UNIVERSITÄT AUGSBURG
Universitätsbibliothek

OPEN ACCESS
OPUS AUGSBURG

Measuring and evaluating the performance of self-organization mechanisms within collective adaptive systems

Benedikt Eberhardinger, Hella Ponsar, Dominik Klumpp, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Eberhardinger, Benedikt, Hella Ponsar, Dominik Klumpp, and Wolfgang Reif. 2018. "Measuring and evaluating the performance of self-organization mechanisms within collective adaptive systems." In *Leveraging Applications of Formal Methods, Verification and Validation - Distributed Systems: 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, proceedings, Part III*, edited by Tiziana Margaria and Bernhard Steffen, 202–20. Cham: Springer. https://doi.org/10.1007/978-3-030-03424-5_14.

Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
Deutsches Urheberrecht
Weitere Informationen finden Sie unter: / For more information see:
https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/

Measuring and Evaluating the Performance of Self-Organization Mechanisms Within Collective Adaptive Systems

Benedikt Eberhardinger(✉), Hella Ponsar, Dominik Klumpp, and Wolfgang Reif

Institute for Software and Systems Engineering, University of Augsburg, Augsburg, Germany
{eberhardinger,ponsar,reif}@isse.de,
dominik.raphael.klumpp@student.uni-augsburg.de

Abstract. By restructuring and reconfiguring itself at run-time, a collective adaptive system (CAS) is able to fulfill its requirements under uncertain, ever-changing environmental conditions. Indeed, this process of self-organization (SO) is of utmost importance for the ability of the CAS to perform. However, it is hard to design high-performing SO mechanisms, because the environmental conditions are partially unpredictable at design time. Thus, a crucial aid for the development of SO mechanisms is a tool set enabling performance evaluations at design time in order to select the best-fitting mechanism and parametrize it. We present a metric for measuring the performance of an SO mechanism as well as a framework that enables evaluation of this metric. The proposed metric is evaluated for different kinds of SO mechanisms in two case studies: a smart energy management system and a self-organizing production cell.

1 Performance of Self-Organization Mechanisms

The performance of software denotes its capabilities in its execution. These capabilities might be determined either by a theoretical analysis or by an experimental evaluation. In general, two measures are of interest: the solution quality and the time taken to achieve the solution [11]. Whereas theoretical analysis are based on abstraction, theorems, and proofs in order to find an asymptotic bound on the dominant operation under a worst-case or average-case mode, experimental evaluation relies on execution, logging, and measuring according to a set of metrics. The knowledge gained from the performance analysis, theoretical as well as experimental, is used for engineering efficient and effective software. The gain for the engineers highly depends on the quality of the analysis.. However, achieving a high quality for the analysis is a challenging task [11]. Theoretical as well as experimental analysis are foremost challenged when the system under evaluation is indeterministic, highly parallel, interactive, or highly dependent on

unforeseeable run-time conditions. All these aspects are characteristics of collective adaptive systems (CAS) [4]. A CAS uses its abilities to reconfigure and restructure itself at run-time in order to cope with an ever-changing environment. Self-organization (SO) mechanisms are used to fulfill this reconfiguration and restructuring task. An important aspect, that is exploited by most of the SO mechanisms (cf. [1]), is that mostly that process can be carried out locally, i.e., in a small part of the overall system. This makes the SO mechanisms scalable and effective which has a great impact on the overall system performance. However, it is far from obvious how to design and implement the best performing SO mechanism for a certain system, because SO mechanisms have to operate under ever-changing environmental conditions that are partially unpredictable at design time. This demands a powerful performance analysis to support this task. In our approach, we face the challenge by focusing on an experimental analysis, following van Dyke Parunak and Brueckner [15], who argue that there is a need of experimental evaluation of SO mechanisms, because the concepts of theoretical analysis are stretched to their limits given that the majority of SO systems are formally undecidable.

This paper contains the following contributions for measuring and evaluating the performance of SO mechanisms within CAS: (1) A *performance metric* for distributed SO mechanisms; (2) a *simulation concept* for measuring performance; (3) a *framework for performance evaluation* incorporating these two concepts; (4) an *intensive evaluation* of different mechanisms within different systems.

The rest of the paper is structured as follows: We introduce our two case studies in Sect. 2. The performance metrics of SO mechanisms are derived in Sect. 3. Section 4 describes our evaluation framework that is used to evaluate the metrics. Within the framework we evaluate the two cases studies according to the introduced metrics in Sect. 5. Section 6 places the contribution of this paper into the related work and Sect. 7 finally concludes the paper.

2 Case Studies

We use two different case studies with different SO mechanisms throughout this paper as well as in our evaluation. First, a self-organizing production cell and second, a self-organizing smart grid. Both are categorized as CAS, according to [4]. Our first case study represents the class of systems with *discrete* SO mechanisms and the second case study the class of *continuous* SO mechanisms. Discrete SO mechanisms are working on a discrete input space, e.g., a tool that breaks and forces a reconfiguration, and continuous SO mechanisms are working on a continuous input space, e.g., a quality rating index where the system's configuration depends on.

2.1 Self-Organizing Production Cell

Future production scenarios demand for much more flexibility [12] than today's shop floor design to cope with the trend towards small series production, individualized products and the reuse of production stations for different tasks. These

future CAS will integrate SO mechanisms to resolve the tasks of decentralized decision making, to optimize the systems structure, and to autonomously react to component failures at runtime increasing the system's robustness. We envision self-organizing production cells, where the production stations are modern robots equipped with toolboxes and the ability to change their tools whenever necessary (self-awareness). They are connected via mobile carts that are able to carry workpieces and to reach robots in any order. Thus, the production cell is able to fulfill any task which corresponds to tools (capabilities) available in the cell. This is possible due to the SO mechanisms that reconfigures the carts and robots in a way that the tools are applied to the workpieces in the correct order.

2.2 Self-Organizing Virtual Power Plants in Smart Grids

The wide-spread installation of weather-dependent power plants as well as the bunch of new consumer types like electric vehicles put a lot of strain on power grids. To save expenses, gain more flexibility, and deal with uncertainties, future *autonomous* power management systems have to take advantage of the full potential of dispatchable prosumers[1] by incorporating them into the scheduling scheme. Further, aleatoric uncertainties have to be anticipated when creating schedules and compensated for locally to prevent their propagation through the system. To meet the challenges of future power management systems, Steghöfer et al. [18] presented the concept of Autonomous Virtual Power Plants (AVPPs) which represent self-organizing groups of two or more power plants of various types that form a CAS. The organizational structure represents a *partitioning*, i.e., every power plant is a member of exactly one AVPP, which is established and maintained by a (partitioning-based) SO algorithm. AVPPs autonomously adapt their structure to changing internal or environmental conditions, they are able to live up to the responsibility of maintaining an organizational structure enabling the system to hold the balance between energy supply and demand. In particular, if an AVPP repeatedly cannot satisfy its assigned fraction of the overall demand or compensate for its local uncertainties, it triggers a reconfiguration of the partitioning. To cope with the vast number of dispatchable power plants, the concept of AVPPs proposes a scalable, hierarchical structure in which AVPPs act as intermediaries. This system decomposition reduces the number of dispatchable power plants each AVPP controls resulting in shorter scheduling times for each AVPP and the overall system.

3 Performance Metric for Distributed SO Mechanisms

In our previous work [6], we evaluated in-depth performance metrics given by the literature regarding their ability to cope with SO algorithms and mechanisms. For this purpose, we implemented the six most promising metrics and applied them to an SO mechanism. The results was, that none of the investigated metrics

[1] We use the term "prosumer" to refer to producers as well as consumers.

were fully capable to judge the performance of an SO mechanism. Thus, we derived a set of requirements that metrics have to fulfill [6]:

Req. 1: The **locality** of SO mechanisms has to be taken into account and the aspects of *time* and *solution quality* have to be evaluated within the subsystems (changing over run-time) that are differently affected by the SO mechanisms, e.g., one subsystem can be reconfigured while another one keeps on working. Furthermore, it is important to be able to assess the performance of the entire system based on the performance of the subsystems.

Req. 2: Since SO mechanisms have control over the system's structure, their performance strongly influences those of the entire system. So the overhead of a reconfiguration can be worthwhile if it sufficiently improves the behavior of the controlled system. Consequently, a metric has to take the **benefit of the reconfiguration** into account.

Req. 3: The interpretation of a value provided by a metric strongly depends on the current state of the system. In self-organizing systems, the possible values for the solution quality can change over time. For instance, a solution quality of 0.7 would be optimal if possible values were defined by the interval $[0, 0.7]$ but quite bad if they stem from the interval $[0, 200]$; the same applies to the parameter *time*. Consequently, there is a need for **dynamic boundaries for the evaluation**—a result from the ever-changing environment of SO mechanisms.

We used these requirements in order to form a metric that is able to cope with decentralized SO mechanisms, is defined locally, respects the benefit of a reconfiguration, and handles dynamic boundaries. The performance of a system is composed of two parts: time performance and quality performance. Thus the performance p of a system sys is defined by the following metric:

$$p(sys) = w_t \cdot tp(sys) + w_q \cdot qp(sys), \tag{1}$$

where $w_t + w_q = 1$ has to be fulfilled. The factors w_t and w_q enable to weight the importance of the time quality $tp(sys)$ and the quality performance $qp(sys)$. The codomain of the metric is $[0, 1]$, where a greater value means a better performance of SO mechanism. The system sys consists of agents (resp. components) $a \in sys$ which are controlled by the SO mechanism that is analyzed in $r \in R$ evaluation runs.

3.1 Time Performance of SO Mechanisms

Evaluating the time performance $tp(sys)$ requires a clear definition of what the time performance of an SO mechanism is. For classical analysis of time performance the answer is: the duration from execution to the appearance of the output. [11] Applying that approach to SO mechanisms is not sufficient, since we have to deal with two aspects (1) SO algorithms are mostly anytime algorithms that are terminated after a certain time and (2) the performance is also depicted by the time to a next reconfiguration. This is due to the fact, that the time performance of the controlled system is effected by the SO mechanism as it might

be slowed down or stopped, that effect needs to be respected, too. Consequently, we are not directly interested in the time to a solution we are interested in its time impact, i.e., the time used for SO compared to the time where the system runs without disturbances. Due to the characteristics of SO mechanisms there is no single point where the time could be measured, because SO mechanisms solve problems in a distributed fashion. Thus, the calculation is for instance achieved by building a coalition of components that are capable to solve the problem without the rest of the system. This is the case in the production cell case study where a group of robots is able to find a new configuration if a capability, e.g., a drill, is broken. The measurement is consequently no central affair. The time of reconfiguration, i.e, the time involved in finding a new system configuration, needs to be measured for each agent $a \in sys$ as follows:

$$tp(a, r) = 1 - \frac{\sum_{s \in r} \text{reconfigurationTime}(r, a, s)}{\text{duration}(r)}, \tag{2}$$

where the time performance is measured by calculating the ratio of the time needed for reconfiguration for a single agent a in a run r consisting of s steps, given by $\text{reconfigurationTime}(r, a, s)$ in unit of time, to the duration of the run r where the measurement has taken place, supplied by $\text{duration}(r)$ in the same unit of time. This is different in two ways from evaluating classical mechanisms or algorithms, like in [11]: the measurement is a time ratio and is measured locally. The metric value of Eq. (2) is prorated with the resulting values of all agents $a \in sys$ in order to gain the time performance of the system by computing the average of all values:

$$tp(r, sys) = avg_a^{sys} tp(a, r) \tag{3}$$

The concrete average function influences the result and has to be chosen with care. The same average function is applied to compute the time performance of the system $tp(sys)$ by prorating the results of all evaluation runs $r \in R$ as follows:

$$tp(sys) = avg_r^R tp(r, sys) \tag{4}$$

The codomain of $tp(sys)$ as well as $tp(r, sys)$ and $tp(a, r)$ is $[0, 1]$, and a value close to 1 indicates a better achieved time performance.

3.2 Quality Performance of SO Mechanisms

The quality performance $qp(sys)$ determines how good the particular solutions of an SO mechanism have been. However, judging the quality of a solution is highly dependent on the particular SO mechanism as well as the system and its environment. This is due to the fact that the quality is measured according to the influence of the SO mechanism on the controlled system. The SO mechanism in the production cell case study controls which robot and which cart has to carry out which task. It consequently influences the ability of the system to produce workpieces and effects the overall output resp. throughput. The quality of the SO

mechanism is determined by the throughput, the number of processing actions, that the system is able to apply within an evaluation run. This value can be measured locally for each robot, that is processing a workpiece, and then aggregated for the entire system. The SO mechanism in the energy grid case study clusters resp. anti-clusters the power plants into virtual power plants. The optimality of this decision could be measured by the mix value of the virtual power plants. The solution quality is evaluated by measuring the domain-dependent quality function quality(r, a) for an evaluation run r of each agent $a \in sys$. As the measured value is dependent upon the context it is normalized by the value which is the best possible one:

$$qp(r, a) = \frac{\text{quality}(r, a)}{\text{quality}_{\max}(r, a)} \tag{5}$$

To prorate $qp(r, a)$ for all $a \in sys$ the average is built according to

$$qp(r, sys) = avg_a^{sys} qp(r, a) \tag{6}$$

The codomain of $qp(r, a)$ as well as $qp(r, sys)$ is $[0, 1]$, and a value close to 1 indicates a better achieved quality, since 1 would imply that the maximum performance has been reached. In order to form the proration for $qp(sys)$ different average functions might be appropriate, as described for the time performance before. Thus, $qp(sys)$ is defined as follows:

$$qp(sys) = avg_r^R qp(r, sys) \tag{7}$$

4 Performance Evaluation Framework

In order to measure and evaluate the performance of an SO mechanism it is of utmost importance to establish a test bed where the performance of the mechanism can be analyzed in a systematic, comprehensible, and representative fashion. For the implementation of an according framework we rely on the requirements derived in [6] that encompass the following two main concerns:

Req. 4: The framework's components should support the generation of evaluation runs, the simulation itself, and the application of performance metrics.

Req. 5: In order to achieve significant results, the evaluation must comprise **simulation runs** that induce an environmental behavior reflecting likely conditions under which the SO mechanisms have to operate.

We base our evaluation framework shown in Fig. 1 on the previously developed testing approaches [5,7], that have originally been used for functional testing. Basically, it consists of an `Evaluation Suite Generator`, an `Execution`, and a `Monitoring and Evaluation` component. The model-based approach is used in all fragments, which is enabled by having executable as well as changeable models. The overall evaluation framework consists of a static and a dynamic part. The static part is used for enabling automated execution as well as automated evaluation. The dynamic part is used for generating evaluation suites.

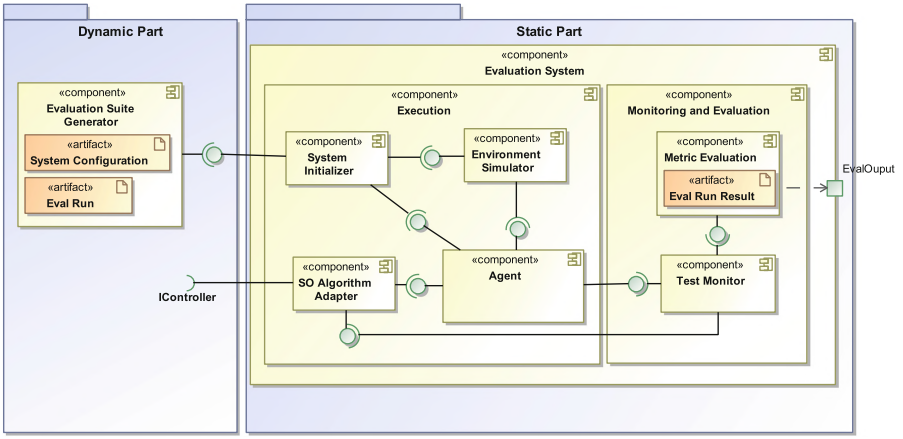

Fig. 1. The *UML* component diagram shows the essential components of the evaluation framework, consisting of two main components.

Based on a description of the `System Configuration` the `Evaluation Suite Generator` derives different possible `System Configuration`s for evaluation. The `Evaluation System` is afterwards started by initializing the static part that uses `Agent`s to set up the environment of the SO mechanism that is plugged into the evaluation framework. The `Evaluation Runs` are generated and executed in the same step (we follow an online testing approach for the evaluation of the performance that enables arbitrary length of evaluation runs). Each action of the SO mechanism is monitored and evaluated due to the implemented performance metrics. The overall evaluation is mainly driven by environmental changes, i.e., changes in the controlled environment of the investigated SO mechanism. In the production cell scenario such a change might be a faulty robot or drill that causes the SO mechanism to reconfigure the production cell. For the energy grid scenario a change can be a fluctuation in the production of a power plant due to a changing weather condition of the solar plant leading to a change in the fulfillment degree of the previously calculated schedule that in turn causes the SO mechanism to reconfigure the AVPP.

4.1 Generating Unbiased Evaluation Runs

In order to supply significant results (fulfilling *Req. 5*), the evaluation runs need to be comprehensible and representative. To illustrate this challenge, let's consider the following example in the production cell scenario: There is only one robot actively using a drill; if the environmental change is to damage the drill, that causes a reconfiguration. However, if the time performance is computed based on the time passed since the last reconfiguration, the environmental action has a direct influence on the performance result of the investigated SO mechanisms. The correlation is unintended, since the evaluation should not influence

the outcome, the system under evaluation should be the only influence factor. But, it is not possible to completely diminish this influence factor within a simulation. To get the best results there are two solutions: (1) to evaluate the system with every combination of environmental conditions or (2) to select the most representative combination of environmental conditions which represent the reality best. Due to the complexity of the environment it is not possible to follow option (1).

There is a necessity to select a representative combination of environmental conditions, i.e., take samples for all possible combination. The result from that sample is a set of evaluation runs. For these evaluation runs we compute a metric, based on Eq. (1). If this metric value is equal to the metric value for option (1) it is called *unbiased*. An unbiased metric value with a low number of samples is called *efficient* [14]. Sampling could be performed in different manners with different effects. There is the possibility to select random samples from the abstract environment states. Random sampling implies that each possible environmental change has the same probability of being chosen at any stage during the sampling process to become part of the evaluation run. However, that does not fully reflect the reality of the environment that is modeled and from which the states are sampled. This is the case for the production cell as well as the energy grid scenario: The failure rates of the production tools, on the one site, and the weather conditions for the power plants, on the other side, are nonuniform in their occurrences if real world processes are observed. There are probability distributions that describe the failure rates, that are well investigated in the field of reliability engineering [13], namely mean-time-to-failure rates as well as mean-time-to-repair rates.

4.2 Modeling the Environment for Evaluating the Performance of SO Mechanisms

Our approach uses a model representation of the environment, where we abstract from concrete states to classes of states. We use a Markov chain model describing how likely it is that a sequence of states occurs or that a state occurs after a particular state, similar to our previous approach [5] where we formed so called environmental profiles. That means there are not just the weather conditions *rainy*, *sunny*, and *cloudy*, but there is also the information whether it is possible to have *rainy* weather after *sunny* weather and how probable it is. The models are distinguished by the class of SO mechanism. For an SO mechanism with a discrete input space, we apply failure models, from the reliability theory, used for a fault injection approach (cf. [7]). The faults are injected into the controlled environment of the SO mechanism. For an SO mechanisms with a continuous input space, we apply the approach of environmental profiles. The first delivers samples as a set of faults (also an empty set is possible) that should be injected and the latter delivers a set of environmental changes. The resulting evaluation runs are used as representative combinations of environmental conditions which represent the reality best and address *Req. 5*.

4.3 Integrating the Evaluation Sequence Selection in the Evaluation Framework

In the evaluation framework (cf. Fig. 1) these models are used to feed the `Environment Simulator` by generating `Eval Runs` in the `Evaluation Suite Generator`. The approach is building upon a model-based testing approach, as described in [5,7], allowing for describing the different parts of the SO mechanism and its environment in order to generate evaluation sequences. The modeling language used at hands is S# [8], an executable modeling language, that is equipped with the ability to execute the model. The execution has direct influence on the `Agents` controlled by the mechanism under evaluation. This online evaluation run generating process allows for endless execution runs. To select the number of all evaluation steps in all runs to be executed, we use the following formula that allows selection of the length by defining the acceptable estimation error $\Delta\mu$ [14]:

$$n \geq z^2 \frac{\sigma^2}{(\Delta\mu)^2}, \tag{8}$$

where n is the number of all evaluation steps in all runs, z is the standard normal distribution (SND) value (taken from an SND table) of the expected distribution, and σ^2 is the to be estimated standard deviation (SD) by taking n evaluation runs. For our evaluation the acceptable estimation error might be $\pm 1\%$ with an confidence interval of 95%. The confidence interval states the probability that the expected metric value $p(sys)$ is within a given symmetric interval of $[p(sys)-b; p(sys)+b]$ where b is called the confidence border. The actual selection of the value $b := z\sigma_{p(sys)} \equiv z\frac{\sigma}{\sqrt{n}}$ is defined by the expected SD and z as the corresponding value of the SND that is expected. Indeed, the number n is the actual number we would like to know and it could be determined by Eq. (8) given the confidence interval and the acceptable estimation error. However, the value σ^2 is unknown, since it is the also unknown SD to be estimated by the evaluation runs. Thus, we have to use a rather gross estimate of σ^2. In order to play it safe, σ^2 should be set to 0.25, leading to a rather to big estimate for the value n, since this the maximum value to be selected [14]. So we select $\sigma^2 = 0.25$ and have the following equation to be solved having given $z = 1.96$ by taking the value form the SND table and the confidence interval of 0.95: $n \geq z^2 \frac{\sigma^2}{(\Delta\mu)^2} = 1.96^2 \frac{0.25}{0.01^2} = 9604$.

5 Evaluation

We used two different self-organizing systems for evaluating the accomplishments of our metric as described in Sect. 3 and the evaluation framework specified in Sect. 4. Within this evaluation we answer the following research questions:

RQ 1: Is it possible to determine differences in the SO mechanisms' performance during the assessment of the performance of the entire CAS?

RQ 2: SO mechanisms have great impact on the overall performance of the controlled system. Is it possible to quantify this impact reasonably for SO mechanisms?

RQ 3: SO systems are faced with an ever-changing environment; their performance depends on the current run time setting of the system. Are these dynamic performance boundaries reflected in the performance evaluation in a way to provide a comprehensive analysis?

RQ 4: The simulation environment might influence the outcome. Is the evaluation framework able to establish conditions for continuous and discrete SO mechanisms that are able to produce comparable results in different settings?

These research questions are derived from a set of requirements for metrics and performance evaluation of SO mechanisms, that we developed in [6]. We selected the case studies as they represent the two different input spaces of SO mechanisms, described in Sect. 2. Thus, we are able to demonstrate the two different possible instance of the evaluation framework. In order to investigate the metric in depth for each case study we used several different SO mechanisms.

5.1 Local and Central Reconfiguration in an SO Production Cell

In the production cell case study, we compare a central SO mechanism working with global knowledge with a coalition-formation mechanism with local knowledge only. The centralized mechanism always stops the entire system when a configuration deficiency is detected. It removes the current configuration entirely, computes a new configuration and distributes it. The localized mechanism on the other hand forms a coalition of agents, starting with the agent that detected the problem. It recruits more and more neighbouring agents, until the agents in the coalition are able to solve the problem at hand among themselves. Only the configurations of those agents within the coalition that must necessarily change their roles is updated. Both mechanisms employ the same algorithm to find a solution within their set of available agents. They differ in the selection of those agents as well as in the method of distribution for the computed solution.

Evaluation Setting. We evaluate both mechanisms within three different setups of the production cell case study: firstly, a setup with few agents (6 robots, 4 carts), and high redundancy with regards to available capabilities (each robot has ≈66.7% of the existing capabilities), we refer to this setup as *FA/HR* below; secondly, a setup with more agents (10 robots, 4 carts), and low redundancy (40%), referred to as *MA/LR*; and lastly one with more agents (10 robots, 4 carts), and high redundancy (70%), *MA/HR*. Each model is simulated with both algorithms in several simulation runs. Within each run, environment faults are activated and deactivated randomly according to their respective MTTF and MTTR. For greater comparability, we simulate the systems with the same random seeds, i.e, expose both algorithms to the same environmental conditions.

The numbers of the evaluation are shown in Table 1. As the shown data is aggregated over 100 runs with 1000 steps each, the shown number are showing

the average value. As outlined in Sect. 4, the concrete choose of the average function is up to the test engineer. One way of choosing is to get more insights in the data. The data is, for this purpose, first tested on normal distribution. The Shapiro-Wilk test, performed on the data, showed p-value < 0.05 which is indicating a nonparametric data set for the measurements. For nonparametric data, it is hard to select a good fitting average functions. For the given data set the arithmetic mean was evaluated as well as the median. The arithmetic mean is selected in the case when additions of the values is meaning full, while the median is more focused on clustering the data in two equal sized data heaps. That makes the arithmetic more prone to outliers compared to the mean value. Thus, the expectation would be, that the values differ, as we have no normal distribution. However, both values are almost equal for all the data. Leading to the assumption that the data is tightly clustered, which is the case. The data as we process it in the metrics is well suited by addition as an average function. Thus, the following data plots the arithmetic average value as the chosen average function. Indeed, it is of further interest to investigate the dispersion of the data. This gives more insights on the state of the data given. A measures for the dispersion is the range of data given. However, for the arithmetic mean the standard deviation (SD) is the average of choice. Indeed, the SD is prone to a normal distribution. However, the data is tightly clustered around the mean, thus, we opt for the SD in the statistics, delivering the best insight on the dispersion of the given data set. For the further results the different results of the different configurations and the different settings have being tested according to the independence of the results. Due to the fact that no distribution is assumed at the given data set the Mann-Whitney-Wilcoxon test has been performed to show the independency. The Mann-Whitney-Wilcoxon test resulted in a p-value < 0.05. Thus, the following discussion of the value is grounded on the given independency.

Discussion of the Evaluation Results

RQ 1: The results of our evaluation as shown in Table 1 clearly indicate that the centralized mechanism yields greater quality, i.e., allows for greater throughput. This is surprising, as one would expect the locality of the coalition-formation mechanism to yield better results, since it allows some parts of the system to be reconfigured while other parts keep working. However, this effect was not pronounced enough in our case study to overcome the negative aspects: the coalition has only a subset of the centralized mechanism's knowledge, and it will always prefer a localized solution, leading to little division of labor. On the other hand, we can see that the coalition-formation mechanism has better time performance, i.e., the relation between working time and reconfiguration time for individual agents is better. This shows the benefit of not involving every agent in every reconfiguration. The coalition-formation is much more efficient in this respect, for the smallest model almost twice as efficient as the centralized mechanism. As a result, agents can perform more production steps between two subsequent reconfigurations. In reality the effect on performance would be even more pronounced because physically stopping production costs more time than

Table 1. Evaluation results for the two SO mechanisms "Centralized", and "Coalition-Formation" with different production cell setups. All values are averages over evaluation runs with 100 steps each; values in parenthesis denote SDs.

Mechanism	Centralized			Coalition-Formation		
Model	FA/HR	MA/LR	MA/HR	FA/HR	MA/LR	MA/HR
$tp(sys)$	0.9728 (3e − 3)	0.8579 (4e − 4)	0.8596 (1e − 2)	0.9963 (6e − 4)	1.0000 (3e − 5)	1.0000 (2e − 6)
$qp(sys)$	0.5909 (0.03)	0.9432 (0.06)	0.7022 (0.04)	0.4987 (0.03)	0.7560 (0.04)	0.5779 (0.04)
$p(sys)$ $w_t = 0.5, w_q = 0.5$	0.7819	0.9005	0.7809	0.7475	0.8780	0.7889
$p(sys)$ $w_t = 0.1, w_q = 0.9$	0.6291	0.9347	0.7180	0.5484	0.7804	0.6200
$p(sys)$ $w_t = 0.9, w_q = 0.1$	0.9347	0.8664	0.8439	0.9466	0.9756	0.9577
#Modified Roles per Reconf.	7.85 (2.65)	16.67 (6.67)	13.73 (5.06)	8.41 (3.83)	11.74 (11.17)	10.71 (7.51)
#Reconf. / #Involved Agents per Reconf.	0.45 (0.12)	0.52 (0.13)	0.46 (0.12)	0.88 (0.16)	0.74 (0.19)	0.76 (0.20)
#Steps between Agent Reconf.	6.87 (6.91)	1.99 (2.41)	3.10 (3.41)	12.84 (21.53)	3.83 (5.10)	6.04 (9.49)

it does in our simulation. Similarly, changing role allocations would correspond to physical tool changes, also requiring great amounts of time. For larger models, the coalition mechanism outperforms the centralized mechanism in this respect as well. We measured the time performance locally for each agent, compute a performance for the complete evaluation run and approximate results for the entire system through a series of runs. The quality is measured similar as the number of processing steps applied by agents within a run, this information is gathered locally for each agent and then aggregated. Hence, to answer *RQ 1*, two different mechanisms are comparable despite their different views on the overall system (local and central). The metrics deliver a clear indication of the advantages of the different mechanisms.

RQ 2: The solution quality can be weighted higher or lower in order to consider the SO mechanism's influence on the system performance. This enables us to control the influence of the quality parameter with the time parameter. The quality is best measured in the MA/LR setting, that indicates, that the quality is depended from the actual setting as well. Nevertheless, it is still possible to quantify a recent difference between the quality of the centralized and the coalition setting. That undermines an effect, that is assumed for decentralized

mechanisms: it is often stuck to local optima. This effect is reasonably quantified here.

RQ 3: Our metric can also account for the dynamic performance boundaries of SO (*RQ 3*). Remember that the quality ratings equal the system's actual productivity, measured by production actions, e.g., drill, compared to the maximal possible productivity. For the maximal possible quality as referenced in Eq. (6), we executed a system run for each setup without any environment faults. For the FA/HR setting of 484 production actions was achieved, 336 production actions for the MA/LR setting, and 343 production actions for the MA/HR setting. One step can encompass at most one production action, hence this value abstracts from the concrete time required. Thus, a larger system with low redundancies is prone to be less productive due to long transits. Here, the same initial system configuration is used for calculating the maximum quality as well as for starting the evaluation. That initial configuration encompasses, amongst other things, the initial role allocation (i.e., which robot and which cart is applying which capability) of the system. However, when the system encounters a faulty environment, the maximal throughput in ideal conditions may in many cases be unreachable even with the best SO mechanism, which explains the relatively low scores for both algorithms.

Further, high time performance ratings can be achieved by both algorithms: even though a faulty environment leads to more time spent on reconfigurations, it influences the total simulation time in the same manner, thus limiting its influence on the quotient. By assigning a lower weight to the quality ratings, we can account for this imbalance to some extent. Similarly, there exists a certain disconnect between the time required for processing steps in our simulations and in reality, the former being much lower. This disconnect does not exist for reconfiguration times, and therefore it affects the time performance as defined in Eq. (2). However, it does so equally for both compared mechanisms.

Hence, while the absolute values in the simulation differ from the realization in a real hardware application scenario the metric has to be taken with a grain of salt for the quantitative comparison, the relation between the two mechanisms remains still the same and thus still allows for a fair qualitative comparison. In order to give a more complete picture, we also included the number of discrete steps the system makes between two reconfigurations involving the same agent, on average.

RQ 4: Lastly, we consider the evaluation system's influence on our results in *RQ 4*. The SD for time performance and quality in our evaluation results is low, less than 0.01 for time performance, and less than 0.06 for the quality rating. We can thus assume the results are unbiased. The high SDs for per-reconfiguration results (the last three rows) is expected: the amount of necessary reconfiguration changes and the frequency of reconfigurations depend on the respectively occurring environment faults, whose frequency and impact vary greatly.

5.2 Self-Organized Creation of Virtual Power Plants in Smart Grids

For the evaluation within the self-organized creation of virtual power plants in a smart grid we used an SO mechanism called *PSOPP* [2] (*P*article *S*warm *O*ptimizer for the *P*artitioning *P*roblem). The PSOPP is a particle swarm optimizer that partitions a set of agents representing a (sub)system into pairwise disjoint and non-empty groups. These groups constitute the (sub)system's configurational structure. Feasible organizational structures can be described by so-called partitioning constraints that restrict the number and the size of these groups. PSOPP is an anytime algorithm and a metaheuristic that optimizes the groups' composition with respect to an objective function. In our evaluation, PSOPP is used to optimize the groups' composition in each so-called separate AVPPs of a hierarchically structured system.

Evaluation Setting. We executed 100 generated evaluation runs, each comprising 300 evaluation steps leading to a size of the evaluation run that is bigger than the smallest useable size calculated using Eq. (8). In order to investigate two different SO mechanisms we instantiated the PSOPP algorithm with two different settings: (1) In the setting *PSOPP HP*, PSOPP established partitionings according to an homogeneous partitioning objective function defined in [2]. (2) In the setting *PSOPP k-means*, PSOPP established heterogeneous partitionings according to the well-known k- means objective function. All evaluation runs have been performed in a distributed cluster of 12 computers with an Intel Core-i5 CPU and 4GB RAM for about a week. We performed each setting on a predefined system structure consisting of 1, 2, and 5 separate subsystems and 1000 controlled power plants within the system that are clustered to AVPPs by the SO mechanism. Each subsystem has one instance of the investigated SO mechanism (PSOPP HP or PSOPP k-means).

The results of our evaluation are summarized in Table 2. Having a closer look at Table 2 the performance metric $p(sys)$ (cf. Eq. (1)) is shown for all instances with three different configurations according to their weights, a balanced weighting, a favor for quality, and a favor for time. As we computed the data from 100 different evaluation runs by using the arithmetic mean value with the according SD. The selection of the average function here followed the same approach as described in the setting of the production cell. The investigated data are also not normal distributed, but tightly clustered, making the mean value a good fit for describing the data. The normal distribution was tested and the p-value resulted in <0.05. Further, the independency between the values to compare was test by the Mann-Whitney-Wilcoxon test with a resulting p-value < 0.05. Overall we have observed very slight variants of the performance over the runs and observed no big outliers.

Discussion of the Evaluation Results

RQ 1: A first observation is that the decrease of agents involved in the reconfiguration has a rather low impact on the $tp(sys)$ value in both types of SO

Table 2. Evaluation results for the two settings "PSOPP HP" and "PSOPP k-means" with different numbers of AVPPs. All values are averages over evaluation runs with 300 steps; values in parenthesis denote SD.

Setting	PSOPP HP			PSOPP k-means		
#Separate Subsystems	1	2	5	1	2	5
$tp(sys)$	0.87 (0.11)	0.86 (0.27)	0.90 (0.02)	0.02 (0.002)	0.02 (0.004)	0.02 (0.007)
$qp(sys)$	0.96 (0.02)	0.96 (0.01)	0.96 (0.01)	0.99 (0.01)	0.99 (0.01)	0.99 (0.01)
$p(sys)$ $w_t = 0.5, w_q = 0.5$	0.92 (0.07)	0.91 (0.15)	0.94 (0.02)	0.51 (0.03)	0.51 (0.05)	0.51 (0.09)
$p(sys)$ $w_t = 0.1, w_q = 0.9$	0.95 (0.04)	0.95 (0.05)	0.95 (0.01)	0.89 (0.03)	0.89 (0.06)	0.89 (0.10)
$p(sys)$ $w_t = 0.9, w_q = 0.1$	0.88 (0.04)	0.87 (0.19)	0.91 (0.02)	0.12 (0.02)	0.12 (0.03)	0.12 (0.06)
#Reorganized Separate Subsystems	1.05 (0.32)	2.88 (0.88)	11.01 (1.98)	141.57 (17.93)	244.11 (43.11)	501.36 (113.11)
#Reconfigured Agents per Reconf.	1000.00 (0.00)	696.78 (400.14)	252.47 (288.34)	1000.00 (0.00)	734.74 (335.52)	499.51 (284.18)

mechanisms. That effect is also shown in the number of reorganizations performed in the different setting compared with the involved number of agents in a reorganizations. The more separate subsystems the less agents are on average involved in a reorganization, but also the more reorganizations are necessary for keeping up the goals of the SO mechanism. This seems to be an effect of the decentralized knowledge that is lower than the central knowledge and thus leads to a higher need for reconfigurations. This effect is reflected in the metric, by having almost the same value despite a changing subsystem size. These local effects are handled in the metric. Same for $qp(sys)$, all values have been gathered locally. The value for the quality function for PSOPP HP setting is stating how similar the AVPPs are in their composition. Thus, the goal it the minimization of the SD of the average state values of the power plants in each AVPPs. For the k-means setting the similarity of the average state is the measure of quality for each AVPP. For $qp(sys)$, in the HP and the k-means setting, a similar effects are shown as for the time performance: the increasing number of separate subsystems has no impact on the quality of the system. Measuring the performance locally is consequently able to judge over the global system without neglecting the structure of the system and the SO mechanisms.

RQ 2: Having $p(sys)$ for the two different SO mechanism in the scenario we can clearly observe that the homogeneous partitioning is in favor. That reflects the fact that homogeneous partitioning is more robust than k-means, as described by Anders et al. [2]. However, the robustness has a slight price in quality, that is overall more optimal with k-means (see the $qp(sys)$ values). Nevertheless, that comes with a high price of a very poor $tp(sys)$ result. To achieve a better rating for PSOPP k-means a possible allocation of the weights is $w_t = 0.03$, $w_q = 0.97$. However, it is not recommended to choose such a strong favor for one part of $p(sys)$ since it ignores one of the two important performance factors. Thus, to answer *RQ 2* the benefit can be considered and even more the influence can be steered individually.

RQ 3: The answer to *RQ 3* is shown in the fact that we observed fluctuations throughout the evaluation runs within the maximum. That is different from the production cell case study, where the maximum for the quality performance was computed for a run not for a step. In this case study the maximum value is dynamically calculated at each step for a single subsystem. Since the value is depended from the current state of the controlled power plants in a subsystem. That is highly necessary to normalize the different achievements in the different system steps.

RQ 4: RQ 4 questions whether the results are adequate regarding the conditions under which they were measured. This question is hard to answer with the resulting data, since we have no gold standard to compare with. However, our argumentation of Eq. (6) indicates that we have an accuracy of at least 0.95 for the measurements. The inaccuracy of 0.05 is within the variation of the $p(sys)$ value according to the SD and consequently negligible. Thus, we have established an adequate evaluation framework.

6 Related Work

In [6], we provided an overview on performance metrics for SO mechanisms and evaluated their abilities. We identified several metrics for adaptation (resp. self-adaptation) algorithms in the literature. And only very few that are focused on SO as we described it in [6]. As is the case with classical algorithms, the metrics can be clustered into *time-oriented metrics* and *solution-quality-oriented metrics*. The research survey of Villegas et al. [20] as well as the criteria for the evaluation of self-* systems of Kaddoum et al. [9] are *time-oriented metrics* that reflect the relationship between time for adaptation and working time. The performance metrics of Becker et al. [3], Tarnu and Tiemann [19], Reinecke et al. [17], and Kantert et al. [10] address the *solution quality* of the algorithm.

Time-Oriented Metrics: The metrics *WAT* [9], *A* [20], and *U* [20] rely on the ratio between working time and adaptivity time resp. the mean time to fail and the mean time to recover. All three focus on the impact of the adaptation on the working system and reflect the stability as well as the robustness of the configurations established by the SO mechanism. Unfortunately, the locality of

SO mechanisms is neglected by the three metrics. Thus, a reconfiguration in a small part of the system is rated as an adaptation period of the entire system as is for a reconfiguration within a huge part of the system. In our approach we address this issue by measuring locally. Considering only the time-oriented metrics, it is possible that an SO mechanism, that causes the system to work inefficiently is rated very good in terms of time if it generates a robust structure. Such a metric is not sufficient to rate the performance of an SO mechanism with all its responsibilities. Therefore, we combined time-oriented metrics with solution-quality-oriented metrics to rate the overall performance of an SO mechanism.

Solution-Quality-Oriented Metrics: To rate the performance of an SO mechanism, the optimality of its solution plays a crucial role. The metrics proposed in [3,19] are quite similar in how they measure the normalized fitness of the SO mechanism over time. Challenges that arise during the evaluation of SO mechanisms with the metrics defined in [3,19] are mainly caused by the locality of the SO mechanisms. This is a major difference to the adaptation algorithm considered in [3] as well as in [19] who regard a central approach of only one adaptation algorithm within the entire system. In case of multiple subsystems, as is the case with our energy grid scenario, the metrics could be applied to the separate subsystems, but it is not obvious how to calculate the performance for the overall system. The *Ad* metric [17] intends to smooth the development of the fitness value. Alas, the metric shows some bad side effects, as elaborated in [6]. Thus, it is hard to use the value for performance evaluation. Quantifying how robust an achieved solution of an SO mechanism is is the focus of [10]. That is measured by how fast the mechanisms are able to recover from disturbances and attacks from outside.

This metric is describing the quality performance by the time for regaining a target output of the system. This approach is, in contrast to our approach, focused on robustness against external attacks. A similar approach was made by Pitt et al. [16], focusing on the aspects of procedural justice as a values for participation, transparency and balancing in an CAS. Compared to the here presented approach [16] is focused on openness and transparency, we are a more generic approach. Indeed, one possibility is to include the metrics of [16] as the quality measure (Eq. (7)) in our approach.

7 Conclusion

We provide a tool set for measuring and evaluating the performance of SO mechanisms. This tool set is able to support the engineering process of developing suitable SO mechanisms for collective adaptive systems (CAS). As we have shown in our evaluation (Sect. 5), different kinds of SO mechanisms for different kinds of systems can be easily compared and rated, enabling to choose and optimize a suitable solution. One important contribution, that is extending the state of the art, is that our metric is able to fully exploit the local behavior of distributed SO mechanisms for time performance as well as quality performance. Reflecting

ever-changing environmental conditions was important for the evaluation framework to select the most representative evaluation runs as well as for rating the achieved solution of the SO mechanism.

There are still some limitations to our approach. The metrics have the advantage of aggregating the performance of the SO mechanisms, but that aggregation is sometimes hiding information about the cause of the resulting performance. Furthermore, the quality performance must be defined by the user of the metric, we are not able to offer a gold standard for SO mechanisms here. Consequently, the results have to be judged by a skilled SO engineer and need some qualified input. Nevertheless, the framework and the development metrics have proven as a valuable tool for assessing the performance of SO mechanisms within CAS.

Acknowledgment. This research is sponsored by the research project *Testing self-organizing, adaptive Systems (TeSOS)* of the *German Research Foundation.*

References

1. Anders, G., Seebach, H., Nafz, F., Steghöfer, J.P., Reif, W.: Decentralized reconfiguration for self-organizing resource-flow systems based on local knowledge. In: Proceedings of the 8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASe 2011), pp. 20–31. IEEE (2011)
2. Anders, G., Siefert, F., Reif, W.: A particle swarm optimizer for solving the set partitioning problem in the presence of partitioning constraints. In: Proceedings of the 7th International Conference on Agents & AI (ICAART) (2015)
3. Becker, M., Luckey, M., Becker, S.: Performance analysis of self-adaptive systems for requirements validation at design-time. In: 9th ACM SIGSOFT International Conference on Quality of Software Architectures (QoSA 2013). ACM (2013)
4. Belzner, L., Hölzl, M., Koch, N., Wirsing, M.: Collective autonomic systems: towards engineering principles and their foundations. In: Steffen, B. (ed.) Transactions on Foundations for Mastering Change I. LNCS, vol. 9960, pp. 180–200. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46508-1_10
5. Eberhardinger, B., Anders, G., Seebach, H., Siefert, F., Knapp, A., Reif, W.: An approach for isolated testing of self-organization algorithms. In: de Lemos, R., Garlan, D., Ghezzi, C., Giese, H. (eds.) Software Engineering for Self-Adaptive Systems III. Assurances. LNCS, vol. 9640, pp. 188–222. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-74183-3_7
6. Eberhardinger, B., Anders, G., Seebach, H., Siefert, F., Reif, W.: A research overview and evaluation of performance metrics for self-organization algorithms. In: Proceedings of the 9th International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 122–127. IEEE (2015)
7. Eberhardinger, B., Habermaier, A., Seebach, H., Reif, W.: Back-to-back testing of self-organization mechanisms. In: Wotawa, F., Nica, M., Kushik, N. (eds.) ICTSS 2016. LNCS, vol. 9976, pp. 18–35. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47443-4_2
8. Habermaier, A., Eberhardinger, B., Seebach, H., Leupolz, J., Reif, W.: Runtime model-based safety analysis of self-organizing systems with S#. In: 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 128–133. IEEE (2015)

9. Kaddoum, E., Raibulet, C., Georgé, J., Picard, G., Gleizes, M.P.: Criteria for the evaluation of self-* systems. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, pp. 29–38 (2010)
10. Kantert, J., Tomforde, S., Müller-Schloer, C., Edenhofer, S., Sick, B.: Quantitative robustness - a generalised approach to compare the impact of disturbances in self-organising systems. In: Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, pp. 39–50 (2017)
11. McGeoch, C.: A Guide to Experimental Algorithmics. Cambridge University Press, Cambridge (2012)
12. Monostori, L.: Cyber-physical production systems: roots, expectations and R&D challenges. Procedia CIRP **17**, 9–13 (2014)
13. Musa, J.D.: A theory of software reliability and its application. IEEE Trans. Softw. Eng. **1**(3), 312–327 (1975)
14. Neyman, J.: Outline of a theory of statistical estimation based on the classical theory of probability. Phil. Trans. R. Soc. Lond. A **236**(767), 333–380 (1937)
15. Parunak, H.V.D., Brueckner, S.A.: Software engineering for self-organizing systems. In: Proceedings of the 12th International Workshops on Agent-Oriented Software Engineering (AOSE 2011), pp. 1–22 (2011)
16. Pitt, J., Busquets, D., Riveret, R.: Procedural justice and 'Fitness for Purpose' of self-organising electronic institutions. In: Boella, G., Elkind, E., Savarimuthu, B.T.R., Dignum, F., Purvis, M.K. (eds.) PRIMA 2013. LNCS (LNAI), vol. 8291, pp. 260–275. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-44927-7_18
17. Reinecke, P., Wolter, K., Van Moorsel, A.: Evaluating the adaptivity of computing systems. Perform. Eval. **67**(8), 676–693 (2010)
18. Steghöfer, J.P., Anders, G., Siefert, F., Reif, W.: A system of systems approach to the evolutionary transformation of power management systems. In: Proceedings of INFORMATIK - Workshops on Smart Grids. LNI. Köllen Verlag (2013)
19. Taranu, S., Tiemann, J.: On assessing self-adaptive systems. In: Proceedings of the 8th International Conference on Pervasive Computing and Communications Workshops, pp. 214–219. IEEE (2010)
20. Villegas, N.M., Müller, H.A., Tamura, G., Duchien, L., Casallas, R.: A framework for evaluating quality-driven self-adaptive software systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 80–89. ACM (2011)