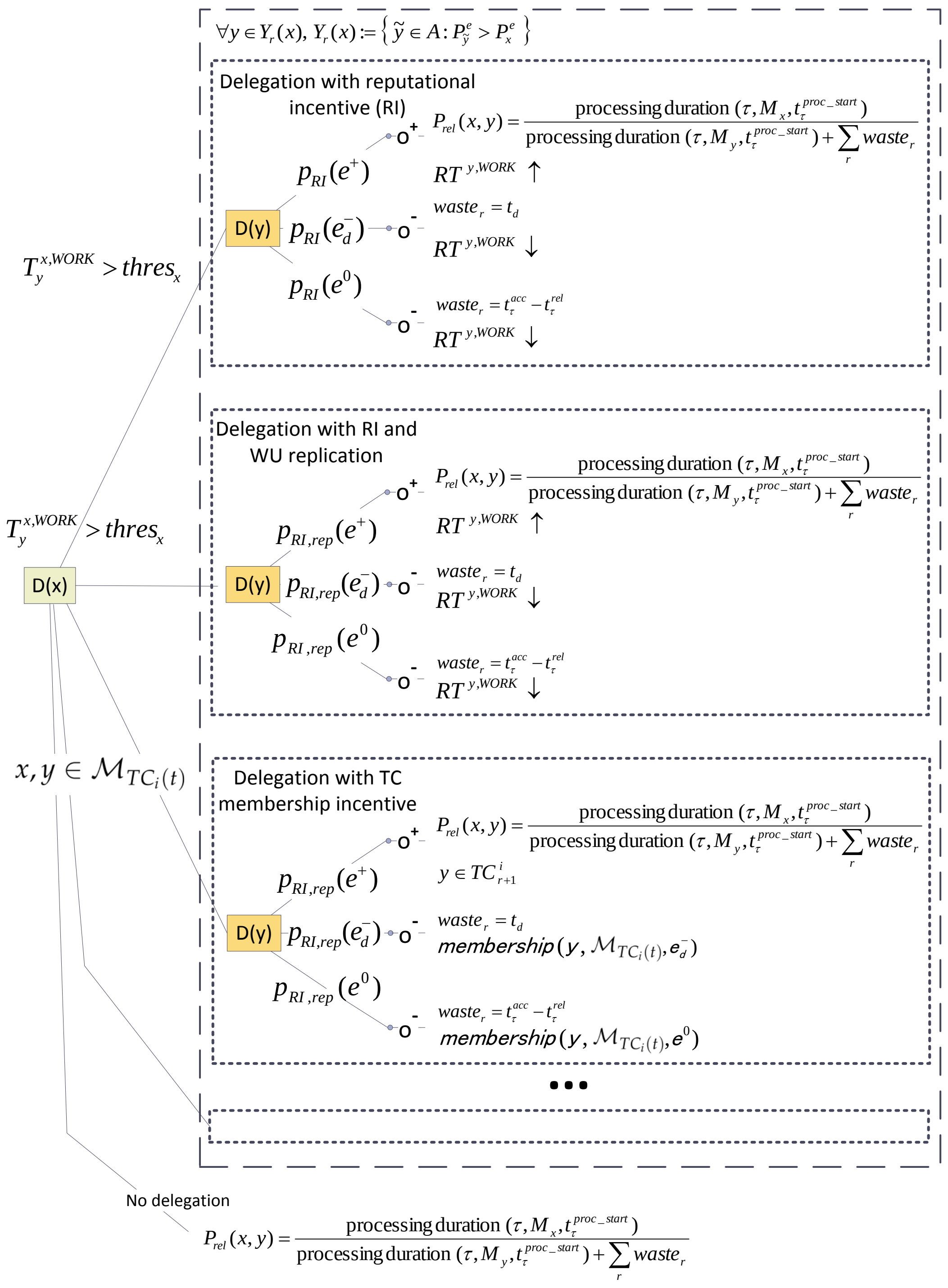
This allows for the following *interaction efficiency* strategy: TC members are obliged to accept processing requests from each other, or in other words, submitters are guaranteed to have the free choice from workers for their WUs among their fellow TC members. **The *worker guarantee* generates an incentive for TC membership, as it reduces the overhead of scheduling and increases its success**. This incentive is realised as follows: As depicted in Sec. 5.9, TC members are equipped with an **additional delegation strategy for interactions with fellow TC members** (*inbound* interactions). When becoming a member of a Trusted Community $TC_i(t)$, an agent y makes a contract with the TCM and all fellow members. This contract is based on the notion of *kinship* and states that, if chosen as delegate by any fellow member x in the round r, the agent y commits to provide the effort e^+. This commitment guarantees the prolonged TC membership of the worker y. This incentive mechanism is based on the fact that each TC member benefits from the worker guarantee when submitting WUs, and that rational agents hence invest this effort. In this, the rationality assumption is based on the composition of the TC by the gathering of highly trustworthy agents (see the description of the *Potential Member Search* strategies in Sec. 4.6.1). However, the TDG is an open system and agent behaviour can change. It must hence be accounted for the fact that TC members refuse to cooperate. This is a case of supervision by the execution of the regulatory *Member Control* strategies by the TCM, as described in Sec. 4.6.7. The effect of such a contract violation by a TC member y is depicted in the decision tree for the efforts e^0 or e_d^-: After being informed about this incident by the submitter x, the execution of the *Member Control* strategies by the TCM re-evaluates the membership privilege of the uncooperative worker y. This is formalised by the evaluation of the following function:

$$\mathcal{M}_{TC_i}(t+1) = membership(y, \mathcal{M}_{TC_i}(t), e^0) \in \{\mathcal{M}_{TC_i}(t), \mathcal{M}_{TC_i}(t) \setminus \{y\}\}\ ^{17} \tag{5.11}$$

Based on the invested effort, the TCM decides whether to exclude the agent y from the TC. In the TDG, the basic implementation of the *Member Control* strategies is utilised, such that these effort levels are assigned *member score* losses (see Sec. 4.6.7), with the effort e^0 resulting in the smallest loss.

This approach has a great benefit over the *reputation incentive* utilised by the *iTC* approach: Submitters that are TC members can always draw workers from the pool of TC members, and are always guaranteed to be granted the effort e^+. This serves as an incentive for cooperation. In sum, it can hence be expected that this increases the probability for an effort e^+ among TC members,

[17] Similar for effort e_d^-.



$\forall y \in Y_r(x),\ Y_r(x) := \{\tilde{y} \in A : P^e_{\tilde{y}} > P^e_x\}$
Delegation with reputational incentive (RI)
$P_{rel}(x,y) = \frac{\text{processing duration}(\tau, M_x, t_\tau^{proc\_start})}{\text{processing duration}(\tau, M_y, t_\tau^{proc\_start}) + \sum_r waste_r}$
$p_{RI}(e^+)$
$RT^{y,WORK} \uparrow$
D(y)
$p_{RI}(e_d^-)$
$waste_r = t_d$
$RT^{y,WORK} \downarrow$
$p_{RI}(e^0)$
$waste_r = t_\tau^{acc} - t_\tau^{rel}$
$T_y^{x,WORK} > thres_x$
Delegation with RI and WU replication
$p_{RI,rep}(e^+)$
$p_{RI,rep}(e_d^-)$
$p_{RI,rep}(e^0)$
D(x)
Delegation with TC membership incentive
$x, y \in \mathcal{M}_{TC_i(t)}$
$y \in TC^i_{r+1}$
$membership(y, \mathcal{M}_{TC_i(t)}, e_d^-)$
$membership(y, \mathcal{M}_{TC_i(t)}, e^0)$
No delegation

Figure 5.9: Decision tree for submitter agents including the TC delegation strategy. This decision tree is an extension of the decision tree depicted in Fig. 5.7. In addition to the delegation strategy with reputation incentive, and the WU replication strategy, a submitter can choose to delegate a WU based on the TC membership incentive. This choice is however only applicable for TC inbound interactions.

as compared to the same probability for non-members, such that $p_{TC}(e^+) > p_{RI}(e^+)$. Additionally, **this incentive mechanism reduces the problem of *over-confidence*, as it does not require the rating of workers with trust- and reputation values. Instead, the TCM regulation allows for quick responses to adversary behaviours**.

Transparent WU validation The *iTC* approach in the TDG utilises *WU replication* for non-validatable work units. A submitter x hence copies a work unit τ and distributes the replicas to different workers. After receiving the results, the submitter performs majority voting to validate them. As discussed in the previous section, such a safety means approach generates overhead: Not only does the workload in the system increase by the replication factor, but also the message overhead, due to the required communication with many more workers. Such an **interaction is more efficient if the submitter does not replicate its work units, but relies on results from single workers**. Trusted Communities allow for such a risky approach for the following reasons: The TC is a closed environment constituted by highly trustworthy agents, and thus reliable interaction partners. The **application of *WU replication* as safety means towards agents that have proven their trustworthiness in many interactions, is considered a form of sub-optimal over-monitoring, a phenomenon also referred to as *too much trust*** in e.g. [20]. The abandonment of *WU replication* for inbound interactions is then a risk that has been mitigated by a multitude of positive interaction experiences. As emphasised many times throughout the thesis though, it must always be assumed that agent behaviour is subject to change in an open system. Other than in the case of validatable WUs, a worker defection cannot be detected by a submitter without *WU replication*. The complete abandonment of *WU replications* would hence not allow to detect defecting TC members, and consequently, the membership itself would turn out as a risk. This is avoided by the **utilisation of combined stochastic and situation-aware TC validation**. For this, each submitter informs the TCM about every WU it delegates. The TCM then decides based on a probability whether to further delegate a single copy of the WU to an additional TC member by assigning an according TC role to it (see *Role-Assignment* Strategies in Sec. 4.6.8). In case of deviating WU results, the TCM informs the submitter, and both workers are marked for spot-checking with precomputed WUs by the TCM. When the defecting worker is revealed, the execution of the *Member Control* strategies (see Sec. 4.6.7) allows for the registration of this misconduct, and eventually leads to the exclusion of the agent from the TC. Unlike with the *Worker Guarantee Incentive*, a TC member returning false WU results for non-validation applications is immediately excluded from the TC, due to the costs of the monitoring.

Note here that this regulated validation is transparent for TC members: Neither submitters nor workers know if, and to whom, the TCM delegates WU copies, as these are not designated as such. This hinders the strategic exploitation of this approach. Additionally, note that the probability for defecting TC members is assumed to be rather low, as the costs of becoming TC members initially involve a great amount of cooperation in order to build up a high reputation and be invited to become TC member. Finally, the application of this validation process is decoupled from the TM in the system, unlike the application of an adaptive *credibility-based WU replication* as delegation strategy for the *iTC* approach. It is hence **not susceptible to a trust breakdown scenario or to the detrimental effects of over-confidence** as described in the previous section.

Submitter replication control In the discussion of drawbacks of the *iTC* approach in the TDG (see Sec. 5.1.6), the issues with *undetectable submitter behaviour* have been examined. Here, mainly the utilisation of the delegation strategy with *WU replication* is problematic: Submitters replicate their WUs, despite *validating* DG applications, in order to minimise the risk of negative interaction outcomes with single workers. While this approach is rational from the point of view of the submitters, it is detrimental for the performance of the *hosting system* when widely used, because it increases redundant processing and hence the workload of system participants. As discussed, the detection of this behaviour requires workers to cooperate, for which there are no incentives in the *iTC* approach as the consequences of this submitter behaviour are only indirectly decreasing the workers' utility. This is a major advantage of the Trusted Community approach: The incentive to remain a TC member promotes cooperation among workers when embedded as *Organisation Benefit* strategy. This is realised by a monitoring scheme that is transparent to the submitters and induces only low costs on the members: **Workers in the TC are obliged to report information on accepted WU processing requests to the *TCM*, which then is able to detect whether submitters have applied WU replication**. Again this can be sanctioned via the *Member Control* strategy and works as incentive to cooperate. This organisation benefit strategy is of the *cooperation* class, as it can not be executed within the entire *hosting system*, due to the centralisation and its scalability issues. **By detecting and sanctioning WU replication, the workload in the TC is lowered and the processing duration decreased**.

5.2.2 TC Strategy Configuration for the TDG

In this thesis, Trusted Communities have been generically presented, such that they can be applied in any open distributed system of the *hosting system* class. This is allowed by the configuration of the TC mechanics with the *basic strategy implementations* as proposed in Sec. 4.6. However, each particular instance of the *hosting system*, such as the *TDG*, allows for improvements to these basic strategies by means of scenario-specific modifications. In the following, the TC configuration used for the evaluation of the application of TCs in the *TDG* is discussed, and such modifications are presented.

The starting point for the *TDG* TC configuration is a composition of the basic strategy implementations. This configurations is however modified in the following aspects:

Non-trust-based strategy alternatives The application of the basic *TC Observer/Controller* strategy allows the TCM to adapt its TC regulation to a trust breakdown scenario by switching to non-trust-based, alternative, implementations of the TCM strategies (high-level adaptation as described in Sec. 4.6.9). Obviously, this requires a repertoire of strategy implementations to let the TCM choose from. In the following, for each TC strategy required within the operation phase of a TC (as depicted in Fig 4.1), such an alternative strategy is presented where necessary.

- *Active TC Expansion* strategy: The basic implementation of this strategy aims at recruiting new TC members from the set of unassociated agents by estimating their trustworthiness based on the collaborative execution of the *Potential Member Search* strategy (see Sec. 4.6.6). This implementation is inherently dependent on an operating Trust Management system, and can therefore not be applied successfully in the event of a trust breakdown. An alternative implementation is hence required that allows to estimate the suitability of agents as TC members. This implementation is based on the approach of *spot-checking* in the literature (cf. e.g. [175]): The TCM generates a spot-checking role that determines the submission of pre-processed work units to potential candidates. These WU delegation requests are openly declared as test tasks for the following reasons: Unassociated agents may be unaware of the presence of a trust breakdown and adhere to the application of trustworthiness thresholds within their worker decision-making (see Sec. 5.1.5). A trust breakdown is also expected to affect TC members, such that their delegation could fail if not openly declared as test-task. This is especially true for the case that a trust breakdown was the consequence of an overload situation and agents have an overall high workload. By openly advertising tests-tasks as TC *invitation with conditions* (see Sec. 4.6.6), workers are motivated to consider the processing in order to become a TC member. The amount of successful spot-checks required to finally invite an unassociated agent as TC member is then determined by the TCM with a threshold. This approach represents in effect an alternative trustworthiness estimation scheme, however based on redundancy as the results of the processed WUs are not further used. This implementation is hence inferior to the basic implementation and is therefore only applied as fallback in the event of a trust breakdown.

- *Member Control* strategy: The basic implementation of this strategy has been described in Sec. 4.6.7 as approach based on the administration of a *member score* system. Each TC member is assigned a current score. This score decreases (as described, the amount is determined by a sanction function) whenever a member behaves uncooperatively towards its fellow members, including the TCM. Such behaviours include the rejecting of inbound processing requests, the return of invalid WU results, or the denial to execute TC roles assigned by the TCM. The *member score* mechanism already represents a behaviour estimation alternative to the Trust Management in the TDG. This is a design choice to increase the robustness of Trusted Communities towards uncooperative agents that threaten the operation of a TC. Due to this decoupling, the basic implementation can be applied unmodified in trust breakdown situations.

- *Role-Assignment* strategy: The basic implementation of a role-assignment strategy, as described in Sec. 4.6.8, allows the TCM to allocate roles to TC members based on the minimisation of the overall costs of this allocation. As the allocation is based on the interpretation of a *cost-matrix*, and role execution costs are assumed to be immutable throughout the TDG states, this implementation is also decoupled from the TM system. It is therefore not necessary to provide an alternative implementation for the application in the TDG.

These strategies are required for the management of a TC and are executed by the TCM. However, the following strategies are also relevant in the operation phase of a TC, albeit being executed by TC members. Here, the TCM instructs its members to exchange their strategy implementation upon detection of a trust breakdown.

- *Distributed Leader Election*: The basic implementation of this strategy aims at electing the most suited TC member as TCM, while ensuring that the amount of private information disclosed in the election process, as well as the message overhead, are minimised (see Sec. 4.6.5). In the event of a trust breakdown, the election based on the highest average trustworthiness is assumed to be not better than any other criterion, due to the fact that the trust values are not reliable (see discussion in Sec. 4.6.9). However, the time and message complexity is worse than that of other distributed leader election algorithms, such that an alternative implementation is preferred here. For the TDG, an ID-based election implementation is utilised as alternative. Such an algorithm requires only one round of message passing in order to allow each agent to determine who the leader is. Additionally, it does not disclose any private information. It has been shown that for this problem algorithms with a time complexity of O(n) exist (cf. e.g. [209]).

- *Membership Evaluation* strategies: The basic membership evaluation strategy described in Sec. 4.6.4 compares the utility of an agent before the association to a TC with the current utility received as TC member. This decision-making is decoupled from the TM and does not require an alternative implementation.

TDG-specific modifications A Trusted Community is a closed environment in which safety means are abandoned in order to improve the utility of member agents by optimising their interactions. As discussed generically for all *hosting systems* in Sec. 4.6.7, and specifically for the TDG in Sec. 5.2.1, the exploitation of this abandonment of safety means by adversary TC members must be compensated by the regulatory execution of *Member Control* strategies by the TCM. As described in Sec. 4.6.7, this control cannot be determined generically, but must be realised scenario-specific. For the TDG, the application of *spot-checking* (cf. e.g. [175]), hence the utilisation of pre-processed test-WUs, is an adequate instrument to validate the worker *willingness* of a TC member. Here, complaints about rejected cooperation by other TC members are processed by the TCM, in that it utilises test-WUs to validate these claims. These test-WUs are not marked as such, and delegated by any member of the TC (assigned as role), in order to prevent the defecting member from being aware of the test. This contrasts the approach presented as alternative *Active TC Expansion* strategy implementation. The validation of test-WUs then determines whether the TC members are sanctioned by a decrease of their *member score*, as described in Sec. 4.6.7. This combined approach of member complaints processing, and their validation by *spot-checking*, then allows to realise the *Worker guarantee incentive* as described at the beginning of this section.

In addition, the *Transparent WU validation* is realised via a combined stochastic/situation-aware validation approach as described in Sec. 5.2.1. For this, the TDG implementation of the *Member Control* Strategy is extended, such that in case of non-validating applications, submitters that delegate WUs to TC members with a *member score* below a threshold are always assigned additional validating workers by the TCM. Additionally, such validating workers are assigned stochastically by the TCM (realised as TC role). Here, missed validations immediately lead to the exclusion from the TC by the TCM, due to the severity of this threat (the sanction function hence allows only the value $score_{max}$).

Finally, the *Submitter replication control* requires the strategy implementation to assign a sanction value on the *member score* for the detection of submitter replication. For this, the TCM utilises a comparison of checksums for delegated WUs by each submitter.

5.2.3 Discussion

In the previous section, the drawbacks of the *iTC* application in the TDG have been discussed. In the following, organisation benefit strategies, as well as a strategic configuration for the application of

TCs in the TDG have been explained. This completes the required specification of the TC mechanics for the TDG, and allows for a comparison of the *iTC* application with the TC application in the TDG, in reference to the drawbacks discussed in Sec. 5.1.6:

- *TM exploitation*: The *iTC* approach does not prevent the exploitation of the TDG Trust Management. This is depicted in Fig. 5.10: A rational TDG agent is motivated to build up its trust-

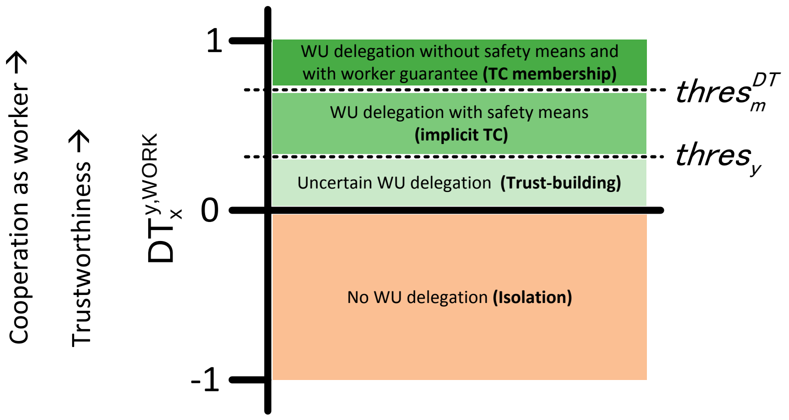

Cooperation as worker →
Trustworthiness →
$DT_x^{y,WORK}$
1
0
-1
WU delegation without safety means and with worker guarantee **(TC membership)**
WU delegation with safety means **(implicit TC)**
Uncertain WU delegation **(Trust-building)**
No WU delegation **(Isolation)**
$thres_m^{DT}$
$thres_y$

Figure 5.10: Comparison of incentives for the TDG between the *reciprocity-based incentive* of the *iTC* approach, and the *TC membership incentive* of the Trusted Community approach. Workers are motivated to cooperate more when TCs are applied, due to the organisation benefits of TC membership and its high entry threshold.

  worthiness by cooperation as worker, in order to prevent isolation as submitter. However, once the trust value is higher than the worker acceptance threshold $thres_y$, there is no motivation to further cooperate, with the exception of occasional cooperation to maintain that trust level. In contrast, the application of Trusted Communities generates an additional incentive, *TC membership*, which allows agents to delegate their WUs without safety means and with the guarantee of successful delegations. TC membership is however granted only to agents with a high trust level. This is quantified with the threshold $thres_m^{DT}$, which is used by the basic *Potential Member Search* strategy, and consequently by the basic *Active TC Expansion* strategy, to determine new TC members in the pool of unassociated agents (see Sec. 4.6.1 and Sec. 4.6.6). Rational TDG agents that want to take advantage of the TC benefits in order to increase their utility, will therefore cooperate longer to build up the required trust level. Additionally, the *iTC* problem of ceasing worker cooperation, once a required threshold is reached, is avoided in the TC approach by the application of the modified *Member Control* strategy: Agents lose TC membership much faster than agents lose reputation. Additionally, agents excluded from a TC are blacklisted, such that they are not invited to become TC members again until the TCM removes them from the blacklist due to forgiveness (see Sec. 4.6.6).

- *Over-confidence*: The application of TM in the TDG is susceptible to the development of *over-confidence*. In the application of the *iTC* approach in the TDG, *over-confidence* results from the slowness of the TM to adequately incorporate negative experiences with workers that have a high reputation. In effect, it leads to submitters that adhere to delegating their WUs to such workers until eventually the trustworthiness sinks below the worker threshold $thres_x$. In the application of Trusted Communities in the TDG, this effect is avoided by the execution of the situation-aware *Member Control* strategy: In case of a report about uncooperative behaviour towards a fellow TC member, an agents' trustworthiness is validated in a focussed but transparent *spot-checking* approach, as described in the previous section. In case of a failed validation, the reaction, *member score* sanctions and eventual TC exclusion, is more immediate (the member score mechanism is usually parameterised very sensitively), and more sustained, than reputation decrease: In the *iTC approach*, defecting agents that have exploited *over-confidence* need only few cooperations to reach the worker threshold again, while the exclusion from a TC results in blacklisting and prevents TC benefits for an extended duration (see Sec. 4.6.6). In sum, the TC approach is less susceptible to the exploitation of *over-confidence*.

- *TM reliance*: The *iTC* approach is based exclusively on trust-based decision-making, as are most approaches in the literature (see discussion in Sec. 2.2). As discussed, the complex dynamics of agent interactions in an open distributed system can lead to the emergence of trust breakdowns. In this event, the trust relationships between most agents are so low (red isolation margin in Fig. 5.10) that the system can get *paralysed*, i.e. agents do not interact with each other due to their perception of the involved risk. In the TDG, this phenomenon manifests as the submitters' inability to determine suited workers (due to the worker trustworthiness threshold $thres_x$), and the isolation of submitters due to the workers' rejection to process WUs of submitters with low trustworthiness values (utilisation of the submitter trustworthiness threshold $thres_y$ for the reciprocity-based reputation incentive).

  This is avoided by the TC approach for agents that are members of operating TCs: Interactions between members, in the TDG mainly WU delegations, are considered safe without restrictions. Hence, for a submitter x, all fellow TC members appear suited as workers, disregarding their current trustworthiness value. This counters the above mentioned submitter inability to find suited workers. On the other hand, TC members are obliged to cooperate with fellow TC members. In the TDG, this is realised via the *Worker Guarantee Incentive* (see Sec. 5.2): TC members that receive a WU processing request from fellow members must accept and invest the effort to produce a valid result, while disregarding the submitters' trustworthiness, in order to remain TC members. This prevents the isolation of submitting TC members. Moreover, the continued cooperation allows for a quick recovery from the trust breakdown, such that at least TC members are eventually considered trustworthy again by unassociated agents (see the evaluation results presented in Sec. 5.3.3). Additionally, the maintained recruitment of additional members from the group of unassociated agents, by application of trust-independent *Active TC Expansion* strategies (see Sec. 5.2.2), reinforces this recovery process.

  In effect, the application of TCs in the TDG generates robust partitions of the agent society in which cooperation prevails despite a trust breakdown. These partitions consist of subgroups of the agent society, the TC members. The number of these agents, $|\mathcal{U}^{\mathcal{H}}(t)|$, then determines the degree of paralysis in the *hosting system*.

- *Sub-optimality due to safety means*: The *iTC* approach for the TDG suffers from sub-optimality due to the utilisation of safety means for non-validating DG applications. These safety means are realised as the replication of work units, and the comparison of their processing results by majority voting. While this invariable validation generates an incentive for workers to cooperate, its utilisation considerably increases the workload of the TDG and degrades its performance.

The application of TCs in the TDG mitigates this performance decrease by the abandonment of regular WU replication in *inbound* interactions. This reduces the workload of TC members and hence increases the performance. As discussed above, the WU replication must however be maintained to some degree in order to account for the dynamics of agent behaviour. This is achieved by an adaptive approach encapsulated by the *Transparent WU validation* strategy (see Sec. 5.2.1). In effect, its execution decreases the workload in the TCs, and consequently in the entire *hosting system*. This improves the performance of the *hosting system* (see evaluation in Sec. 5.3.2).

- *Sub-optimality due to undetectable submitter behaviour*: In Sec. 5.1.6 the susceptibility of the *iTC* approach to undetectable submitter behaviour has been discussed. The conclusion of this discussion was that the utilisation of the WU replication strategy by risk-minimising submitters is not detected by other agents in the system. This is due to the fact that such detection must be executed collaboratively, that self-interested agents have no *direct* utility gain from this collaboration, and that there is no incentive mechanism to enforce it. This is different in Trusted Communities: TC members accept the delegation of control to the TCM and allow it to allocate TC management roles to them. This mechanism is used to enforce member collaboration to detect submitter behaviour that goes unnoticed for single interaction partners: By pledging TC members to inform the TCM about each accepted WU processing request, the TCM can compare (checksums of) submitted WUs and decide whether WU replication has been executed. To discourage its usage, the TCM has a sanction mechanism based on the execution of the modified *Member Control* Strategy (as described in Sec. 5.2.2) at its disposal. Note that such an approach requires closed environments, such as a TC, to be executable. This is due to the centrality of this approach and its lack of scalability in a distributed system, such as the *hosting system*. In sum, the application of TCs in the TDG avoids the problem of undetectable submitter behaviour, by applying a centralised monitoring of their delegation activity and enforcing good conduct through sanctions. In effect, TC members are discouraged from the application of WU replication as means of risk reduction. This avoids the additional workload involved in this behaviour and hence improves the performance of the *hosting system*.

This completes the discussion of the benefits of TC application in the TDG, as opposed to the application of the *iTC*, or similar trust-based approaches. The elaborated improvements of the *hosting system* can however only be realised when the TC application is successful in the TDG. This refers to the actual formation, and sustained operation of Trusted Communities. In Sec. 3.2.4 the generic assumptions with respect to such a successful application have been laid out. Finally, the remaining specification required to apply Trusted Communities in the TDG is the utility function $U^x(t)$ of agents. This function is defined in appendix D which introduces an analysis on performance and robustness metrics suited for the TDG. In the following, the evaluation results are presented and discussed.

5.3 Evaluation Results of the TC Application in the TDG

This section starts with the discussion of the experimental setup used for the evaluation. In the following, the **performance** of agents in the TDG is compared when applying the *iTC*-, the *Clan*-, and the TC-approach. Finally, the **robustness** of the TDG in case of collusion attacks is compared for the three organisation forms.

5.3.1 Experimental setup

The evaluations have been conducted with the following setup (cases of deviation are documented in the text):

Agent stereotypes The configuration of agents has been limited to a few stereotypes to account for the application of default component configurations (as described in Sec. 3.1.4 and Sec. 5.1.4), while additionally allowing to evaluate uncooperative behaviours. The stereotypes used were:

- *Adaptive agents (ADA)*: The default implementation of an agent using only default components. This agent type uses the trustworthiness threshold as defined by the *iTC*-approach.
- *Freerider agents (FRE)*: These agents represent the type of user that wants its own grid jobs to be processed in the TDG as fast as possible, while not being willing to invest any effort as worker. In result these agents consequently reject WU processing requests.
- *Egoistic agents (EGO)*: These agents represent a malicious and/or faulty type of behaviour. Unlike FRE-agents, EGO-agents accept WU processing requests, but do not return any result with a certain probability (0.2 percent for the experimental setup).
- *TM-exploiting agents (CAA)*: These agents are based on the ADA-agents, but illustrate the susceptibility of the *iTC*-approach to TM exploitation: CAA-agents behave like regular *iTC*-agents as long as they do not reach a reputation threshold (0.5). Then these agents switch to a free-riding mode, rejecting all processing requests until again, they reach a lower reputation threshold (0.1). Then these agents start building up reputation again, by accepting WU processing requests based on the rules of an ADA-agent. The effect of this behaviour is depicted in Fig. 5.11, for a single agent in an exemplary simulation run.
- *Defecting agents (DAA)*: This is a behavioural stereotype that can be imposed on any other agent type and demonstrates the threat of false WU results for non-validating DG applications. Agents that are combined with this stereotype accept and process WUs just as their supertype (e.g. ADA), but have a probability (0.2) of returning a false WU result in the processing. It is further assumed that agents of this type collude by always producing the same false result, such that the majority-voting for a set of results returned by such agents can lead to the acceptance of a false WU result (see the description of the *accuracy*-metric in appendix D.1.4).

All agents are additionally equipped with a TC organisation component configured for the TDG application (as described in Sec. 5.2.2), and allowing them to form Trusted Communities. In alternative experimental runs, the agents are equipped with a respective component to allow for *Clan*-formation.

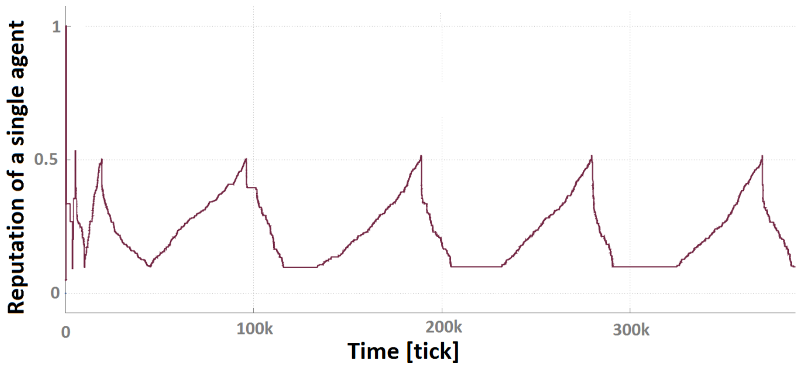


Figure 5.11: Reputation of a TM-exploiting agent: When its reputation is above the threshold 0.5, the exploiting agent starts to free-ride, rejecting all processing requests. This behaviour is changed to cooperation again when the reputation of the agent reaches the lower threshold 0.1. The course of the reputation shows that firstly this agent has been consequently requested as worker which allowed for reputation build-up, and secondly that the time to reach the upper threshold was much higher than the time to fall back to the lower threshold.

Clan comparison In Sec. 2.5.1 the MAS organisation *Clan* (cf. [118]) has been summarised and its relevance as related work for Trusted Communities has been emphasised. Consequently, *Clans* have been used in the evaluation of TCs as comparable form of agent organisation. In that, *Clans* and TCs have been evaluated separately, but under identical system conditions, such that the set of organisations in the hosting system is either interpreted as $\mathcal{O}^{TC}(t)$, or as $\mathcal{O}^{Clan}(t)$.

In [118], the article proposing *Clans*, the mechanics of *Clans* are given in algorithmic form such, that an implementation could be derived for this thesis. However, the author does not provide any information on the evaluation of the approach, especially not in a Desktop Grid scenario. The *Clan* implementation has hence been tailored for the application in the TDG by the author of this thesis. In that, the following aspects required special implementations to the provided generalised concept described by the article:

- *Goals and plans*: The goals of agents participating in the TDG have been described as the rapid processing of DG jobs throughout the thesis. For this several metrics have been introduced. As discussed in the rationale (see appendix D.3), the *speedup* has been identified as the most relevant metric. A concrete *plan* to fulfil the goal is generated for each DG job produced. Plans are hence understood as the mapping of workers to every WU contained in such a job.

- *Formation criteria:* The formation of Clans has been described as a check of criteria against thresholds, esp. trustworthiness thresholds. For this, the same values have been used as for TCs. Additionally, the criteria have been interpreted as follows for the TDG application:

  Missed opportunities: This criterion allows for a straight interpretation in the TDG system: Whenever an agent has rejected a processing request of another agent, but has requested the same agent to process one of its own WUs and was also rejected, a *missed opportunity* is registered.

  Scalability: The definition of the scalability criterion in [118] is such that in systems with many cooperation requests and a large number of agents, scalability is an issue as the search for interaction partners is more expensive than in small systems. However, it is assumed for the TDG, that this criterion is always fulfilled.

Lack of information: This formation criterion is described dependent on a system state in which a large group of agents is not trusted or not known, while a small group of agents is very trusted. *Clan* membership is then aspired to gain access to trust and capability information about the group of unknown agents. This criterion can be generically applied.

High failure rate: This criterion describes agent states in which the agent has many unsuccessful interactions. In the TDG this can be directly translated into the waste (see Sec. D.1.3) metric: The higher the waste, the more interactions were unsuccessful and the greater the motivation to form or join a Clan.

Additionally, in [118], agents are described as seeking *Clan*-membership whenever they have a plan that requires cooperation, and any of the above mentioned criteria is met. In a strict interpretation, a TDG agent would seek *Clan*-membership only if it had a DG job to distribute as submitter and be in active search for workers. In a less strict interpretation, agents expect the generation of consecutive DG jobs and seek *Clan*-membership exclusively based on the formation criteria. In this thesis, this less strict interpretation has been chosen as this is seen to more reflect the agent rationality assumptions made by the author of the article on *Clans*.

- *Kinship motivation:* For the application in the TDG, the motivation to cooperate when requested by a fellow *Clan* member is interpreted as situation-aware probability, as depicted in Tab. 5.3: This table extends the worker decision making described in Sec. 5.1.5: When using *Clans*, this

	Currently no jobs	**Unprocessed job**
Worker decision: False	0.5	0.8
Worker decision: True	1.0	1.0

Table 5.3: Values for kinship motivation.

decision is not considered final, but can be overridden by situations with a high motivation for cooperation. The table primarily accounts for the fact that there is a *base probability (0.5) to cooperate* among *Clan* members even when the requested agent has currently no job and it would normally not accept the request. If the requested agent has currently a job to distribute (a *plan* requiring cooperation from fellow members), the motivation is higher (0.8), due to reciprocity expectation. Finally, fellow *Clan* members are never discriminated against non-members, i.e. when the decision to cooperate as worker is positive, the motivation does never decrease the probability to cooperate. Note here, that the author of the *Clan* approach does not provide any information on the quantification of motivational values, and that motivation is ascribed a subjective quality (while the values provided in this thesis are applied by all agents disregarding any disposition for more specific quantifications).

- *Preferred Clan size:* The initiator of a *Clan* formation considers the number of agents required to execute the tasks of its current plan and a small redundancy in order to determine how many agents to invite to a Clan. In the TDG, this is expressed by the average number of WUs in a job. However, the *Clan* approach describes the (seemingly unconstrained) extension of a *Clan* by invitations of members towards unassociated agents. This has been interpreted such that members invite other agents only when the preferred *Clan* size is not already exceeded.

- *Adaptation to avoid bias:* The benefits of *Clan*-membership are mainly determined by cooperation resulting from the kinship motivation. In the TDG, this is the high probability that a fellow member will accept a processing request as a worker. In Trusted Communities, this is related to the *Worker Guarantee Incentive* discussed in Sec. 5.2.1. However, TCs also allow for another, even more substantial benefit: The *Transparent WU Validation* introduced in Sec. 5.2.1. In a comparison of the performance of both approaches, this additional benefit strategy is likely to bias the results of the organisations as such. This is especially valid for the case that the *Transparent WU Validation* strategy is applied without using TCs. In order to avoid such a biased comparison, the performance evaluations of *Clans* were hence conducted with an adapted implementation that uses the additional *transparent WU validation* benefit within *Clans*, as is done in TCs.

TDG composition In general, the evaluations used the following setup:

- The agent society $\mathcal{A}$ consisted of 250 agents of which 10 % were FRE-agents, 10 % were EGO-agents and the remaining 80 % of agents were of the ADA-type (for the undisturbed case, deviations are documented below).
- Agents had processing capabilities from the range of a performance level (PL) of 2, to PL 5.
- *Non-validating* DG applications were used, and agents produced a job in average after 4500 ticks. The jobs contained an average of 11 work units (as defined in Sec. 5.1.2) with average processing costs of 350 (the according processing duration for a single WU copy was between 175 ticks (PL 2) and 70 ticks (PL 5)). The quorum was set to 3, such that each original WU had in general to be replicated 2 times to allow for the majority voting. An exemplary job processing scheme resulting from this setup is also illustrated in Fig. 5.12. Finally, this setup had the following implications on the preferred clan size (as described above): The max. WU number for a job (15) was taken as base number for the preferred size, adding a redundancy of 10. This resulted in preferred *Clan* size of 25 agents.
- The Trust Management system was parameterised with the rating values defined in Sec. 5.1.4. Additionally, the initial reputation of each agent was 0.05.

Metrics The metrics used are discussed in appendix D.3. In the evaluation result presentations the average (mean) of these metrics (e.g. *speedup*) is provided, along with the *standard deviation*. Here, the notation s_{N-1} is applied for the *standard deviation*, based on the *bias-corrected variance*, as defined by:

$$s_{N-1}^2 = \frac{1}{N-1} \cdot \sum_{N}^{i=1} (x_i - \overline{x})^2 \quad \text{and then} \quad s_{N-1} = \sqrt{s_{N-1}^2}$$

5.3.2 Performance Evaluation

The main goal in the development of the Trusted Community approach has been described throughout the thesis as the increase of the systems' performance by TC application. In the following discussion of the evaluation results, this performance improvement, compared to the *iTC*-approach and the state-of-the-art approach *Clans*, is documented for the TDG. As described in the introduction of the section, the agent society used in the evaluations was comprised of agents based on stereotypes: Cooperative agents (most ADA-agents), as well as various stereotypes of uncooperative agents. The aim in the evaluation was to achieve a high performance of cooperative agents, while allowing only for a low performance of uncooperative agents (isolation). The latter aim is motivated by the potential of the control of open systems through cooperation incentives. The isolation of uncooperative agents through the utilisation of the *iTC* approach in the TDG has been evaluated and documented in various publications (see e.g. [8], [18]). The main focus in this thesis is therefore on the performance of cooperative agents.

The setup of the performance evaluations in the TDG allowed for two cases: The utilisation of validating, or non-validating DG applications by the participating agents. In the former case, a submitter can easily validate the correctness of a WU result without having to re-process it. In the latter case, several WU results must be compared in order to allow for statements about the correctness. Only few types of applications allow for programmatic validation. Additionally, such validating applications have lower demands on submitter-strategies and are therefore less challenging. As a consequence, the evaluations for this thesis have been conducted with non-validating applications. For the sake of completeness, evaluation results for validating applications have been published in [18].

In the following, experimental results for a variety of conditions are summarised in Tab. 5.4, and then analysed for each condition. The metric values are averages for the group of cooperative agents in the system, as described above. Each condition has been evaluated in an experiment containing 100 simulation runs for each organisation type (standard deviation depicted in brackets). The parameterisation of the approaches was based on parameter studies conducted for each experiment and organisation approach. In each case, the best parameterisation was used to produce the results. A more detailed visualisation of the results, depicting the average speedup in each experimental run, can be found in appendix E. Note that as described above, an adapted, more competitive, *Clan*-implementation was used in the evaluation.

Conditions	**Metrics** (s_{N-1})	**iTC**	**Clan**	**TC**
Undisturbed	Avg. speedup	5.411 (0.222)	7.223 (0.373)	**8.491** (0.623)
	Accuracy	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
	Avg. organisation utility	-	1.770 (0.461)	**2.827** (0.494)
	Avg. operating organisations	-	5.280 (0.877)	3.730 (1.100)
20 % defecting (DAA) agents	Avg. speedup	5.327 (0.203)	7.066 (0.309)	**7.108** (0.353)
	Accuracy	0.994 (0.001)	**0.998** (0.001)	0.996 (0.001)
	Avg. organisation utility	-	1.667 (0.332)	**2.284** (0.406)
	Avg. operating organisations	-	4.880 (1.387)	3.360 (0.746)

Conditions	Metrics (s_{N-1})	iTC	Clan	TC
30 % defecting (DAA) agents	Avg. speedup	5.316 (0.207)	**6.936** (0.309)	6.702 (0.234)
	Accuracy	0.987 (0.002)	**0.995** (0.002)	0.991 (0.002)
	Avg. organisation utility	-	1.567 (0.458)	**1.920** (0.178)
	Avg. operating organisations	-	4.350 (1.794)	3.820 (0.626)
20 % TM-exploiting (CAA) agents	Avg. speedup	4.962 (0.195)	6.785 (0.209)	**8.102** (0.418)
	Accuracy	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
	Avg. organisation utility	-	2.042 (0.254)	**3.530** (0.387)
	Avg. operating organisations	-	4.980 (0.284)	2.500 (0.659)
30 % TM-exploiting (CAA) agents	Avg. speedup	4.686 (0.184)	6.393 (0.202)	**7.346** (0.366)
	Accuracy	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
	Avg. organisation utility	-	2.158 (0.196)	**3.517** (0.369)
	Avg. operating organisations	-	4.650 (0.479)	2.470 (0.540)

Table 5.4: Performance evaluation results of the *iTC*-, the *Clan*-, and the TC approach for non-validating DG applications in various conditions. Best values are depicted in bold where applicable. The average metric values refer to the group of cooperative agents and are based on 100 experimental runs each.

Undisturbed Experiment

In this experiment, the three organisation approaches were compared for the case of an agent society without additional uncooperative agents (apart from the initial 10% FRE-, and 10% EGO-agents). The major aim here was to quantify how Trusted Communities improve the TDG by organising agents and allowing them to take advandage of the organisation benefit strategies (as described in Sec. 5.2.1). This performance is compared to the performance of *Clans* which pursue a similar aim by providing members with the benefits of *kinship motivation*. As described above, the *Clan*-implementation used here also allows *Clan*-members to benefit from the *Transparent WU Validation* as presented in Sec. 5.2.1.

This experiment was conducted to demonstrate the ability of Trusted Communities to optimise the interactions of agents within the TDG. In Fig. 5.12 the times agents were active as workers are depicted for the case of *iTC*-application. The diagram shows firstly that the 20% uncooperative agents (on the right side) are isolated and forced to process their own WUs (red marks). Additionally, the diagram shows that the cooperative ADA-agents have a high workload. Especially agents with a high performance level are sought as workers and are constantly processing WUs with little idle time.

In contrast, the application of TCs allows TC member agents to submit WUs without replicating them. As there is no uncooperative behaviour among the agents (undisturbed case), the majority of agents is organised in TCs. This optimises the interactions among the agents and results in a

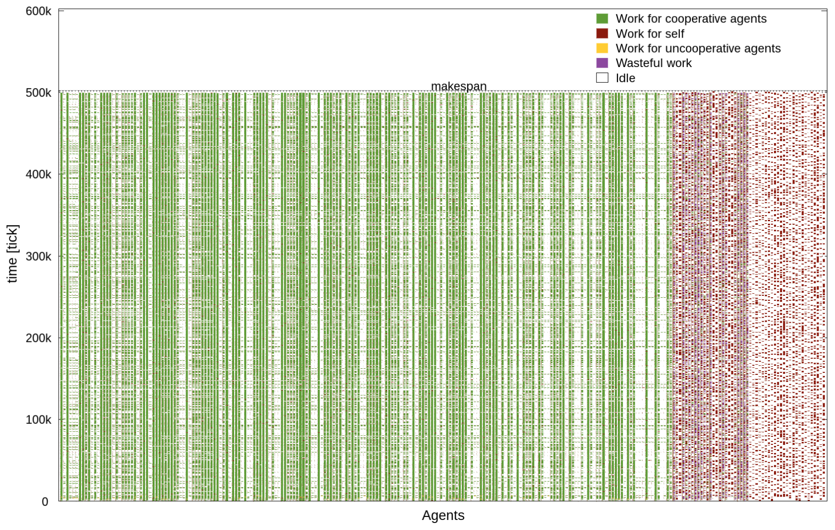


Figure 5.12: Worker diagram for the *iTC*-approach in the undisturbed performance experiment: The majority of agents has a high workload, working for other cooperative agents, throughout the duration of the simulation run. The second group of agents (right side) is the group of 10% EGO and FRE-agents. These agents are isolated and forced to process their own WUs.

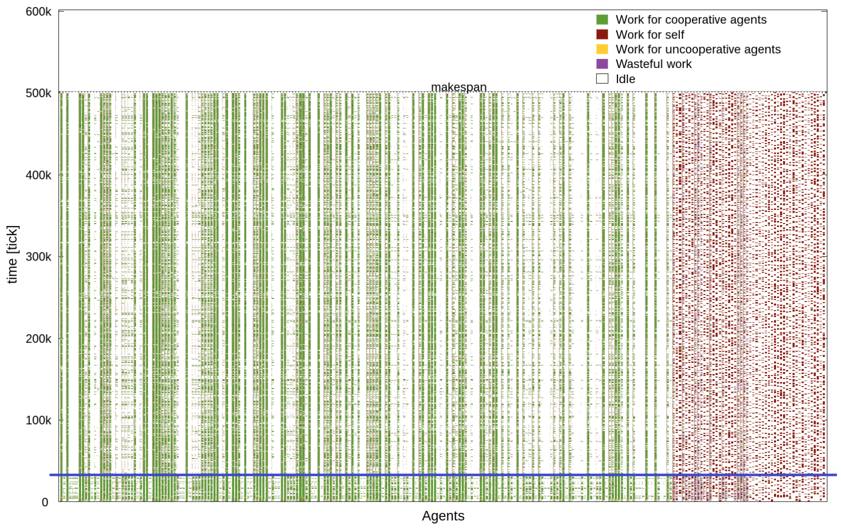
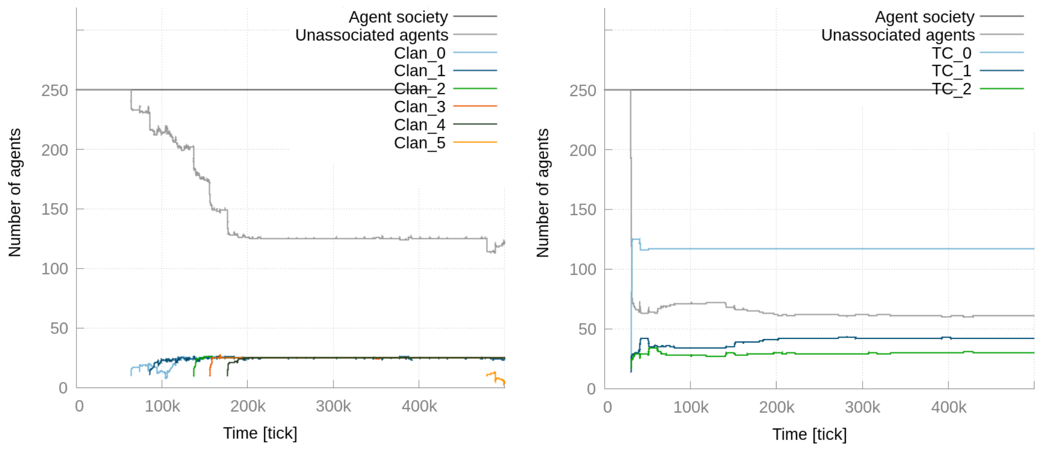


Figure 5.13: Worker diagram for the TC-approach in the undisturbed performance experiment: This diagram shows how TC members reduce their workload due to the *Transparent WU Validation* strategy (see Sec. 5.2.1) while maintaining the isolation of uncooperative agents (right side). The difference shows best in comparison to the time before TC formation (at tick 30k, depicted by blue line) and the diagram for the *iTC*-approach application (depicted above).

reduced workload, as depicted in the lower of the two graphs. Consequently, the throughput is higher and the speedup increases. This shows in the average speedup values for the 100 experimental runs summarised in Tab. 5.4: *iTCs* provide an average speedup that is only 63,73% of the speedup achieved by applying TCs. *Clans* also provide the benefit of abondoning the WU replication to their members. Despite that, the average speedup for *Clans* is lower (85,07%) than the average speedup achieved by TCs. This is mainly a result of the composition of the organisation forms: As depicted

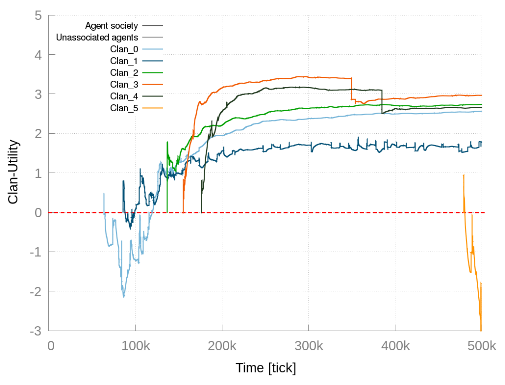


(a) Clan (b) TC

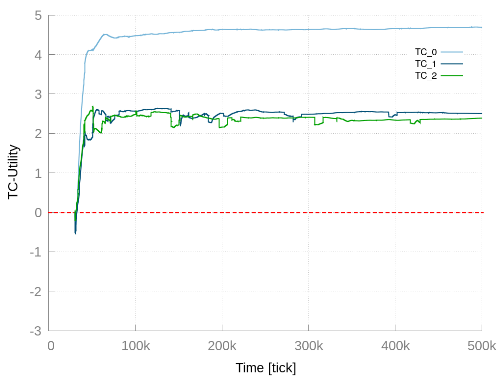
Figure 5.14: Comparison of the number of agents associated to a Clan/TC in case of the undisturbed performance experiment: When *Clans* are applied, the amount of agents associated to any *Clan* is approx. 50%. With a total amount of 20% uncooperative agents, approx. 30% of cooperative ADA-agents remain unasassociated and do not benefit from organisation membership. When TCs are applied, the number of members is significantly higher with approx. 75%. From the remaining 25% of agents, only 5% are cooperative ADA-agents.

in Fig. 5.14a the total amount of agents associated to any *Clan* is approx. 50%. Hence 30% of the agents do not gain *Clan*-membership despite being cooperative. In contrast, the amount of cooperative agents remaining unassociated is only 5% when TCs are used. The difference of agents being able to submit their WUs without replicating them hence amounts to 25%. This obviously influences the speedup. Additionally, the size of the organisation forms is a key factor in the average speedup: While *Clans* are constrained by a preferred size (see discussion in Sec. 5.3.1), TCs are unconstrained. The more members an organisation has, the more workers are available for WU processing without replication, hence the greater the speedup improvement. This is also supported by the graphs depicted in Fig. 5.15: The relative organisation utility (as defined in Eq. 4.21), hence the average speedup during membership in relation to the average speedup as unassociated agent, mainly depends on the number of members in the organisation. This is best demonstrated by the utility of the largest TC (TC_0): Having approx. 120 members and achieving a utility of approx. 4.5 this TC provides its members with a significant benefit (the average organisation utility being 2.6). Averaged over the 100 experimental runs, the relative organisation utility between the approaches amounts to 1.770 in case of *Clans*, and 2.827 for TC application.

Finally, the speedup of TCs is decreased by the application of a *Member Control* strategy that utilises an number of randomly chosen TC members for whom WU replication is applied despite their



(a) Clan



(b) TC

Figure 5.15: Comparison of the relative organisation utilities of Clans/TCs (as defined in Eq. 4.21) for the undisturbed performance experiment: Both approaches show significant speedup improvements for organisation members. In that, the comparison of the relative utility of single formed *Clans*/TCs with their size (as depicted in Fig. 5.14) shows a strong positive correlation. This is due to the number of available workers that do not require the application of WU replication.

membership in order to allow for the detection of defecting agents. This strategy is not executed by *Clan* members as *Clans* are not regulated by a manager and members cannot be excluded even in the presence of uncooperative behaviour. However, this experiment does not include defecting agents and thus the execution of this strategy in TCs manifests as overhead and decreases the advantage of TCs over *Clans*.

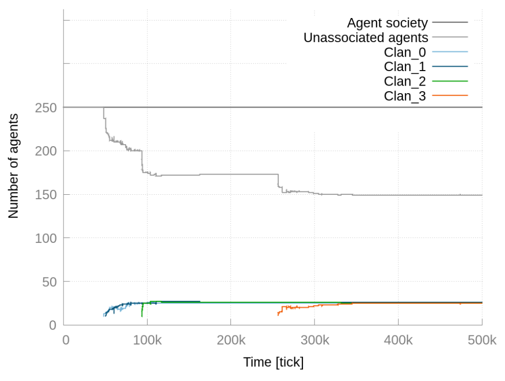
In summary, this experiment demonstrates that the optimisation of agent interactions by the *Transparent WU Validation* strategy yields significant benefits. This shows in an increased speedup of both explicit organisation approaches *Clan* and TC, as opposed to the *iTC*-approach. In addition, the diverging organisation mechanics allow TCs to outperform *Clans*, both in terms of average speedup and average relative organisation utility.

Experiments With Defecting Agents

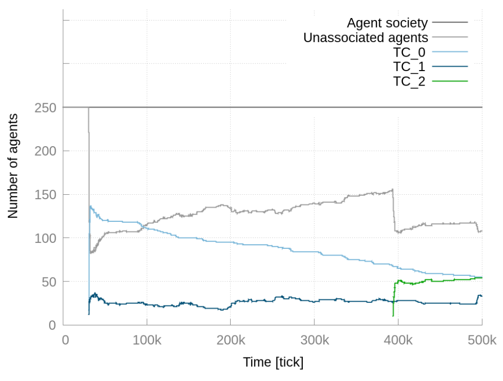
In these two experiments, the composition of the agent society has been changed, such that 20 % (30 % respectively) of the agents were defecting (of type DAA, see Sec. 5.3.1), hence producing false WU results with a certain probability (here 0.2). This behaviour was static among the agents, such that they were not influenced by incentives like organisation membership. The motivation to conduct these experiments was the measurement of the risk of the *Transparent WU Validation* benefit: By abandoning the safety means replication, organisation members could accept false WU results as valid. Additionally, the agents were colluding in the production of false WU results, such that majority voting could also be overcome. The impact of the defecting behaviour was quantified with the metric *accuracy*, as defined in appendix D.1.4. Additionally, the identification of such defecting behaviours was aided by negative trust assessments, consequently reducing the reputation of these agents. This resulted in a reduced number of suited workers in the system by 20% and 30% respectively. An additional motivation for these experiments was therefore the measurement of the impact on the *speedup* resulting from this reduced number of cooperative workers.

The results summarised in Tab. 5.4 show that *Clans* and TCs achieve a comparable speedup for the case of defecting agents. Both approaches outperform the *iTC*-approach. However, the experiments also show that while the speedup of *iTCs* and *Clans* is hardly reduced compared to the undisturbed case, the TC-approach achieves only 83.71% (20% DAA), and 78.93% (30% DAA) respectively of the speedup in the former case. Additionally, both experiments show that the explicit forms of organisation (as opposed to the *iTC*-approach) increase the accuracy in the system. Here, TCs are outperformed by the *Clan*-approach. Finally, the results demonstrate that in case of TC-application the average relative utility of the organisations is higher than in the application of *Clans*. In consequence, this means that there is a stronger incentive for TC-membership in such system states than for *Clan*-membership.

In the following, the reasons for these results are examined by analysis of a single exemplary run from the 20%-DAA experiment. The formation of *Clans* and TCs differs in the required criteria:



(a) Clan



(b) TC

Figure 5.16: Comparison of the number of agents associated to a Clan/TC in case of the presence of 20% DAA-agents: These graphs show how the amount of organisation members decreases in comparison to the experiment with an undisturbed agent society (see Fig. 5.14). Also, the amount of formed *Clans* decreases, and the formation of *Clans* and TCs is initiated later. Additionally, the TC graph shows how the largest operating TC (*TC_0*) constantly loses members. This is due to the exclusion of defecting members which are identified during the course of the operation phase.

While both approaches require a high trustworthiness among the potential members, *Clans* require additionally the fulfilment of at least one secondary criterion (see summary in Sec. 5.3.1). As depicted in Fig. 5.17, this results in a slower *Clan*-formation as compared to the TC-formation. This time is well invested in case of defecting agents: As described above, the probability to produce a false WU result amounts to 0.2. In consequence, this means that the defecting DAA-agents can be identified only after an extended period of time. Each forming *Clan* is comprised of the 25 (see *preferred clan size* described in Sec. 5.3.1) most trusted agents known to the initiator. Subsequent *Clans* are formed when the next set of agents reach the formation criteria, mainly determined by the time they reach a high trustworthiness. DAA-agents never reach that threshold as their defecting behaviour is identified through the application of majority voting resulting in lower trustworthiness values. Consequently, when *Clans* are applied, DAA-agents have a low chance of becoming organisation members and reducing the accuracy of other members.

In contrast, TCs have an unconstrained size (in case of the applied strategy configuration) and lower formation thresholds. This results in earlier TC formation and the inclusion of DAA-agents as members due to the lack of negative experiences with these agents. However, TCs are also robust due to the regulation of the TCM and allow for the exclusion of uncooperative members. This is depicted in Fig. 5.16b where mainly the largest TC (TC_0) is affected by this phenomenon: Shortly after its formation the TC comprises 136 members. However, many of these members are DAA-agents. With time passing, these are subsequently identified and excluded from the TC. This shows as a decline in the number of members throughout the run duration. At the end of the run, 108 agents are unassociated and 142 are members. The 20% EGO-/FRE-agents and max. additional 20% DAA-agents (some are of both types) sum up to max. 100 uncooperative agents. This supports the claim that in the long term, TCs are comprised only of cooperative agents.

Finally, the speedup decrease of the TC approach (as compared to the undisturbed experiment) is a result of the membership of DAA-agents and their subsequent identification and exclusion. This process is associated with a higher overhead due to the situation-aware application of WU replication among TC-members (as described in Sec. 5.2.1) and consequently lower throughput and speedup. The process has also an influence on the accuracy achieved by TC application: For this experiment, the accuracy values for TCs are always between the values of the *iTC*- and the *Clan*-approach (see Tab. 5.4). This is a result of the duration of being exposed to the DAA-agents that produce false WU results. While in the *iTC*-approach the DAA-agents can hardly be avoided, *TCs* lead to their omission as workers in a lengthy process. This is contrasted by the *Clan* mechanism discussed above: Here DAA-agents have only small chances of becoming *Clan*-members. As *Clan*-members search for workers almost exclusively among fellow members, they are least exposed to this threat.

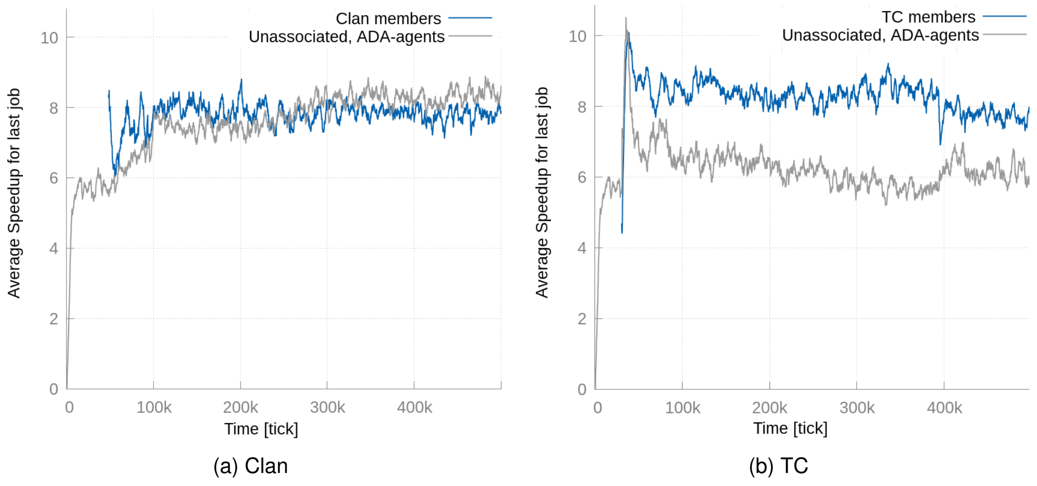


Figure 5.17: Comparison of the average speedup for the last job of Clan-/TC-members and unassociated agents in case of 20% defecting agents. While the speedup of *Clan*/TC-members is approx. at the same level in this experiment (see also Tab. 5.4), the average speedup of unassociated agents is lower in case of TC application. As unassociated agents in this scenario are mostly defecting agents, this lower speedup is intended to serve as incentive for cooperative behaviour and eventually TC membership.

The evaluation results summarised in Tab. 5.4 show that *Clans* and TCs achieve a comparable average speedup. In the following, the contribution to the average speedup of unassociated agents and organisation members is examined. The graphs depicted in Fig. 5.17 show the average speedup of unassociated ADA-agents (including defecting agents) in comparison to the average speedup of TC members. Here the results for *Clans* and TCs vary more than in the average speedup: While the application of *Clans* results in a comparable speedup for members and unassociated agents, the application of TCs leads to a significantly higher member speedup. This has the following reasons: *Clan*-members search for workers exclusively within their organisation to reduce the costs of finding cooperative interaction partners (cf. [118]). Due to the kinship motivation, they have a very high probability of finding a worker for each WU among the other members. This leads to a significant reduction of the workload within the group of unassociated agents as these submit their own WUs both to *Clan*-members, as well as among themselves, but do not receive processing requests from *Clan*-members. As examined for the undisturbed experiment, a reduction in the workload obviously increases the throughput and thus speedup of the agents. In *Clans*, unassociated agents hence profit from the inbound interactions of *Clan*-members. When Trusted Communities are applied, unassociated agents do evidently not profit from the operation of TCs, resulting in a significantly lower speedup for them. This is due to the different submitter behaviours: TC members choose workers strictly based on performance and a trust threshold (see 5.2.1). This means that they also choose workers from other TCs and from the group of unassociated agents when this promises a shorter processing duration although it involves the necessity to replicate the WUs. The effect is that, the workload in the group of unassociated agents is reduced less in comparison to *Clan*-application. In addition, TC-members refuse to process WUs from non-members whenever they are also requested to process WUs from fellow members. The unassociated agents hence have in general fewer available workers to choose from. In summary, the operation of TCs does not provide any advantage for unassociated agents. Instead, the higher speedup of members provides an incentive to seek TC membership.

In comparison of the organisation forms, these phenomena are the reason for the higher average relative organisation utility of TCs (as depicted in Tab. 5.4). In effect, this increased utility also contributes to the benefits of TC application by providing a strong incentive to agents to abstain from uncooperative behaviour in order to become TC member. As the behaviour of DAA-agents has been applied statically in these experiments, the incentive had however no measurable effect. In a more complete consideration, especially with rational and adaptive DAA-agents, this additional incentive mechanism is expected to lead to a greater system performance.

Finally, the graphs depicted in Fig. 5.18 show exemplary TC results for the case of 30% DAA-agents in the agent society. In this scenario, the increased amount of uncooperative agents had a negative effect on the average speedup and especially on the accuracy (see Tab. 5.4). While the (above discussed) phenomena that were responsible for these results are the same as for the scenario with 20%, these additional graphs show intensified characteristics: The TC composition shows that here all operating TCs suffered from DAA-agents that had to be identified and excluded over time. In addition, the speedup comparison between unassociated agents and TC members for this scenario is depicted in Fig. 5.18b. This graph reveals the same quality of speedup divergence between the groups of agents as the graph for 20% DAA-agents depicted in Fig. 5.17b. However, the speedup achieved in this case is lower for both groups of agents.

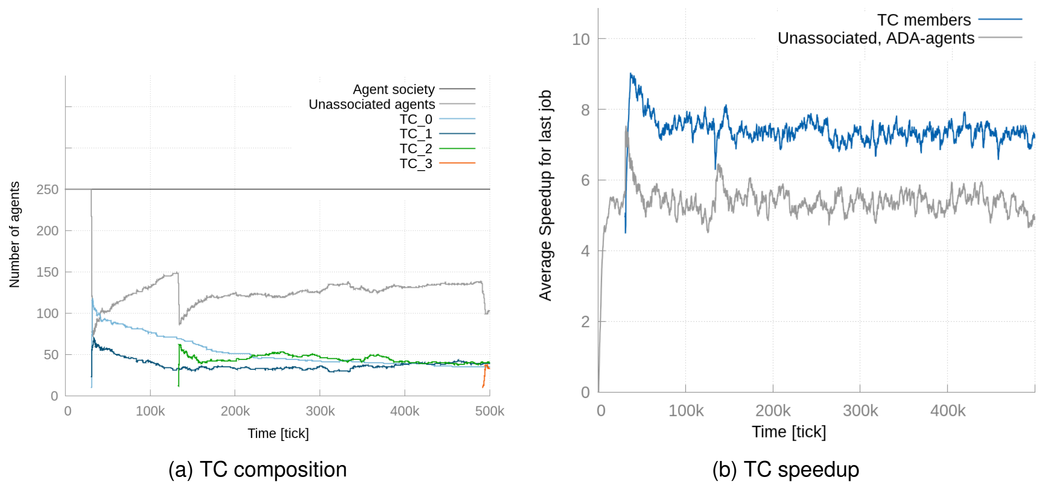


Figure 5.18: TC composition and speedup for the experiment with 30% defecting agents. These graphs show the impact of an increased amount of defecting agents: In Fig. *(a)*, the phenonmenon of continuous identification and exclusion of defecting TC members, as shown for a single TC in Fig. 5.16b, manifests here for the majority of operating TCs. However, the number of unassociated agents increases unproportionally. In Fig.*(b)*, the speedup of unassociated agents is contrasted with the speedup of members. Here, the speedup of unassociated agents is further decreased in comparison to the experiment with 20% defecting agents. TC member speedup remains approx. equal.

In summary, these experiments show that the application of either of the two explicit organisation forms *Clan* and TC increases the speedup (to a comparable degree) in an open system that suffers from defecting agents. Due to the differences in these two approaches, the benefits of their application (in addition to the speedup increase) vary: *Clans* achieve a higher accuracy, while TCs achieve a higher relative organisation utility and hence provide an incentive to abstain from the defecting behaviour.

Experiments With TM-Exploiting Agents

These two experiments evaluated the susceptibility of the organisation approaches towards TM-exploiting agents (of type CAA). Here, 20 % (30 % respectively) of the agent society were of that type, meaning that they refused cooperation (freeriding) once their reputation reached the threshold 0.5, and remained in this state until their reputation reached the lower threshold 0.1. The presence of such agents has several implications: Firstly, the number of willing and suited workers is reduced due to the freeriding phases of the CAA-agents, decreasing the achievable speedup in the system. Secondly, this also means that the number of suited organisation members is decreased and the organisations are expected to be fewer/smaller than in the undisturbed case. Finally, as with the defecting agents described above, CAA-agents are either penalised for their uncooperative behaviour by the organisation approaches or not. This penalty shows in terms of a reduced speedup as compared to the speedup of ADA-agents and is an incentive for increased cooperation. In the following, the results depicted in Tab. 5.4 are first set in relation and then examined with the help of graphs from an exemplary experimental run.

The results show the expected decrease in the performance as compared to the undisturbed experiment: With 20% CAA-agents, *iTCs* achieve 91,70%, *Clans* 96,03%, and TCs 95,42% of the former speedup. In case of 30% CAA-agents, the results show 86,60% (*iTC*), 88,51% (*Clans*), and 86,52% (TCs) of the undisturbed speedup. The amount of the speedup decrease is similar for the three approaches, consequently TCs outperform the other approaches as in the undisturbed experiment. Here, the speedup benefit when applying TCs, as opposed to applying *Clans*, amounts to 16,26% (20% CAA) and respectively 12,97% (30% CAA). In addition, Tab. 5.4 shows that indeed the average number of operating organisations has decreased due to the reduction in cooperative ADA-agents. The benefit for organisation has however increased (significantly when TCs are applied).

In the two graphs depicted by Fig. 5.20, the association to *Clans* and TCs for a regular run from the 20% CAA-agents experiment is shown. The graphs visualise for each of the 250 agents in the agent society if, and for how long, it has been associated to an organisation. The begin of an association duration is initiated by either formation or invitation to the organisation in both approaches. Complementary, the end of an association duration is caused by the dissolution of the organisation, members leaving it due to lack of benefit and, only for TCs, members being excluded from the organisation by the TCM. A comparison of the two diagrams reveals a similar phenomenon as seen in the undisturbed experiment: *Clan*-formation is slower than TC-formation, and more ADA-agents remain unsassociated. In addition, CAA-agents do not gain *Clan*-membership (with one exception). This has the following reasons: As in the experiment with defecting agents, the ADA-agents always have a higher trustworthiness than the (partially uncooperative) CAA-agents. When *Clans* are formed, these agents are therefore not considered suited as members as long as there are sufficiently ADA-agents to reach the preferred clan size. Finally, in the state when most ADA-agents are associated to *Clans*, the remaining group of agents does not include enough trustworthy agents to reach the preferred clan size, hence no additional *Clan* is formed. The lower of the two graphs shown by Fig. 5.20 depicts the same run for the application of TCs: Here the number of ADA-agents associated to an organisation is higher than in the *Clan* run. In addition, some of the CAA-agents are also associated to TCs. While this may seem as a disadvantage at first, the reasons for this association characteristics are that in fact CAA-agents are not constantly uncooperative. Instead, the freeriding phases depend on their reputation. Due to the varying processing capabilities, not all CAA-agents are equally attractive as workers. In effect, CAA-agents with weaker capabilities hardly reach a sufficiently high reputation to start freeriding. On the other hand, even if CAA-agents freeride, there are some ADA-agents that have sent processing requests to single CAA-agents only at cooperative time intervals. As a result, they have a strong direct trust relationship to these agents and propose to invite them as TC members (see the *Active TC Expansion* strategies in Sec. 4.6.6). In contrast to *Clans*, where the preferred clan size does not permit the expansion of the *Clan* as long as other ADA-agents do not leave it, a TC makes such agents members. Finally, the graph also shows that most of these accepted CAA-agents have only a short association duration. This demonstrates the regulatory effect of TCs: When these agents are in the freeriding phase and reject to process WUs from members, the *Member Control* strategies help to identify and punish this behaviour. Ultimately, the CAA-agents are exluded when the rejections are too numerous. However, the application of *forgiveness* allows them to become members again at a later time which also shows for some agents in the graph.

These association patterns have the following effect on the speedup: More associated agents

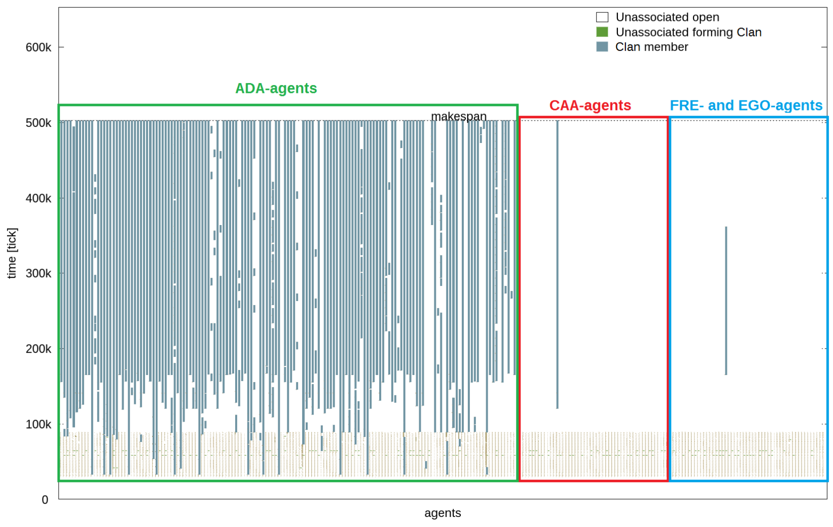


Figure 5.19: Association times diagram for *Clans* in the presence of CAA-agents: The TM-exploiting CAA-agents are not considered during *Clan*-formation as they do not reach the necessary trustworthiness value. Later during the simulation, CAA-agents are not invited to become *Clan*-members (by members with positive interaction histories) as the operating *Clans* already have the preferred size. The majority of ADA-agents is organised in *Clans* throughout the simulation run while EGO- and FRE-agents are isolated and have no access to *Clans* (with one exception).

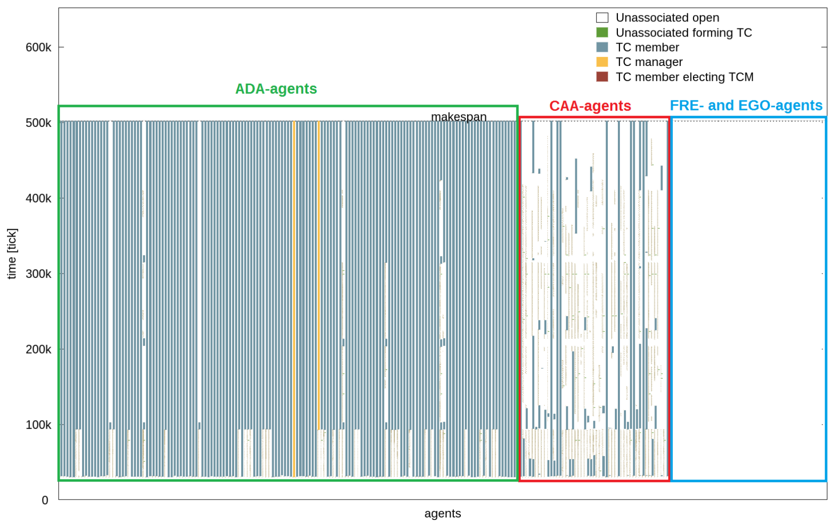
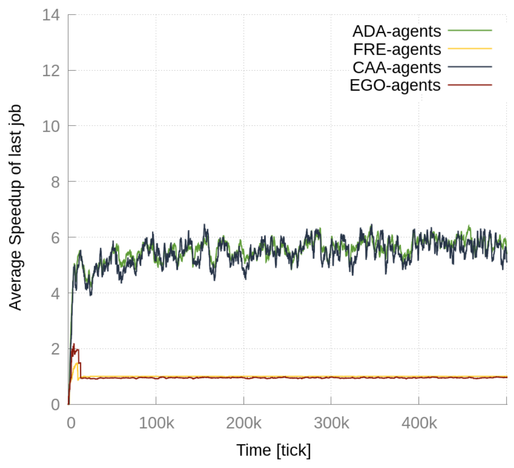


Figure 5.20: Association times diagram for TCs in the presence of CAA-agents: In contrast to *Clan* association (as depicted above), the size of TCs is not constrained, thus CAA-agents can become TC-members in case there are other agents with a positive interaction history. This allows to take advantage of the worker power of these agents as long as they are not in the freeriding state. TCs however adapt to the state change and exclude CAA-agents that frequently refuse cooperation based on the regulation via *Member Control Strategies*. The application of forgiveness however allows CAA-agents to gain membership again.

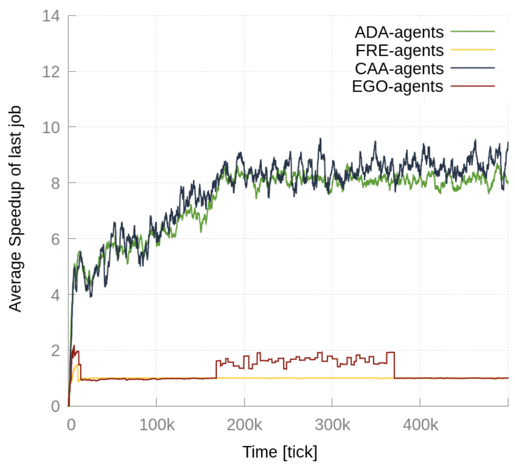
and larger organisations allow for more agents taking advantage of the *Transparent WU Validation* benefit. TCs hence outperform *Clans* here, because not only do they include more ADA-agents as members, but also adapt to the behaviour of CAA-agents. When these agents are uncooperative, they are excluded from the TCs, but for as long as they cooperate, they are seen as valuable members that further increase the relative utility of the TCs.

Finally, the graphs depicted in Fig. 5.21 show that TCs are the only organisation form in this comparison that establishes an incentive for cooperation: As seen with defecting agents, when TCs are applied the speedup of cooperative agents is significantly higher than the speedup of uncooperative agents (see Fig. 5.21c). The reasons for this phenomenon are similar to the reasons examined in case of defecting agents: The formation of *Clans* leads to a reduction of the workload in the group of unassociated agents, and consequently to an improvement of their speedup (see Fig. 5.21b). Note especially how the speedup of the CAA-agents increases when *Clans* are applied, as opposed to the application of *iTCs* (see Fig. 5.21a). The ADA-agents also benefit from *Clan*-application because of the *Transparent WU Validation* and the *kinship motivation*. However, due to the scalability-motivated approach of finding workers only among fellow *Clan*-members and the constrained size of the *Clans*, the extent of these benefits is limited. In comparison, less, but larger TCs are formed in these experiments (see Tab. 5.4). This is not only due to the varying forming criteria (as discussed for the previous experiments), but also due to the inclusion of CAA-agents as TC members. In effect, TC members have a higher gain from the *Organisation Benefit* strategies. On the other hand, CAA-agents that remain unassociated when TCs are applied do not profit from a significant workload reduction as is the case for *Clan*-application. As described for the case of defecting agents, this is due to the performance-based submitter behaviour of TC members. This behaviour results in processing requests towards CAA-agents from TC members. Additionally, unassociated agents have a lower probability to get their WUs processed from TC members as these reject those requests whenever fellow members also pose processing requests. In effect, when TCs are applied, the unassociated CAA-agents are forced to let their WUs be processed within a relatively small group of workers who in addition express freeriding behaviour (rejecting requests) whenever their reputation is high.

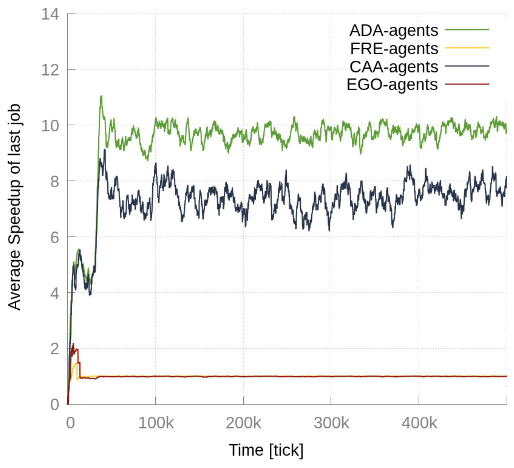
In summary, the experiments with TM-exploiting agents show that not only do TCs achieve the highest speedup in this scenario, but also that they establish an incentive for abstaining from uncooperative behaviour. In that, the speedup differences between the three approaches are similar to the differences in the undisturbed experiment.



(a) iTC



(b) Clan



(c) TC

Figure 5.21: Comparison of the average speedup for the last job of agent stereotypes for the case of 20% TM-exploiting (CAA-) agents and iTC/Clan/TC application. While the *iTC*- and *Clan*- approaches allow the CAA-agents to have a high speedup, TCs penalise their uncooperative behaviour. This shows in the lower speedup of these agents and is an incentive to act cooperatively. In addition, the speedup of TC-associated agents is higher than the speedup of *iTC*-agents and *Clan*-members.

5.3.3 Robustness Evaluation

Apart from the increase of a hosting systems' performance, the second major goal in the development of Trusted Communities was the increase of the systems' robustness towards disturbances. While the mitigation of the effects of minor disturbances resulting from uncooperative behaviour of **single agents** (TM-exploiting, egoistic, freeriding and defecting behaviour) through the application of TCs has been examined in the previous section, the evaluations here were aimed at substantial **attacks of colluding agents**. For this, the following experimental setup has been used:

The experiment used non-validating DG applications. The agent society was then composed as follows: Initially, the composition was consisted of 250 agents with 70% ADA-agents, 10% CAA-agents, 10% FRE-agents and 10% EGO-agents, with an additional, random 10% of agents with the defecting trait (DAA-agents). This composition hence included agents with each type of uncooperative behaviour from the previous performance experiment, albeit in smaller numbers. After the time of 100,000 ticks a collusion attack was initiated: Here, a group of FRE-agents entered the system and started to send processing requests for their DG jobs. These agents colluded in that they knew each other and did not send request among each other, such that they did not decrease their trustworthiness. As these attackers entered the system, they had an initial reputation value and could therefore not be detected immediately. Such a disturbance was quantified by comparing the amount of attackers with the size of the agent society before the attack. This is referred to as the *disturbance size (ds)* in the following. As the agent society size before the attack was always 250 agents, a disturbance size of 0.5 here means that 125 attacking agents entered the system. The following evaluations examined the hosting system in the event of disturbances with the size ds. 0.1 to ds. 1.0 for 25 different simulation seeds and each organisation option (iTC, Clan, TC), hence in a series of 750 experimental runs. These were evaluated by using the robustness metrics *(speedup) collapse fraction*, *relative recovery costs*, and *recovery duration* (defined in appendix D.2).

In the following, the course of such a single experiment run is analysed. This is concluded by the presentation and discussion of the evaluation results for the entire experiment.

The TDG system under a collusion attack

As described earlier in this thesis (e.g. in Sec. 5.1.6), the control of a system as the TDG by the application of Trust Management must account for the event of an emergent *trust breakdown* to be truly robust. By starting a collusion attack, a trust breakdown is provoked in the system: FRE-agents behave as *producers* of WUs, but not as *consumers*. Hence, the system is flooded with additional processing requests and agents trying to build up a reputation accept these processing requests which increase their workload. After a certain number of accepted WU processing requests, the agents stop accepting further requests and receive negative ratings from attackers, as well as initial agents alike. These processing request rejection ratings continue as long as the accepted and queued WUs of the agents are processed. Additionally, these ratings lead to a decreased reputation of the agents and consequently a system state where the reputation of most agents is so low that the system becomes *paralysed* (see Sec. 5.1.6).

This is depicted in Fig. 5.22 for a disturbance size of ds. 0.6 (150 attackers). This graph shows how the reputation of the various agent stereotypes behaves right after the attack at tick 100,000. While the initially high reputation of ADA-agents collapses substantially, the reputation of the CAA-

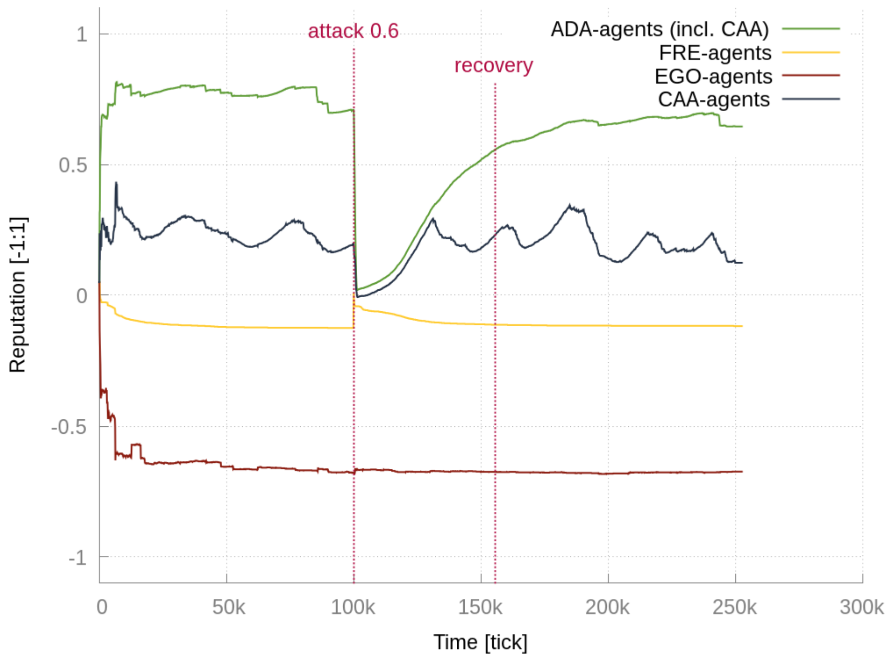


Figure 5.22: Reputation for agent stereotypes for the disturbance size 0.6. While the average reputation of ADA- and CAA-agents varies due to their behaviour, both suffer from a substantial decline in the event of the attack. With the average reputation approaching 0, trust within the agent society breaks down. Following the attack, the reputation recovers and eventually reaches an equivalent value to the time before the attack. Note that the end of the recovery is based on the speedup which recovers quicker than the reputation.

agents is not very high initially and hence the collapse is not that strong. However, the typical course of the CAA-agent reputation, generated by their TM-exploiting behaviour is broken through the attack. Additionally, the reputation of EGO-agents is not affected by the attack as these agents are already isolated and are not perceived as competent workers by the attackers in the first place. Also, their reputation is already very low even before the attack. Finally, the reputation of FRE-agents increases in the event of the attack. This is due to the fact that the attackers are FRE-agents entering the system with an initial reputation higher than the average reputation of the isolated FRE-agents before the attack. The graph also shows how the agent society recovers from the attack and re-establishes the pre-attack reputation values after a recovery time. The duration of the recovery time is however based on the influence of the attack on the *speedup* and not on the reputation which is why the end of the recovery cannot directly be interpreted from the course of the reputation functions in the graph.

So far, the emergence of a trust breakdown by the collusion attack has been examined. The system enters a state where paralysis prevails due to the low reputation. This is best depicted by the worker perspective presented for the *iTC*-approach in Fig. 5.23. Here, the following phenomenon is demonstrated: The collusion attack and the resulting trust breakdown lead to a phase where ADA-agents are isolated and forced to process their WUs on their own (red margin after the attack) hence do not benefit from the TDG participation. This is where the benefits of TC application can be best

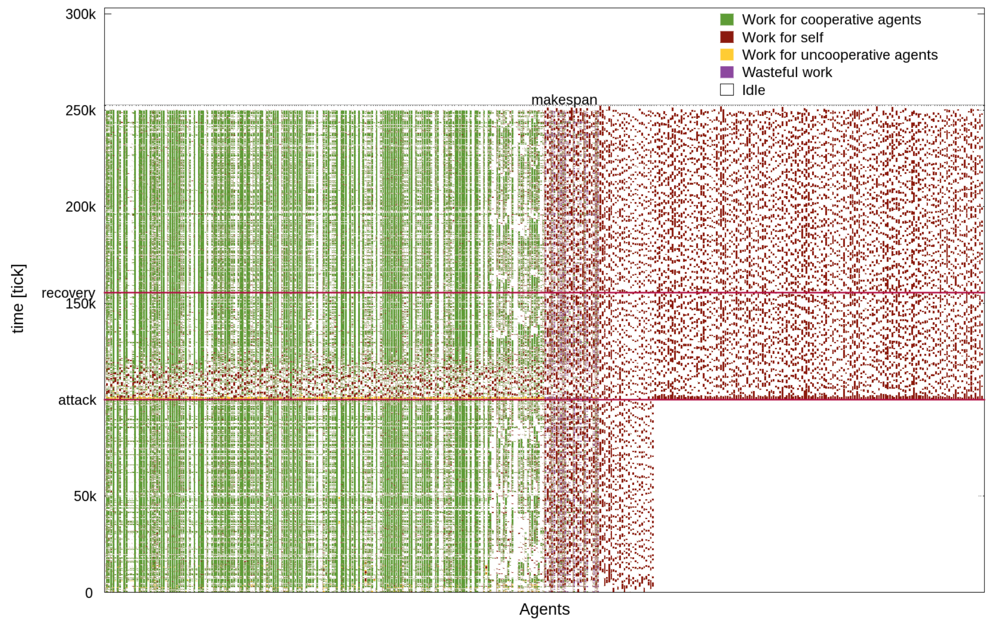


Figure 5.23: Worker diagram for disturbance size 0.6 and *iTC* application: The impact of the attack can clearly be recognized by the appearance of a period of massive owner processing of WUs among ADA-agents (left side). Only after trust among the agents is slowly restored, are WUs processed by other workers again.

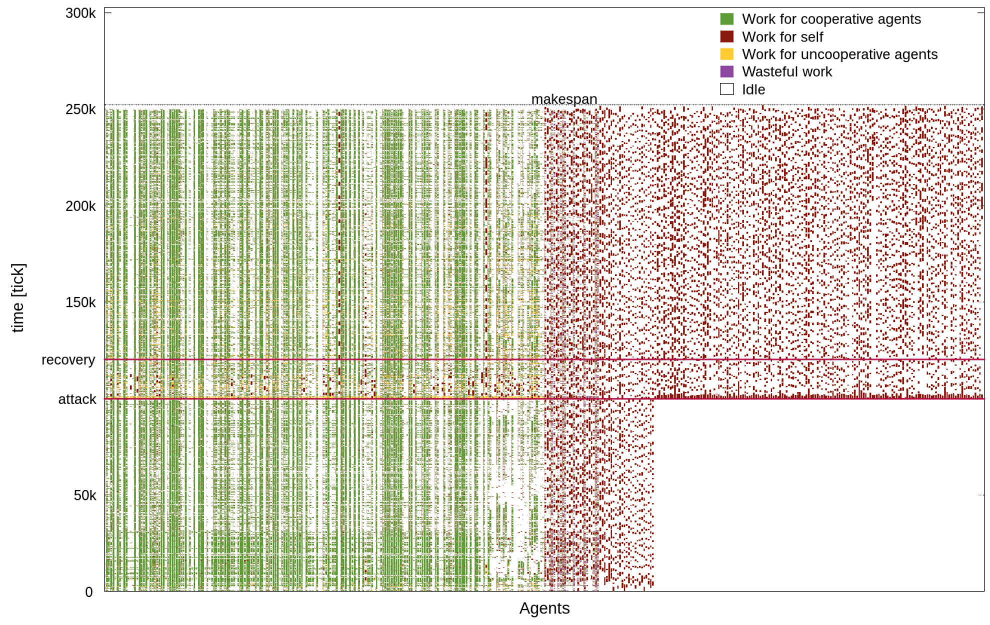


Figure 5.24: Worker diagram for disturbance size 0.6 and TCs: In contrast to the application of *iTCs*, the impact of the attack on ADA-agents (left side) is less significant. While self-processing of WUs also appears after the attack, it is limited to unassociated agents and the duration is substantially shorter.

demonstrated: As depicted by the lower of the two graphs, this paralysis substantially less affects a system where TCs are allowed to form. Most of the agents (TC members) continue to cooperate (green phases) despite the attack, while single agents (not members of a TC at the time of the attack) are isolated. However, their isolation does not prevail as long as in the *iTC*-case. The reason for the continued inbound TC member cooperation lies in the independence of the trustworthiness: Members accept processing requests for each other due to the TC membership incentive which is not affected by the collusion attack. Only the cooperation with non-members is reduced by the attack, such that still the attack has a negative influence of the TDG performance.

Until here, the effect of a FRE-agent collusion attack has been shown to lead to a trust breakdown. In consequence, a period of paralysis has been shown to prevail, which manifests as the isolation of a majority of agents (in the *iTC*-approach) or unassociated agents (for TC application). The focus in the evaluation of the robustness is however how these phenomena affect the *performance*, i.e. the *speedup*, of the agents in the TDG. That is, how strong the *speedup* collapses, how long it takes to recover to the value before the attack, and how costly that recovery has been (see appendix D.2). The results for these metrics are depicted as a comparison of the *iTC*-, the *Clan*- and the *TC*-approach in Fig. 5.25 for the experimental run discussed here. The general structure of the graphs is as follows: The green curve shows the *speedup* over time (speedup of last job as defined in Eq. D.20). The blue curve shows how the speedup has been in the equivalent experimental run (same seed) without the disturbance, and the red area shows the recovery costs.

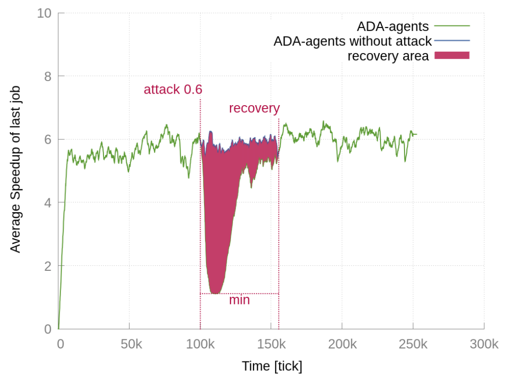
In the comparison, the speedup of the *iTC*- and the *Clan*-approach is in general lower than the speedup when applying TCs (this has been analysed in the performance evaluation in Sec. 5.3.2). With respect to the attack, the following additional results are demonstrated by the graph:

- The *(speedup) collapse fraction* is significantly lower when using TCs: 0.201 compared to 0.592 (Clans) and 0.814 (iTC).

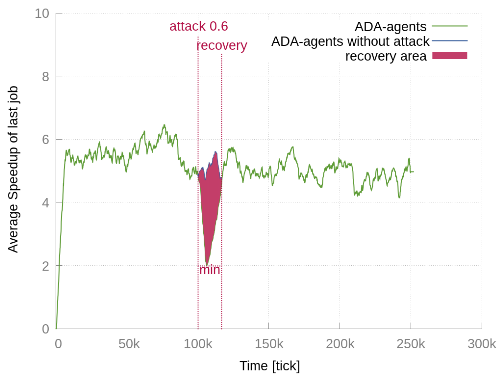
- While all three approaches allow for the recovery from the attack, TCs have a substantially shorter *recovery duration* than the *iTC*-approach: 20410 ticks compared to 55547 ticks (iTC). Although *Clans* have an even short recovery duration in this particular example (16736 ticks), in average, *TCs* outperform *Clans* in this metric, too (see Tab. 5.5).

- The *relative recovery costs* are significantly lower when TCs are applied: 0.152 compared to 0.343 (iTC) and 0.351 (Clans).

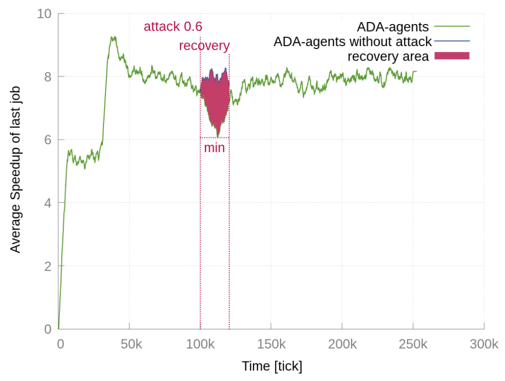
The reasons for this high robustness of the TC approach lies within the continued cooperation despite the attack and the resulting trust breakdown. Such cooperation requires explicit organisation between members, which is why the robustness of the *iTC*-approach is low. However organising agents alone (as is done in *Clans*) does not suffice: Unlike TC members, Clan members still rely on correct trustworthiness information in their decision making. A trust breakdown hence negatively impacts *Clan* member cooperation. Also, members leave a *Clan* when their overall trust in the other members becomes too low, disregarding that this decrease can result from an attack. As depicted in Fig. 5.26a, the trust breakdown eventually leads to a breakdown of Clan performance as members leave the *Clans* until finally the *Clans* are dissolved. Only after the recovery, when trust among the agents has reached a high value again, do new *Clans* form. In contrast, Fig. 5.26b shows how TCs mitigate the effects of the attack: The most significant difference is seen between unassociated



(a) iTC



(b) Clan

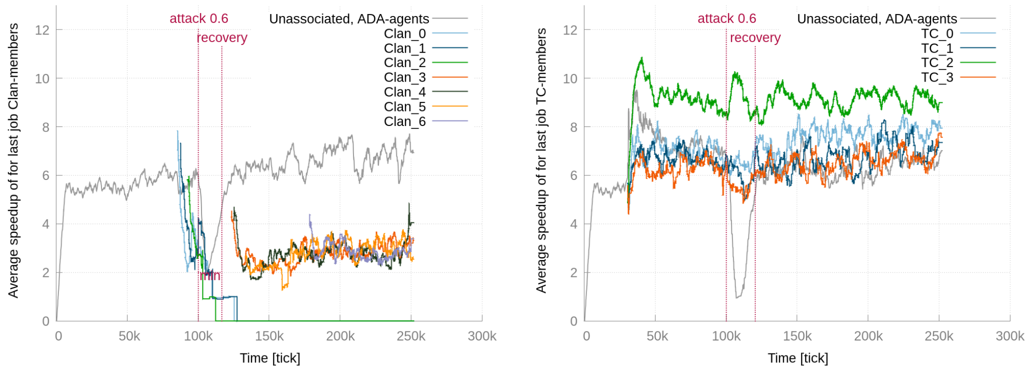


(c) TC

Figure 5.25: Speedup comparison for the disturbance size 0.6 for *iTC*-/*Clan*-/TC-application: The attack causes the speedup of ADA-agents to collapse for all three organisation forms. The greatest impact is seen when applying the *iTC*-approach, hence when no explicit form of organisation is used. *Clan*- and TC-application show comparable recovery durations, while the speedup collapse fraction is more substantial for *Clans*. The difference grows for increased disturbance sizes, as presented in Tab. 5.5.

agents and TC members. Here, the speedup of the unassociated agents collapses to a very low value due to the attack. In comparison, the speedup of the single TCs is not significantly affected. In case of *TC_2*, the speedup is even increased due to the attack. These phenomena have the following reasons: TC members continue to cooperate with other members of their own TC. The difference to the phase before the attack is however, that TC members do not accept processing requests from non-members. While the largest *TC_2* profits from this as the members' capacity for inbound jobs is higher, the smaller TCs suffer from not being able to submit WUs to agents organised in other TCs. It is hence in particular the robustness of the large TC that has a positive influence of the robustness of the TDG in total.

This completes the consideration of a single comparative experimental run. In the following, the evaluation results of the complete experiment are presented and discussed.



(a) Members of single Clans vs. unassociated agents (b) Members of single TCs vs. unassociated agents

Figure 5.26: Speedup comparison for disturbance size 0.6 according to TC/Clan association: When *Clans* are applied, the attack causes the speedup of *Clan*-members and unassociated agents alike to collapse. This eventually leads to the dissolution of the *Clans*. In contrast, TC application allows the members to maintain their speedup despite the attack and only impacts unassociated agents.

Collusion attacks in the TDG and agent organisations

The following table Tab. 5.5 contains the results for the robustness evaluation of the three approaches *iTC*, *Clans*, and Trusted Communities. The experiment contained a series of 10 disturbance sizes from ds. 0.1 to ds. 1.0, hence for 25 to 250 attackers for an initial agent society size of 250 agents. The results are based on 25 runs with unique seeds for each disturbance size. Each organisation option was evaluated under each disturbance size and for the same 25 seeds. The results contained in the table are based on the robustness metrics defined in appendix D.2. For all three metrics lower values imply higher robustness. The best result for each disturbance size, among the three organisation options, is depicted in bold. The textual representation is followed by a graphical representation (Figures: 5.27, 5.28 and 5.29) for each metric. It also contains the *min* and *max* values not contained in the table.

Conditions	**Metrics** (s_{N-1})	**iTC**	**Clan**	**TC**
ds. 0.1	Recovery duration	5755 (1321)	28476 (47837)	**5318** (1082)
	Rel. recovery costs	0.069 (0.020)	0.069 (0.033)	**0.035** (0.020)
	Speedup collapse	0.057 (0.046)	0.122 (0.113)	**0.026** (0.031)
ds. 0.2	Recovery duration	11096 (3848)	25377 (40097)	**5640** (1046)
	Rel. recovery costs	0.130 (0.043)	0.123 (0.037)	**0.053** (0.025)
	Speedup collapse	0.166 (0.064)	0.205 (0.099)	**0.047** (0.037)

Conditions	**Metrics (s_{N-1})**	**iTC**	**Clan**	**TC**
ds. 0.3	Recovery duration	16918 (3895)	24412 (37948)	**6300** (1520)
	Rel. recovery costs	0.191 (0.059)	0.169 (0.048)	**0.061** (0.026)
	Speedup collapse	0.284 (0.080)	0.287 (0.094)	**0.061** (0.040)
ds. 0.4	Recovery duration	20730 (6943)	15367 (4695)	**8353** (4942)
	Rel. recovery costs	0.249 (0.084)	0.241 (0.070)	**0.070** (0.029)
	Speedup collapse	0.392 (0.139)	0.395 (0.129)	**0.080** (0.053)
ds. 0.5	Recovery duration	27923 (10091)	18724 (10402)	**12536** (7367)
	Rel. recovery costs	0.298 (0.082)	0.268 (0.061)	**0.077** (0.032)
	Speedup collapse	0.505 (0.166)	0.451 (0.139)	**0.094** (0.055)
ds. 0.6	Recovery duration	32176 (9214)	29147 (30796)	**14526** (10528)
	Rel. recovery costs	0.360 (0.088)	0.291 (0.070)	**0.082** (0.035)
	Speedup collapse	0.611 (0.159)	0.538 (0.130)	**0.111** (0.060)
ds. 0.7	Recovery duration	38619 (8861)	31093 (27556)	**14501** (11731)
	Rel. recovery costs	0.385 (0.076)	0.336 (0.062)	**0.078** (0.037)
	Speedup collapse	0.692 (0.130)	0.627 (0.105)	**0.101** (0.065)
ds. 0.8	Recovery duration	39075 (7835)	34731 (26865)	**19616** (11392)
	Rel. recovery costs	0.410 (0.085)	0.344 (0.071)	**0.095** (0.040)
	Speedup collapse	0.728 (0.123)	0.660 (0.083)	**0.137** (0.061)
ds. 0.9	Recovery duration	44534 (7137)	37878 (34825)	**20418** (13043)
	Rel. recovery costs	0.404 (0.074)	0.364 (0.084)	**0.092** (0.041)
	Speedup collapse	0.773 (0.074)	0.687 (0.078)	**0.132** (0.060)
ds. 1.0	Recovery duration	44113 (5298)	35139 (26976)	**20028** (11764)
	Rel. recovery costs	0.426 (0.053)	0.375 (0.085)	**0.092** (0.044)
	Speedup collapse	0.799 (0.030)	0.692 (0.075)	**0.133** (0.066)

Table 5.5: Robustness evaluation of the TC approach for different disturbance sizes (ds.).

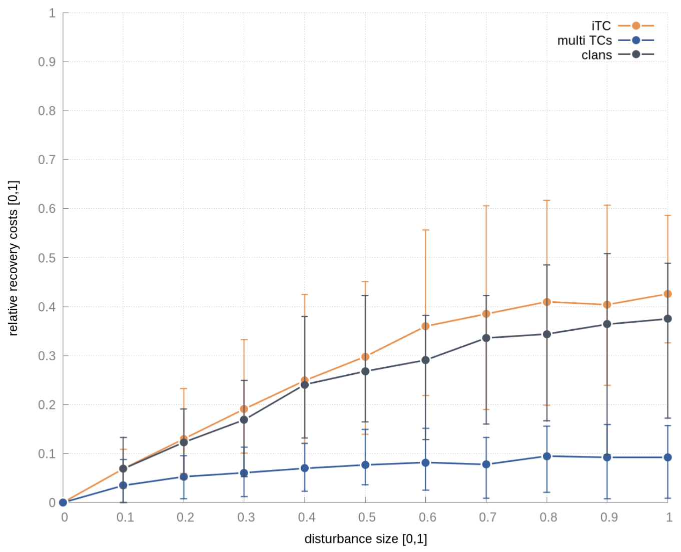


Figure 5.27: Relative recovery costs at collusion attacks: Lower values are better. As expected, the recovery costs are roughly proportional to the disturbance size. The graph also shows that TCs are substantially more robust compared to *iTCs*, as well to *Clans*. The difference increases with the disturbance size.

The nature of the robustness of each of the organisation approaches has been examined in detail for the case of a disturbance size of ds. 0.6. The experimental results depicted in the table and figures complement this evaluation by providing values for various other disturbance sizes. The general quality of the metric results reflects that the expected performance is proportionally dependent on the disturbance sizes. As discussed throughout the thesis, the robustness of an organisation approach is here understood as its ability to recover the system from a disturbance, such that eventually the performance of the system from before the attack is reached. The results depicted in the figures 5.27, 5.28 and 5.29 show that all three approaches are robust to some degree. This is founded in the course of the metric functions: All functions show a linear or sub-linear behaviour for the increasing disturbance sizes. An approach not providing any robustness would result in an exponential behaviour. However, it can be clearly seen that the robustness of the three approaches varies largely. The *iTC*-approach is in general outperformed by the *Clan*-approach, albeit only for disturbance sizes greater ds. 0.4 for two out of three metrics. Both approaches are however of equal robustness compared to the robustness of the TC approach which significantly outperforms the other approaches in all three metrics. The greatest difference shows in the speedup collapse fraction: TCs have an average value of 0.13 in the worst case (ds. 1.0), while *Clans* and *iTCs* have an average value of 0.69 and 0.8 respectively. The smallest difference shows in case of the recovery duration. This metric is however dependent on a parameterisation of the exact definition for the end of the recovery time. Depending on the set parameter, the difference can be greater.

In sum, the TC approach makes the TDG significantly more robust than the other two state-of-the-art approaches. The difference between the achieved metric results of the three approaches is also larger than in the performance evaluation, where TCs also outperformed the other approaches albeit not in each case (see experiment with defecting agents in Sec. 5.3.2).

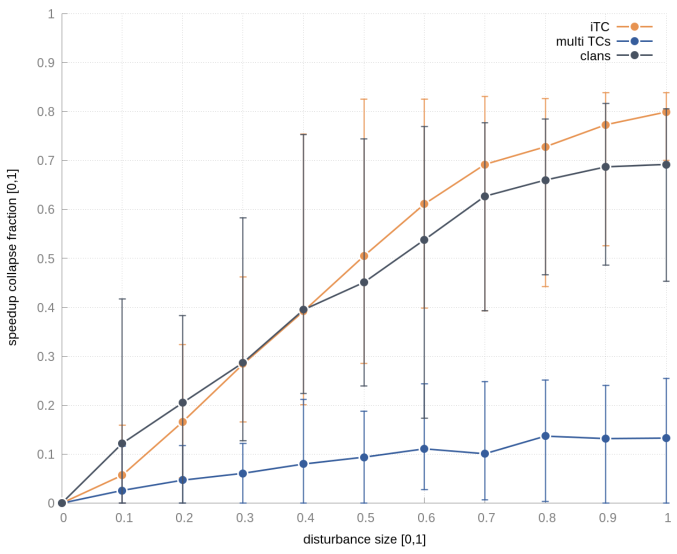


Figure 5.28: Summarised speedup collapse at collusion attacks: Lower values are better. Again it shows that the metric results are increasing with the disturbance size. While *Clans* are comparable to *iTCs*, TCs are substantially more robust, the difference being even greater than for the relative recovery costs metric (see Fig. 5.27).

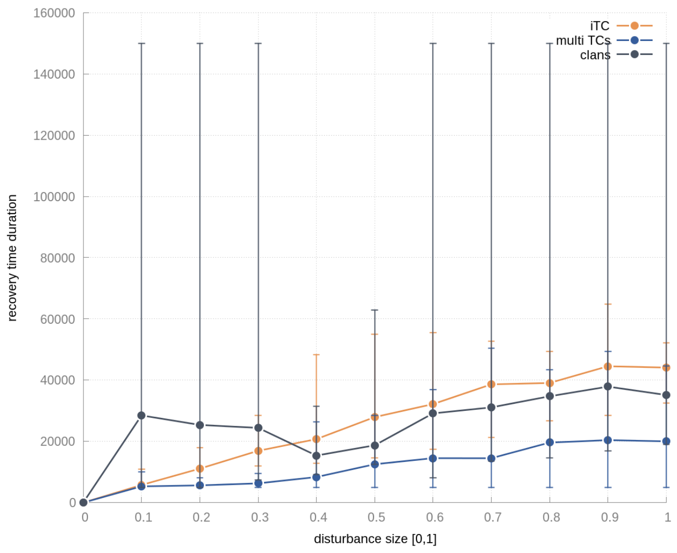


Figure 5.29: Summarised recovery duration at collusion attacks: Lower values are better. For *iTC*- and TC-application the metric values are again proportional to the disturbance size. Also it shows that TCs are more robust than the other two forms of organisation. The average values for the *Clan*-approach are affected by simulation runs in which no complete recovery was achieved after an attack (maximum values of 150000 ticks).

5.4 Summary

This chapter first presented a formal definition and classification of the evaluation scenario *Trusted Desktop Grid*. It was shown how a job- and work unit model is used to specify the execution assumptions of this system and how the behaviour of system participants is defined. In conclusion, it has been discussed that this system is subject to challenging issues due to uncertainty in the behaviour of the agents. Subsequently, it has been analysed how a trust and reputation model can be applied to tackle some of these issues and why this does not suffice. Then the applicability of Trusted Communities has been motivated and defined. Finally, the main part of this chapter presented and discussed the evaluation results achieved in an implementation of this system according to the models presented in this thesis.

This final section first described the implementation of the related *Clan* approach and some general assumptions. The actual presentation of the evaluation results was then divided into *performance evaluations* (no attacks) and *robustness evaluations* (with collusion attacks). The performance evaluations were conducted for an undisturbed TDG, as well as with an agent society containing 20-30% defecting, and respectively TM-exploiting, agents. In these evaluations, TCs have been compared with a modified version of the *Clan*-approach (with the Transparent WU validation benefit for members to avoid bias), as well as with the *iTC*-approach. The evaluations showed that TCs substantially outperform the *iTC*-approach in each experiment. As for the *Clan*-approach, it has been shown that TCs result in a higher preformance in the undisturbed and TM-exploiting agents experiment, while being on par in the defecting agents experiment. Additionally, the robustness of the three organisation approaches has been evaluated in a series of experiments with increasing disturbance sizes. The diusturbances were caused by attacks from colluding freeriders. The results for these experiments have shown that the TC-approach is significantly more robust than the other approaches, being reflected in a lower speedup collapse fraction, shorter recovery duration and lower recovery costs.

In sum, the evaluation results have demonstrated that Trusted Communities significantly increase the performance and robustness of an Open Distributed System, as compared to state-of-the-art approaches.

6 | Conclusion

This chapter summarises the preceding chapters, and discusses the contributions of the approach presented in this thesis. Finally, it presents an overview over future research opportunities in the context of the proposed approach.

6.1 Thesis Summary

The objective of this thesis was the performance and robustness improvement of open, distributed systems, accommodating a society of self-interested, heterogeneous and autonomous agents. After an analysis of the related work, a system model has been presented. This model defines the class of targeted systems through the specification of system participants, characteristics, agent models and Trust Management models. In the following, the challenging issues in such systems have been examined. Based on these issues, a novel MAS organisation, the *Trusted Community*, has been proposed. The subsequent chapter presented the modular design of this approach: The focus has been on the presentation of the state-based realisation as agent component, the configuration of the decision-making through the utilisation of dedicated strategies, and the proposition of a set of basic strategy implementation that in sum realise the TC approach. In conclusion, an instance of the targeted system class, the *Trusted Desktop Grid*, has been defined with great detail regarding the underlying assumptions, mechanisms, and challenging issues. The application of the TC approach in the TDG has then been motivated and the required realisations examined. Finally, the evaluation results for the application of *Trusted Communities* in the *Trusted Desktop Grid* have been presented and discussed.

The Trusted Community approach presented in this thesis is the result of a focussed combination of techniques from the fields of *MAS organisations*, *Trust Management*, and *Organic Computing*. Contributions to the state of the art from this approach can be summarised as the explicit consideration of the following aspects:

- **Consideration of emergent system states:** Most TM-based approaches in the literature implicitly assume that the operation of the TM as such is never at risk, and consequently do not account for system states in which agents cannot base their decision-making on trust. Here, the TC approach is heavily influenced by mechanisms from the field of *Organic Computing* systems: Complex systems can drift into emergent states at runtime that have not been predicted and accounted for at design time. The elements of the systems must hence be equipped with instruments to observe such states and react to them in order to ensure a successful operation. Trusted Communities have been designed by incorporating respective instruments, both for TC members, as well as hierarchically for the TCM.

- **Adaptive optimisation of trust-aware decision-making:** Most approaches in the literature that describe the utilisation of TM in agent decision-making do not explicitly account for the overhead and sub-optimality of the resulting interactions. Trust is only applied where risk is involved, but not always does risk actually lead to interaction partners deceiving each other. In these cases, trust can be *too much*, preventing optimal interactions without safety means. On the other hand, TM can make agents slow to react on behaviour changes because of the phenomenon of *over-confidence*. The Trusted Community approach contributes to the state of the art here, by allowing for a situation-aware utilisation of TM towards agents with uncertain behaviours, an abandonment of TM and other safety means in a closed environment for highly trusted agents, and fast reactions to behaviour changes as regulatory means within this environment.

- **Alternatives to reputation-based incentive mechanisms:** Approaches to control agents in open systems often describe the requirement for incentives in order to motivate self-interested agents to cooperate. Many approaches describe the utilisation of a reputation system to realise this incentive mechanism. However, as described above, these approaches do not account for a malfunctioning reputation system, and provide no solutions for this case. Additionally, the requirement to provide incentives for the collaborative detection of phenomena that cannot be observed locally by single agents are often neglected. These phenomena, such as *collusion*, can threaten the operation of a system, but self-interested agents need to be motivated to cooperate in their observation, as these agents have no *direct* consequences because of such behaviours and thus see no need to invest effort. The TC approach provides an environment in which such cooperation can be fostered by incentivising member cooperation through the absolute goal of agents in the system, the gain in utility.

- **Extensive evaluation of controls for open distributed Desktop Grid Systems:** The classification of the TDG according to the taxonomy presented in [80] revealed that this particular class of Desktop Grid systems has seldom been evaluated by the research community. Most approaches in the literature are applicable in either centralised, or enterprise-based systems, while there is a lack of approaches for the application in open, decentralised, and volunteer-based systems. This thesis contributes to the state of the art by providing an analysis of such a system instance. This analysis is comprised by the evaluations of Trust Management, as well as the application of MAS organisations in general, and Trusted Communities in particular for such systems. Additionally, the developed threat model examines the challenges of such a system and can be used as a starting point for the development of new approaches for this system class.

6.2 Future research opportunities

The Trusted Community approach proposed in this thesis has been presented as extensible set of components, workflows, strategies and decisions. In that, the thesis aimed at providing an overall view on the design, possible implementations and their applicability in a particular hosting system, the TDG. Some aspects of this approach have however only been briefly brought up. The following is a short examination of these aspects and the contained future research opportunities:

- **Advanced TC strategy implementations**: The Trusted Community approach has been designed with extensibility in mind. Especially the TC strategies, encapsulating specific decision-making, are an instrument to adapt the approach to the requirements of a hosting system and to allow for more runtime-adaptations. While this thesis has provided details about *basic* implementations of the strategies in Sec. 4.6, refined implementations have only been discussed on an abstract level. The design of advanced strategies offers many opportunities for improved TC behaviour, be it by the mentioned applications of *trust-prediction* (cf. e.g. [76]) in the implementation of an *Active TC Expansion* strategy, and *test-tasks* (cf. e.g. [175]) in an advanced implementation of a *Member Control* strategy, or entirely different influences on the decision-making within the strategies.

- **Incorporation of confidence in the trust model**: The specification of the trust model in Sec. 3.1.3 included the requirement to provide aggregation functions for direct trust, reputation trust and aggregated trust values. In the literature, often trust models are extended by a meta value for the *confidence* for the correct estimation of a trust value (cf. e.g. [210]). The application of confidence is suited to further improve the TC approach, especially where agents are accepted as TC members based on their trustworthiness. Consider for example the application of the threshold $thres_m^{DT}$ in the *Basic Potential Member Search* strategy (see Sec. 4.6): Here, the additional assignment of a confidence value can help to decrease the risk of accepting agents as potential members that have a high trust value ($> thres_m^{DT}$) only due to outdated or too few ratings.

- **Machine-learning within the O/C-loops**: The agent model presented in Sec. 3.1.2 of this thesis has been described as constituted by an *Observer- and Controller component* as proposed in the literature on *Organic Computing*. Additionally, the TC repertoire of TCM-strategies has been designed including an additional, hierarchical O/C-loop. While the application of both loops for the runtime-adaptation of the comprising agents has been described and motivated, a central part of the O/C-loops has been neglected in this thesis: The application of machine-learning techniques, such as *Learning Classifier Systems* (cf. e.g. [50]), provides a wide field of additional research opportunities for the situation-aware self-adaptation of agents. Such adaptations can for example include the runtime exchange of agent component implementations and their effect on the dynamics within the hosting system.

- **Protection from adversary TCMs**: The design of the TC strategies and compromising states has focussed on the evaluation and sanctioning of member behaviour in order to allow for a stable operation of a TC. This design has only rudimentarily accounted for the threat of an adversary TC manager: The upcoming of an adversary agent as TCM has been tried to avoid by *Distributed Leader Election* strategies that make it costly for adversary agents to be elected (e.g. by election based on trustworthiness), and the strategy repertoire of TC members has been designed such that it allows members to leave a TC and form a new without the adversary TCM. However, the actual influence of an adversary TCM has not been evaluated thoroughly and the necessity of a dedicated *TCM Control* strategy (applied by TC members) has not been explored.

- **Consideration of fairness**: The contribution of TC application to the operation of the hosting system has been demonstrated as the improvement of *performance* and *robustness* of the system. Besides these two aspects, often the *fairness* of an approach is considered in the literature (cf. e.g. [35]). The utilisation of fairness as optimisation target for TC application has the following motivation: In an agent society composed of self-interested agents with heterogeneous capabilities, often the best-performing agents build up reputation the fastest and are then frequently requested for the delegation of tasks by the majority of agents (cf. e.g. [24]). Other agents, having lower capabilities, benefit from such potent agents, while being ignored as delegation partners having much less overhead. This asymmetry provides incentives for agents to give false testimonies about own capabilities in order to receive less requests. The contribution of TC application to more fairness in the system can be further explored, with the regulatory strategies and TC incentives providing strong starting points for an according realisation.

- **Utilisation of agent norms**: In the design of the TC approach, the communication of TCM instructions to members has only been implicitly approached by the definition of roles and the expectancy of their adherence. Additionally, the *Basic Member Control* strategy (see Sec. 4.6.7) has been designed with *self-explainability* in mind, in that it provides TC members the possibility to request their current score from the TCM and derive the consequences of their actions. Such expected behaviours and according sanctions are often explicitly formulated by *behavioural norms* (cf. e.g. [211]). The extension of the TC approach by a stringent formalisation of norms and their sanctioning could further elaborate on expected TC membership benefits and allow for more purposeful behavioural adaptations of TC members and potential member alike. This yields the potential to make Trusted Communities more *self-explaining* and hence more robust.

- **TC applicability in other systems**: The TC approach has been designed as a generic approach for the class of systems that match the specification of the hosting system in Sec. 3.1. While the applicability has been extensively evaluated in a particular instance of this class, the *Trusted Desktop Grid*, this class contains many other instances, as summarised in Sec. 2.1, that have been left unexplored. Here, particularly *Vehicular Ad-Hoc Networks*, and *Decentralised Power Grid Systems* are recent fields of research that promise many opportunities for improvement through TC application. The evaluation of the TC applicability in *Decentralised Power Grid Systems* is then congruously scheduled as part of the final phase of the DFG research unit *OC-Trust* (FOR 1085).

- **Advanced Organisation Benefit Strategies**: The application of TCs in the Trusted Desktop Grid presented in Sec. 5.2 has been evaluated with a set of *Organisation Benefit* strategies that capture the most common approaches in this field. However, further research opportunities can be derived from the examination of additional strategies of that type: Especially the application of a TCM-coordinated, central (TC-wide) scheduling for the members promises to further exploit the closed environment of a TC. Here, the members would further give up a part of their autonomy, but would gain in an optimised scheduling of their work units. The TCM could apply machine-learning techniques to predict resource and host availabilities, as well as job generation patterns of its members. This could then provide indications for optimal schedules.

Bibliography

[20] C. Castelfranchi and R. Falcone, *Trust Theory - A socio-Coginitive and Computational Model*. John Wiley & Sons Ltd., 2010, ISBN: 9780470028759.

[21] T. Malsch, 'Naming the Unnamable: Socionics or the Sociological Turn of/to Distributed Artificial Intelligence', *Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 3, pp. 155–186, 2001, ISSN: 1387-2532.

[22] M. Schillo, P. Funk and M. Rovatsos, 'Using trust for detecting deceitful agents in artificial societies', *Applied Artificial Intelligence*, vol. 14, no. 8, pp. 825–849, 2000.

[23] S. D. Ramchurn, D. Huynh and N. R. Jennings, 'Trust in multi-agent systems', *The Knowledge Engineering Review*, vol. 19, no. 01, pp. 1–25, Apr. 2004, ISSN: 1469-8005.

[24] H. Yu, Z. Shen, C. Leung, C. Miao and V. R. Lesser, 'A Survey of Multi-Agent Trust Management Systems', *IEEE Access*, vol. 1, pp. 35–50, 2013, ISSN: 2169-3536.

[25] I. Pinyol and J. Sabater-Mir, 'Computational trust and reputation models for open multi-agent systems: a review', *Artificial Intelligence Review*, vol. 40, no. 1, pp. 1–25, 2013, ISSN: 0269-2821.

[26] A. Jøsang, 'The right type of trust for distributed systems', in *Proceedings of the 1996 workshop on New security paradigms - NSPW '96*, New York, New York, USA: ACM Press, Sep. 1996, pp. 119–131.

[27] M. Blaze, J. Feigenbaum, J. Ioannidis and A. D. Keromytis, 'The role of trust management in distributed systems security', in *Secure Internet programming*, J. Vitek and C. D. Jensen, Eds., Springer-Verlag, Jun. 1999, pp. 185–210, ISBN: 3-540-66130-1.

[28] J. Sabater and C. Sierra, 'REGRET: A reputation model for gregarious societies', in *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, C. Castelfranchi and L. Johnson, Eds., ACM Press, May 2002, pp. 475–482.

[29] C. Burnett, T. J. Norman and K. Sycara, 'Trust decision-making in multi-agent systems', in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, vol. 1, Jul. 2011, pp. 115–120, ISBN: 978-1-57735-513-7.

[30] C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., *Organic Computing - A Paradigm Shift for Complex Systems.* Basel: Birkhäuser Verlag, 2011, ISBN: 978-3034-801-294.

[31] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, C. Müller-Schloer and H. Schmeck, 'Organic Traffic Control', in *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., Birkhäuser Verlag, 2011, pp. 431–446, ISBN: 978-3-0348-0129-4.

[32] B. Horling and V. Lesser, 'A Survey of Multi-Agent Organizational Paradigms', *The Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, 2005.

[33] P. Mathieu, J.-C. Routier and Y. Secq, 'Principles for dynamic multi-agent organizations', in *Intelligent Agents and Multi-Agent Systems*, vol. 2413, ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2002, ISBN: 978-3-540-44026-0.

[34] J. Ferber, O. Gutknecht and F. Michel, 'From agents to organizations: An organizational view of multi-agent systems', in *Agent-Oriented Software Engineering IV*, P. Giorgini, J. P. Müller and J. Odell, Eds., 2004, pp. 214–230, ISBN: 978-3-540-24620-6.

[35] A. Wierzbicki, *Trust and Fairness in Open, Distributed Systems*, ser. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 298, ISBN: 978-3-642-13450-0.

[36] F. G. Mármol, 'Trust and Reputation Management in Distributed and Heterogeneous Systems', PhD thesis, University of Murcia, 2010, p. 166, ISBN: 9780387097503.

[37] R. P. Würtz, Ed., *Organic Computing (Understanding Complex Systems).* Springer, 2008, p. 356, ISBN: 978-3540776567.

[38] J. Branke, M. Mnif, C. Müller-Schloer, H. Prothmann, U. Richter, F. Rochner and H. Schmeck, ‘Organic Computing - Addressing Complexity by Controlled Self-Organization’, in *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (isola 2006)*, IEEE, Nov. 2006, pp. 185–191, ISBN: 978-0-7695-3071-0.

[39] H. Kasinger and B. Bauer, ‘Combining Multi-Agent-System Methodologies for Organic Computing Systems’, in *16th International Workshop on Database and Expert Systems Applications (DEXA’05)*, IEEE, 2005, pp. 160–164, ISBN: 0-7695-2424-9.

[40] C. Castelfranchi, ‘Earthquakes in Trust Networks: Basic Dynamic Principles’, *SSRN Electronic Journal*, 2012, ISSN: 1556-5068.

[41] R. S. Seymour and G. L. Peterson, ‘Responding to Sneaky Agents in Multi-agent Domains.’, in *Proceedings of the 22nd International Florida Artificial Intelligence Research Society (FLAIRS) Conference*, H. C. Lane and H. W. Guesgen, Eds., AAAI Press, 2009, ISBN: 978-1-57735-419-2.

[42] L. Atzori, A. Iera and G. Morabito, ‘The internet of things: a survey’, *Computer Networks*, vol. 54, no. 15, pp. 2787 –2805, 2010, ISSN: 1389-1286.

[43] N. R. Jennings, ‘An agent-based approach for building complex software systems’, vol. 44, pp. 35–41, 2001, ISSN: 00010782.

[44] M. Luck, P. McBurney and C. Preist, ‘A manifesto for agent technology: towards next generation computing’, *Autonomous Agents and Multi-Agent Systems*, vol. 9, no. 3, pp. 203–252, Nov. 2004, ISSN: 1387-2532.

[45] I. Foster and N. R. Jennings, ‘Brain Meets Brawn: Why Grid and Agents Need Each Other’, in *International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 1, IEEE Computer Society, 2004, pp. 8–15.

[46] M. Luck, P. McBurney, O. Shehory and S. Willmott, *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.

[47] R. Falcone and C. Castelfranchi, ‘The human in the loop of a delegated agent: the theory of adjustable social autonomy’, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 31, no. 5, pp. 406–418, 2001, ISSN: 10834427.

[48] C. Carabelea, O. Boissier and A. Florea, ‘Autonomy in multi-agent systems: a classification attempt’, in *Agents and Computational Autonomy: Potential, Risks, and Solutions. Vol. 2969.* M. Nickles, M. Rovatsos and G. Weiss, Eds., ser. Lecture Notes in Computer Science, 2004, pp. 103–113.

[49] M. Wooldridge, *An Introduction to Multiagent Systems*, 2nd ed. Chichester, UK: Wiley, 2009, ISBN: 978-0-470-51946-2.

[50] J. H. Holland, *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press, 1992, ISBN: 0-262-58111-6.

[51] R. G. Smith, ‘The contract net protocol: high-level communication and control in a distributed problem solver’, *IEEE Trans. Comput.*, vol. 29, no. 12, pp. 1104–1113, Dec. 1980, ISSN: 0018-9340.

[52] C. Brooks and E. Durfee, ‘Congregation formation in multiagent systems’, *Autonomous Agents and Multi-Agent Systems*, vol. 7, no. 1, 2003.

[53] R. Patil, D. Mckay, T. Finin, R. Fikes, T. Gruber, P. F. Patel-Schneider and R. Neches, ‘The darpa knowledge sharing effort: progress report’, in *Proceedings on the Third International Conference on Principles of Knowledge Representation and Reasoning (KR92)*, Morgan Kaufman, 1998, pp. 777–788.

[54] A. S. Rao and M. P. Georgeff, ‘BDI Agents : From Theory to Practice’, *Practice*, vol. 95, pp. 312–319, 1995.

[55] B. Savarimuthu and S. Cranefield, ‘Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems’, *Multiagent and Grid Systems*, vol. 7, no. 1, pp. 21–54, 2011.

[56] M. Wooldridge, N. Jennings and D. Kinny, ‘The gaia methodology for agent-oriented analysis and design’, *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000, ISSN: 1387-2532.

[57] M. D. Oliveira and M. Purvis, ‘A distributed model for institutions in open multi-agent systems’, in *Knowledge-Based Intelligent Information and Engineering Systems*, M. Negoita, R. Howlett and L. Jain, Eds., ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2004, pp. 1172–1178, ISBN: 978-3-540-23206-3.

[58] R. Centeno, H. Billhardt and R. Hermoso, ‘An Adaptive Sanctioning Mechanism for Open Multi-agent Systems Regulated by Norms’, in *23rd IEEE International Conference on Tools with Artificial Intelligence*, IEEE, Nov. 2011, pp. 523–530, ISBN: 978-1-4577-2068-0.

[59] G. Di Marzo Seregundo, M.-P. Gleizes and A. Karageorgos, ‘Self-organization in multi-agent systems’, *The Knowledge Engineering Review*, vol. 20, no. 02, pp. 165–189, Jun. 2005, ISSN: 1469-8005.

[60] R. Centeno and H. Billhardt, ‘Using incentive mechanisms for an adaptive regulation of open multi-agent systems’, in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, vol. 1, Barcelona, Catalonia, Spain, Jul. 2011, pp. 139–145, ISBN: 978-1-57735-513-7.

[61] R. Hermoso, H. Billhardt and S. Ossowski, ‘Role evolution in Open Multi-Agent Systems as an information source for trust’, in *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 217–224.

[62] A. Artikis, ‘Dynamic protocols for open agent systems’, in *Proceedings of The Eighth International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’09, vol. 1, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 97–104, ISBN: 978-0-9817381-6-1.

[63] M. Feldman and J. Chuang, ‘Overcoming free-riding behavior in peer-to-peer systems’, *ACM SIGecom Exchanges*, vol. 5, no. 4, pp. 41–50, Jul. 2005, ISSN: 15519031.

[64] J. Pitt, J. Schaumeier and A. Artikis, ‘Coordination, conventions and the self-organisation of sustainable institutions’, in *Agents in Principle, Agents in Practice*, D. Kinny, J.-j. Hsu, G. Governatori and A. Ghose, Eds., vol. 7047, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 202–217, ISBN: 978-3-642-25043-9.

[65] H. J. LeBlanc and X. D. Koutsoukos, ‘Consensus in networked multi-agent systems with adversaries’, in *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control (HSCC ’11)*, New York, USA: ACM Press, Apr. 2011, p. 281, ISBN: 978-1-450-30629-4.

[66] K. Barber and J. Kim, ‘Soft security: isolating unreliable agents from society’, in *Trust, Reputation, and Security: Theories and Practice*, R. Falcone, S. Barber, L. Korba and M. Singh, Eds., vol. 2631, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2003, pp. 224–233, ISBN: 978-3-540-00988-7.

[67] F. Zambonelli, N. R. Jennings and M. Wooldridge, ‘Organisational rules as an abstraction for the analysis and design of multi-agent systems’, *International Journal of Software and Knowledge Engineering*, vol. 11, no. 03, pp. 303–328, 2001.

[68] D. Gambetta, ‘Can we trust trust?’, *Trust: Making and Breaking Cooperative Relations*, pp. 213–237, 1988.

[69] S. P. Marsh, ‘Formalising Trust as a Computational Concept’, PhD thesis, 1994, p. 184.

[70] Z. Yan and S. Holtmanns, ‘Trust Modeling and Management: from Social Trust to Digital Trust’, *Computer Security Privacy and Politics Current Issues Challenges and Solutions*, pp. 290–323, 2007.

[71] F. G. Mármol and G. M. Pérez, ‘Trust and reputation models comparison’, *Internet Research*, vol. 21, no. 2, pp. 138–153, 2011, ISSN: 1066-2243.

[72] A. Jøsang, ‘Robustness of Trust and Reputation Systems: Does It Matter?’, in *Trust Management VI*, T. Dimitrakos, R. Moona, D. Patel and D. McKnight, Eds., vol. 374, ser. IFIP Advances in Information and Communication Technology, Springer Berlin Heidelberg, 2012, pp. 253–262, ISBN: 978-3-642-29851-6.

[73] D. Elgesem, ‘Normative Structures in Trust Management 1 The Puzzle of Trust Management’, *Lecture Notes in Computer Science*, vol. 3986/2006, pp. 48–61, 2006.

[74] A. Koster, J. Sabater-Mir and M. Schorlemmer, ‘Trust alignment: A Sine Qua Non of Open Multi-agent Systems’, in *On the Move to Meaningful Internet Systems: OTM 2011*, vol. 7044, ser. Lecture Notes in Computer Science, Springer-Verlag, Oct. 2011, pp. 182–199, ISBN: 978-3-642-25108-5.

[75] I. Pinyol and J. Sabater-Mir, ‘Pragmatic-strategic reputation-based decisions in bdi agents’, in *Proceedings of The Eighth International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’09, vol. 2, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 1001–1008, ISBN: 978-0-9817381-7-8.

[76] C. Burnett, T. Norman and K. Sycara, ‘Bootstrapping trust evaluations through stereotypes’, in *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, vol. 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 241 –248, ISBN: 978-0-9826571-1-9.

[77] C. Burnett, T. Norman and K. Sycara, ‘Stereotypical trust and bias in dynamic multi-agent systems’, *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 2011.

[78] W. T. L. Teacy, G. Chalkiadakis, A. Rogers and N. R. Jennings, ‘Sequential decision making with untrustworthy service providers’, in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’08, vol. 2, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 755–762, ISBN: 978-0-9817381-1-6.

[79] A. E. Arenas, B. Aziz and G. C. Silaghi, ‘Reputation management in collaborative computing systems’, in *Security and Communication Networks*, vol. 3, Wiley Online Library, 2010, pp. 546–564.

[80] S. Choi, R. Buyya, H. Kim and E. Byun, ‘A Taxonomy of Desktop Grids and its Mapping to State of the Art Systems’, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Tech. Rep., 2008.

[81] K. K. Fullam, M. Voss, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein and L. Vercouter, ‘A specification of the Agent Reputation and Trust (ART) testbed’, in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS ’05)*, New York, New York, USA: ACM Press, Jul. 2005, p. 512, ISBN: 1595930930.

[82] L. Shang, Z. Wang, X. Zhou, X. Huang and Y. Cheng, ‘TM-DG: A Trust Model Based On Computer User’s Daily Behavior for Desktop Grid Platform’, in *Proceedings of the 2007 symposium on Component and framework technology in high-performance and scientific computing - CompFrame ’07*, ACM Press, Oct. 2007, p. 59, ISBN: 9781595938671.

[83] P. Domingues, B. Sousa and L. Moura Silva, ‘Sabotage-tolerance and trust management in desktop grid computing’, *Future Generation Computer Systems*, vol. 23, no. 7, pp. 904–912, Aug. 2007, ISSN: 0167-739X.

[84] N. Andrade, F. Brasileiro, W. Cirne and M. Mowbray, ‘Discouraging Free Riding in a Peer-to-Peer CPU-Sharing Grid’, in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, Washington, DC, USA: IEEE Computer Society, 2004, pp. 129–137, ISBN: 0-7803-2175-4.

[85] G. C. Silaghi, F. Araujo, L. M. Silva, P. Domingues and A. E. Arenas, ‘Defeating colluding nodes in Desktop Grid computing platforms’, *2008 IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8, Apr. 2008, ISSN: 1530-2075.

[86] J. Dyson, N. Griffiths, H. Lim, S. Jarvis and G. Nudd, ‘Trusting agents for grid computing’, *IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, 2004.

[87] M. Vladoiu and Z. Constantinescu, ‘A Taxonomy for Desktop Grids from Users’ Perspective’, in *Proceedings of the 2008 World Congress on Engineering (WCE)*, vol. 1, London, U.K., 2008, pp. 1–5, ISBN: 978-9-8898-6719-5.

[88] H. Zhao and X. Li, ‘H-Trust: A Robust and Lightweight Group Reputation System for Peer-to-Peer Desktop Grid’, *The 28th International Conference on Distributed Computing Systems Workshops*, pp. 235–240, Jun. 2008.

[89] L.-C. Canon, E. Jeannot and J. Weissman, ‘A dynamic approach for characterizing collusion in desktop grids’, *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pp. 1–12, 2010.

[90] F. Azzedin and M. Maheswaran, ‘Evolving and managing trust in grid computing systems’, in *Canadian IEE Conference on Electrical and Computer Engineering (CCECE 2002)*, vol. 3, IEEE, 2002, pp. 1424–1429, ISBN: 0-7803-7514-9.

[91] A. Arenas, M. Wilson and B. Matthews, ‘On trust management in grids’, in *Proceedings of the First International Conference on Autonomic Computing and Communication Systems*, Rome, Italy: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Oct. 2007, p. 7, ISBN: 978-963-9799-09-7.

[92] T. G. Papaioannou and G. D. Stamoulis, ‘Reputation-based estimation of individual performance in collaborative and competitive grids’, *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1327–1335, Oct. 2010, ISSN: 0167739X.

[93] J. Patel, ‘A Trust and Reputation Model for Agent-Based Virtual Organisations’, PhD thesis, University of Southampton, 2006, p. 184.

[94] G. Anders, J.-P. Steghöfer, H. Seebach, F. Nafz and W. Reif, ‘Structuring and Controlling Distributed Power Sources by Autonomous Virtual Power Plants’, in *Proceedings of the IEEE Power and Energy Student Summit (PESS)*, IEEE, 2010, pp. 40–42.

[95] G. Chalkiadakis, V. Robu and R. Kota, ‘Cooperatives of distributed energy resources for efficient virtual power plants’, in *The 10th International Conference on Autonomous Agents and Multiagent Systems*, vol. 2, Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 787–794, ISBN: 0-9826571-6-1.

[96] G. Anders, F. Siefert, J.-P. Steghöfer and W. Reif, ‘A decentralized multi-agent algorithm for the set partitioning problem’, in *PRIMA 2012: Principles and Practice of Multi-Agent Systems*, I. Rahwan, W. Wobcke, S. Sen and T. Sugawara, Eds., vol. 7455, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 107–121, ISBN: 978-3-642-32728-5.

[97] M. Vasirani, R. Kota, R. L. Cavalcante, S. Ossowski and N. R. Jennings, ‘Technical Report: Virtual Power Plants of Wind Power Generators and Electric Vehicles’, Tech. Rep., 2012, p. 21.

[98] J. Zhang, ‘A Survey on Trust Management for VANETs’, in *2011 IEEE International Conference on Advanced Information Networking and Applications*, IEEE, 2011, pp. 105–112, ISBN: 978-1-61284-313-1.

[99] H. Chen, H. Wu, X. Zhou and C. Gao, ‘Agent-based trust model in wireless sensor networks’, in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 3, 2007, pp. 119–124.

[100] A. Boukerche and X. Li, ‘An agent-based trust and reputation management scheme for wireless sensor networks’, in *IEEE Conference on Global Telecommunications (GLOBECOM '05)*, vol. 3, 2005.

[101] N. Sahli, G. Lenzini and H. Eertink, ‘Trustworthy agent-based recommender system in a mobile p2p environment’, in *Proceedings of the Seventh International Conference on Agents and Peer-to-Peer Computing*, D. Beneventano, Z. Despotovic, F. Guerra, S. Joseph, G. Moro and A. P. Pinninck, Eds., ser. Lecture Notes in Computer Science, vol. 6573, Berlin, Heidelberg: Springer Berlin Heidelberg, May 2012, pp. 59–70, ISBN: 978-3-642-31808-5.

[102] N. Griffiths, ‘Enhancing peer-to-peer collaboration using trust’, *Expert Systems with Applications*, vol. 31, no. 4, pp. 849–858, Nov. 2006, ISSN: 09574174.

[103] X. Ding, W. Yu and Y. Pan, ‘A Dynamic Trust Management Scheme to Mitigate Malware Proliferation in P2P Networks’, in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, 2008, pp. 1605–1609.

[104] J. Hu, Q. Wu and B. Zhou, ‘RBTrust: A Recommendation Belief Based Distributed Trust Management Model for P2P Networks’, *10th IEEE International Conference on High Performance Computing and Communications*, pp. 950–957, Sep. 2008.

[105] K. Aberer and Z. Despotovic, ‘Managing Trust in a Peer-2-Peer Information System’, in *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM01)*, P. Henrique, L. Liu and D. Grossman, Eds., 2001, pp. 310–317.

[106] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, ‘The Eigentrust algorithm for reputation management in P2P networks’, in *12th International Conference on World Wide Web (WWW)*, 2003, p. 640, ISBN: 1581136803.

[107] Y.-H. Tan and W. Thoen, ‘Toward a generic model of trust for electronic commerce’, *International Journal of Electronic Commerce*, vol. 5, no. 2, pp. 61–74, Dec. 2000, ISSN: 1086-4415.

[108] A. Hnativ and S. A. Ludwig, ‘Evaluation of trust in an ecommerce multi-agent system using fuzzy reasoning’, in *IEEE International Conference on Fuzzy Systems*, IEEE, Aug. 2009, pp. 757–763, ISBN: 978-1-4244-3596-8.

[109] A. Gutowska and K. Buckley, ‘Computing Reputation Metric in Multi-Agent E-Commerce Reputation System’, in *28th International Conference on Distributed Computing Systems Workshops*, IEEE, Jun. 2008, pp. 255–260.

[110] S. Zhao, H. Liu and Z. Sun, ‘Scalable Trust in Multi-agent E-commerce System’, in *2008 International Symposium on Electronic Commerce and Security*, IEEE, 2008, pp. 990–993, ISBN: 978-0-7695-3258-5.

[111] Z. Gan, Q. Ding and V. Varadharajan, ‘Reputation-Based Trust Network Modelling and Simplification in Multiagent-Based E-Commerce Systems’, in *Fifth International Conference on Next Generation Web Services Practices*, IEEE, Sep. 2009, pp. 60–67, ISBN: 978-0-7695-3821-1.

[112] L. Gasser, ‘Distributed artificial intelligence’, in, N. M. Avouris and L. Gasser, Eds., Norwell, MA, USA: Kluwer Academic Publishers, 1992, ch. An Overview of DAI, pp. 9–30, ISBN: 0-7923-1585-5.

[113] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999, ISBN: 0201360489.

[114] K. M. Carley and L. Gasser, ‘Multiagent systems’, in, G. Weiss, Ed., Cambridge, MA, USA: MIT Press, 1999, ch. Computational Organization Theory, pp. 299–330, ISBN: 0-262-23203-0.

[115] D. Grossi, F. Dignum, M. Dastani and L. Royakkers, ‘Foundations of organizational structures in multiagent systems’, in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, New York, New York, USA: ACM Press, Jul. 2005, p. 690, ISBN: 1595930930.

[116] N. R. Jennings, ‘On agent-based software engineering’, *Artificial Intelligence*, vol. 117, no. 2, pp. 277–296, Mar. 2000, ISSN: 0004-3702.

[117] S. A. DeLoach, ‘Multiagent systems engineering of organization-based multiagent systems’, in *Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems (SELMAS '05)*, ser. SELMAS '05, New York, New York, USA: ACM Press, 2005, p. 1, ISBN: 1595931163.

[118] N. Griffiths, ‘Cooperative clans’, *Kybernetes*, vol. 34, no. 9/10, 2005, ISSN: 0368-492X.

[119] R. Centeno, H. Billhardt, R. Hermoso and S. Ossowski, ‘Organising mas: a formal model based on organisational mechanisms’, in *Proceedings of the 2009 ACM Symposium on Applied Computing*, ser. SAC '09, New York, NY, USA: ACM, 2009, pp. 740–746, ISBN: 978-1-60558-166-8.

[120] Z. Guessoum, M. Ziane and N. Faci, ‘Monitoring and Organizational-Level Adaptation of Multi-Agent Systems’, in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 2, New York, New York, USA, Jul. 2004, pp. 514–521, ISBN: 1-58113-864-4.

[121] D. Grossi, F. Dignum, V. Dignum, M. Dastani and L. Royakkers, ‘Structural evaluation of agent organizations’, in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS ’06)*, New York, New York, USA: ACM Press, May 2006, p. 1110, ISBN: 1595933034.

[122] K. R. T. Larsen and C. R. McInerney, ‘Preparing to work in the virtual organization’, *Information and Management*, vol. 39, no. 6, pp. 445–456, May 2002, ISSN: 0378-7206.

[123] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. Gray and N. J. Fiddian, ‘Agent-based formation of virtual organisations’, *Knowledge-Based Systems*, vol. 17, no. 2-4, pp. 103 –111, 2004, ISSN: 0950-7051.

[124] A. Amorim, ‘Virtual organization theory: current status and demands’, in *Integration and Innovation Orient to E-Society Volume 1*, W. Wang, Y. Li, Z. Duan, L. Yan, H. Li and X. Yang, Eds., vol. 251, ser. IFIP - The International Federation for Information Processing, Springer US, 2007, pp. 1–8, ISBN: 978-0-387-75465-9.

[125] E. Oliveira and A. P. Rocha, ‘Agents Advanced Features for Negotiation in Electronic Commerce and Virtual Organisations Formation Processes’, in *Agent Mediated Electronic Commerce, The European AgentLink Perspective*, Jan. 2001, pp. 78–97, ISBN: 3-540-41671-4.

[126] J. McGinnis, K. Stathis and F. Toni, ‘A Formal Framework of Virtual Organisations as Agent Societies’, *Electronic Proceedings in Theoretical Computer Science*, vol. 16, pp. 1–14, Jan. 2010, ISSN: 2075-2180.

[127] I. Foster, ‘The Anatomy of the Grid: Enabling Scalable Virtual Organizations’, *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, Aug. 2001, ISSN: 1094-3420.

[128] J. Patel, G. Shercliff, P. J. Stockreisser, J. Shao, W. A. Gray, N. J. Fiddian, S. Thompson, W. T. L. Teacy, N. R. Jennings, M. Luck, S. Chalmers, N. Oren, T. J. Norman, A. Preece and P. M. D. Gray, ‘Agent-based virtual organisations for the Grid’, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS ’05)*, p. 1151, 2005.

[129] M. Coppola, Y. Jégou, B. Matthews, C. Morin, L. P. Prieto, Ó. D. Sánchez, E. Y. Yang and H. Yu, ‘Virtual Organization Support within a Grid-Wide Operating System’, *IEEE Internet Computing*, vol. 12, no. 2, pp. 20–28, Mar. 2008, ISSN: 1089-7801.

[130] The OC Initiative, *Organic Computing Website*. [Online]. Available: http://www.organic-computing.org/ (visited on 27/12/2013).

[131] J. Kephart and D. Chess, ‘The vision of autonomic computing’, *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003, ISSN: 0018-9162.

[132] H. Kitano, ‘Biological robustness.’, *Nature reviews. Genetics*, vol. 5, no. 11, pp. 826–37, Nov. 2004, ISSN: 1471-0056.

[133] C. Müller-Schloer, ‘Organic computing - on the feasibility of controlled emergence’, in *International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS ’04)*, 2004, pp. 2–5.

[134] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter and H. Schmeck, ‘Observation and Control of Organic Systems’, in *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., vol. 1, ser. Autonomic Systems, Springer Basel, 2011, pp. 325–338, ISBN: 978-3-0348-0129-4.

[135] H. Schmeck, C. Müller-Schloer, E. Çakar, M. Mnif and U. Richter, ‘Adaptivity and self-organization in organic computing systems’, *ACM Transactions on Autonomous and Adaptive Systems*, vol. 5, no. 3, pp. 1–32, Sep. 2010, ISSN: 15564665.

[136] S. Tomforde, ‘Runtime adaptation of technical systems: An architectural framework for self-configuration and self-improvement at runtime’, PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, Fakultät für Elektrotechnik und Informatik, 2012, p. 356, ISBN: 978-3-8381-3133-7.

[137] U. M. Richter, ‘Controlled self-organisation using learning classifier systems’, PhD thesis, Universität Karlsruhe, Fakultät für Wirtschaftswissenschaften, 2009, p. 248, ISBN: 978-3-86644-431-7.

[138] E. Çakar, J. Hähner and C. Müller-Schloer, ‘Investigation of generic observer/controller architectures in a traffic scenario’, in *INFORMATIK 2008: Beherrschbare Systeme - dank Informatik*, ser. Lecture Notes in Computer Science, Köllen Verlag, 2008, pp. 733–738.

[139] M. Mnif and C. Müller-Schloer, ‘Quantitative emergence’, in *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., vol. 1, ser. Autonomic Systems, Springer Basel, 2011, pp. 39–52, ISBN: 978-3-0348-0129-4.

[140] F. Nafz, H. Seebach, J.-P. Steghöfer, S. Bäumler and W. Reif, ‘A formal framework for compositional verification of organic computing systems’, in *Autonomic and Trusted Computing*, B. Xie, J. Branke, S. Sadjadi, D. Zhang and X. Zhou, Eds., vol. 6407, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 17–31, ISBN: 978-3-642-16575-7.

[141] E. Çakar, N. Fredivianus, J. Hähner, J. Branke, C. Müller-Schloer and H. Schmeck, ‘Aspects of learning in oc systems’, in *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., vol. 1, ser. Autonomic Systems, Springer Basel, 2011, pp. 237–251, ISBN: 978-3-0348-0129-4.

[142] H. Prothmann, J. Branke, H. Schmeck, S. Tomforde, F. Rochner, J. Hähner and C. Müller-Schloer, ‘Organic traffic light control for urban road networks’, *International Journal of Autonomic and Adaptative Communication Systems*, vol. 2, no. 3, pp. 203–225, Jun. 2009, ISSN: 1754-8632.

[143] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, C. Müller-Schloer and H. Schmeck, ‘Possibilities and limitations of decentralised traffic control systems’, in *2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–9.

[144] B. Jakimovski, B. Meyer and E. Maehle, ‘Swarm intelligence for self-reconfiguring walking robot’, in *IEEE Symposium on Swarm Intelligence (SIS 2008)*, 2008, pp. 1–8.

[145] B. Satzger, A. Pietzowski, W. Trumler and T. Ungerer, ‘Using automated planning for trusted self-organising organic computing systems’, in *Autonomic and Trusted Computing*, C. Rong, M. Jaatun, F. Sandnes, L. Yang and J. Ma, Eds., vol. 5060, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 60–72, ISBN: 978-3-540-69294-2.

[146] S. Tomforde and J. Hähner, ‘Organic Network Control: Turning Standard Protocols into Evolving Systems’, in *Biologically Inspired Networking and Sensing: Algorithms and Architectures*, 2012, pp. 11–35, ISBN: 9781613500927.

[147] J. Salzmann, S. Kubisch, F. Reichenbach and D. Timmermann, ‘Energy and coverage aware routing algorithm in self organized sensor networks’, in *Fourth International Conference on Networked Sensing Systems (INSS ’07)*, 2007, pp. 77–80.

[148] M. Wittke, C. Grenz and J. Hähner, ‘Towards organic active vision systems for visual surveillance’, in *Architecture of Computing Systems - ARCS 2011*, M. Berekovic, W. Fornaciari, U. Brinkschulte and C. Silvano, Eds., vol. 6566, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 195–206, ISBN: 978-3-642-19136-7.

[149] J. Branke, J. Hähner, C. Müller-Schloer and H. Schmeck, *Organic Traffic Control Website*. [Online]. Available: http://projects.aifb.kit.edu/effalg/otcqe/index.htm (visited on 27/12/2013).

[150] B. Bauer and H. Kasinger, ‘Aose and organic computing - how can they benefit from each other? position paper’, in *Perspectives in Conceptual Modeling*, J. Akoka, S. Liddle, Y. Song, M. Bertolotto, I. Comyn-Wattiau, W.-J. Heuvel, M. Kolp, J. Trujillo, C. Kop and H. Mayr, Eds., vol. 3770, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 109–118, ISBN: 978-3-540-29395-8.

[151] W. van Oortmerssen, *Trusted communities: The only way one can effectively combat cheating (article in "Wouter’s Wiki")*, 2013. [Online]. Available: http://strlen.com/trusted-communities (visited on 27/12/2013).

[152] Y. Wang and J. Vassileva, ‘Trust-based community formation in peer-to-peer file sharing networks’, in *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI ’04, Washington, DC, USA: IEEE Computer Society, 2004, ISBN: 0-7695-2100-2.

[153] R. Ghanea-Hercock, ‘Dynamic trust formation in multi-agent systems’, in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), 10th International Workshop on Trust in Agent Societies*, 2007.

[154] D. G. Mikulski, F. L. Lewis, E. Y. Gu and G. R. Hudas, ‘Trust dynamics in multi-agent coalition formation’, in *SPIE Proceedings Vol. 8045, Unmanned Systems Technology XIII*, vol. 8045, Society of Photo-Optical Instrumentation Engineers (SPIE), May 2011,

[155] L. Nardin and J. Sichman, ‘Simulating the impact of trust in coalition formation: a preliminary analysis’, in *Second Brazilian Workshop on Social Simulation (BWSS)*, 2010, pp. 33–40.

[156] M. Kim and M. Kim, ‘Group organization algorithm based on trust and reputation in agent society’, *Proceedings of the Sixth International Conference on Advances in Mobile Computing and Multimedia (MoMM ’08)*, no. c, p. 387, 2008.

[157] N. Griffiths and M. Luck, ‘Coalition formation through motivation and trust’, *Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS ’03*, p. 17, 2003.

[158] B. Hoelz and C. Ralha, ‘A coalition formation mechanism for trust and reputation-aware multi-agent systems’, in *Advances in Artificial Intelligence (SBIA 2012)*, L. Barros, M. Finger, A. Pozo, G. Gimenénez-Lugo and M. Castilho, Eds., ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2012, pp. 162–171, ISBN: 978-3-642-34458-9.

[159] Z. Qing-hua, W. Chong-jun and X. Jun-yuan, 'Core: a trust model for agent coalition formation', in *Fifth International Conference on Natural Computation (ICNC '09)*, vol. 5, 2009, pp. 541–545.

[160] A. Ghaffarizadeh and V. H. Allan, 'History Based Coalition Formation in Hedonic Context Using Trust', *International Journal of Artificial Intelligence & Applications*, vol. 4, no. 4, pp. 1–8, Jul. 2013, ISSN: 0975-900X.

[161] S. Breban and J. Vassileva, 'Using inter-agent trust relationships for efficient coalition formation', in *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, ser. AI '02, London, UK: Springer-Verlag, 2002, pp. 221–236, ISBN: 3-540-43724-X.

[162] S. Breban and J. Vassileva, 'Long-term coalitions for the electronic marketplace', in *Proceedings of the E-Commerce Applications Workshop, Canadian AI Conference*, 2001.

[163] R. Hermoso, H. Billhardt and S. Ossowski, 'Integrating trust in virtual organisations', in *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, P. Noriega, J. Vázquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara and E. Matson, Eds., vol. 4386, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 19–31, ISBN: 978-3-540-74457-3.

[164] F. Kerschbaum, J. Haller, Y. Karabulut and P. Robinson, 'Pathtrust: A trust-based reputation service for virtual organization formation', in *Trust Management*, K. Stø len, W. Winsborough, F. Martinelli and F. Massacci, Eds., vol. 3986, ser. Lecture Notes in Computer Science Cd, Springer Berlin / Heidelberg, 2006, pp. 193–205, ISBN: 978-3-540-34295-3.

[165] A. Avila-Rosas and M. Luck, 'A direct reputation model for vo formation', in *Multi-Agent Systems and Applications IV*, M. Pěchouček, P. Petta and L. Z. Varga, Eds., vol. 3690, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, Sep. 2005, pp. 460–469, ISBN: 978-3-540-29046-9.

[166] W. Brockmann, A. Buschermöhle, J. Hülsmann and N. Rosemann, 'Trust management - handling uncertainties in embedded systems', in *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., vol. 1, ser. Autonomic Systems, Springer Basel, 2011, pp. 589–591, ISBN: 978-3-0348-0129-4.

[167] W. Reif, C. Müller-Schloer, J. Hähner, T. Ungerer and E. André, *Oc trust project website*. [Online]. Available: http://www.informatik.uni-augsburg.de/de/lehrstuehle/swt/se/projects/oc-trust/ (visited on 27/12/2013).

[168] F. Nafz, H. Seebach, J.-P. Steghöfer, G. Anders and W. Reif, 'Constraining self-organisation through corridors of correct behaviour: the restore invariant approach', in *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck and T. Ungerer, Eds., vol. 1, ser. Autonomic Systems, Springer Basel, 2011, pp. 79–93, ISBN: 978-3-0348-0129-4.

[169] C. Cérin and G. Fedak, Eds., *Desktop Grid Computing*, ser. Chapman & Hall/CRC Numerical Analy & Scient Comp. Series. Chapman and Hall/CRC, 2012, ISBN: 978-1-4398-6214-8.

[170] T. Kiss and G. Terstyanszky, 'Programming Applications for Desktop Grids', in *Desktop Grid Computing*, C. Cérin and G. Fedak, Eds., ser. Chapman & Hall/CRC Numerical Analy & Scient Comp. Series, Chapman and Hall/CRC, Jun. 2012, ch. 14, pp. 309–331, ISBN: 978-1-4398-6214-8.

[171] D. W. Walker and J. J. Dongarra, 'Mpi: a standard message passing interface', *Supercomputer*, vol. 12, pp. 56–68, 1996.

[172] I. Foster and C. Kesselman, 'Globus: a metacomputing infrastructure toolkit', *International Journal of Supercomputer Applications*, vol. 11, pp. 115–128, 1996.

[173] I. Foster, C. Kesselman and Globus Alliance Board, *Globus Toolkit*. [Online]. Available: http://www.globus.org/toolkit/ (visited on 27/12/2013).

[174] D. Kondo, M. Taufer, C. L. Brooks, H. Casanova and A. A. Chien, 'Characterizing and Evaluating Desktop Grids: An Empirical Study', in *Proceedings of the International Parallel and Distributed Processing Symposium*, 2004.

[175] L. Sarmenta, 'Sabotage-tolerance mechanisms for volunteer computing systems', in *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society, 2001, pp. 337–346, ISBN: 0-7695-1010-8.

[176] K. Watanabe and M. Fukushi, 'Generalized Spot-Checking for Sabotage-Tolerance in Volunteer Computing Systems', in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, IEEE, 2010, pp. 655–660, ISBN: 978-1-4244-6987-1.

[177] K. Ranganathan, M. Ripeanu, A. Sarin and I. Foster, 'Incentive mechanisms for large collaborative resource sharing', in *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, IEEE Computer Society, Apr. 2004, pp. 1–8, ISBN: 0-7803-8430-X.

[178] P. Merz, F. Kolter and M. Priebe, ‘Free-riding prevention in super-peer desktop grids’, in *Third International Multi-Conference on Computing in the Global Information Technology (ICCGI ’08)*, 2008, pp. 297–302.

[179] Z. Shanyu and V. Lo, ‘Result Verification and Trust-Based Scheduling in Peer-to-Peer Grids’, in *Fifth IEEE International Conference on Peer-to-Peer Computing (P2P’05)*, IEEE, Aug. 2005, pp. 31–38, ISBN: 0-7695-2376-5.

[180] I. Thabet, I. Bouslimi, C. Hanachi and K. Ghédira, ‘A multi-agent organizational model for grid scheduling’, in *Agent and Multi-Agent Systems: Technologies and Applications*, ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, ISBN: 978-3-642-21999-3.

[181] S. Abdallah, H. Zhang and V. Lesser, ‘The role of an agent organization in a grid computing environment’, in *Proceedings of the 14th International Conference on Automated Planning and Scheduling, Workshop on Planning and Scheduling for Web and Grid Services*, Jun. 2004.

[182] A. Chakravarti, G. Baumgartner and M. Lauria, ‘The organic grid: self-organizing computation on a peer-to-peer network’, in *Proceedings of the International Conference on Autonomic Computing*, IEEE, 2004, pp. 96–103, ISBN: 0-7695-2114-2.

[183] H. Kim, S. Kim, E. Byun, C. Hwang and J. Choi, ‘Agent-Based Autonomous Scheduling Mechanism Using Availability in Desktop Grid Systems’, in *15th International Conference on Computing*, IEEE, Nov. 2006, pp. 174–179, ISBN: 0-7695-2708-6.

[184] S. Choi, ‘Group-based adaptive scheduling mechanism in desktop grid’, PhD thesis, Korea University, 2007, p. 194.

[185] W. Jamroga, A. M?ski and M. Szreter, ‘Modularity and Openness in Modeling Multi-Agent Systems’, *Electronic Proceedings in Theoretical Computer Science*, vol. 119, pp. 224–239, Jul. 2013, ISSN: 2075-2180.

[186] M. Viroli and A. Omicini, ‘Specifying agent observable behaviour’, in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems Part 2 (AAMAS ’02)*, New York, New York, USA: ACM Press, Jul. 2002, p. 712.

[187] R. Dumke, S. Mencke and C. Wille, *Quality Assurance of Agent-Based and Self-Managed Systems*, 1st. Boca Raton, FL, USA: CRC Press, Inc., 2009, ISBN: 1439812667, 9781439812662.

[188] W. T. L. Teacy, J. Patel, N. R. Jennings and M. Luck, ‘TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources’, *Autonomous Agents and MultiAgent Systems*, vol. 12, pp. 183–198, 2006.

[189] D. D. Heckathorn, ‘Collective Action and the Second-Order Free-Rider Problem’, *Rationality and Society*, vol. 1, no. 1, pp. 78–100, Jul. 1989, ISSN: 1043-4631.

[190] G. Anders, F. Siefert, J.-P. Steghöfer and W. Reif, ‘Trust-Based Scenarios - Predicting Future Agent Behavior in Open Self-Organizing Systems’, in *Proceedings of the Seventh International Workshop on Self-Organizing Systems (IWSOS 2013)*, 2013.

[191] O. Shehory and K. Sycara, ‘Multi-agent coordination through coalition formation’, *Intelligent Agents IV Agent Theories*, Lecture Notes in Computer Science, vol. 1365, pp. 1–12, 1998.

[192] I. Gupta, R. van Renesse and K. P. Birman, ‘A Probabilistically Correct Leader Election Protocol for Large Groups’, in *Proceedings of the 14th International Conference on Distributed Computing*, Springer-Verlag, Oct. 2000, pp. 89–103, ISBN: 3-540-41143-7.

[193] A. Vasalou and J. Pitt, ‘Reinventing forgiveness: a formal investigation of moral facilitation’, in *Proceedings of the Third International Conference on Trust Management*, P. Herrmann, V. Issarny and S. Shiu, Eds., ser. Lecture Notes in Computer Science, vol. 3477, Berlin, Heidelberg: Springer Berlin Heidelberg, May 2005, pp. 146–160, ISBN: 978-3-540-26042-4.

[194] P. Tosic and G. Agha, ‘Maximal Clique Based Distributed Group Formation for Autonomous Agent Coalitions’, in *Proceedings of the Third International Joint Conference on Agents and Multi-Agent Systems, Coalitions and Teams Workshop (W10)*, 2004.

[195] A. Campbell and A. S. Wu, ‘Multi-agent role allocation: issues, approaches, and multiple perspectives’, *Autonomous Agents and Multi-Agent Systems*, vol. 22, no. 2, pp. 317–355, Apr. 2010, ISSN: 1387-2532.

[196] M. Litzkow, M. Livny and M. Mutka, ‘Condor-a hunter of idle workstations’, in *Proceedings of the Eighth International Conference on Distributed Computing Systems*, IEEE Computer Society Press, 1988, pp. 104–111, ISBN: 0-8186-0865-X.

[197] Space Sciences Laboratory at the Berkeley University of California, *The BOINC Project*. [Online]. Available: http://boinc.berkeley.edu/ (visited on 27/12/2013).

[198] Z. Patoli, M. Gkion, A. Al-Barakati, W. Zhang, P. Newbury and M. White, ‘How to build an open source render farm based on desktop grid computing’, in *Wireless Networks, Information Processing and Systems*, D. Hussain, A. Rajput, B. Chowdhry and Q. Gee, Eds., vol. 20, ser. Communications in Computer and Information Science, Springer Berlin Heidelberg, 2009, pp. 268–278, ISBN: 978-3-540-89852-8.

[199] D. Kondo, G. Fedak, F. Cappello, A. A. Chien and H. Casanova, ‘Characterizing resource availability in enterprise desktop grids’, *Future Generation Computer Systems*, vol. 23, no. 7, pp. 888–903, Aug. 2007, ISSN: 0167739X.

[200] C. Anglano, J. Brevik, M. Canonico, D. Nurmi and R. Wolski, ‘Fault-aware scheduling for Bag-of-Tasks applications on Desktop Grids’, in *2006 Seventh IEEE/ACM International Conference on Grid Computing*, IEEE, 2006, pp. 56–63, ISBN: 1-4244-0343-X.

[201] D. Kondo and H. Casanova, ‘Computing the optimal makespan for jobs with identical and independent tasks scheduled on volatile hosts’, Technical Report CS2004-0796, Dept. of Computer Science and Engineering, University of California at San Diego, San Diego, Tech. Rep., 2004, pp. 1–9.

[202] H. Tianfield and R. Unland, ‘Towards self-organization in multi-agent systems and Grid computing’, *Multiagent Grid Systems*, vol. 1, no. 2, pp. 89–95, 2005, ISSN: 1574-1702.

[203] K. Pruhs, J. Sgall and E. Torng, ‘Online Scheduling’, in *Handbook of Scheduling: Algorithms, models, and performance analysis*, J. Y.-T. Leung and J. H. Anderson, Eds., CRC Press, Boca Raton, 2004, pp. 196–231, ISBN: 1584883979.

[204] L. F. G. Sarmenta, ‘Volunteer Computing’, PhD thesis, Massachusetts Institute of Technology, 2001, p. 216.

[205] F. Araujo and P. Domingues, ‘Security and Result Certification’, in *Desktop Grid Computing*, C. Cérin and G. Fedak, Eds., ser. Chapman & Hall/CRC Numerical Analy & Scient Comp. Series, Chapman and Hall/CRC, Jun. 2012, pp. 211–235, ISBN: 978-1-4398-6214-8.

[206] K. Budati, J. Sonnek, A. Chandra and J. Weissman, ‘RIDGE: Combining Reliability and Performance in Open Grid Platforms’, in *Proceedings of the 16th international symposium on High performance distributed computing - HPDC '07*, New York, New York, USA: ACM Press, Jun. 2007, p. 55, ISBN: 9781595936738.

[207] P. Cremonesi and R. Turrin, ‘Performance models for desktop grids’, in *Proceedings of the 10th International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, Citeseer, 2007.

[208] F. Brasileiro and N. Andrade, ‘Open, Scalable and Self-Regulated Federations of Desktop Grids with OurGrid’, in *Desktop Grid Computing*, C. Cérin and G. Fedak, Eds., ser. Chapman & Hall/CRC Numerical Analy & Scient Comp. Series, Chapman and Hall/CRC, Jun. 2012, pp. 29–51, ISBN: 978-1-4398-6214-8.

[209] B. Awerbuch, ‘Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems’, in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, ser. STOC '87, New York, NY, USA: ACM, 1987, pp. 230–240, ISBN: 0-89791-221-7.

[210] R. Kiefhaber, G. Anders, F. Siefert, T. Ungerer and W. Reif, ‘Confidence as a means to assess the accuracy of trust values’, in *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2012, pp. 690–697.

[211] C. D. Hollander and A. S. Wu, ‘The current state of normative agent-based systems’, *Journal of Artificial Societies and Social Simulation*, vol. 14, no. 2, 2011.

[212] H. W. Kuhn, ‘The Hungarian method for the assignment problem’, *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955, ISSN: 00281441.

[213] R. E. Burkard and E. Cela, ‘Linear Assignment Problems and Extensions’, *Handbook of Combinatorial Optimization*, vol. 4, pp. 1–54, 1999.

[214] L. Ramshaw and R. E. Tarjan, ‘A Weight-Scaling Algorithm for Min-Cost Imperfect Matchings in Bipartite Graphs’, in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, IEEE, Oct. 2012, pp. 581–590, ISBN: 978-0-7695-4874-6.

[215] A. Kumar, ‘A modified method for solving the unbalanced assignment problems’, *Applied Mathematics and Computation*, vol. 176, no. 1, pp. 76–82, 2006, ISSN: 0096-3003.

[216] E. Gamma, R. Helm, R. Johnson and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Addison-Wesley Professional, 1994, ISBN: 0201633612.

[217] S. Poslad, P. Charlton and M. Calisti, ‘Specifying standard security mechanisms in multi-agent systems’, in *Proceedings of the 2002 International Conference on Trust, Reputation, and Security: Theories and Practice*, ser. AAMAS'02, Berlin, Heidelberg: Springer-Verlag, 2003, pp. 163–176, ISBN: 3-540-00988-4.

[218] F. Mármol and G. M. Gómez Pérez, ‘Security threats scenarios in trust and reputation models for distributed systems’, *Computers & Security*, vol. 28, no. 7, pp. 545–556, 2009.

[219] S. Hernan, S. Lambert, T. Ostwald and A. Shostack, ‘Uncover Security Design Flaws Using The STRIDE Approach’, *MSDN Magazine*, 2006.

[220] K. Lee, J. Y.-T. Leung and M. L. Pinedo, ‘Makespan minimization in online scheduling with machine eligibility’, *4OR*, vol. 8, no. 4, pp. 331–364, Nov. 2010, ISSN: 1619-4500.

[221] C. Jiang, C. Wang, X. Liu and Y. Zhao, 'A survey of job scheduling in grids', in *Advances in Data and Web Management, Proceedings of the Joint Ninth Asia-Pacific Web and Eighth International Conference on Web-Age Information Management*, G. Dong, X. Lin, W. Wang, Y. Yang and J. Yu, Eds., Springer Berlin / Heidelberg, Jun. 2007, pp. 419–427, ISBN: 978-3-540-72483-4.

[222] K. Shudo, Y. Tanaka and S. Sekiguchi, 'P3: p2p-based middleware enabling transfer and aggregation of computational resources', in *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid '05)*, vol. 1, 2005.

[223] A. Chien, 'Entropia: architecture and performance of an enterprise desktop grid system', *Journal of Parallel and Distributed Computing*, vol. 63, no. 5, pp. 597–610, 2003, ISSN: 07437315.

[224] D. Kondo, A. Chien and H. Casanova, 'Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids', in *Proceedings of the ACM/IEEE SC2004 Conference*, IEEE, 2004, p. 14, ISBN: 0-7695-2153-3.

[225] D. Zhou and V. Lo, 'WaveGrid: a scalable fast-turnaround heterogeneous peer-based desktop grid system', in *Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium*, IEEE, 2006, ISBN: 1-4244-0054-6.

[226] H. Wang, H. Takizawa and H. Kobayashi, 'A dependable Peer-to-Peer computing platform', *Future Generation Computer Systems*, vol. 23, no. 8, pp. 939–955, 2007.

[227] D. Kondo, D. P. Anderson and J. M. Vii, 'Performance Evaluation of Scheduling Policies for Volunteer Computing', in *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, IEEE, 2007, pp. 415–422, ISBN: 0-7695-3064-8.

[228] Q. Zhu and G. Agrawal, 'Supporting fault-tolerance for time-critical events in distributed environments', in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*, New York, New York, USA: ACM Press, Nov. 2009, p. 1, ISBN: 9781605587448.

[229] Y. Yu and H. Jin, 'A workload balancing based approach to discourage free riding in Peer-to-Peer network', *Advances in Web and Network Technologies, and Information Management*, vol. 4537, pp. 144–155, 2007.

[230] E. Jen, 'Stable or robust? What's the difference?', *Complexity*, vol. 8, no. 3, pp. 12–18, 2003.

[231] S. Ali, A. Maciejewski and H. Siegel, 'Measuring the robustness of a resource allocation', *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, Jul. 2004, ISSN: 1045-9219.

[232] J.-S. Kim, B. Nam, M. Marsh, P. Keleher, B. Bhattacharjee, D. Richardson, D. Wellnitz and A. Sussman, 'Creating a Robust Desktop Grid using Peer-to-Peer Services', in *2007 IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2007, pp. 1–7, ISBN: 1-4244-0909-8.

Appendices

A | TC Strategy Algorithm Details

A.1 Basic Distributed Leader Election Strategy

This section provides details about the implementation of the basic leader election strategy introduced in Sec. 4.6.5: The algorithm 1 following on the next page details the *Highest TC reputation Election Strategy*. Note that for the sake of comprehensibility, code addressed at verifying that no agent has left the system during the election process is left out here. The same applies to code that ensures asynchrony of the algorithm (round message receive order consideration and caching).

The algorithm is split up in three rounds, each round again consisting of up to three actions an agent $a_i \in \mathcal{M}_{TC_i}(t)$ performs: (1) Receive messages, (2) make local computations and (3) send messages. The algorithm is executed periodically for as long as it returns *UNDECIDED*, each time entering in one of the rounds. In the first round, a_i determines (based on ids) the agent a_r^i. This is the agent the electing agent a_i is responsible for. This means that the agent a_i collects the direct trust values of all other agents about this agent in order to compute its reputation among the TC members. In order to allow for all agents computing such a value for their responsible agents, the agent must send its own direct trust values $DT_{a_i}^{a_x,\hat{c}}$ about each other agent $a_x \in \mathcal{M}_{TC_i}(t)$ to the respective responsible agent (determined accordingly to own responsibility). Each agent a_i then receives messages containing direct trust values $DT_{a_x}^{a_r,\hat{c}}$ about the agent a_r^i it is responsible for, sent by the all other agents a_x. When a_i has received messages from all agents except a_r, it enters round 2. In round 2, a_i can compute the average direct trust value $RT_{\mathcal{M}_{TC_i}(t)\backslash\{a_r\}}^{a_r,\hat{c}}$ for a_r by evaluating the messages sent by all other agents and its own direct trust value for a_r^i. It then sends messages with this value to all other agents (this time including a_r). Also, other average direct trust messages are received - if messages from all agents were received, a_i enters round 3. In the final round, the agent a_{TCM} is determined by scanning all received messages and the average direct trust value for a_r^i for the highest average direct trust value $RT_{\mathcal{M}_{TC_i}(t)\backslash\{a_{TCM}\}}^{a_{TCM},\hat{c}}$. Based on the comparison whether a_i equals a_{TCM}, each agent can then go into one of the two state *decided(TCM)* or *decided(member)*.

This algorithm has a time complexity of $\mathcal{O}(n)$ (3 rounds, local computation dependent on electing member number) and a message complexity of $\mathcal{O}(2 \cdot n \cdot (n-1)) = \mathcal{O}(n^2)$. Although algorithms with better message complexities are known - these do not incorporate the two rounds of message sending necessary here - the number of electing members n is usually limited (see discussion in Sec. 4.6.6), which renders the introduced message overhead acceptable.

Note that this algorithm implies that at the end of the execution each member knows whether it has been elected as a TCM, and - if not - it can identify the TCM without further communication.

Algorithm 1 Highest TC reputation Election Strategy

```
Require: Own unique identifier $a_i$ of the executing agent
Require: Round variable initialised: $ROUND := 0$
Require: Set $\mathcal{M}_{TC_i}(t) = \left\{a_1, .., a_{|\mathcal{M}_{TC_i}(t)|}\right\}$ of electing TC members with an unique identifiers
Require: Set $\left\{DT_{a_i}^{a_1,\hat{c}}, .., DT_{a_i}^{a_{|\mathcal{M}_{TC_i}(t)|},\hat{c}}\right\}$ of direct trust values towards all agents $a_x \in \mathcal{M}_{TC_i}(t) \setminus \{a_i\}$

Each electing agent $a_i$ executes the following code:
if $ROUND = 0$ then
    Determine responsible agent $a_r^i$ with $r$ being the next highest id after $i$ or the smallest id if $i$ is the largest.
    for all $a_j \in \mathcal{M}_{TC_i}(t), j \neq i \wedge j \neq r$ do
        Determine responsible agent $a_j$ for agent $a_r^j$
    end for
    $ROUND := 1$
    return UNDECIDED
end if
if $ROUND = 1$ then
    if NOT sent $|\mathcal{M}_{TC_i}(t)| - 1$ distinct $msg^1$-messages then
        for all $a_j \in \mathcal{M}_{TC_i}(t), j \neq r$ do
            Send $msg_{a_j}^1(DT_{a_i}^{a_r^j,\hat{c}})$ with direct trust value for agent $a_r^j$ to its responsible agent $a_j$
        end for
    else if received $|\mathcal{M}_{TC_i}(t)| - 2$ distinct $msg^1$-messages then
        Compute TC reputation $RT_{\mathcal{M}_{TC_i}(t)\setminus\{a_r^i\}}^{a_r^i,\hat{c}}$ for responsible agent $a_r^i$, based on received values $DT_{a_j}^{a_r^i,\hat{c}}$ from all agents $a_j \in \mathcal{M}_{TC_i}(t), j \neq r \wedge j \neq i$
        $ROUND := 2$
        return UNDECIDED
    end if
else if $ROUND = 2$ then
    if NOT sent $|\mathcal{M}_{TC_i}(t)| - 1$ distinct $msg^2$-messages then
        for all $a_j \in \mathcal{M}_{TC_i}(t), j \neq i$ do
            Send $msg_{a_j}^2(T_r^{avg})$ containing the average trust value of the responsible agent $T_r^{avg}$
        end for
    else if received $|\mathcal{M}_{TC_i}(t)| - 1$ distinct $msg^2$-messages then
        Determine the id $j = TCM$, with the agent $a_j$ being the member with the highest average trust value, $max\left(RT_{\mathcal{M}_{TC_i}(t)\setminus\{a_1\}}^{a_1,\hat{c}}, .., RT_{\mathcal{M}_{TC_i}(t)\setminus\left\{a_{|\mathcal{M}_{TC_i}(t)|}\right\}}^{a_{|\mathcal{M}_{TC_i}(t)|},\hat{c}}\right)$
        $ROUND := 3$
        return UNDECIDED
    end if
else if $ROUND = 3$ then
    if $TCM = i$ then
        return DECIDED(TCM)
    else
        return DECIDED(MEMBER)
    end if
end if
```

A.2 Basic Role Assignment Strategy

In Sec. 4.6.8, the implementation of a basic role assignment strategy based on the formulation as *linear assignment problem* has been discussed. The following provides a short review on related work on this problem and discusses how it can be applied for the assignment of roles to TC members by a TCM.

The *(linear) assignment problem* comes in two variants: The *balanced assignment problem (BAP)* describes the problem to assign n roles to n agents, such that each agent is executing only one role and the total costs for the execution of all roles are minimised. This special case in the role assignment can be solved for example by application of the *Hungarian Method* (cf. [212]) with a time complexity of $O(n^4)$, or its modification reducing it to $O(n^3)$ (cf. e.g. [213]). The application of this algorithm requires the formulation of a *cost matrix*:

Let $\left|\Psi_{TC_i}(t)\right| = n$ be the number of roles to assign, $\left|\mathcal{M}_{TC_i}(t)\right| = k$ the number of TC members to assign the roles to, and c_{m_i,r_j} be the costs for agent m_i to act in the role r_j, then the *cost matrix* for the problem is:

$$\begin{array}{c|cccc|}
 & r_1 & r_2 & \dots & r_n \\
m_1 & c_{m_1,r_1} & c_{m_1,r_2} & \dots & c_{m_1,r_n} \\
m_2 & c_{m_2,r_1} & c_{m_2,r_2} & \dots & c_{m_2,r_n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
m_k & c_{m_k,r_1} & c_{m_k,r_2} & \dots & c_{m_k,r_n}
\end{array}$$

Let additionally a_{m_i,r_j} be a binary variable denoting whether m_i is assigned r_j ($a_{m_i,r_j} = 1$) or not ($a_{m_i,r_j} = 0$). If $n = k$, the following steps of the *Hungarian Method* (cf. [212]) will then minimise the costs

$$\sum_{i=1}^{k} \sum_{j=1}^{n} a_{m_i,r_j} \cdot c_{m_i,r_j}$$

for the assignment of each role to exactly one agent:

1. Subtract the smallest entry c_{m_i,r_j} in each row i from the other entries in this row.
2. Subtract the smallest entry c_{m_i,r_j} in each column j from the other entries in this column. The resulting matrix is called *opportunity cost matrix*.
3. Check if optimal assignment can be made: Cover all rows i and columns j containing a 0 with a minimum number of lines possible. If the number of obtained covered rows and columns is equal to the number of rows (or columns) stop, else continue with the next step.
4. Revise *opportunity cost matrix* by subtracting the minimum uncovered entry from each uncovered entry and adding it to each entry covered by two lines. Go back to step 3.

Note here that if the costs for executing a role are identical for each agent, this problem is trivial as the minimum is reached for any assignment of roles. This is however a special case for *BAP*.

So far, the case of matching a set of roles with an equally sized set of agents has been covered. However, the number of roles cannot be expected to match the number of TC members in general.

In this second variant, referred to as *unbalanced assignment problem (UBAP)*, the problem is hence to assign n roles to k agents with $n \neq k$. Here, two cases are discerned:

- $n > k$: There are more roles than available TC members. Some (if not all) agents must be assigned more than one role such that the summed assignment costs are minimised.
- $n < k$: There are less roles than available TC members. Not all members will be assigned a role. The assignment must be done such that the summed assignment costs are minimised.

The latter case can be trivially converted to a *BAP* by filling up the n roles with d dummy roles that are assigned costs of 0, such that $n + d = k$ (cf. e.g. [213]). This can then again be solved by the *Hungarian Method* as described above. In the resulting assignments, some agents do not have any roles assigned to them, which is not a problem for the operation of a TC. If however the first case is encountered, such that there are more roles than agents, the solution of adding dummy agents is not acceptable, as it leaves roles unassigned. There are two types of solutions to this problem: First, the division of management tasks to roles can be re-executed. In general, this will mean to iteratively concentrate two excess roles r_i and r_j to a single role r_k until the number of roles in $\Psi_{TC_i}(t)$ matches the number of members again. As the roles are containing associated tasks, a new role r_k combines the execution of the tasks for both roles. The costs for the execution of this role by an agent m_a can hence be assumed to be the sum of the costs of these roles, such that: $c_{m_a,r_k} = c_{m_a,r_i} + c_{m_a,r_j}$. The balancing of the number of roles to the number of TC members is however problematic in a highly dynamic TC: Each time an agent leaves the TC or is excluded from it by the TCM, a re-balancing needs to be executed and the entire set of roles must be assigned again. This can be acceptable in environments where the TC composition does not change often, in general however the assignment of roles to agents should also be allowed for the unbalanced case where there are more roles than agents. This is hence the second approach to this case. In the literature some approaches to this case have been proposed. For example in [214] a very deep analysis of the UBAP is presented, and modifications of known algorithms for BAP to the application for UBAP are discussed. The algorithm *FlowAssign*, being a modification of the *Hungarian*-algorithm, is then proposed and the performance analysed. Another solution is proposed in [215], describing a 18-step algorithm again modifying the *Hungarian*-algorithm. With such an *UBAP*-capable algorithm, the number of roles can exceed the number of agents and still a minimum cost assignment can be computed. This assignment does not involve the restructuring of roles and can be executed even for TCs with high composition dynamics.

In summary, if the role model allows to formulate a *cost matrix* to quantify the costs for the execution of a role by an agent, then a *BAP* or *UBAP*-algorithm can implement the *Role Assignment* strategy. As long as the number of roles is smaller or equal to the number of agents in a TC, the *Hungarian Method* can be applied for a quick and cost optimal solution. The output of the strategy for the case $|\Psi_{TC_i}(t)| \leq |\mathcal{M}_{TC_i}(t)|$ will then be a set of tuples

$$\left\{ (m, \Psi_m(t)) \;\middle|\; m \in \mathcal{M}_{TC_i}(t), \Psi_m(t) \subseteq \Psi_{TC_i}(t) \right\}$$

with each set of roles assigned to a member m containing either one or no role, such that $|\Psi_m(t)| \leq 1$. If such a balancing of roles cannot be performed by the TCM, either because the costs of this operation are too high due to the TC dynamics, or because the roles do not allow for (further) concentration, the implementation of this strategy must use a *UBAP*-algorithm (e.g. the ones in [214] or [215]).

B | The Design of the TC Organisation Agent Component

In the thesis, the TC approach has been discussed from a functional view: The dynamics of the TC have been introduced with organisation phases constituting a TC lifecycle (as depicted in Fig. 4.1), and the decision-making strategies required by the approach have been assigned to each phase. This functional view, detached from the agent model (see Sec. 3.1.2) and TC organisation agent component (see Sec. 3.2.3), is abandoned here in favour of a complete consideration of the architecture and workflows of the TC decision-making within an agent.

The design of the TC organisation agent component $Comp^x_{TCO}$ is based on *states*, as defined in the well-known behavioural design pattern *State Pattern* (cf. e.g. [216]). Here, a state encapsulates all TC-related agent behaviours, esp. the execution of the TC strategies, required by an agent in a certain situation. The TC organisation component is always in a current state and is invoked periodically. An invocation executes the workflow defined in the state. Based on the actions and decisions within the workflow, the state is either maintained, or the transition to a next state is performed.

A state-based agent component $Comp^x_{TCO}$ has several advantages: Firstly, it allows for a clean encapsulation of agent behaviours that are exchangeable at runtime. This supports the concept of the Observer/Controller-architecture in that it allows to exchange the implementation of TC strategies based on observations of the environment. Consider for example a TCM that adapts its TC regulation by situation-aware adaptations of TC-management strategies. Secondly, this encapsulation allows to define a set of observables O^x_{TCO} for exactly the information needed to execute the workflow within a state. If for example an agent is unassociated, it requires different information than an agent being the TCM of a Trusted Community.

All states that the $Comp^x_{TCO}$ can adopt are based on the separation of the decision-making required for the TC approach. The assignment to separate states is based on the *association status* of agents. This status is defined as follows: Each agent in the society of the hosting system is always either a member of a Trusted Community, or unassociated. This is represented with two according states, the *Unassociated Open*-state and the *TC Member*-state. The *Unassociated Open*-state is the initial state for agents entering the system. Here, agents process all decisions regarding association status changes, i.e. they reason about accepting TC membership invitations, forming a TC etc. The *TC member state* is entered by agents when they form a TC as initiating agent, or are granted TC membership by a TCM. Here, all behaviours regarding interactions with other members and the TCM are encapsulated. In addition, each TC member can be elected as manager of a TC, which requires

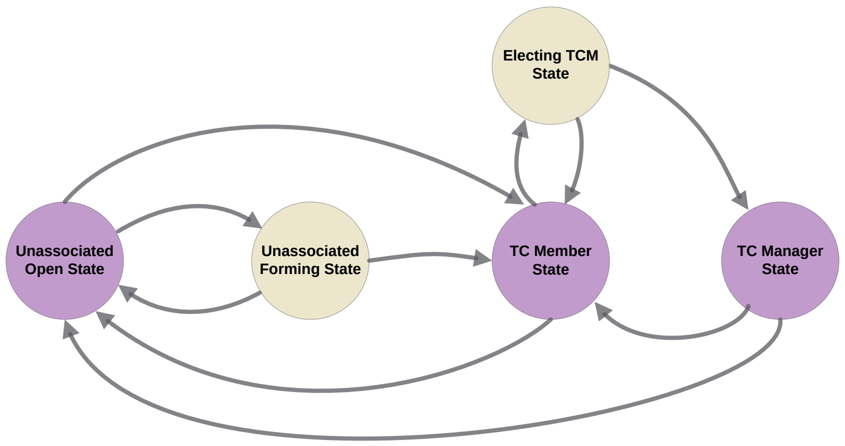
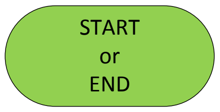


Figure B.1: States of the TC organisation agent component and their transitions. The three long-term states are depicted in darker colour, while the transitional states are depicted in lighter colour.

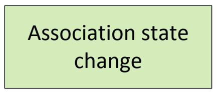
a third *TC Manager*-state to encapsulate TCM functionality. These three states are in general maintained long-term by the agents that is, the agents usually do not switch between them frequently. If, for example, being unassociated, and then becoming a TC member, an agent will usually not leave the TC (and thus become unassociated again) for at least the amount of time it takes to make a thorough assessment of the experienced utility.

Besides long-term states, the behaviour assignment is further refined by two additional, transitional states. These states encapsulate behaviours required only for a limited amount of time: The *Unassociated Forming*-state and the *Electing TCM*-state. The *Unassociated Forming*-state is entered when an agent decided to either initiate or join the formation of a Trusted Community, and abandoned after the formation is either successful or fails. It serves the purpose to capture behaviours that are solemnly related to decisions regarding the formation, like the invitation process of potential members, negotiations between them etc. The *Electing TCM*-state is entered by members of a TC whenever there is no TC Manager, thus after TC formation or when the previous TCM left the TC. Agents leave this state after a new TCM has been elected. All valid transitions between the five states of the TC organisation component are depicted in Fig. B.1.

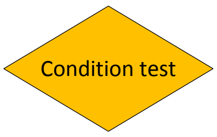
In the following, the detailed, generic workflow within the $\mathcal{C}omp^{x}_{TCO}$ states is presented. This presentation starts with the long-term states and is followed by the transitional states. Workflows are depicted with flow charts utilising the following syntax:



Start/End symbol to show the entry/exit point of the states. State transitions are explicit via association change actions, thus if no change action was called, the agent will begin the next run at the start symbol of the current state. In case of a transition, the agent will begin the next run at the start symbol of the new state.



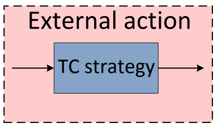
Explicit **association state change actions**, these change the current association state of the agent to the given association state.



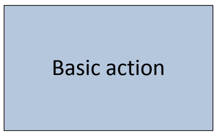
Condition tests to determine the valid branch in a workflow.

TC strategy

Strategy actions execute the TC strategies as depicted in Fig. 4.1. These are the most important and configurable building blocks of the workflow within the states. The utilisation of these strategies requires precisely defined inputs and outputs (which are described in more detail in the following section).



External actions are actions by other agents (especially a TCM) that influence the workflow of the executing agent.



Basic actions define management activities that are necessary, but not expensive and variable, thus not configurable like strategy actions.

Unassociated Open State

Each agent in the agent society of the hosting system starts its participation in the unassociated state. In this state, an agent determines whether to remain unassociated or become a the member of a TC. The workflow, depicted in Fig. B.2, starts with the execution of the potential member search strategy in order to analyse the trust relationships of this agent towards other agents in the society. The output of this analysis is a set of agents that are trustworthy enough to rely on them as fellow TC members. This is followed by the execution of the association evaluation strategy. Here, the agent decides whether to remain unassociated, initiate the formation of a new TC, follow the invitation to join the formation of a new TC (not as initiator), or join an inviting TC as member. The last two decisions require invitations from other agents. The according messages are hence processed in the association evaluation strategy. If this strategy determines that the agent should follow an invitation, this invitation is accepted, all other invitations are rejected, and the agent changes the association state of its TC organisation component. The first two decisions are based on the analysis of the set of trustworthy agents determined by the first strategy. If this set is auspicious, the association evaluation strategy determines to form a new TC, leading to a change of states to the unassociated forming state. If the agent trusts only few other agents and has not received any convincing invitations, it remains unassociated and can eventually leave this state only if its environment changes, for example if it receives an invitation.

TC Member State

The TC organisation component changes its state to the TC member state when an agent is granted membership by a TCM or through the formation of a new TC. A Trusted Community must be managed by a TCM, hence the first test in the workflow depicted in Fig. B.3 is whether the associated TC has a TCM. In case this does not apply, the agent informs the other members about it and changes the state of the TC organisation component to the transitional state of TC election. The following steps in the workflow hence assume that there is a TCM managing the TC. The workflow is continued by the execution of the membership evaluation strategy. Here, the benefit of the TC membership is analysed: If the strategy determines that the agent has no utility gain in being a member, the membership is cancelled. This action involves the notification of the TCM, which can in turn inform the other members about the leaving agent. The agent then changes its state back to the unassociated open state. Apart from own reasons to leave a TC, the member can also receive instructions from its TCM to leave the TC. Such instructions arise from the regulatory reasoning of the TCM and come into effect when the member has shown adversary behaviours, or when the TC is dissolved.

If the membership evaluation strategy determines that the agent shall remain member and the TCM does not object, the workflow continues with the processing of updates regarding the TC composition. These updates contain information about new and leaving members. Subsequently, the agent processes information about updates on roles assigned to it by the TCM. Finally, the member agent can execute interactions based on the organisation benefit strategies and/or execute tasks delegated to it by role assignment. This completes the workflow of a TC member.

TC Manager State

The TCM state is initiated in the TC organisation component only when an agent becomes the TC manager of an associated TC. This status is granted by election in the according state. The TC manager state is the last of the long-term states, in general preserved by agents over a longer period in time. The workflow depicted in Fig. B.4 starts with the test whether it is executed for the first time by the agent. In this case, the TCM performs initialising actions such as the information exchange with a former TCM (which is possible only in case the former TCM is known and still online).

A usual workflow of a TCM starts with the execution of the TC observer which generates an up-to-date situation description of the TC. The TC controller is then executed to adapt the configuration of the TCM according to the observed situation. The most influencing reasoning in this loop regards the decision whether the operation of the TC cannot be maintained any more. The criteria for this condition are dependent on the exact realisation of the TC controller, however the most obvious criterion is a minimum number of members. In case the controller determines to dissolve the TC, the TCM first informs its members about this. This step has been referred to as external action in the TC members state as depicted in Fig. B.3. In addition to the notification of its members, the TCM informs the agent society about the dissolution of the TC. Finally, the TCM ends its own membership and the TC organisation component initiates a state change to the unassociated open state.

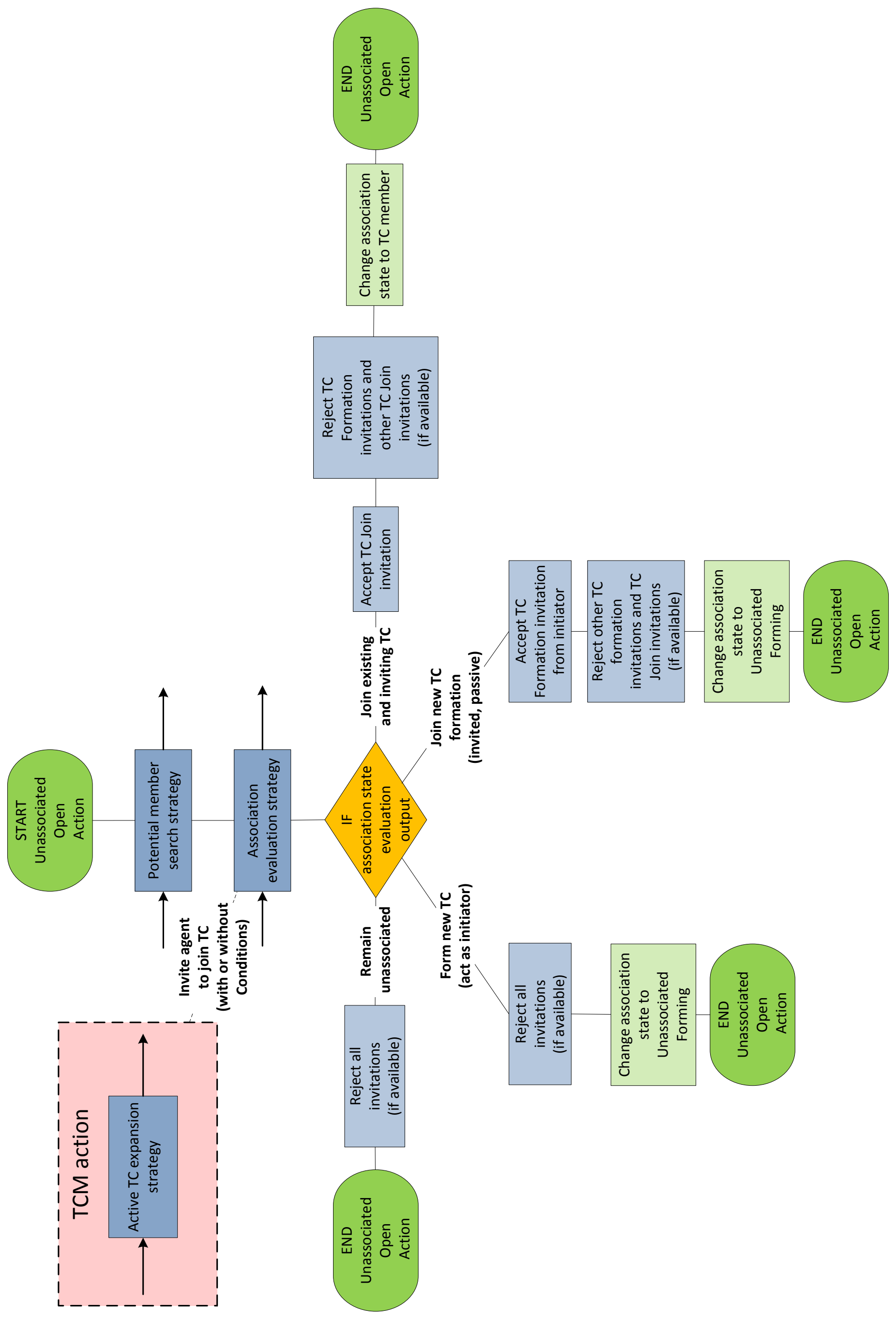

START Unassociated Open Action
Potential member search strategy
Association evaluation strategy
IF association state evaluation output
Join existing and inviting TC
Accept TC Join invitation
Reject TC Formation invitations and other TC Join invitations (if available)
Change association state to TC member
END Unassociated Open Action
Join new TC formation (invited, passive)
Accept TC Formation invitation from initiator
Reject other TC formation invitations and TC Join invitations (if available)
Change association state to Unassociated Forming
END Unassociated Open Action
Form new TC (act as initiator)
Reject all invitations (if available)
Change association state to Unassociated Forming
END Unassociated Open Action
Remain unassociated
Reject all invitations (if available)
END Unassociated Open Action
Invite agent to join TC (with or without Conditions)
TCM action
Active TC expansion strategy

Figure B.2: Workflow of the **Unassociated Open State** and composing basic actions, condition test, strategy actions, and state change actions. The key part of the workflow is the evaluation of the association decision as obtained from the execution of the according strategy.

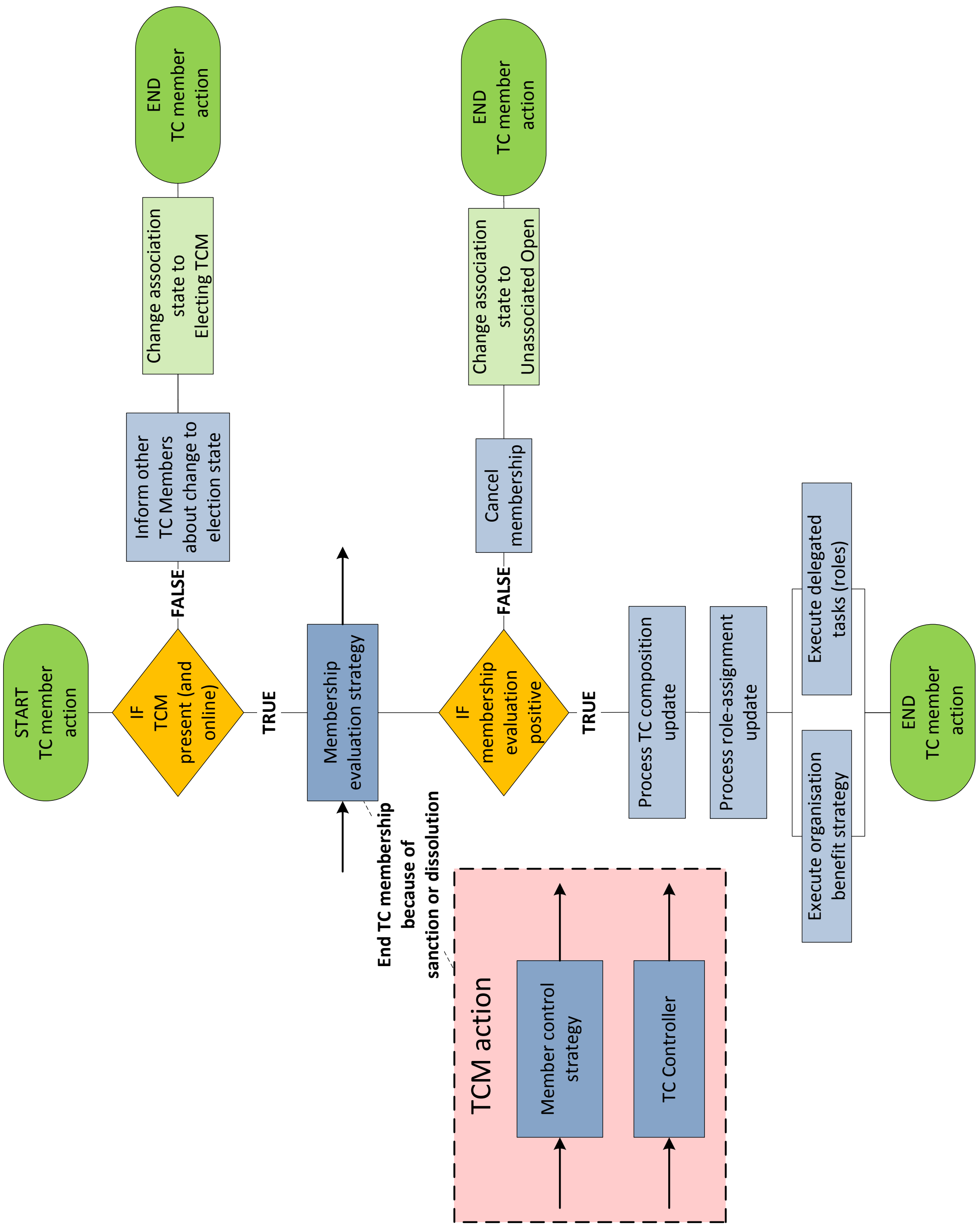

START TC member action
IF TCM present (and online)
FALSE
Inform other TC Members about change to election state
Change association state to Electing TCM
END TC member action
TRUE
Membership evaluation strategy
End TC membership because of sanction or dissolution
IF membership evaluation positive
FALSE
Cancel membership
Change association state to Unassociated Open
END TC member action
TRUE
Process TC composition update
Process role-assignment update
Execute organisation benefit strategy
Execute delegated tasks (roles)
END TC member action
TCM action
Member control strategy
TC Controller

Figure B.3: Workflow of the **TC Member State** with composing actions and condition tests. Key parts of the workflow are the check for the TCM presence, as well as the execution of TC member actions after a check of the membership benefits.

In the usual case, the TC can maintain its operation and must be managed by the TCM. The decision-making for this is divided into two subsequent blocks in the workflow: First, the current TC memberships are managed. Here, the TCM applies the member control, as well as role-assignment strategies. The purpose of this execution is to detect and sanction undesired member behaviour, to (re-)assign management roles to members, to process messages about agents leaving the TC, and finally to inform the members about this. The second block regards non-members and is constituted by the execution of the active TC expansion strategy to recruit new members from the TC, and related actions (invitation with and without conditions). Consequently, invitation answers are processed, and in case of positive response, the unassociated senders are made members by the TCM. Finally, a TCM is always also a TC member, hence it additionally executes (parts of) the TC member state, most importantly interactions based on the organisation benefit strategies. This concludes the TCM state workflow.

Unassociated Forming State

The unassociated forming state is a transitional state which in entered by the TC organisation component when an agent in the unassociated open state decides to initiate the formation of a new TC, or join such a formation when invited. The relevant workflow, as depicted in Fig. B.6, is divided according to these two cases. If the agent in this state is the initiator of the formation, it starts the execution with a check whether it has already sent invitations to the potential members. If this is the case, the initiator executes the initiation strategy with the invitations answers as input. The strategy determines whether the accepting agents constitute a suitable TC composition, in which case the initiator informs the other agents about the successful formation, advertises the new TC in the agent society and changes the state of its component to the TC member state. Otherwise, the formation is a failure and the initiator likewise informs the other agents accordingly and changes its state back to the unassociated open state. Agents not initiating, but joining a TC formation have already informed the initiator about the accepted formation invitation (this triggers the transition to this state as depicted in Fig. B.2). They hence remain in this state checking whether the initiator of the formation has decided about the success of the formation. Finally, the processing of the decision allows to change the state to either TC member state (formation success) or to unassociated open state (formation failure).

Electing TCM State

Finally, the last state of the TC organisation is the transitional Electing TCM state. It is triggered by TC members upon detection of the absence of an (online) TCM in the TC, as depicted in Fig. B.3. The workflow of this state, depicted in Fig. B.6, starts with the check if all other members of the associated TC are already in the election state. This is necessary, as in general the election of a TCM may require some amount of synchronisation between the electing agents and the time the agents become aware of the necessity to elect a TCM varies. The workflow is hence postponed until all members are in this state. If all agents entered the state, the election is undecided and the distributed leader election strategy is executed for the first time. Normally, such a strategy operates in rounds and the execution ends with either of the three results: election undecided, election decided (not elected as leader), and election decided (elected as leader). In case one of the latter results is obtained from the strategy, the state is changed to either TC member state, or to TCM state (with

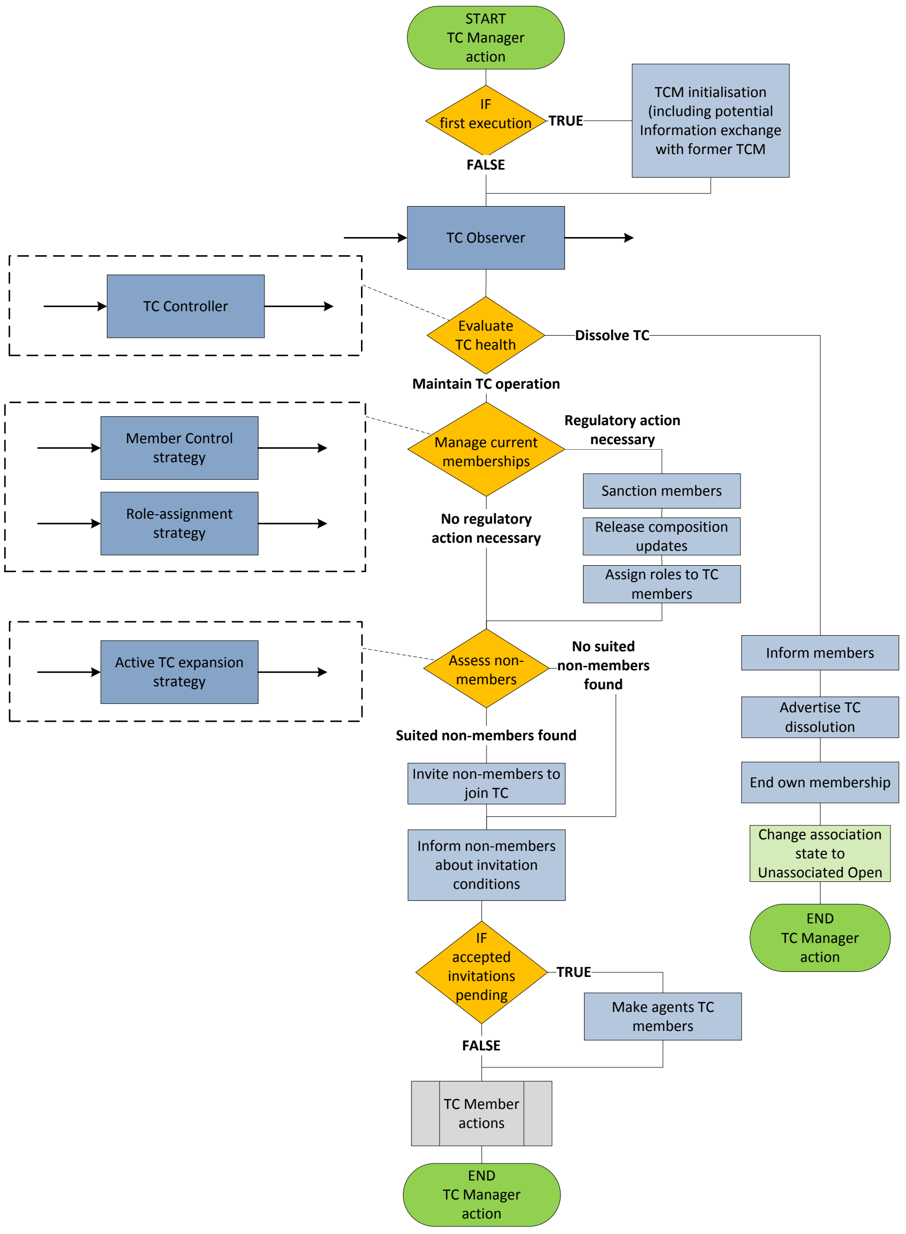

START
TC Manager
action
IF
first execution
TRUE
TCM initialisation (including potential Information exchange with former TCM
FALSE
TC Observer
TC Controller
Evaluate
TC health
Dissolve TC
Maintain TC operation
Member Control strategy
Role-assignment strategy
Manage current memberships
Regulatory action necessary
Sanction members
Release composition updates
Assign roles to TC members
No regulatory action necessary
Active TC expansion strategy
Assess non-members
No suited non-members found
Inform members
Advertise TC dissolution
End own membership
Change association state to Unassociated Open
END
TC Manager
action
Suited non-members found
Invite non-members to join TC
Inform non-members about invitation conditions
IF
accepted invitations pending
TRUE
Make agents TC members
FALSE
TC Member actions
END
TC Manager
action

Figure B.4: Workflow of the **TC Manager State**, and composing actions and condition tests. The key parts are the execution of the TCM O/C-loop, the separate assessment of TC members and unassociated agents, and the execution of the TC members state.

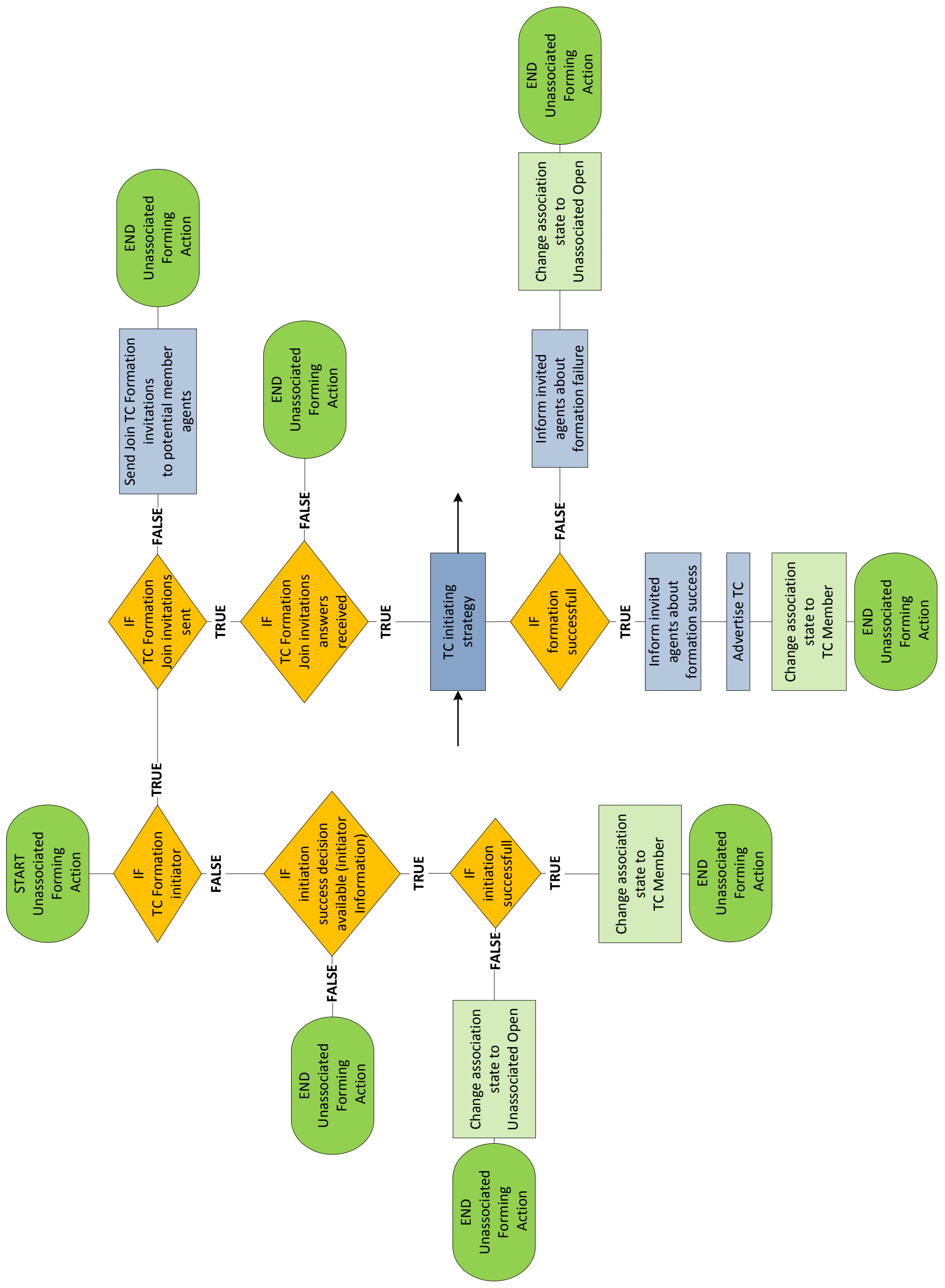

START Unassociated Forming Action
IF TC Formation initiator
TRUE
FALSE
IF TC Formation Join invitations sent
FALSE
Send Join TC Formation invitations to potential member agents
END Unassociated Forming Action
TRUE
IF TC Formation Join invitations answers received
FALSE
END Unassociated Forming Action
TRUE
TC initiating strategy
IF formation successfull
FALSE
Inform invited agents about formation failure
Change association state to Unassociated Open
END Unassociated Forming Action
TRUE
Inform invited agents about formation success
Advertise TC
Change association state to TC Member
END Unassociated Forming Action
IF initiation success decision available (initiator Information)
FALSE
END Unassociated Forming Action
TRUE
IF initiation successfull
FALSE
Change association state to Unassociated Open
END Unassociated Forming Action
TRUE
Change association state to TC Member
END Unassociated Forming Action

Figure B.5: Workflow of the **Unassociated Forming State**, and composing actions and condition tests. The workflow is divided into two main parts for the separate execution by the initiator of the formation and agents that joined the formation.

prior information of the other electing agents about the result) with the next execution of the workflow. If the result is undecided, the execution of the strategy is performed for another round and the state is not left.

Discussion and Summary

The visualisation of the workflows makes the following implicit assumptions for the sake of comprehensibility:

The execution of the workflows within the states is performed periodically. In Sec. 5.1.2, it is formalised that the notion of *time* is understood as a discrete sequence of time steps t throughout this thesis. The execution frequency is hence defined in terms of these time steps with the general assumption that the states are executed in *each* time step t. Such a periodic execution is indeed not required for each of the actions within the workflow. Consider for example the workflow of an unassociated agent (as depicted in Fig. B.2) that executes the potential members strategy in order to determine whether it has strong trust relationships in the agent society. The environment in which the agents act is not assumed to change so fast that this action is required in *each* time step. On the other hand, other actions, such as the check whether an important message has been received, may very well require such frequent executions. As such, the presented workflows are simplified with respect to the representation of time. It is rather implicitly assumed that the TC strategies apply their own execution constraints. An exemplary constraint is the frequency with which the associated evaluation strategy (see Fig. B.2) is allowed to determine the formation of a new TC as initiator. This is usually dependent on past unsuccessful attempts to form a new TC instead of being executed in each time step without consideration of the overhead such a formation involves.

Apart from the lack of a temporal representation, the workflows are further simplified in that they do not depict the application of timeouts required due to the open nature of the system. This application of timeouts can be best exemplified with the following case: An agent initiating the formation of a new TC, can get locked in the execution of the according workflow (as depicted in Fig. B.5), if it relies on the fact that each agent it has sent a formation join invitation will eventually answer this invitation. Agents not interested in the formation of a new TC, have no incentive to answer such messages, as they do not have any direct influence on their utility from this. Also an invited agent can have left the system in the meantime. The initiator is hence forced to apply a timeout mechanism for the processing of invitation answers, to assume all unanswered requests as rejected, and to continue with the workflow after the timeout.

Finally, the TC organisation component $Comp^{x}_{TCO}$ is always in either of the described states and all decision-making, required to allow for the TC application in a hosting system, is contained in the workflows of the states. This allows to complete the specification of the set of required interactions $C^{x}_{TCO} = \{c_1, .., c_k\}$ for this component. An agent x must utilise an implementation Λ^{impl}_{TCO} that allows other agents to interact with x according to the following interaction interfaces:

- c^{UO}_{1}: Receive TC join invitation (*unassociated open state*)
- c^{UO}_{2}: Receive join TC formation invitation (*unassociated open state*)
- c^{UF}_{1}: Receive TC formation join invitation answers (*unassociated forming state*)

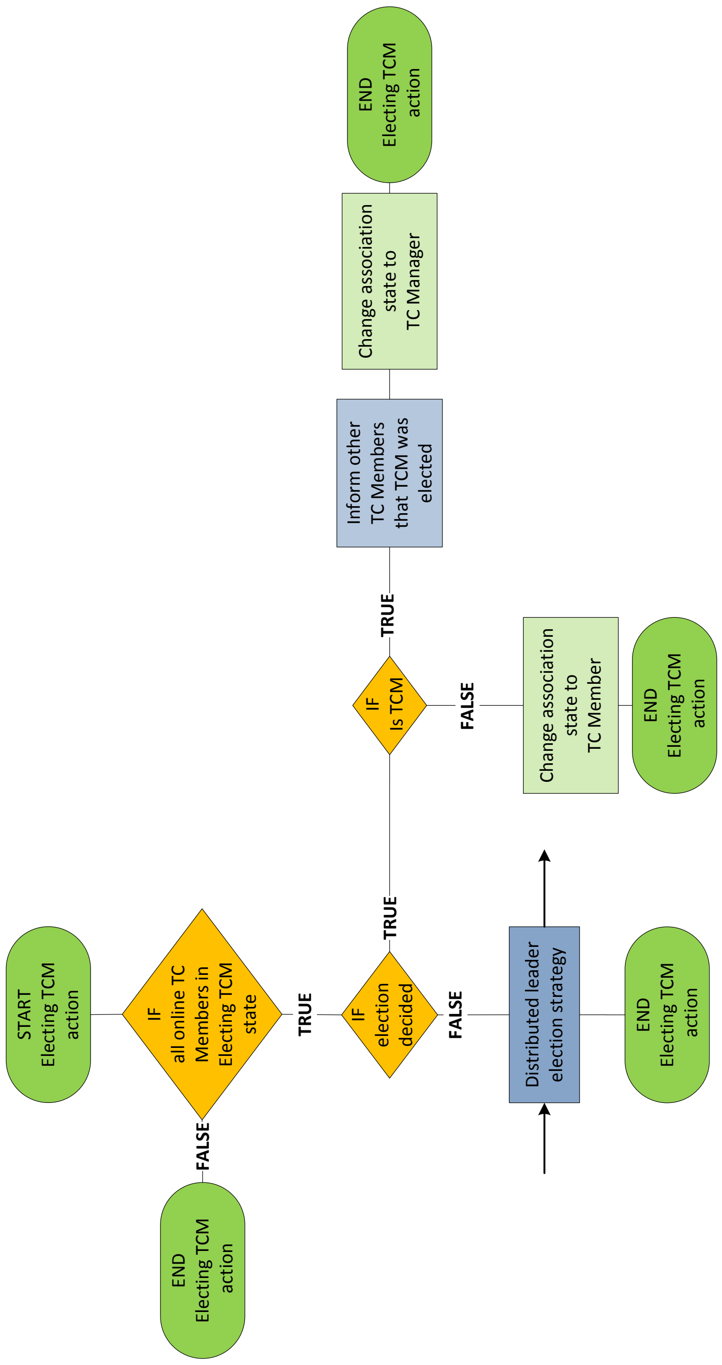

START Electing TCM action
IF all online TC Members in Electing TCM state
FALSE
END Electing TCM action
TRUE
IF election decided
FALSE
Distributed leader election strategy
END Electing TCM action
TRUE
IF Is TCM
FALSE
Change association state to TC Member
END Electing TCM action
TRUE
Inform other TC Members that TCM was elected
Change association state to TC Manager
END Electing TCM action

Figure B.6: Workflow of the **Electing TCM State**, and composing actions and condition tests. The workflow is executed as long as the election is decided, with the discrimination of the two results *not-TCM* and *TCM*.

- c_2^{UF}: Receive TC formation success/failure information (*unassociated forming state*)
- c_1^{M}: Receive TC join invitation answer (*TCM state*)
- c_2^{M}: Receive TC composition update (*TC member state*)
- c_3^{M}: Receive TC exclusion information (sanction or dissolution) from TCM (*TC member state*)
- c_4^{M}: Receive role-assignment instruction from TCM (*TC member state*)
- c_1^{E}: Receive TCM election information (sanction or dissolution) from TCM (*electing TCM state*)
- c_1^{TCM}: Receive TC membership cancellation information (*TCM state*)
- c_2^{TCM}: Receive TC join invitation answer (*TCM state*)
- c_1^{A}: Receive advertisements for the formation/dissolution of a TC (*all states*)
- c_2^{A}: Receive information request about association state (*all states*)

The last interactions c_i^A allow agents to perceive other agents in the agent society as (fellow) TC members, TCM, unassociated agent etc. and to remain up-to-date about the operation of TCs in the hosting system. The complete set of interactions for C_{TCO}^x is then:

$$\begin{aligned} C_{TCO}^x \in \Gamma^x := &\left\{c_1^{UO}, c_2^{UO}, c_1^{UF}, c_2^{UF}, c_1^{M}, c_2^{M}, c_3^{M}, c_4^{M}, c_1^{E}, c_1^{TCM}, c_2^{TCM}, c_1^{A}, c_2^{A}\right\} \\ &\cup \left\{c_1^{opt}, .., c_i^{opt}\right\} \cup \left\{c_1^{coop}, .., c_j^{inf}\right\} \cup \left\{c_1^{inf}, .., c_k^{inf}\right\} \end{aligned} \tag{B.1}$$

with the latter sets denoting interactions defined by the organisation benefit strategies as described in Sec. 4.3.

Additionally, the operations defined in Sec. 4.2 are encapsulated in the following actions of an agent x within the workflows of the states:

- The action *Advertise TC* in the unassociated forming state of a formation initiator $x \in \mathcal{F}$ executes the operation $\mathcal{O}(t) \lhd^{\mathcal{F}} TC_i(t)$ to form a TC with the group of agents $\mathcal{F}$.
- The action *Advertise TC dissolution* in the TC manager state of a TCM x executes the operation $\mathcal{O}(t) \rhd^{x} TC_i(t)$ to dissolve a TC.
- The action *Cancel membership* in the TC member state of an agent x executes the operation $TC_i(t) \ominus^x x$ to end its own membership.
- The action *Make agents TC members* in the TCM state of agent x executes the operation $TC_i(t) \oplus^x y$ to include agents y as TC members.
- The action *Sanction members* in the TCM state of an agent x executes the operation $TC_i(t) \ominus^x y$ to end the membership of an agent y.
- The action *Assign roles to TC members* in the TCM state of an agent x executes the operation $r \rightsquigarrow^x y$ to assign a management role to an agent y.

The set of observables $O^x_{TCO} \in \Theta^x$ required by the component $\mathcal{C}omp^x_{TCO}$ is composed of information required as input for the strategies that are embedded within the workflows. Consider for example the definition of the basic *Membership Evaluation* strategy in Sec. 4.6.4: The according function requires the information $U^x(t)$ and $U^x(t_a)$, being the current utility and the utility at the time of TC association. The *observation model* must hence contain observables that provide the current information, whenever the agent executes the strategy. Consequently, the utilisation of TC strategy implementations determines the set of observables for this component. Due to the runtime adaptations, this set is time-dependent. For detailed information about this input data see the description of strategies presented in Sec. 4.6.

C | Threat Model For Open Desktop Grids

Sec. 5.1.4 of this thesis has introduced the detailed process of work unit scheduling in the TDG, and in particular the application of Trust Management to reduce the impact of adversary worker behaviour on submitter agents. This behaviour has been so far limited to disturbances in the processing of work units, as seen from the view of a submitter. However, the openness of the system, along with the autonomy of agents, allow for more types of undesirable behaviour. In the following, possible adversary behaviours in the Trusted Desktop Grid are classified with a threat model. This allows to define the responsibility of the Trust Management system in the TDG and shows what types of security means have to be assumed as given, in order to guarantee a robust system.

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 types of threats in ODS, as well as threats specific for Desktop Grid Systems (cf. e.g. [83]). 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, a threat model is presented here, classifying threats that can be countered by TM in such a system, as well as threats that cannot be countered by these means. In this, the argumentation in e.g. [20] is followed, i.e. trust and security are two conceptually different notions, and 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 (cf. e.g. [217], [3]). This is shown 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 an improperly configured TM system. For this reason, threats to the system participants that have to be targeted by such a TM system are additionally examined.

As presented in the previous section, the Trust Management in the TDG is applied to influence the submitter and worker decisions. As such, the most obvious threats to be countered here are those that negatively influence these decisions if not detected. Take for example an agent that does not return valid WU results - if this is not considered by the submitter component, then the performance of the WU owner will decrease. However, as the agents have information about this TM system, it can be used to the opposite effect, for example by manipulation of the own reputation. As the other agents' worker decisions are based on the reputation of a submitter requesting a WU processing, a manipulated reputation would lead to undetected advantages of adversary agents. In the following threat list, indirect threats, realised by the manipulation of the Trust Management system, are also included where such a manipulation would circumvention the application of this system. While other authors in the literature provide elaborate analysis of generic methods of such manipulation (cf. e.g. [72], [218]), the model presented here is restricted to those manipulations that induce a direct advantage in the TDG. As discussed in Sec. 2.5.2, the application of Trust Management to counter adversary actions in Desktop Grid Systems is an active field of research. Examples of threats mentioned in the threat model that were also seen as challenging in the research community are the presence of freeriders (cf. e.g. [84], [63], [177]) or the presence of agents that return invalid WU results (cf. e.g. [83], [204]).

As for the modelling of the security threats, the well-known STRIDE approach is related here (cf. [219]). In this, security threats are characterised according to the categories depicted in Tab. C.1.

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 authorized to perform.

Table C.1: The STRIDE threat classification, taken from [219]

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, the threats are grouped according to the capabilities the attacking entity owns:

- *Inside threats* are executed with capabilities an agent in the system owns. In the context of the TDG, this refers to the agent components an agent has according to the model presented in Sec. 5.1.4. Such an adversary agent utilises custom versions $\Lambda^{i}_{SUB} \neq \Lambda^{default}_{SUB}, \Lambda^{j}_{WORK} \neq \Lambda^{default}_{WORK}, \Lambda^{j}_{WORK} \neq \Lambda^{default}_{TM}$ of the submitter and worker components that encode malevolent behaviours. This is possible due to the openness of the system. Consequently, attackers here are agents that represent their users in the system.

- *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 here do not necessarily own an agent participating in the system.

Secondly, the attackers are grouped 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 threats that appear as attacks although the "attacker" has no respective objective are not considered here. This is for example the case when faulty hardware is involved and hence communication affected. The threats listed here are classified with respect to the capabilities and objectives of the attacker as depicted in Tab. C.2.

In this model, the focus is on threats in the TDG system. Additionally, the STRIDE classification for outside threats is provided. In the following, threats comprising the classes are listed in Tab. C.3.

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

Table C.2: Threat classification

Class	Threat	Target	STRIDE
Class 1	Refuse to process WUs for other clients (freeriding)	-	-
	Do not return WU results despite accepting WU processing request (hidden freeriding)	-	-
	Delegate accepted WUs to other workers, acting as owner (hidden freeriding)	-	-
	Provide false private information (performance level, work load) to avoid WU processing requests	-	-
	Submit false positive trust ratings (collusion to improve submitter success)	-	-
Class 2	Cancel the processing of accepted WUs	C	-
	Return false WU results	C	-
	Execute DoS-attack via WU replication and submission	S	-
	Join other clients to execute a distributed DoS-attack (collusion)	S	-
	Execute DoS-attack or slow-down with fake WUs	S	-
	Execute DoS-attack via excessive messaging	S/C	-
	Distribute malware via WUs (or results)	S/C	-
	Submit WUs with unrealistic processing requirements (to legitimate negative trust ratings and improve submitter success)	C	-
	Submit false negative trust ratings (discrediting attack to improve submitter success)	S/C	-
Class 3	Manipulate reputation system to improve own or deteriorate rival reputation (improve submitter success)	-	T
	Sybil attack: Change own identity to appear as new client (with no reputation history)	-	S
	Manipulate other clients' processing queues to place (or prioritize) own WUs	-	T
	Manipulate direct experience history of interaction partners to hide misbehaviour and/or appear trustworthy	-	T

Class	Threat	Target	STRIDE
	Manipulate messages with negative trust ratings to avoid being detected as misconducting client	-	T
	Appear as client with high reputation (impersonation)	-	S
	Manipulate messages, WU results or information of other clients to damage their reputation (relative improvement of own reputation)	-	T
	Manipulate messages from other submitting clients to keep preferred worker clients from having a high workload	-	T
Class 4	Manipulate reputation system (e.g. by rendering all clients untrustworthy)	S	E,T
	Generate fake rating messages or manipulate existing messages	S/C	T
	Execute a man in the middle attack to access WU results or data	C	I
	Execute DoS-attack to damage the communication between clients	S/C	D
	Generate Class 2 clients (bots) and control them	S/C	R

Table C.3: Threats in the Trusted Desktop Grid, classified according to Tab. C.2, STRIDE and target (S)ystem or (C)lient

In the following, the control approach in the TDG regarding these threats is discussed: In the TDG, an underlying security sub-system is assumed that prevents all outside threats (class 3 and 4). This assumption allowed to focus the research in the TDG on the application of trust-based MAS- and OC-technology, while disregarding the well-documented research approaches to the security of such systems (e.g. distributed cryptographic authentication to prevent Sybil attacks). As for the threats of exploiting or damaging the operation of the system from within (class 1 and 2), the agent behaviours behind these threats are either detectable by single agents (e.g. the return of no or false WU results can be detected by the WU owner), or only detected cooperatively (e.g. delegation of WUs to other workers requires dedicated communication among workers to be detected.). This is due to the lack of central control in such a system.

Agents utilise Trust Management to detect agents that show such behaviours, but only if it would be their disadvantage not to do so. Here, the self-interested and autonomous nature of the agents generates no motivation to punish these adversary agents, if there is no perceivable drawback from their behaviour (sometimes also referred to as *second order free-riding*, cf. e.g. [189]). It is hence the degree to which this ability of strategic hindsight and perception exists in the agents that decides whether such selfish and exploiting behaviours by adversary agents can be successful in the system.

D | Performance and Robustness Metrics For Open Desktop Grids

The research and evaluation of Desktop Grid Systems mostly aims at improving the performance and robustness of these systems. In this thesis, the application of Trusted Communities is proposed for this aim. In the following, it is defined under which criteria this improvement has been evaluated.

It is especially interesting in the context of volunteer-based systems how much participants benefit from the performance of the system (cf. e.g. [204]). This involves to consider the influence of the openness of the system on the performance. This is discussed in detail in the following section on *performance metrics*.

Additionally, the openness also introduces the threat of undesired system states. Therefore, also metrics are needed to measure the systems' robustness against these kind of threats. These metrics are discussed in the final part of this section.

D.1 Performance Metrics

For this thesis, a research literature survey on performance metrics for DG and similar systems has been conducted, based on the previous TDG classification and scheduling problem definition presented in Sec. 5.1.1. A set of relevant metrics from different system views has then been identified, as depicted in Fig. D.1. The metrics discussed here have been chosen because of their widespread use in the research community. Other, less frequently used metrics have been neglected here. The requirement for a relevant metric was that it had to be evaluable by a single host. This is because of the distributed nature of the considered system and the self-interest of the participating hosts. In the following, each of these metrics is defined, referenced and discussed. In that, the formalisation defined in Sec. 5.1.2 is used.

D.1.1 Theoretic Scheduling Problems

The performance of a distributed computing system is mainly determined by the performance of the contained scheduling sub-system. Consider for example the life-cycle of a work unit as depicted in Fig. 5.4: When an unreliable host is chosen for the execution of the work unit, the time waited during the processing of the WU is wasted and another attempt has to be started to process the work unit. This increases the time the owner of the work unit has to wait for its completion and reduces the benefit of participating in the Desktop Grid System.

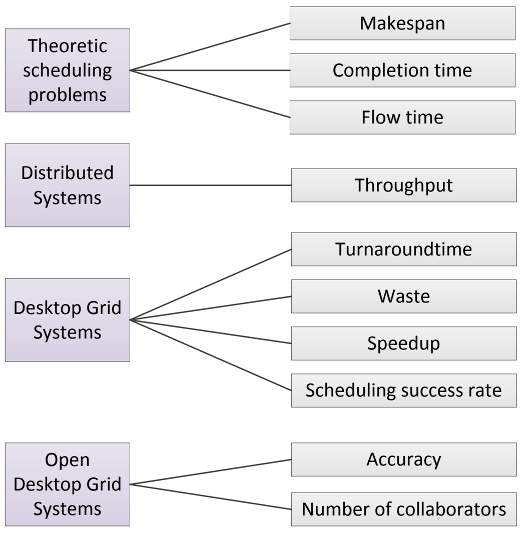


Figure D.1: Overview of relevant TDG performance metrics for distributed systems. The metrics are grouped according to application domains in the literature.

Formally, a scheduling sub-system, which is referred to as *scheduler* from here on, solves the problem of scheduling n atomic pieces of work, called *jobs*[1] for the execution on m machines, such that a given performance criterion is optimised (cf. e.g. [220]). Note that the origin of the jobs (owner), as well as migration and validation are neglected. This theoretic scheduling problem has been studied for at least 50 years, with ever increasing interest since the advent of affordable distributed computing systems. The basic version of this problem is usually refined by further specifying some properties: The processing speed of the machines is defined as either equal for all machines and jobs, for all jobs only, or as being truly depended on the combination of job and executing machine - respectively this is referred to as either Identical (P), Uniform (Q) or Unrelated (R) machines. For the purpose of transferring this to the modelling of a Desktop Grid System, from here on only unrelated machines are considered, as machines in Desktop Grids are heterogeneous in terms of processing capabilities. Further differentiation is made with respect to offline (static) scheduling, if all information about jobs is known to the scheduler at the start of the scheduling process (release times of all jobs are $t = 0$), or (clairvoyant) online scheduling, when information about jobs becomes available only at the release times of the jobs (not necessarily at $t = 0$). Many other variations of the problem exist (cf. [220], [203]), regarding aspects like the quality of the information that becomes available (online vs. semi-online, clairvoyant vs. non-clairvoyant), restrictions on the set of machines allowed to process a job (machine eligibility constraints) or preemptiness options (checkpointing allowed or not-allowed, restart-preemption).

Common to all variations is the formulation of one or more performance metrics under which an optimal solution to the problem is searched. Obviously, because of the global scope with anonymous and atomic jobs, this cannot be directly applied to the previous problem definition. Therefore, the following addition is made: For the purpose of conformity, jobs, as in theoretic scheduling problems, are

[1]Note here, that previously jobs were referred to as Bags-of-tasks, while this definition of job equals the previous definition of work units.

equated with and referred to, as work units (as in BoT-applications). In doing so, owners of the work units, as well as their assignment to jobs are not regarded. Most importantly, the local scheduling at the owner is substituted with a central scheduling of all work units for these considerations. In the following, the most important performance metrics in theoretic scheduling problems are introduced:

Makespan is defined as the point in time where all work units from a finite set $\{\tau_1, \tau_2, ..., \tau_n\}$ provided to the central scheduler, are completely processed, hence:

$$makespan = max(t^{compl}_{\tau_1}, t^{compl}_{\tau_2}, .., t^{compl}_{\tau_n}) \tag{D.1}$$

This metric is especially useful to evaluate the ability of a scheduler to completely carry out the scheduling of a set of work units released at the same time, by applying resource and work unit prioritisation. The performance of the scheduler can then be contrasted with an optimal (minimal) makespan. Makespan is a very common performance metric, used for example in [220], [203] and [206]. In the context of Desktop Grid Systems, sometimes other metrics are denoted as makespan - this is referred to in the according descriptions.

Completion time is the total, or average, of the single completion times of the work units. It is hence defined as follows for a set $\{\tau_1, \tau_2, ..., \tau_n\}$ of work units:

$$total\ completion\ time = \sum_{i=1}^{n} t^{compl}_{\tau_i} \tag{D.2}$$

$$average\ completion\ time = \frac{\sum_{i=1}^{n} t^{compl}_{\tau_i}}{n} \tag{D.3}$$

By minimising the completion time, a scheduler can be evaluated with respect to fair task allocation (average completion time) or efficient task allocation (total completion time). Note that the completion time is not set in relation to the release time of the tasks. The completion time is for example regarded in [220], [203], [221] and [79].

Flow time is a measure to relate the completion times of tasks to their release times and thus provide an estimate of their residing time in the scheduling system. Again, flow time can be measured as total or average for a set $\{\tau_1, \tau_2, ..., \tau_n\}$ of work units and their respective release times $\left\{t^{rel}_{\tau_1}, t^{rel}_{\tau_2}, ..., t^{rel}_{\tau_n}\right\}$, such that:

$$total\ flow\ time = \sum_{i=1}^{n} t^{compl}_{\tau_i} - t^{rel}_{\tau_i} \tag{D.4}$$

$$average\ flow\ time = \frac{\sum_{i=1}^{n} t^{compl}_{\tau_i - t^{rel}_{\tau_i}}}{n} \tag{D.5}$$

The use of flow time metrics is for example discussed in [220] and [203].

D.1.2 Distributed Systems

The class of Desktop Grid systems referred here are distributed systems. As such, metrics from this domain can also be applied in this context. However, most of these metrics are already in use in Desktop Grid systems, often in a specialised definition and under a different name. See for example the discussion of the *turnaround time*, which is a rephrase of the more generic metric *response time*.

Throughput The throughput in a distributed system is the amount of successfully processed messages or tasks in a given time interval. For DG systems, this usually refers to the amount of work units that are completely processed in a time interval (cf. e.g. [222]), and accordingly, these systems are described as platform for high-throughput applications (cf. e.g. [174]). The following definition is applied in this thesis:

$$throughput(t_{start}, t_{end}) = \frac{\text{number of completed work units } \tau_i}{t_{end} - t_{start}} \tag{D.6}$$

with:

$$\tau_i \text{ completed in the time interval when } t_{start} \leq t^{rel}_{\tau_i} \text{ and } t^{compl}_{\tau_i} \leq t_{end}$$

Note that the length of the time interval will usually be in minutes or hours (cf. e.g. [223]). As opposed to previously discussed metrics, the maximum of this metric is the optimum. Throughput-based Desktop Grid evaluations are described in cf. e.g. [223], [207] and [222].

D.1.3 Desktop Grid Systems

In DG systems, metrics from the domain of theoretic scheduling problems are often applied, however redefined to reflect the view of single hosts with decentralised scheduling and self-interested goals. As the evaluation scenario is from the DG systems domain, the following metrics are the most relevant for the evaluation.

Turnaround time The turnaround time denotes the time interval from the release of a work unit t_{τ_i} until the work unit is completed. It is therefore defined as:

$$turnaroundtime(\tau_i) = t^{compl}_{\tau_i} - t^{rel}_{\tau_i} \tag{D.7}$$

As such, the turnaround time is the sum of all intervals of the WU life-cycle presented in Fig. 5.4, however not providing information about where a possible delay occurred (e.g. high waiting duration vs. high processing duration). Again, turnaroundtime can be measured as average (locally over all WUs τ^a of an owner a or globally over all WUs in the system), or as local or global total. Turnaround time is a very common metric for Desktop Grid Systems, its application for the evaluation of e.g. scheduling approaches is described in [80], [206], [224], [225].

Turnaround time is also sometimes referred to as *makespan* (cf. e.g. [206]) in the context of DG Systems, however the scope is on single work units, not the global set of jobs as described by the makespan-metric in theoretic scheduling problems. Additionally, turnaround time can be regarded as *response time* of the system, when taking the scope of single work units as reference tasks.

Waste In an open DG system, workers can fail the successful processing of work units, due to errors or the withdrawal of donated resources. On the submitter side, this means that the time beginning with the distribution of the respective WUs to the unreliable worker to the time the notification of the processing failure was received, is wasted. In reference to the WU life-cycle presented in Fig. 5.4, each round except the final round leading to a valid work unit result is comprised of waste. Obviously, a high waste amount constitutes a long turnaround time. Waste is therefore a good indicator metric to analyse the reasons for unsatisfying turnaround times. In this thesis, the waste metric is defined as follows: For a work unit τ_i, that was completed in r rounds, let $t_{\tau_i}^{begin_r}$ denote the begin of round r (re-scheduling of the WU). Then the amount of waste in the turnaround time of τ_i is:

$$waste(\tau_i) = \frac{t_{\tau_i}^{begin_r} - t_{\tau_i}^{rel}}{turnaroundtime(\tau_i)} \tag{D.8}$$

with:

$$t_{\tau_i}^{rel} \leq t_{\tau_i}^{begin_r} < t_{\tau_i}^{compl}$$

A host can locally compute the average waste for its work units and indicate the schedulers' ability to avoid delegation to unreliable workers. Globally, the average waste over all work units can be computed, indicating the impact and amount of unreliable workers in the system. Apart from indicating performance issues due to suboptimal worker choice, waste is also applied when work unit replication is involved (see definition of *accuracy* below). Here, the processing of replica is also counted as waste (cf. e.g. [226]) in order to evaluate the efficiency of validation mechanisms. Waste is applied as performance metric for example in [200], [227] and [226].

Speedup Speedup is a metric known from distributed and multicore systems where parallelisation can be applied to process tasks faster than on single machines/cores. In DG computing, the speedup is usually measured for jobs composed of atomic WUs [2]. In this thesis, the speedup for a job ϕ_i^a containing $|\phi_i^a| = n$ WUs of an owner a is defined as:

$$\begin{aligned} speedup(\phi_i^a) &= \frac{\textit{summed estimated owner turnaround time}}{\textit{max real turnaround time}} \\ &= \frac{\sum\limits_{\tau_i \in \phi^a} turnaroundtime_{owner}(\tau_i)}{max\left\{turnaroundtime(\tau_1), .., turnaroundtime(\tau_{|\phi^a|})\right\}} \end{aligned} \tag{D.9}$$

with $turnaroundtime_{owner}(\tau_i)$ being the a posteriori estimate of the turnaroundtime the owner would have achieved, had it processed the work unit τ_i on its own. The following assumptions regarding this time are made in this thesis:

- The owner is processing its own WUs from the respective job exclusively, that is, other jobs (own or from other hosts) do not interfere with the processing.

- The processing is not interrupted, thus no waste is generated.

[2]Sometimes the speedup is also measured for single WUs, here no parallelisation is utilised, but the serial processing speed of the job owner is compared with the speed of a chosen worker.

- The available resources delimiting the processing time are exactly known, as this is calculated a posteriori.

These assumptions result in a strict definition of the owner turnaroundtime that is lower than it might be in a non-idealised environment (if for example several jobs are being submitted at the same time). This means that the speedup measure applied here is a worst case measure, and that the speedup is in general lower than it could be expected to be in a real environment.

Speedup is an important metric in DG computing as it puts the raw performance of job processing in the system in relation to the relative speed of the owning host. In this, also the benefit of the system participation is expressed. Speedup is a common metric in the literature, for example applied in [204], [222] and [207].

Scheduling success rate In the literature, the (scheduling) success rate is most commonly referred to as the rate of work units that were scheduled to workers that processed them correctly and in time, as opposed to being scheduled to workers that generate waste. Additionally, a successful scheduling is sometimes attested only in case of the first chosen worker being the one to produce a correct result in time, hence penalising rescheduling (cf. e.g. [206]). In case of an unexpectedly large turnaroundtime, the scheduling success can be a strong indicator where the delay stems from. This is especially valid for Open Desktop Grid Systems, where workers that behave adversely (and thus generate waste) are to avoid when scheduling. The scheduling success rate is defined with reference to Fig. 5.4:

$$\textit{scheduling success rate} = \frac{\textit{total number WUs with processing in 1 round}}{\textit{total number WUs with processing in r>1 rounds}} \tag{D.10}$$

, where total can be understood as local total or global total. This makes the scheduling success rate suited for the assessment of a local scheduling strategy by a host. The metric is for example used for evaluations in [221], [228] and [206].

D.1.4 Open Desktop Grid Systems

Open DG systems introduce challenges encountered in general Open Distributed Systems to the domain of DG systems. As such, a number of additional metrics have been applied in the literature to verify approaches countering these challenges.

Accuracy Open Desktop Grid systems are based on the processing of Work Units by volunteers. This processing can lead to invalid results, due to faulty hardware, network infrastructure or adversary behaviour (also referred to as *sabotage* in this context, cf. e.g. [176]). Often WU results cannot be validated without reprocessing them entirely, in which case replication and then following majority voting are applied to disseminate correct from invalid results (cf. e.g. [175]). Evidently, majority voting is a frail validation approach: In case of colluding workers, an incorrect result can achieve a majority and be accepted by the submitter (cf. e.g. [179]). The metric *accuracy* is hence applied to verify the performance of validation approaches like spot-checking (cf. e.g. [176]) by measuring the amount of wrongly accepted WU results. The metric can be expressed as follows:

$$accuracy = \frac{\textit{number rightly accepted WU results}}{\textit{total number accepted WU results}} \tag{D.11}$$

Obviously, *accuracy* can only be observed theoretically by the designer of a validation approach, and hence not estimated by participating hosts. *Accuracy* is also referred to as *error rate* in the literature (cf. e.g. [175]).

Number of collaborators In Open Systems as the TDG, the participants are free to chose their cooperation strategy. However, these systems are often designed such that there are incentives to collaborate as opposed to exploiting the system without providing compensation (freeriding). In such systems, the metric *number of collaborators*, also referred to as *number of contributors*, can be used to measure the success of the incentive mechanisms. The metric can be defined as follows:

$$numberofcollaborators(t) = \textit{total number system participants} - \textit{number freeriders} \tag{D.12}$$

Note that this metric depends on a reliable identification of freeriders and is thus, similar to the metric accuracy, not applicable by single hosts. This metric is for example used in [84], [177] and [229]. This metric is additionally related to the *faulty fraction* metric defined in [175] measuring the amount of hosts that produce invalid WU results on purpose, in comparison with cooperative hosts.

D.2 Robustness Metrics

Robustness is a fundamental property of complex systems (cf. e.g. [132]), observed in systems from biological, psychological and technical (among others) domains. Though the exact definition of robustness varies in the literature, it is commonly agreed that robustness is the ability of a system to maintain a stable or high-performance state, or recover to such a state, in the event of specific disturbances in the system (cf. e.g. [135], [230]). High-performance refers to system features or characteristics that must be present in such a state and be quantifiable, e.g. by a performance metric (cf. e.g. [231]). To maintain or recover to such a state means thus, to preserve or recover a system characteristic despite disturbances (cf. e.g. [121]). System characteristics related to the work presented are for example to protect the operation of the Trust Management system, such that it cannot be manipulated (cf. e.g. [98], [72]), or to guarantee a certain performance (measured in e.g. *makespan*, see above) when submitting jobs to workers in a Distributed Computing System (cf. e.g. [231], [232]).

In this thesis, the robustness for the evaluation scenario Trusted Desktop Grid is defined as follows:

- The system characteristic that is to preserve is the global average of one of the performance metrics as defined in the previous section (e.g. *speedup*). It is denoted as $\lambda(t)$.
- The perturbations under which $\lambda(t)$ is to preserve are defined via a threat model (see appendix C). These perturbations $\delta_{i,t_{act}}$ have an intensity i and an activation time t_{act}. To preserve $\lambda(t)$ means thus to ensure that $\lambda(t)$ does not collapse for $t > t_{act}$. Note that it is assumed here, that once activated, a perturbation remains active, i.e. if the system does not provide any means to detect and neutralise the disturbance, it will have a constant influence on the system performance $\lambda(t)$. A set of disturbances D_δ is further defined to denote the variety in intensities for a given perturbation.
- Finally, the robustness of the system is quantified as the function $r_m(\lambda, D_\delta)$: This function expresses the systems' ability to preserve the system characteristic λ against various intensities

of the perturbation δ. In that, m denotes a specific robustness metric, thus a metric to measure whether the system is within an accepted state.

Consider the example depicted in Fig. D.2: Here, the system characteristic to preserve, $\lambda(t)$, is the average speedup for the last job among the system participants. The time t_{attack} marks the

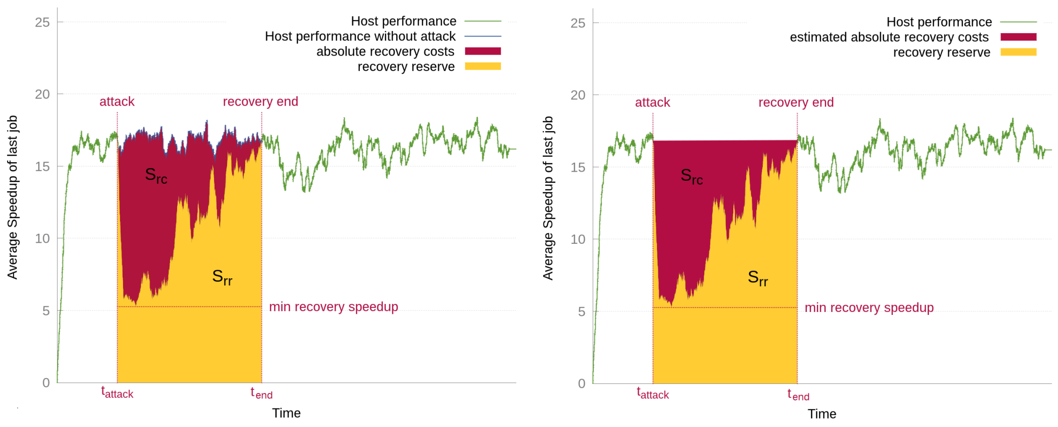


Figure D.2: Robustness example: Speedup collapse and recovery due to attack. The left figure shows an example where the reference performance without perturbation can be determined, while the right figure depicts an interpolated performance.

activation of a perturbation $\delta_{i,t_{act}}$, here the arrival of a group of colluding freeriders in the system. The intensity i of this collusion attack is the fraction of freeriders of the total number of system participants. Here, the evaluation focuses on the impact of this perturbation. In the progression of the example, the speedup drops due to the attack and recovers after a certain amount of time. To evaluate the general robustness $r_m(\lambda, D_\delta)$ of this system against this type of attack, it is necessary to evaluate the system with a set D_δ of various attack intensities, as well as robustness metrics m. For the evaluations in this thesis, the following robustness metrics, referring to Fig. D.2, are applied:

Recovery duration After the activation of $\delta_{i,t_{act}}$ at t_{attack}, a robust system will eventually recover from the attack. The recovery duration is hence the time to reach a value for $\lambda(t)$ that is acceptable (formalised by $f_{acc} \cdot \lambda(t)$) and maintain it stable for a time interval (formalised as t_{sta}). This requirement to the recovery is expressed as the following function:

$$a(\lambda(t_{end}), \lambda(t_{act}), f_{acc}, t_{sta}) = \begin{cases} 1, & \text{if } t_{end} \geq t_{act} \wedge \\ & \forall t_i \in \{t_{end}, .., t_{end} + t_{sta}\} : \lambda(t_i) \geq f_{acc} \cdot \lambda(t_{act}) \\ 0, & \text{else} \end{cases} \tag{D.13}$$

The recovery duration is then based on the existence of a time t_{end} at which the system recovered from the disturbance:

$$recoveryduration = \begin{cases} t_{end} - t_{act}, & \text{if } \exists\, t_{end} : a(\lambda(t_{end}), \lambda(t_{act}), f_{acc}, t_{sta}) = 1 \\ undefined, & \text{else} \end{cases} \tag{D.14}$$

In the example above, f_{acc} is 1, that is the speedup has to reach the value from before the attack in order to constitute a recovery, while t_{sta} is set to 0, that is, the speedup has only to be reached and not maintained. Obviously, a full recovery cannot always be expected, consider for example disturbances that lead to a complete breakdown of a system (for example by disabling a central component in a non-robust system). Here, [135] is referred to, in which the following understanding of robustness is proposed: The ideal set of states for an adaptive system constitute the target space. This is where the system is before the disturbance. When a disturbance occurs, in general the system leaves the target space, and depending on the type and intensity of the disturbance, enters a state where the performance is degraded:

- but recovery is possible without external control actions (acceptance space),
- and recovery is only possible with external control actions (survival space),
- and recovery is not possible at all (dead space).

The acceptance state is defined over the compliance of constraints. A typical constraint is the threshold for allowed values of the system characteristic $\lambda(t)$, for example, the constraint that the average speedup in a DG system has to be greater 1, i.e. there is a benefit in system participation. This definition provides a way to complete the definition of the recovery duration: The end of the recovery, constrained by f_{acc}, can only be reached when the system is in the acceptance space, and f_{acc} can be dimensioned by providing for $f_{acc} \cdot \lambda(t_{act})$ being greater than the threshold for an accepted state. Finally, this allows to define a recovery phase for systems that do not fully recover to $\lambda(t_{act})$ and leave the recovery duration metric undefined only for cases in which the system did not (re-)enter an acceptance or target space.

Collapse fraction While the recovery duration can capture how long the performance of a system was degraded due to a disturbance, it does not measure the extent of the degradation. This is achieved by the metric *collapse fraction* which relates the values of a system characteristic $\lambda(t)$ during the recovery phase to the value at the begin of the disturbance. Here, the minimum value is essential and the metric is hence defined as:

$$collapsefraction = 1 - \frac{min\,\{\lambda(t_{act}+1), .., \lambda(t_{end})\}}{\lambda(t_{act})} \tag{D.15}$$

If the system did not recover, t_{end} marks the end of the evaluation interval.

Relative recovery costs Though generally useful, the two previously defined metrics are limited with respect to the comparison of two robustness approaches in a system: One of the approaches could reduce the recovery duration, but have a higher performance collapse fraction (and the other way round). Therefore, a more elaborate metric, the relative recovery costs, is introduced here: The (absolute) costs for the recovery are the surface S_{rc}, bound by the performance values of $\lambda(t)$ during the recovery phase, and the performance without a disturbance (see Fig. D.2). The latter is either known from reference measurements or interpolated by using the datapoints $\lambda(t_{act})$ and $\lambda(t_{end})$. The absolute recovery costs neglect how the performance was before the attack and could thus bias the results for comparisons. As such, in this thesis the relative recovery costs are applied, defined as:

$$recoverycosts_{rel} = \frac{S_{rc}}{S_{rc} + S_{rr}} \tag{D.16}$$

The relative recovery costs are bound by 0 (no recovery costs) and 1 (the system performance completely collapsed and did not recover).

D.3 Rationale for the Utilisation of Metrics in the TDG

The system model presented in Sec. 3.1 introduced agents that control the production engine (here the DG client) such that they achieve their users' goals. These goals are reflected in the agent utility $U^x(t)$, a function that can be evaluated by the agents (self-awareness), as well as the user of the agent (benefit of grid participation). The self-evaluation is necessary to allow for adaptations of the agent in the O/C-cycle, as well as for decision making in the context of TC membership. Consequently, the choice of a suited metric for the utility $U^x(t)$ of a TDG agent depends on the possibility to locally evaluate this metric. The second aim of this analysis is to choose a metric that allows for the measurement of a system performance: Here, abstracted from the view of single self-interested agents, the system as such is analysed. This allows for making statements about the improvements of Trusted Community application in such a hosting system.

The discussion on performance metrics has first referred to metrics derived from the domain of theoretic scheduling problems. The metrics referred there are useful as they focus on a complete consideration of the whole life-cycle (see Fig. 5.4) without breaking it down into sub-phases. This has the advantage of definite statements about the performance of a scheduling scheme. Additionally, these metrics have a distinct optimal value with respect to a given problem size - the set of all jobs generated during a finite execution time. This optimum can however only be determined offline (a posteriori), as the problem is NP-complete (cf. e.g. [221]) and of unmanageable scale for DG systems. In addition, agent autonomy does not allow an agent to determine the exact situation in which its submitter component has made delegation decisions, hence an offline optimisation is always biased by the observations a particular agent has made. Optima are thus only of minor practical relevance for the utilisation as *agent utility*, while also being too restrictive for the use as *system metric*.

The throughput metric is applied in distributed systems in general, as well as in DG systems in particular. While this metric is suited as a system-wide performance metric, it requires a precisely defined job generation function as benchmark. Only by relating the number and specification of generated jobs is the throughput resulting from two different approaches comparable to each other. Note also that the throughput is a global measurement, not evaluable by a single agent. Additionally, the throughput of jobs/WUs is linked to their completion times in a time interval (cf. e.g. [207]). As the completion is also covered by other metrics, the throughput is neglected as performance metric in this thesis.

The next set of metrics discussed was from the domain of Desktop Grid systems. As expected, these metrics are regarded as the most relevant for the evaluations in this thesis. The metric turn-around time reflects the users' goal to process own work as fast as possible in the grid. Also, it hides the single phases of the work unit life-cycle (see Fig. 5.4) as it focuses only on the resulting summed time. This metric can hence be applied by TDG agents to evaluate their submitter decision-making and make adaptations to it based on the resulting performance. However, the summed turnaround time is insufficient when the deviation of an expected turnaround time needs to be analysed. This is for example the case in the distinction of an extensive waiting duration from a processing duration in system states with a high load.

The amount of waste in the turnaround time of a work unit is particularly suited to indicate whether a submitter component is delegating the workers with uncooperative behaviours. This is highly relevant in an open system as the TDG, where agents with adversary behaviours try and disturb other agents. The waste is hence used in the evaluations, where the isolation of such agents is involved. In that, the waste is used in the form of an *average waste* over all agents and all of their work units.

Turnaround time and waste are absolute measurements of the performance received in the DG system. These metrics are not suited to give feedback to the agent (and user) whether the delegation of work units to other agents really pay off. Consider for example an agent that has exhibited adversary behaviour (possible because of an exploration of behavioural strategies) that has a bad reputation value and hence spends a long time waiting for workers to accept its work units. In that time, the agent might have processed the respective WUs by itself, depending on the processing capabilities of its machine. Additionally, turnaround time and waste refer to single work units. The aim of the agent is however to process a job composed of these work units as fast as possible. In that, the turnaround time of single WUs in the job can vary substantially. Also, the ability to schedule WUs of a single job efficiently to several workers in parallel is not determined by a WU metric as the turnaround time. It is therefore useful for the decision-making of the agents, as well as for the estimation of the benefit of DG participation for a user, to relate the absolute performance for an entire job to a personal performance for that job. This is only achieved with the speedup metric. This metric is hence regarded as the central metric for the evaluations in this thesis, being measured in the following forms:

The system-wide performance is given as *average speedup* over all jobs and agents. This is used to make statements about the benefits of the self-organisation within the whole system. Especially, the introduction of Trusted Communities in a system is shown to raise the average speedup in the whole system, as well as the average speedup of TC members in particular when compared to a simulation run without TCs. Formally, this metric is defined as:

$$\mathcal{S}(\mathcal{A}, \Phi^{a_1}, .., \Phi^{a_{|\mathcal{A}|}}) := \frac{1}{\left| \bigcup\limits_{a_i \in \mathcal{A}} \Phi^{a_i} \right|} \cdot \sum_{\phi^{a_i} \in \bigcup\limits_{a_i \in \mathcal{A}} \Phi^{a_i}} speedup\,(\phi^{a_i}) \tag{D.17}$$

with:

Φ^{a_i} being the set of all jobs that the agent $a_i \in \mathcal{A}$ generated throughout its participation time in the system, and referring to the speedup definition presented in Eq. D.9.

On the agent level, hence for agents to estimate the performance of their decision-making with respect to their users' goals, the *speedup of the last job of a single agent* is used as utility function $U^x(t)$. This measure is defined for an agent x as follows:

$$U^x(t) := speedup(last(\Phi^x, t)) \tag{D.18}$$

with $x \in \mathcal{A}$ and $last(\Phi^x, t)$ being the last completed job of agent x, such that:

$$last(\Phi^x, t) := \phi^x, \forall \widetilde{\phi}^x \in \Phi^x \setminus \{\phi^x\} : t \geq completiontime_{\phi^x} > completiontime_{\widetilde{\phi}^x} \tag{D.19}$$

Where effects with an activation time in the system and their influence on the speedup are regarded, the *average speedup for the last job*, a time dependent form of the *average speedup*, is used as a system-wide metric. Formally, this function is defined in

$$\mathcal{S}(\mathcal{A}, \Phi^{a_1}, .., \Phi^{a_{|\mathcal{A}|}}, t) := \frac{1}{|\mathcal{A}|} \cdot \sum_{a_i \in \mathcal{A}} speedup(last(\phi^{a_i}, t)) \tag{D.20}$$

As the speedup is used for the performance measurements in the following evaluation, the time-dependent form of it is consequently also applied as system characteristic $\lambda(t)$ for the robustness metrics, hence:

$$\lambda(t) := \mathcal{S}(\mathcal{A}, \Phi^{a_1}, .., \Phi^{a_{|\mathcal{A}|}}, t) \tag{D.21}$$

The openness of the TDG allows for the evaluation of agents with adversary behaviour. As such, the performance metrics used in open Desktop Grid Systems are highly relevant. Especially the metric *accuracy* is applied for the evaluations of the self-protection abilities of the TC approach with respect to the exploitation of the *Transparent WU validation* (see Sec. 5.2.1).

E | Additional Evaluation Results

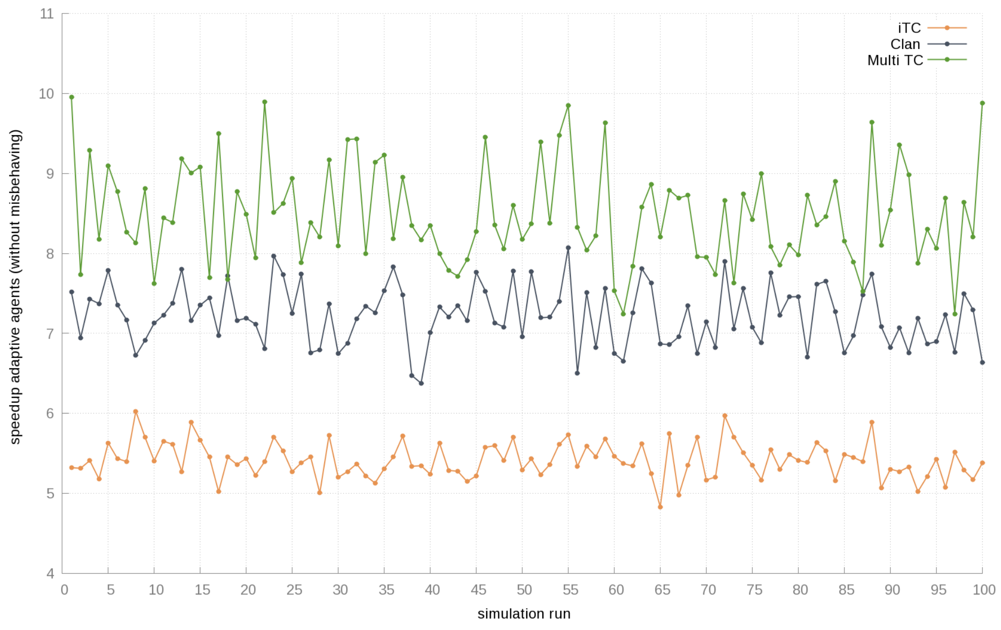


Figure E.1: Speedup comparison of the *iTC*, TC and *Clan* approaches for the *undisturbed* case and *non-validating* DG applications. TCs formed in each experiment run and outperformed the *iTC* and *Clan* results in each run.

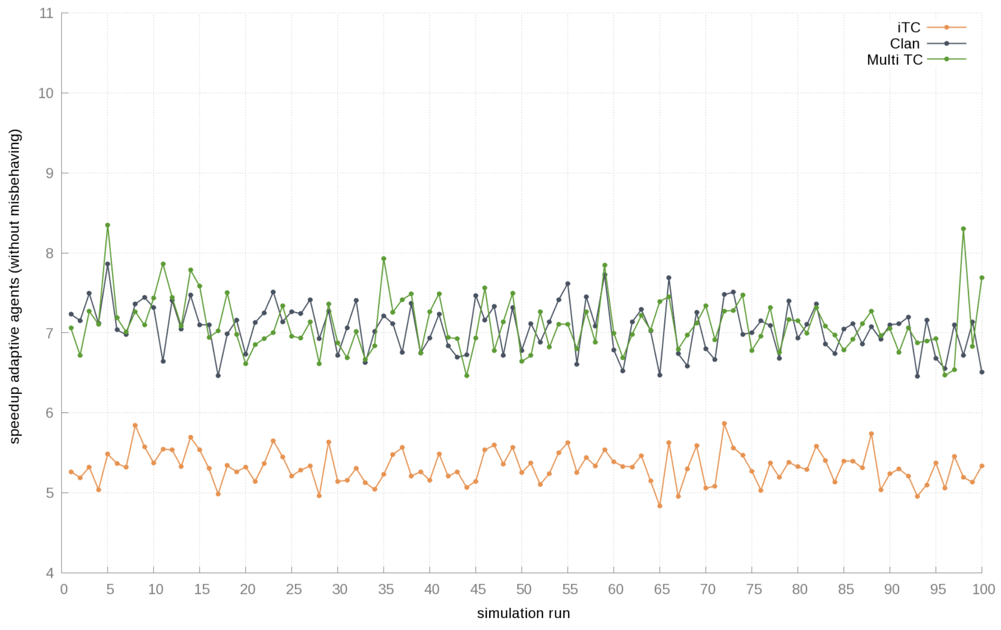


Figure E.2: Speedup comparison of the *iTC*, TC and *Clan* approaches for the case with 20 % defecting agents and *non-validating* DG applications. TCs formed in each experiment run and outperformed the iTC results in each run. The difference was however smaller than in the *undisturbed* case. The performance of *Clans* was on par with the TC performance.

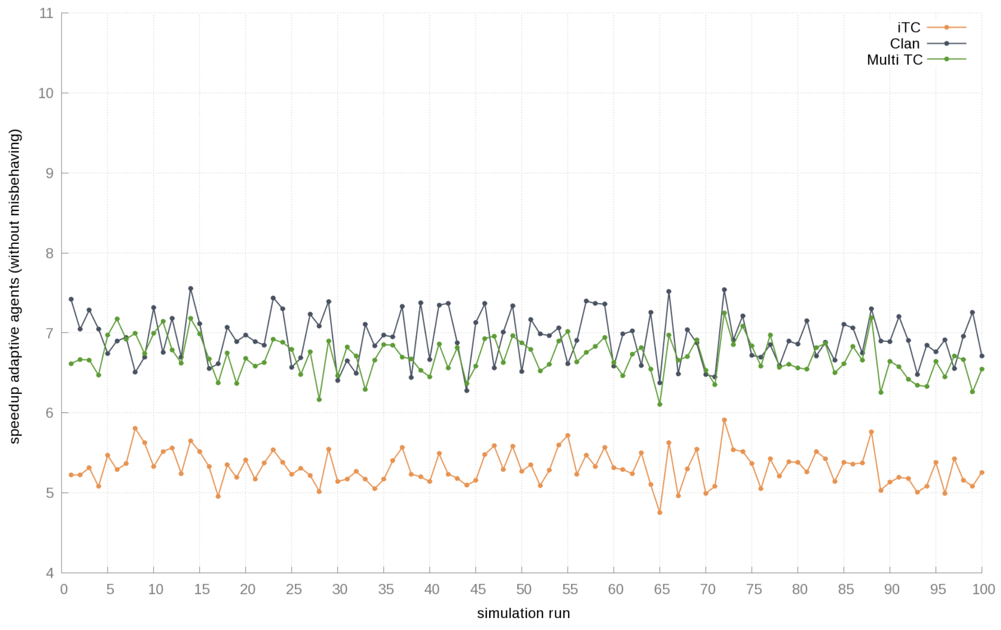


Figure E.3: Speedup comparison of the *iTC*, TC and *Clan* approaches for the case with 30 % defecting agents and *non-validating* DG applications. TCs formed in each experiment run and the outperformed the iTC results in each run. The difference was however smaller than in the other cases. The performance of *Clans* was on par with the TC performance.

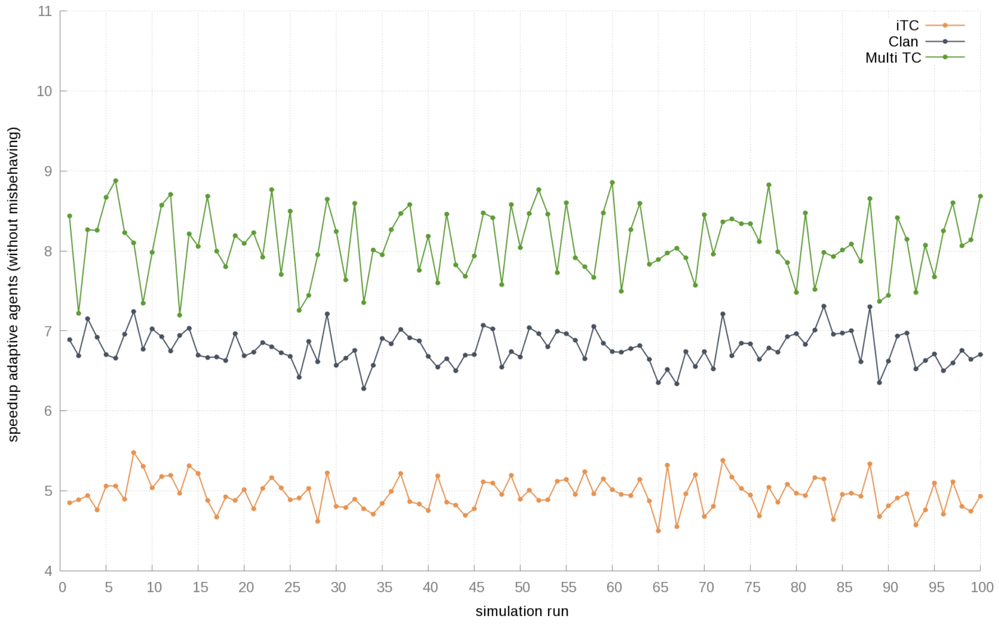


Figure E.4: Speedup comparison of the *iTC*, TC and *Clan* approaches for the case with 20 % TM-exploiting agents and *non-validating* DG applications. TCs formed in each experiment run and out-performed the *iTC* and *Clan* results in each run.

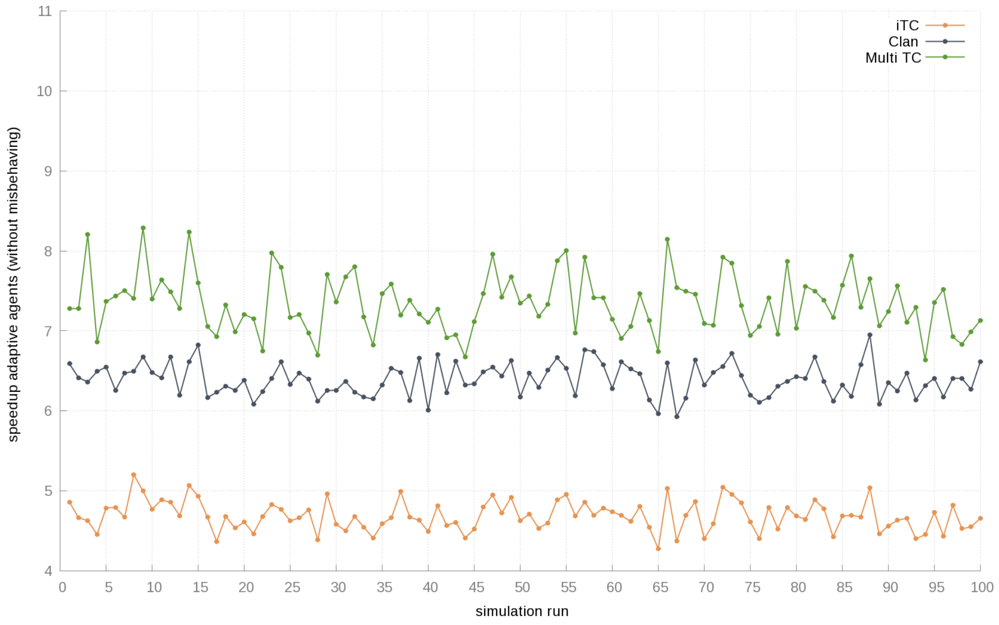


Figure E.5: Speedup comparison of the *iTC*, TC and *Clan* approaches for the case with 30 % TM-exploiting agents and *non-validating* DG applications. TCs formed in each experiment run and out-performed *iTCs* and *Clans*.

LEBENSLAUF - LUKAS KLEJNOWSKI

PERSÖNLICHE DATEN

Geboren	21. November 1981 in Piekar, Polen
Staatsangehörigkeit	Deutsch
Familienstand	Verheiratet, 1 Kind

BERUFLICHER WERDEGANG

03.2009–09.2013 Wissenschaftlicher Mitarbeiter

Leibniz Universität Hannover

Zeitlich befristete Anstellung als wissenschaftlicher Mitarbeiter im Institut für Systems Engineering, Fachgebiet System- und Rechnerarchitektur. Tätigkeiten: Forschung und Entwicklung im Bereich dezentraler Kontrolle von offenen, verteilten Systemen im Rahmen des DFG-Projektes OC-Trust (FOR 1085), sowie Lehre im Universitätsbetrieb.
http://www.sra.uni-hannover.de

12.2008–02.2009 Praktikum als Java-Entwickler

SoftwareLoft IT-Solutions GmbH, Hamburg

Praktikum im Bereich der Java-Entwicklung. Das Praktikum umfasste die Anforderungsanalyse, den Entwurf, die Implementierung und Integration einer Softwarekomponente in das bestehende System des Arbeitgebers. Auftraggeber war ein Kunde aus dem Finanzdienstleistungsbereich.
http://www.softwareloft.de/

BILDUNGSWEG

03.2009–03.2014 Promotion im Fachgebiet Informatik

Leibniz Universität Hannover

Promotion an der Universität Hannover im Institut für Systems Engineering, Fachgebiet System- und Rechnerarchitektur. Dissertation bereits eingereicht, Prüfung und Abschluss mit Titel *Dr.-Ing.* am 24.02.2014.

10.2006–03.2009 Studium Master Informatik

Leibniz Universität Hannover

Studium im Studiengang Master Informatik. Abschluss mit Titel *M.Sc.*.

Masterarbeit:
Design and Implementation of an Algorithm for the Distributed Detection of Disturbances in Traffic Networks
Fachbereich System- und Rechnerarchitektur.
Prüfer und Betreuer: Prof. Dr. rer. nat. J. Hähner · Dr.-Ing. S. Tomforde.

10.2002–09.2006 Studium Bachelor Informatik

Leibniz Universität Hannover

Studium im Studiengang Bachelor Informatik. Abschluss mit Titel *B.Sc.*.

Bachelorarbeit:
Entwurf und Implementierung eines XForms-Interpreters für Java Swing
Fachbereich Software Engineering.
Prüfer und Betreuer: Prof. Dr. K. Schneider · Dr.-Ing. D. Lübke.

08.1994–07.2001 Schüler am Gymnasium

Wilhelm-Raabe-Schule Hannover

Schüler der Wilhelm-Raabe-Schule Hannover, Abschluss mit allgemeiner Hochschulreife (Abitur).