Quantitative Robustness – A Generalised Approach to Compare the Impact of Disturbances in Self-organising Systems

Jan Kantert[1], Sven Tomforde[2], Christian Müller-Schloer[1], Sarah Edenhofer[3] and Bernhard Sick[2]
[1]*Institute of Systems Engineering, Leibniz University Hanover, Appelstr. 4, 30167 Hanover, Germany*
[2]*Intelligent Embedded Systems Group, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany*
[3]*Organic Computing Group, University of Augsburg, Eichleitnerstr. 30, 86159 Augsburg, Germany*

Keywords: Robustness, Organic Computing, Multi-agent-systems, Self-organisation.

Abstract: Organic Computing (OC) and Autonomic Computing (AC) systems are distinct from conventional systems through their ability to self-adapt and to self-organise. However, these properties are just means and not the end. What really makes OC and AC systems useful is their ability to survive in a real world, i.e. to recover from disturbances and attacks from the outside world. This property is called robustness. In this paper, we propose a metric to gauge robustness in order to be able to quantitatively compare the effectiveness of different self-organising and self-adaptive system designs with each other. In the following, we apply this metric to three experimental application scenarios and discuss their usefulness.

1 INTRODUCTION AND MOTIVATION

Organic Computing (Tomforde et al., 2011) (OC) (and Autonomic Computing (IBM, 2005) (AC)) systems are inspired by nature. They mimic architectural and behavioural characteristics, such as self-organisation, self-adaptation or decentralised control, in order to avoid a single point of failure and to achieve desirable properties like e.g. self-healing, self-protection, and self-optimisation. Moreover, OC and AC systems organise themselves bottom-up; this can eventually lead to the formation of macroscopic patterns from microscopic behaviours. Such „emergent“ effects (Mnif and Müller-Schloer, 2006) can be beneficial or detrimental and have to be understood and controlled.

It is a misconception, however, that the goal of building OC systems is primarily the construction of self-adaptive, emergent or self-organising systems. Self-organisation and self-adaptation are just means to make technical systems resistant against external or internal disturbances. It is also a misconception to assume that OC systems (or self-adaptive and self-organising systems in general) generally achieve a higher performance (e.g. higher speed) than conventional systems. OC systems are not per se faster than conventional systems but they return faster to a certain corridor of an acceptable performance in the presence of disturbances. The ultimate goal of OC systems is to become more resilient against disturbances and attacks from outside. We call this property ”robustness”.

In a variety of experiments in different application fields, such as wireless sensor networks (Kantert et al., 2016b), open distributed grid-computing systems (Choi et al., 2008), and urban traffic control (Prothmann et al., 2011; Tomforde et al., 2010), we have observed a characteristic behaviour of OC systems under attack[1]: They show a fast drop in utility after a disturbance (or an intentional attack), followed by a somewhat slower recovery to the original performance provided that there are suitable OC mechanisms (such as a learning observer/controller architecture Tomforde et al. (2011)) in place. The exact characteristic of this utility drop and recovery curve is an important indicator for the effectiveness of the observer/controller mechanism. If we can quantify robustness, we can compare different observer/controller designs in terms of their ability to provide resilience to external disturbances.

The general idea for measuring robustness is to

[1]An attack is a certain instance of the broader class of disturbances. In the remainder of this paper, we will use the term "attack" but the discussion is valid in general for all kinds of disturbances.

Kantert J., Tomforde S., MÃijller-Schloer C., Edenhofer S. and Sick B.
Quantitative Robustness âĂŞ A Generalised Approach to Compare the Impact of Disturbances in Self-organising Systems.
DOI: 10.5220/0006137300390050
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 39-50
ISBN: 978-989-758-219-6

use the area of the characteristic utility degradation over time because this area captures the depth of the utility drop as well as the duration of the recovery. A degradation of zero corresponds to an ideally robust system. In a more detailed view, it is interesting to analyse the gradient of the drop phase and the bottom level of the degraded utility. These figures allow the characterisation of the so-called *passive robustness*. As soon as the recovery mechanisms are activated (usually this happens as fast as possible after the disturbance), the upward gradient of the utility is an indicator of the effectiveness of the system's *active robustness*. „Utility" is an application-specific metric. It is used as a generalised term for the targeted effect of the system as defined by the user. It can be a speed (in case of robots or autonomous cars), a performance or throughput (in case of computing systems), or a transfer rate (in case of communication systems).

We have outlined a first draft of this robustness metric in (Kantert et al., 2016a) in a preliminary form. In this paper, we will review and refine this metric (Section II). In the following, we will apply it to a wireless sensor network (III), a desktop grid-computing system (IV), and an urban traffic management system (V). Then, we will discuss its usefulness (VI) and conclude with a discussion of future work.

2 APPROACH – MEASURING ROBUSTNESS

In this approch, we extend our previous work (Kantert et al., 2016a) by classifying and quantifying the robustness.

2.1 Passive and Active Robustness

We assume a system S in an undisturbed state to show a certain target performance. More generally, we rate a system by a utility measure U, which can take the form of a performance or a throughput (in case of a computing system), a speed (in case of car), or any other application-specific metric. Typically, a system reacts to a disturbance by deviating from its target utility U_{target} by ΔU. Passively robust systems, such as flexible posts or towers under wind pressure, react to the disturbance by a deflection $\Delta U = D_x$. This deflection remains constant as long as the disturbance remains. Active robustness mechanisms (such as self-organisation effected by a control mechanism like an observer/controller) counteract the deviation and guide the system back to the undisturbed state with $\Delta U = 0$ or U_{target}. If we want to quantify robustness (for comparison between different systems), we have to take into account the following observables:

1. The strength of the disturbance, z
2. The drop of the system utility from the acceptable utility U_{acc}, ΔU, and
3. The duration of the deviation (the recovery time $t_{\text{rec}} - t_z$).

We will introduce the developed method that takes all three aspects into consideration in the following part of this paper.

2.2 Measuring Robustness

We assume that it is generally feasible to measure the utility over time (at least from the point of view of an external observer). However, it is often hard to quantify the strength of a disturbance z. Therefore, depending on the application, an estimation of z is required for our model. Furthermore, the system has to know a target utility value U_{target} (maybe the highest possible utility) and an acceptable utility value U_{acc} (a minimal value where the system is still useful to the user). The goal of the model is to measure the response of a system to a certain disturbance in terms of its utility and compare it to other disturbances or systems.

In Figure 1, we show an exemplary typical utility function $U(t)$. In the beginning, U is at the target value U_{target}. At time t_z, a disturbance of strength z happens (displayed in green) and the utility (red) decreases. Once it drops below the acceptance threshold U_{acc} at t_{cm}, a control mechanism (CM) starts to intervene. At t_{low}, U reaches $U_{\text{low,perm}}$ without an effective recovery mechanism and $U_{\text{low,cm}}$ if a control mechanism is acting against the impact of the disturbance. With a CM, U starts to recover at t_{rec} and passes U_{acc} at t_{acc}. However, without a CM, U does not recover.

We can differentiate between two classes of behaviour during an attack:

1) An effective CM is started and the system recovers to at least U_{acc}.
2) The system does not recover during attack (or during a disturbance).

Furthermore, we see two different types of behaviour when the attack ends at $t_{\overline{z}}$. Either (a) the utility reaches the same value as before the attack (here U_{target} at t_{target}; solid red line in Figure 1), or (b) it stays at the same level as during the attack (dashed line). In total, this results in four different stereotypes of behaviour:

1a) The system S recovers during the attack to $U \geq U_{\text{acc}}$ and returns to $U \geq U_{\text{target}}$ when the attack ends. This is a *strongly robust system*.

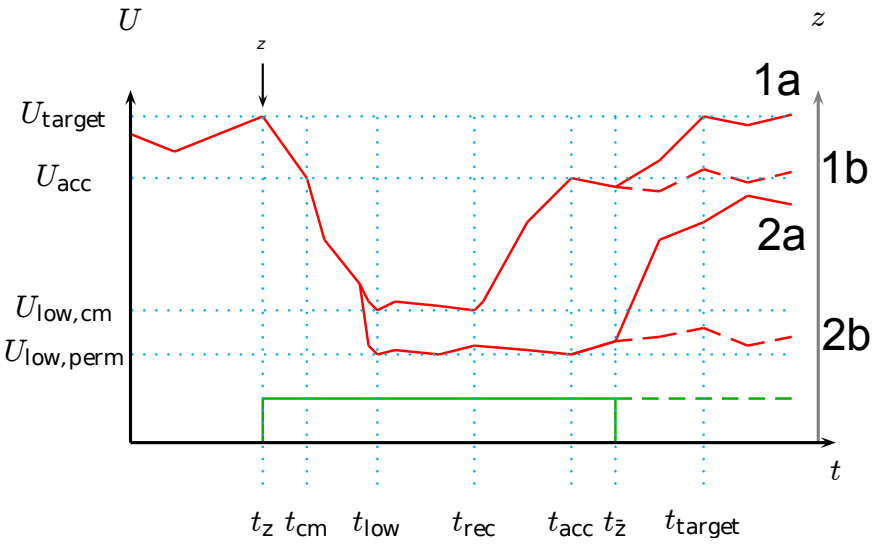

Figure 1: Utility degradation over time. At t_z a disturbance with strength z occurs. The utility U drops. When it reaches U_{acc} (i.e. a utility value that is mapped on the acceptance space of the considered system, see Schmeck et al. (2010)), a control mechanism (CM) is activated for (1) which only decreases to $U_{\text{low,cm}}$ (i.e. the lowest utility value under presence of the control mechanism CM). At t_{rec}, recovery starts and (1) passes U_{acc} at t_{acc}. For (2) no CM is activated and utility drops to $U_{\text{low,perm}}$ (with *perm* referring to a system-inherent permanent robustness level) and no recovery occurs. When the attack ends (a), the utility recovers U_{target} eventually (i.e. it reaches the target space again, see Schmeck et al. (2010)). If the attack prevails or did permanent damage to the system (b), utility may stay at a lower value.

1b) S recovers during the attack to $U \geq U_{\text{acc}}$ and stays there when the attack ends. This is a *weakly robust system*.

2a) S does not recover during the attack but returns to the previous value after the attack ends. Such a system shows just a certain "elasticity", we call it *partially robust*.

2b) S does not recover during the attack, it stays at the low utility level $U_{\text{low,perm}}$. Such a system is not robust at all.

While important, this only serves as a classification of different typical behaviours. However, to compare the effect of different disturbances z or different CMs, we have to quantify the utility degradation. We define the utility degradation D_U as the area between a baseline utility (U_{baseline}) and $U(t)$ for the time when $U \leq U\text{acc}$ (see Equation 1):

$$D_U := \int_{t_z}^{t_{\bar{z}}} (U_{\text{baseline}}(t) - U(t))\,\mathrm{d}t \qquad (1)$$

First, we need to define a baseline U_{baseline} for the measurement. This can be either hypothetical by using U_{target} or U_{acc} (in Figure 1) or we can run a reference experiment (as we will show later in Scenario 2, see Section 4). In order to maximise the system robustness, we have to minimise D_U. A system, which never drops below U_{baseline} apparently has maximal robustness. We define the robustness during attack R_a as:

$$R_a := \frac{\int_{t_z}^{t_{\bar{z}}} U(t)\,\mathrm{d}t}{\int_{t_z}^{t_{\bar{z}}} U_{\text{baseline}}(t)\,\mathrm{d}t}$$

For each CM to be compared, we measure a utility degradation D_U. This allows comparing the behaviour of two CMs during an attack (such as cases (1) and (2) from above) or the effectiveness of one CM for different attacks.

Furthermore, when the attack ended, we can measure the long-term utility degradation $D_{U,\text{long\_term}}$ which is the area between U_{baseline} and the actual U for the time after $t_{\bar{z}}$. This allows us to compare the cases (1a/2a) and (1b/2b) from above. Hence, the long-term robustness R_l is defined as:

$$R_l := \frac{\int_{\bar{z}}^{t_{\text{target}}} U(t)\,\mathrm{d}t}{\int_{\bar{z}}^{t_{\text{target}}} U_{\text{baseline}}(t)\,\mathrm{d}t}$$

In cases where $U(t)$ never reaches U_{target} again, t_{target} is ∞ (also U_D is ∞) but we can calculate the open integral:

$$t_{\text{target}} = \infty \rightarrow R_l := \frac{\int_{\bar{z}}^{t_{\text{target}}} U(t)\,\mathrm{d}t}{\int_{\bar{z}}^{t_{\text{target}}} U_{\text{baseline}}(t)\,\mathrm{d}t} = \lim_{t\to\infty} \frac{U(\infty)}{U_{\text{baseline}}(\infty)}$$

In all normal cases, R is assumed to be in the interval $[0,1]$. It can never be negative and will only

be larger than 1 if $U(t)$ improves through the disturbance which happens only under very specific conditions (i.e. the system settled in a local optimum before the disturbance occurs, and is able to leave this local optimum as a result of the CM's intervention). Even if the utility $U(t)$ never recovers, we get an asymptotic robustness value. However, values of R_l with $t_{\text{target}} = \infty$ are not comparable to values for R_l with $t_{\text{target}} \neq \infty$ because the open integral would be always 1 in this case.

2.3 Related Work

The term "robustness" is widely used with different meanings in literature, mostly depending on the particular context or underlying research initiative. Typical definitions include the ability of a system to maintain its functionality even in the presence of changes in their internal structure or external environment (sometimes also called *resilient* or *dependable* systems) Callaway et al. (2000), or the degree to which a system is insensitive to effects that have not been explicitly considered in the design Slotine et al. (1991).

When especially considering engineering of information and communication technology-driven solutions, the term "robust" typically refers to a basic design concept that allows the system to function correctly (or, at the very minimum, not failing completely) under a large range of conditions or disturbances. This also includes dealing with manufacturing tolerances. Due to this wide scope of related work, the corresponding literature is immense, which is e.g. expressed by detailed reports that range back to the 90ies Taguchi (1993). For instance, in the context of scheduling systems, robustness of a schedule refers to its capability to be executable – and leading to satisfying results despite changes in the environment conditions Scholl et al. (2000). In contrast, the systems we are interested in (i.e. self-adaptive and self-organising OC systems) typically define robustness in terms of fault tolerance (e.g. Jalote (1994)).

In computer networks, robustness is often used as a concept to describe how well a set of service level agreements are satisfied. For instance, Menasce et al. explicitly investigate the robustness of a web server controller in terms of workloads that exhibit some sort of high variability in their intensity and/or service demands at the different resources Menascé et al. (2005).

More specifically and beyond these general definitions of the notion of robustness, we shift the focus towards quantification attempts. Only a few approaches known in literature aim at a generalised method to quantify robustness; in a majority of cases, self-organised systems are either shown to perform better (i.e. achieve a better system-inherent utility function) or react better in specific cases (or in the presence of certain disturbances), see e.g. ICAC (2015) and SASO (2015). . In the following, we discuss the most important approaches to robustness quantification.

In the context of OC, a first concept for a classification method has been presented in Schmeck et al. (2010). Here, the idea is (as in our approach) to take the system utility into account. Based on a predefined separation of different classes of goal achievement (i.e. distinguishing between target, acceptance, survival, and dead spaces in a state space model of the system S), the corresponding states are assigned to different degrees of robustness. Consequently, different systems are either strongly robust (i.e. not leaving the target space), robust (i.e. not leaving the acceptance space), or weakly robust (i.e. returning from the survival space in a defined interval). In contrast to our method, a quantitative comparison is not possible. In particular, this robustness classification does not take the recovery time into account.

Closely related is the approach by Nafz et al. presented in Nafz et al. (2011), where the internal self-adaptation mechanism of each element in a superior self-organising system has the goal to keep the element's behaviour within a pre-defined corridor of acceptable states. Using this formal idea, robustness can be estimated by the resulting goal violations at runtime. Given that system elements have to obey the same corridors, this would also result in a comparable metric. Recently, the underlying concept has been taken up again to develop a generalised approach for testing self-organised systems, where the behaviour of the system under test has to be expressable quantitatively Eberhardinger et al. (2015). However, this depends on the underlying state variables and invariants that are considered – which might be more difficult to assess at application level (compared to considering the utility function in our approach).

In contrast, Holzer et al. nodes are considered as stochastic automaton in a network and model the nodes' configuration as a random variable Holzer and de Meer (2011). Based on this approach, they compute the level of resilience (the term is used there similarly to robustness in this paper) depending on the network's correct functioning in the presence of malfunctioning nodes that are again modelled as stochastic automatons Holzer and de Meer (2009). In contrast to our approach, this does not result in a comparable metric and limits the scope of applicability due to the underlying modelling technique.

From a multi-agent perspective, Di Marzo Seru-

gendo approached a quantification of robustness using the accessible system properties Di Marzo Serugendo (2009). In general, properties are assumed to consists of invariants, robustness attributes, and dependability attributes. By counting or estimating the configuration variability of the robustness attributes, systems can be compared with respect to robustness. Albeit the authors discuss an interesting general idea, a detailed metric is still open.

Also in the context of multi-agent systems, Nimis and Lockemann presented an approach based on transactions Nimis and Lockemann (2004). They model a multi-agent system as a layered architecture (i.e. seven layers from communication at the bottom to the user at the top layer). Of particular interest is the third layer, i.e. the conversation layer. The key idea of their approach is to treat agent conversations as distributed transactions. The system is then assumed to be robust, if guarantees regarding these transactions can be formulated. This requires technical prerequisites, i.e. the states of the participating agents and the environment must be stored in a database – this serves as basis for formulating the guarantees. Obviously, such a concept assumes hard requirements that are seldom available, especially not in open, distributed systems where system elements are not under control of a centralised element and might behave unpredictably.

To conclude this discussion, we can observe that a generalised approach to quantify robustness is needed that: (a) works on externally measurable values, (b) does not need additional information sources (e.g. transactional data-bases), (c) distinguishes between system-inherent (or passive) and system-added (or active) robustness (to allow for an estimation of the effectiveness of the particular mechanism) and (d) comes up with a measure that allows for a comparison of different systems for the same problem instance. We claim that our approach as outlined before fulfils all of these aspects. In order to demonstrate the effectiveness, we apply it to three different use cases in the following part of this paper.

3 APPLICATION SCENARIO 1: WIRELESS SENSOR NETWORKS

Wireless Sensor Networks (WSNs) consist of spatially distributed nodes which communicate over a radio interface. Nodes sense locally and send the result to the root node in the network. Since most nodes cannot reach root directly, other nodes have to relay packets. To find a path to root, the *Routing Protocol for Lossy and Low Power Networks* (RPL) (Winter et al., 2012) is used. The primary objective (O1) in such networks is to reach a high Packet Delivery Rate (PDR; ranges from 0 to 1). Since nodes are battery-powered, the secondary objective (O2) is to minimise the number of Transmitted Packets (#TX) because sending data over the air causes most power consumption in WSNs. These two objectives translate into two utility functions, which we have to investigate with respect to their robustness. Utility function 1 is PDR(t), utility function 2 is #TX(t).

In open distributed sensor networks, attacks by malicious or broken nodes can occur, which lead to poor PDR. To counter such threats, we introduced end-to-end trust in RPL in previous work (Kantert et al., 2016b). In this trust-enhanced approach, the nodes assess the trustworthiness of their parents and isolate bad-behaving nodes. This constitutes a specific self-organised control mechanism CM. We are interested in a comparison of different variants of CMs:

- CM0 OF0. This is the default routing mechanism in RPL as described in (Winter et al., 2012). It selects parents by the smallest rank.
- CM1 Trust + ETX. Nodes use a trust metric to rate and isolate bad-performing parents. See (Kantert et al., 2016b) for more details. The particular method is not of interest in the context of this paper, it serves as a representative for a more powerful, self-organised control mechanism.
- CM2 Trust + ETX + Second Chance. This approach is similar to the previous one but also incorporates a mechanism to retry previously isolated parents occasionally (i.e. after re-stabilising the system, see again (Kantert et al., 2016b) for details). This approach serves as a representative with even more decision freedom of the CM.

In an undisturbed RPL network, our trust-enhanced system implemented as CM1 behaves very similar to standard RPL (CM0). However, when an attack occurs, standard RPL looses numerous packets and PDR drops because it cannot handle (intentional or unintentional) malicious behaviour. When enabling our approach, nodes start to identify and isolate bad-behaving parents. Hence, the PDR recovers to nearly 100% (i.e. U_{target}) while the attack happens. Standard RPL only recovers after the attack ends (see Figure 2).

For O1 (PDR(t)), D_U is 112 for CM0, 13.7 for CM1 and 17.9 for CM2. Quantitatively, D_U of CM1

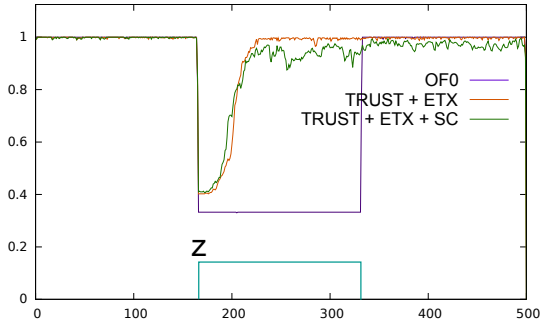

Figure 2: Objective 1, Utility function 1: Packet Delivery Rate (PDR) over sequences (time). At time step 160, an attack starts and ends at time step 340. PDR drops to about 30% for OF0 during the attack and recovers afterwards. TRUST + ETX and TRUST + ETX + Second-Chance recover during the attack and stay near 100% PDR until the end of the experiment.

is 87.7% better than CM0 and CM2 is 84% better than CM0. Also, CM1 has 23.5% smaller D_U than CM2. The baseline is 1 (see Equation 2). R_a for CM0 has a value of 70% (3), CM1 has a value of 91% (4) and CM2 has a value of 89%(5).

$$U_{\text{baseline}}(t) := 1 \tag{2}$$

$$R_{a,\text{CM0}} \approx 70\% \tag{3}$$

$$R_{a,\text{CM1}} := \frac{146.3}{160} \approx 91\% \tag{4}$$

$$R_{a,\text{CM2}} := \frac{142.1}{160} \approx 89\% \tag{5}$$

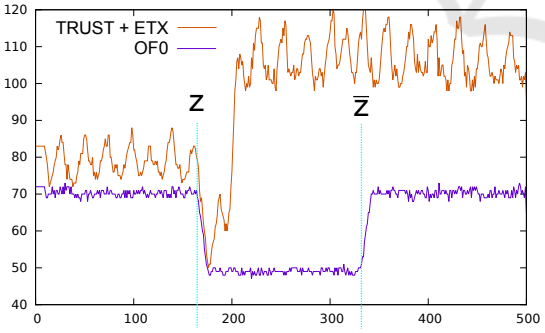

Figure 3: Objective 2, Utility function 2: Transmitted Packets (#TX) over time. At time step 160, an attack starts, and ends it at time step 340. During the attack #TX increases for the TRUST + ETX control mechanism (CM1) and stays at that level after the attack. For OF0 (CM0), #TX drops during the attack and recovers to the undisturbed level afterwards.

For objective O1 (i.e. PDR(t)) CM1 recovers perfectly during the attack and returns to the same value as before the attack. Therefore, in this scenario, RPL with Trust + ETX is rated as strongly robust. However, this changes when we look at the second objective O2 represented by utility function 2: metric #TX(t) (the number of transmitted packets, see Figure 3). When the attack starts, #TX drops for all CMs (which would be perfect if PDR stayed at a constant level). For Trust + ETX it increases to a higher level because the new routes are longer and require more transmissions. This is expected since some parents failed. However, after the attack ends, only standard RPL returns to its previous #TX. Trust + ETX stays at a high level. This is acceptable when dealing with intentional malicious attackers but this is bad when disturbances are only temporary. Therefore, Trust + ETX is not robust regarding the second utility function.

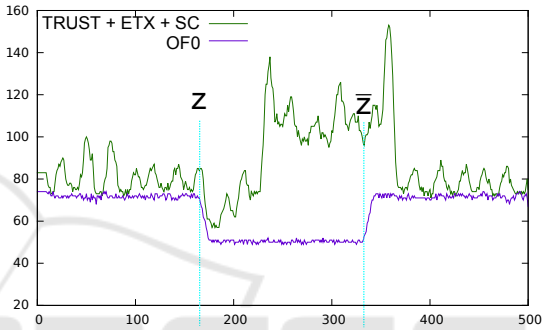

Figure 4: Objective 2, Utility function 2: Transmitted Packets (#TX) over time. At time step 160, an attack starts, and ends at time step 340. During the attack #TX(t) increases for the TRUST + ETX + Second-Chance (CM2). For OF0 (CM0), #TX(t) drops during the attack and recovers to the same level as before afterwards. Unlike in Figure 3, TRUST + ETX + Second-Chance (CM2) recovers to a similar level as before the attack.

Most energy in wireless sensor nodes is used when sending packets. Thus, to recover the energy consumption and #TX after an attack, we introduced TRUST + ETX + Second-Chance as a third control mechanism CM2 which retries parents when the system has stabilised (from a node's local perspective). This leads to a slight decrease of the PDR during the attack because nodes loose packets when retrying during an attack. However, after the attack the PDR is very similar and #TX recovers to its level from before the attack (see Figures 2 and 4).

For O2, we measure D_U only after the attack. Quantitatively, D_U for CM1 is ∞ because it does not return to the previous values. For CM0 D_U is about 0. CM2 needs some time to recover and has a D_U of 1998.6. Thus, CM0 is the best metric when only considering O2 (since it is very bad for O1). In our robustness metric, we have to assume that a higher utility is better. In this case, the best utility for O2 would be a value of 0 (which is unrealistic). There-

fore, we invert $U(t)$ to get a utility function for the calculation of R_l (see Equation 6). This results in a baseline U_{baseline} of 43 (saved packets per second; see Equation 7). For CM1 this results in an asymptotic long-term robustness R_l of 35% (8). CM2 recovers and has a robustness value R_l of 47% (not comparable to CM1; Equation 9).

$$U_{\text{normalised}}(t) \coloneqq 120 - U(t) \tag{6}$$

$$U_{\text{baseline}} \coloneqq 120 - 77 = 43 \frac{\text{packets}}{s} \tag{7}$$

$$R_{l,\text{CM1}} \approx \frac{120 - 105}{43} \approx 35\% \tag{8}$$

$$R_{l,\text{CM2}} \approx \frac{401.4}{860} \approx 47\% \tag{9}$$

CM2 (TRUST + ETX + Second-Chance) is strongly robust for PDR. Also, it is robust regarding #TX. It does not recover during the attack (because better routes do not exist) but it recovers to the previous state when the attack ends.

4 APPLICATION SCENARIO 2: OPEN DISTRIBUTED SYSTEMS

Another scenario we measured robustness in is the simulation of an open, distributed Multi-agent System with applied trust metrics, the *Trusted Desktop Grid* (TDG) (Edenhofer et al., 2015). In this grid, jobs J are created by the agents and split into work units (WU), which are calculated in a distributed way by the agents. The success (or utility) is expressed as the speedup $\sigma(t)$. $\sigma(t)$ is defined as the time the agent would have needed to process all work units on its own ($self$) divided by the real time it took to calculate the job in a distributed way in the system ($dist$), see Equation 10. A job J is released in time step t_J^{rel} and completed in t_J^{compl}.

$$\sigma = \frac{\sum_J (t_{self}^{compl} - t_{self}^{rel})}{\sum_J (t_{dist}^{compl} - t_{dist}^{rel})} \tag{10}$$

In this scenario, our objective is to maximise the utility function speedup $\sigma(t)$ (see Equation 10). Nodes make decisions based on a local trust metric and isolate bad-behaving agents using this Control Mechanism (CM1; see (Klejnowski, 2014) for details). We compare different attacks:

A0 No attack. Used as baseline.

A1 A short attack which ends at $t_{\overline{z}}$.

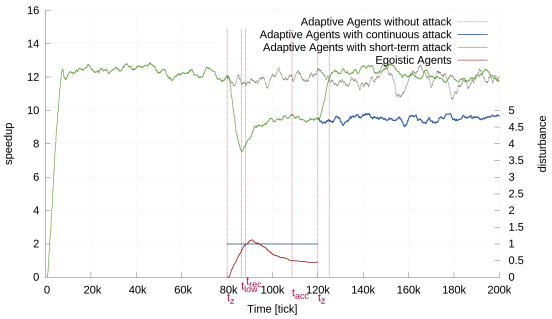

Figure 5: Disturbance D1: Robustness in the TDG. Attack of 100 EGO from tick 80,000 to 120,000 and continuous from 80,0000 (A2). t_{low} marks the point in time, where $\sigma(t)$ is at its lowest. The speedup starts to recover (t_{rec}) during the attack until it reaches U_{acc} (at t_{acc}). After the attack has stopped, the speedup returns to the level it had before for A1. However, it stays at about 10 for A2.

A2 Permanent attack which continues until the experiment ends.

Additionally, we consider two different disturbances by stereo-type attacker behaviour:

D1 *Egoistic agents* (EGO) are accepting all WU, but abort 80% of them after some time, which decreases $\sigma(t)$ because these WUs have to be redistributed.

D2 *Freeriding agents* (FRE) do not accept WU at all. They reject WUs right away but try to distribute their WUs at the same time.

For evaluation purposes, we investigated exemplary scenarios that incorporate disturbed system states. More precisely, we simulated 100 well-behaving adaptive agents (ADA). At time t_z, 100 bad-behaving egoistic agents (EGO; D1) join the system, simulating a colluding attack. To show the effect of robustness in the system, we calculated the average speedup of 10 runs with a length of 200,000 ticks each. We want to compare the system behaviour (i.e. its robustness) under two different attacks. Attack A1 starts at tick 80,000 and lasts 40,000 ticks. Attack A2 is a continuous attack of 100 EGO starting at tick 80,000. As baseline, we ran the same experiment without an attack (A1; see Figure 5).

In both attacks (A1 and A2), the average speedup of the ADA during the attack of the EGO is decreased, due to the malevolent behaviour of the EGO. t_{low} marks the point in time, where σ of the ADA is at its lowest. During the attack, once the average speedup of ADA increases by 5% (t_{rec}) over $\sigma(t)$ at t_{low}, the recovery phase is said to start; recovery is defined to be reached, if $\sigma(t)$ is at least 75% (at t_{acc}) of $\sigma(t)$ before the attack in t_z.

In the first attack, after tick 120,000, the ADA have to redistribute the WUs formerly occupied by

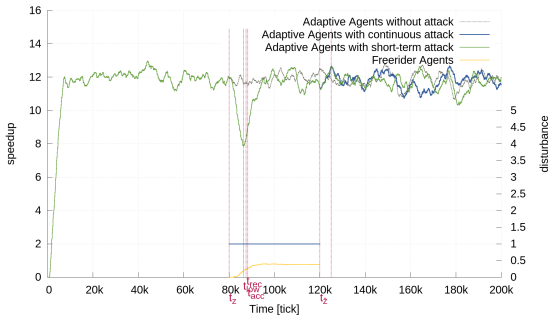

Figure 6: Disturbance D2: Robustness in the TDG. Attack of 100 FRE from tick 80,000 to 120,000 (A1) and continuous from 80,0000 (A2) t_{low} marks the point in time, where $\sigma(t)$ is at its lowest. The speedup starts to recover (t_{rec}) during the attack and quickly reaches U_{acc} and U_{target} for both A1 and A2. At tick 120,000 A1 and A2 are the same because the attackers are fully isolated and have no further influence.

EGO. After these WUs have been successfully calculated, $\sigma(t)$ recovers to the level it had before the attack. In both attacks, $\sigma(t)$ of the ADA recovers during the attack until it reaches the acceptance space (i.e. system states where the utility is above U_{acc}). This is due to the EGOs getting low trust ratings because they abort their assigned WUs. Yet, they manage to retain a high enough reputation to still get some WUs computed by other agents. As 80% of the WUs held by EGOs have to be redistributed, $\sigma(t)$ of the ADA cannot recover to the level it had before the attack.

Quantitatively, D_U is 130,105 below the baseline for both A1 and A2. After the attack, D_U is 6250 for A2. D1 has a permanent influence and CM1 is not able to fully mitigate the effect. We measured the baseline utility U_{baseline} in a reference experiment. R_a for A1 and A2 is with 72% (see Equation 11). Similar to, O2 in Scenario 1, only A2 fully recovers. For A1, we calculate R_l in an open integral in Equation 12. Again, not comparable, R_l for A2 can be calculated in a closed integral (see Equation 13; the open integral would have a value of 1).

$$R_{a,\text{A1}} = R_{a,\text{A2}} \coloneqq \frac{349,895}{480,000} \approx 73\% \tag{11}$$

$$R_{l,\text{A1}} \approx \frac{9.5}{12} \approx 79\% \tag{12}$$

$$R_{l,\text{A2}} \coloneqq \frac{113,750}{120,000} \approx 95\% \tag{13}$$

Similarly, we run experiments for disturbance D2 with free-riding agents (see Figure 6). Since those agents are simpler to detect, they are isolated within the attack period of A1 and the system fully recovers to U_{target} within the attack. U_D is 13,056 for both A1 and A2 during the attack. After the attack ends, A1 is already fully recovered and the utility for A1 and A2 is similar. CM1 is able to fully mitigate this disturbance D2. R_a is 97% (see Equation 14). Since $U(t)$ fully recovers for both attacks, R_l is about 100% for A1 and A2.

$$R_{a,\text{D2}} \coloneqq \frac{466,944}{480,000} \approx 97\% \tag{14}$$

5 APPLICATION SCENARIO 3: URBAN TRAFFIC MANAGEMENT

A third scenario that is often used as basis to investigate a self-organisation mechanism due its inherent dynamics and distributed nature is the control of traffic lights in urban areas, see (Bazzan and Klügl, 2009; Dinopoulou et al., 2006) for instance. One of the major OC-based contributions in this context is the *Organic Traffic Control* (OTC) system (Prothmann et al., 2011) that applies the observer/controller approach as well as an autonomous and safety-oriented learning process to the traffic domain.

In OTC, each intersection of the inner-city road network is managed by an observer/controller instance that gathers detector data about the underlying traffic conditions (in terms of flows passing each turning movement) and reacts by adapting green durations. The success of the control strategy is typically expressed as flow-weighted delays:

$$t_D = \frac{\sum_i (M_i \times t_{d,i})}{\sum_i M_i} \tag{15}$$

Where M_i corresponds to the current traffic flow at the i-th turning of the observed intersection and $t_{d,i}$ denotes the average waiting time with respect to a single turning t_i. This metric is also referred to as *Level of Service* (LoS) Transportation Research Board (2000). In addition, the same metric is used to determine the most promising progressive signal system Tomforde et al. (2010).

Adapting traffic signalisation to changing demands and coordinating intersection controllers provides a first step towards robustness – but this does not dissolve the initial disturbance. In terms of the previously developed notion, this accounts as passive robustness. Re-routing of traffic participants counters the disturbance (i.e. a blocked road or traffic jams) directly and can be considered as active robustness mechanism in this context. In previous work, we equipped OTC with such a mechanism Prothmann et al. (2012). Based on ideas resembling the Link State or Distance Vector Routing

protocols as known from the computer networks domain(Tanenbaum, 2002), information about the shortest routes are exchanged between intersection controllers and provided to drivers at each incoming intersection, see Prothmann et al. (2011, 2012) for details.

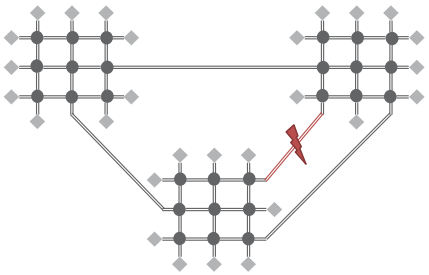

Figure 7: An exemplary model of 27 intersections (depicted as grey circles), forming three connected Manhattan-style road networks. Traffic enters and leaves the network at locations referred to as destinations (depicted as light-grey diamonds). In the evaluation, the red road is blocked during attack A1.

In this scenario, the objective is to maximise the utility function *traffic flow* which can be measured globally. At t_z, one road is blocked until $t_{\overline{z}}$ (attack A1, red road in Figure 7). We compare three different control mechanisms:

CM0 No active control mechanism. Used as baseline.

CM1 OTC without routing.

CM2 OTC with routing.

The evaluation has been conducted for a simulated network that is illustrated in Figure 7. The network consists of three Manhattan-type sub-networks. It contains 27 signaled intersections (depicted as circles) and 28 prominent destinations (depicted as diamonds). Within each sub-network, the intersections are connected by one-laned road segments of 250m length that provide two additional turning lanes starting 100m before an intersection. Regions are connected by two-laned roads. Signalised intersections are operated by an observer/controller (i.e. OTC) and can provide route recommendations for the prominent destinations. Each destination also serves as origin for traffic entering the network.

We configured the simulation as follows: eight vehicles per hour travel from every origin to every destination. Since this demand does not cause significant jams at the network's intersections, the scenario allows evaluating the impact of routing mechanisms under uncongested conditions. As disturbance, we temporarily blocked one of the roads connecting two sub-networks due to an incident (see mark in Figure 7). The blockage affects both directions of the road, occurs after 25 min and lasts for 40 min within the simulation period. The incident scenario allows analysing the impact of the two different adaptation mechanisms (CM1 = OTC without routing, and CM2 = OTC with routing) in comparison to a standard fixed-time control approach (= CM0).

Within the professional simulator Aimsun Barceló et al. (2005), we performed five runs of experiments for each instance under evaluation and computed averages. Figure 8 depicts the achieved results. We can clearly observe that the typical system behaviour of a disturbed and recovering system is visible in this scenario. When considering the OTC system without routing in comparison to a standard fixed-time control, we can distinguish two effects: a) the fixed-time controller (i.e. CM0) defines the base-line (i.e. the maximum loss of utility) and b) the standard OTC system (CM1) already provides a permanent robustness increase that improves the behaviour even if no disturbance takes place. If activating the routing mechanism (CM2), an active robustness is added. Quantitatively, U_D for CM1 is about 5.4% better than the reference solution (CM0) (i.e. 5534 vs 5250 vehicles per hour) and routing (CM2) adds further utility, i.e. an improvement in U_D of 14.0% (i.e. 6312 vehicles per hour).

$$U_{\text{baseline}}(t) \coloneqq 7{,}000 \frac{\text{vehices}}{h} = 116.7 \frac{\text{vehicles}}{\text{min}} \tag{16}$$

$$R_{a,\text{CM0}} \coloneqq \frac{87.5}{116.7} \approx 75\% \tag{17}$$

$$R_{a,\text{CM1}} \coloneqq \frac{92.2}{116.7} \approx 79\% \tag{18}$$

$$R_{a,\text{CM2}} \coloneqq \frac{105.2}{116.7} \approx 90\% \tag{19}$$

To calculate the robustness, we use a baseline based on U_{target} (see Equation 16). Using the U_D values from above, we calculate the robustness R_a for all CMs in Equations (17) to (19). What we can see in the figure is that a passive robustness is already in place with CM0, since most of the network is still operating without strong impact of the disturbance (in terms of traffic flow through the network). This changes if we just consider the blocked link: Here, for all three mechanisms the utility drops to zero and recovers immediately when the blockade is removed (not shown in figure). Hence, the robustness is achieved at network-level, since participants are routed using the best available link (i.e. with routing mechanism, CM2) or at least the non-affected participants benefit due to longer green durations (i.e. a blocked road receives a lower fraction of the phase cycle time when using OTC, CM1 and CM2). With this mechanism at hand, we are now able to compare the developed

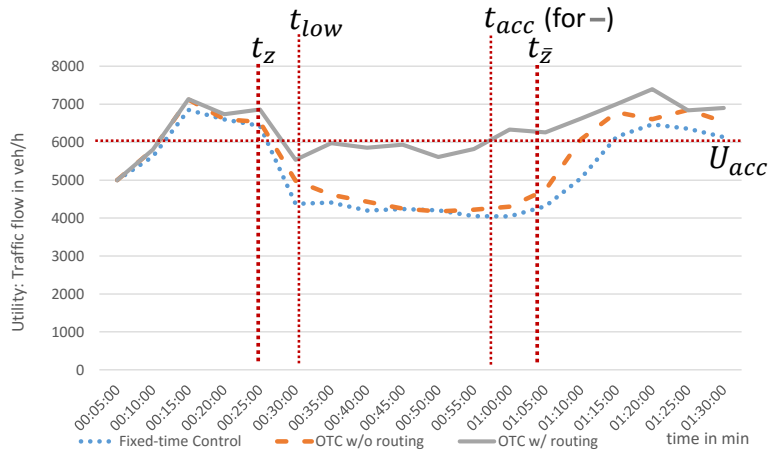

Figure 8: Utility over time for OTC. At t_z one road is blocked (attack A1) and traffic has to be redirected. With fixed-time control (CM0), utility drops by nearly 3,000 vehicles per hour. OTC without routing (CM1) performs slightly better. Compared, OTC with routing (CM2) only drops about 1,000 vehicles per hour. After the attack A1 ends, all CMs recover the system to their previous utility.

mechanisms to other solutions from the state-of-the-art (such as Wedde et al. (2007) for self-organised routing or Dinopoulou et al. (2006) for autonomous traffic controllers) and quantify the varying robustness levels in different disturbance scenarios.

6 DISCUSSION

We demonstrated that our approach is applicable to three varying scenarios from different application domains. In Scenario 1, we compared three different control mechanisms with two different objective functions for wireless sensor networks. The resulting values are reasonable and allow for a simple and effective comparison of robustness levels for the considered CMs. In Scenario 2, we compared two different attack types and two different disturbances in a grid scenario. We measured the long-term influence for the disturbances and found that only one of the disturbances has a permanent effect. Afterwards, in Scenario 3, we measured the robustness for three CMs in a traffic management scenario.

The general shape of the utility graph was very similar in all our scenarios. Still, the utility degradation U_D is not comparable between scenarios because it has an application specific unit (i.e. it is hard to compare apples and oranges). However, our robustness metrics R_a and R_l are unit free and always have the same boundaries. Thus, they allow some comparison between different systems.

Nevertheless, some limitations apply: We have to set a fixed attack length $t_{\bar{z}} - t_z$. This limits the applicability of the concepts to those cases where a disturbance is observable in the first place. Current research addresses this topic by developing techniques to deal with this issue. Also, it influences the value of R as illustrated by Scenario 2. If we had set $t_{\bar{z}}$, earlier the robustness R_a for both CMs would be lower. A similar challenge occurs when calculating R_l for two CMs which reach U_{target} at different times. To make their robustness R_l values comparable, a later t_{target} has to be chosen. This issue can be solved but it requires some caution when designing experiments.

We noticed another challenge when calculating R_l for O2 in Scenario 1 because our goal was to minimise the metric. However, for a utility function $U(t)$ a higher value should be better. Hence, we had to invert the objective function to calculate the utility. There is no need to normalise the utility function but larger values must be better which is important to keep the robustness values inside the interval between 0 and 1.

Finally, we saw in Scenario 1 that it is crucial to choose the right utility function. If we only consider O1, A1 has the best robustness. However, if we also take into account O2, A2 is clearly better. The remaining challenge here is to properly combine multiple metrics.

In future work, we will focus on quantitatively comparing similar mechanisms in different application scenarios. Also, we would like to extend the selection of integration limits for better comparison.

7 CONCLUSION

Self-adaptation and self-organisation are concepts increasingly incorporated in system design. This is

mostly due to one key motivation: We want to improve the robustness of technical systems in terms of a utility preservation in order to maintain the system's functionality even under harsh external conditions or the presence of internal failures (we refer to such effects in general as *disturbances*). Hence, one of the most important aspects to judge whether one specific solution is more beneficial than another is to estimate which system is more robust. Such a decision process needs a quantitative basis to come up with a meaningful statement. In this paper, we presented a novel method that estimates such a measurement at runtime.

We discussed the state-of-the-art and explained that existing approaches have drawbacks, e.g. requiring too much internal information, measuring only certain aspects (such as the time), being application-specific, or abstracting the robustness too far (i.e. coming up with a discretisation of a few classes only). With our method, we focus on externally measurable attributes only and allow for a generalised concept for comparing robustness. We further distinguish between a permanent part of robustness that is system inherent and a part that is generated by internal adaptation mechanisms. We demonstrated the expressiveness of the developed approach in terms of three case studies, i.e. from the desktop grid, the wireless sensor network, and the traffic control domains.

In future work, we will investigate how our method behaves when comparing different systems within the same application domain. In addition, we focus on questions regarding the heterogeneity of occurring disturbances: Where are the drawbacks and advantages of the developed technique and what needs to be improved to come up with a fully generally applicable method?

REFERENCES

(2015). *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems, Cambridge, MA, USA, September 21-25, 2015*. IEEE Computer Society.

(2015). *2015 IEEE International Conference on Autonomic Computing, Grenoble, France, July 7-10, 2015*. IEEE Computer Society.

Barceló, J., Codina, E., Casas, J., Ferrer, J., and García, D. (2005). Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *Journal of Intelligent and Robotic Systems*, 41(2–3):173–203.

Bazzan, A. L. C. and Klügl, F., editors (2009). *Multi-Agent Systems for Traffic and Transportation Engineering*. Idea Group Publishing, Hershey, PA, US.

Callaway, D. S., Newman, M. E., Strogatz, S. H., and Watts, D. J. (2000). Network robustness and fragility: Percolation on random graphs. *Physical review letters*, 85(25):5468.

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

Di Marzo Serugendo, G. (2009). *Stabilization, Safety, and Security of Distributed Systems: 11th International Symposium, SSS 2009, Lyon, France, November 3-6, 2009. Proceedings*, chapter Robustness and Dependability of Self-Organizing Systems - A Safety Engineering Perspective, pages 254–268. Springer Berlin Heidelberg, Berlin, Heidelberg.

Dinopoulou, V., Diakaki, C., and Papageorgiou, M. (2006). Applications of the urban traffic control strategy tuc. *European Journal of Operational Research*, 175(3):1652–1665.

Eberhardinger, B., Anders, G., Seebach, H., Siefert, F., and Reif, W. (2015). A research overview and evaluation of performance metrics for self-organization algorithms. In *2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops 2015, Cambridge, MA, USA, September 21-25, 2015*, pages 122–127.

Edenhofer, S., Stifter, C., Jänen, U., Kantert, J., Tomforde, S., Hähner, J., and Müller-Schloer, C. (2015). An accusation-based strategy to handle undesirable behaviour in multi-agent systems. In *2015 IEEE International Conference on Autonomic Computing, Grenoble, France, July 7-10, 2015*, pages 243–248.

Holzer, R. and de Meer, H. (2009). Quantitative modeling of self-organizing properties. In *Self-Organizing Systems, 4th IFIP TC 6 International Workshop, IWSOS 2009, Zurich, Switzerland, December 9-11, 2009. Proceedings*, pages 149–161.

Holzer, R. and de Meer, H. (2011). Methods for approximations of quantitative measures in self-organizing systems. In *Self-Organizing Systems - 5th International Workshop, IWSOS 2011, Karlsruhe, Germany, February 23-24. 2011. Proceedings*, pages 1–15.

IBM (2005). An architectural blueprint for autonomic computing. Technical report, IBM.

Jalote, P. (1994). *Fault tolerance in distributed systems*. Prentice-Hall, Inc.

Kantert, J., Edenhofer, S., Tomforde, S., Hähner, J., and Müller-Schloer, C. (2016a). *Normative Control – Controlling Open Distributed Systems with Autonomous Entities*, volume 7 of *Autonomic Systems*, pages 87–123. Springer International Publishing.

Kantert, J., Reinhard, F., von Zengen, G., Tomforde, S., Wolf, L., and Müller-Schloer, C. (2016b). Combining Trust and ETX to Provide Robust Wireless Sensor Networks. In Varbanescu, A. L., editor, *Workshop Proceedings of the 29th International Conference on Architecture of Computing Systems*, chapter 16, pages 1–7. VDE Verlag GmbH, Berlin, Offenbach, DE, Nuremberg, Germany.

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

Menascé, D. A., Bennani, M. N., and Ruan, H. (2005). *Self-star Properties in Complex Information Systems: Conceptual and Practical Foundations*, chapter On the Use of Online Analytic Performance Models, in Self-Managing and Self-Organizing Computer Systems, pages 128–142. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mnif, M. and Müller-Schloer, C. (2006). Quantitative Emergence. In *Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals 2006), held 24 Jul - 26 Jul 2006, Utah State University College of Engineering Logan, UT, USA*, pages 78–84, Piscataway, NJ, USA. IEEE.

Nafz, F., Seebach, H., Steghöfer, J.-P., Anders, G., and Reif, W. (2011). Constraining Self-organisation Through Corridors of Correct Behaviour: The Restore Invariant Approach. In Müller-Schloer, C., Schmeck, H., and Ungerer, T., editors, *Organic Computing – A Paradigm Shift for Complex Systems*, Autonomic Systems, pages 79 – 93. Birkhäuser Verlag, Basel, Switzerland.

Nimis, J. and Lockemann, P. C. (2004). Robust multi-agent systems: The transactional conversation approach. In *First International Workshop on Safety and Security in Multiagent Systems (SASEMAS'04), S*, pages 73–84.

Prothmann, H., Tomforde, S., Branke, J., Hähner, J., Müller-Schloer, C., and Schmeck, H. (2011). Organic Traffic Control. In *Organic Computing – A Paradigm Shift for Complex Systems*, pages 431 – 446. Birkhäuser Verlag.

Prothmann, H., Tomforde, S., Lyda, J., Branke, J., Hähner, J., Müller-Schloer, C., and Schmeck, H. (2012). Self-organised Routing for Road Networks . In *Proc. of the International Workshop on Self-Organising Systems (IWSOS'12), held in Delft, The Netherlands, March 15 - 16, 2012*, number 7166 in LNCS, pages 48 – 59. Springer Verlag.

Schmeck, H., Müller-Schloer, C., Çakar, E., Mnif, M., and Richter, U. (2010). Adaptivity and Self-organisation in Organic Computing Systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 5(3):1–32.

Scholl, A. et al. (2000). Robuste planung und optimierung: Grundlagen, konzepte und methoden; experimentelle untersuchungen. Technical report, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).

Slotine, J.-J. E., Li, W., et al. (1991). *Applied nonlinear control*, volume 199. Prentice-Hall Englewood Cliffs, NJ.

Taguchi, G. (1993). Robust technology development. *Mechanical Engineering-CIME*, 115(3):60–63.

Tanenbaum, A. S. (2002). *Computer Networks*. Pearson Education, 4th edition.

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., and Schmeck, H. (2011). Observation and Control of Organic Systems. In *Organic Computing - A Paradigm Shift for Complex Systems*, pages 325 – 338. Birkhäuser Verlag.

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Müller-Schloer, C., and Schmeck, H. (2010). Possibilities and limitations of decentralised traffic control systems. In *2010 IEEE World Congress on Computational Intelligence (IEEE WCCI 2010)*, pages 3298–3306. IEEE.

Transportation Research Board (2000). Highway capacity manual. Technical report, National Research Council, Washington D.C., US.

Wedde, H. F., Lehnhoff, S., et al. (2007). Highly dynamic and adaptive traffic congestion avoidance in real-time inspired by honey bee behavior. In Holleczek, P. and Vogel-Heuser, B., editors, *Mobilität und Echtzeit – Fachtagung der GI-Fachgruppe Echtzeitsysteme*, pages 21–31. Springer.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard).