Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 1261–1268

ISBN 978-83-60810-51-4

Scalability and Robustness Analysis of a Multi–Agent based Self–healing Resource–flow System

Thomas Preisler and Wolfgang Renz
Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik, Labor für Multimediale Systeme
Berliner Tor 7, 20099 Hamburg, Germany
{thomas.preisler, wolfgang.renz}@haw-hamburg.de

***Abstract*—In resource–flow systems e.g. production lines resources are processed by agents applying certain capabilities to them. Such systems profit from self–organization like self–healing as they become more robust against failures. In this paper the development of a decentralized coordination process for such a system is described. The system is realized as a multi–agent system for the purpose of simulating large systems. Furthermore, a stochastic model is developed and compared to the simulation results. The scalability and robustness of the proposed coordination process is shown in good agreement of simulation results and analytic results for the stochastic model.**

I. Introduction

In the last decade, self–healing systems have attracted large attention, since they can adapt to failures. As a particular class of such systems *self–organizing resource–flow systems* have been identified to be tractable as well as relevant to applications like web service orchestration or transport logistic and production logistic scenarios where the transportation of resources is carried out by autonomous guided vehicles (AGVs) [1]. As a simple example, Fig. 1 shows such a resource–flow system where robots process car bodies which are transported between them by AGVs. The first Robot (A) welds the body, the second Robot (B) inserts electronic parts, the third Robot (C) insert the motor, the fourth Robot (D) attaches the exhaust and the last Robot (E) attaches the wheels, while the processed car is transported from one robot to an other by the AGVs. The figure also shows that the robots have multiple capabilities which they can apply, while currently only applying the highlighted one. The self–healing mechanism relies on the redundant availability of capabilities owned by the robots, which allows to reconfigure the system following the organic design pattern (ODP) [2]. Recently we have shown that such automated self–healing reconfiguration processes can be realized by a decentralized coordination process, i.e. through self–organization at runtime [3]. In this paper we introduce a modified, more realistic self–organizing resource–flow system that also includes resource buffers at each robot and with a consequently refined self–healing reconfiguration process that includes workload and show scalability and robustness of the reconfiguration process.

The following Coordination Development Process is based on work in the research project "Self–organization based on decentralized Coordination in Distributed Systems" (SodekoVS) [4]. Where a programming technique was developed that allows to equip software systems with self–organizing features based on the conceptional separation of agent functionality and coordination. The systematic integration of decentralized coordination strategies in Multi–Agent Systems (MAS) has been discussed in [5]. The appropriate coordination is approached by modeling the problematic, unintended behavior of applications. Based on the identification of the *Problem Dynamic* a corresponding *Solution Dynamic* is derived that supplements the application behavior with additional interdependencies and inter–element feedbacks to correct the system's behavior and alleviate unintended effects [3].
Two important aspects when developing coordination strategies for self–healing systems that have to be considered are scalability and robustness. The developed solution has to be robust in the meaning that it should be able to restore the system's functionality after a failure. Also the solution should scale well. The resource requirements for the reconfiguration process should increase linear according to an increasing system size. To analyze the scalability and robustness of the coordination process the resource–flow system was realized as a multi–agent simulation. Related Work is presented in the next section. The development of the system and the coordination process is described in section III. In section IV a stochastic model for the coordination is described and in section V this model and the simulation results are used for the scalability and robustness analysis. A conclusion is drawn in section VI.

II. Related Work

A general overview about multi–agent based simulation (MABS) is given in [6]. Interesting properties of MABS are pointed out and compared to other approaches like traditional discrete event simulation, object–orientated simulation and dynamic micro simulation. It is argued that MABS is a useful technique for simulation scenarios that are distributed

978-83-60810-51-4/$25.00 © 2012 IEEE

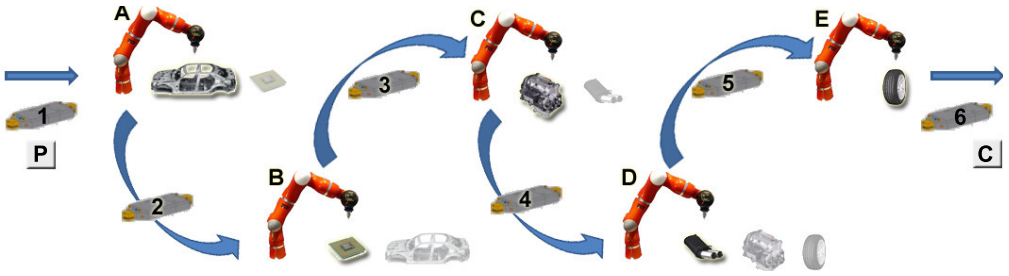


Fig. 1. Robots with different capabilities (icons to the right of the robot) process a car by applying one of the capabilities each (highlighted icon) [3].

in a technical way and involve complex interactions between humans and machines. Related work in the field of multi–agent based simulation of self–organizing systems is presented in [7]. There, a multi–environment simulation framework for building self–organizing multi–agents systems is introduced. The proposed framework provides higher abstractions and components to support the development of self–organizing systems with multiple environments. Furthermore the framework provides a coordination component and self–organizing mechanisms to be instantiated and also flexibility to evolve the framework with more complex ones. In [8] an agent–based simulation tool for decision making about automatic warehouses management is presented. The there proposed MAS should be used in a real environment and therefore a simulation framework was developed in order to study problems, constraints and issues of truck unload operations. Similar to the work presented in this paper is the use of AGVs in the application domain, but with the focus of optimizing the suitable number of AGVs used to unload containers arrived to a warehouse. Other work regarding the use of AGVs in production lines or related environments is presented in [9]. Where an agent–based simulator for evaluating the use of AGVs in the handling of containers within terminals was developed. In [10] a decentralized control system for AGVs based on a situated MAS is presented. Their work mentions the advantages and disadvantages of centralized and decentralized control systems. Based on real world AGVs for automated logistics services in warehouses and manufacturing which originally were controlled by a centralized system, a decentralized control system was modeled, implemented and tested on the same hardware that was use by the centralized system. While this paper does not focus on the realization of the AGVs and considers them as black box systems, the work of [10] focuses on the actual realization of these vehicles and the therefore encountered problems. An other agent–based approach to control flexible resource–flow systems is given in [11]. In contrast to the approach presented in this publication the there presented approach tries to solve a capacity bottleneck in a resource–flow system with high volume production. It is based on an auctioning algorithm where workpiece agents auction off their current task, while machine agents bid on them. The special feature of this approach is that a capacity bottleneck is automatically propagated in opposite direction of the resource–flow because workpiece agents not only take into account the machine's current workload when awarding a machine but also the workpieces which are leaving the machine. Therefore if the stream of workpieces leaving the machine is blocked for some reason, eventually the machine agent ceases to accept new workpieces and blocks it's input stream as well. Because the workpieces are transported between the machines via conveyor belts, the input stream of one machine is influenced by the output stream of a previous machine. So the capacity bottleneck is automatically propagated in opposite direction of the resource–flow. [12] focuses on agent–based modeling and simulation. Challenges of validation and verification of agent–based simulation models are identified and therefore a developed generic testing framework for agent–based simulation models to conduct validation and verification of models is presented.

III. Development of a Self–healing Resource–flow System

Before the reconfiguration process can be explained, the role–concept of this system has to be introduced: The information which capability a robot has to apply on a received resource is stored in its *role*. The role also contains information from which AGV the robot should receive a resource, to which AGV it should give it after processing it and the state the resource should have before and after the processing.

The reconfiguration process is triggered when due to a random error one or more capabilities in one or more robots fails so that the affected robots can no longer apply their roles to the resources. In this case if the system would not be able to heal itself the production of resources would stop. But assuming that each robot is capable to exhibit a set of capabilities a correct resource flow can be re–established by a swapping of roles. Defect robots adopt suitable roles from other robots and in return the other robots help out by adopting the affected roles. The defect robot emits a wave of reallocations by sending requests for help along a predefined communication graph designed as a token ring. Each receiving robot has to decide locally if it is capable to swap roles. A swap of roles is called direct if one swap of roles is enough to get the system back into a working state. Such a swap is characterized by the fact that the receiving robot can apply all the roles from the deficient robot, meaning that it owns the required capabilities and also the deficient robot can apply all the roles the other robot has to swap with it in order to reconfigure. But not in

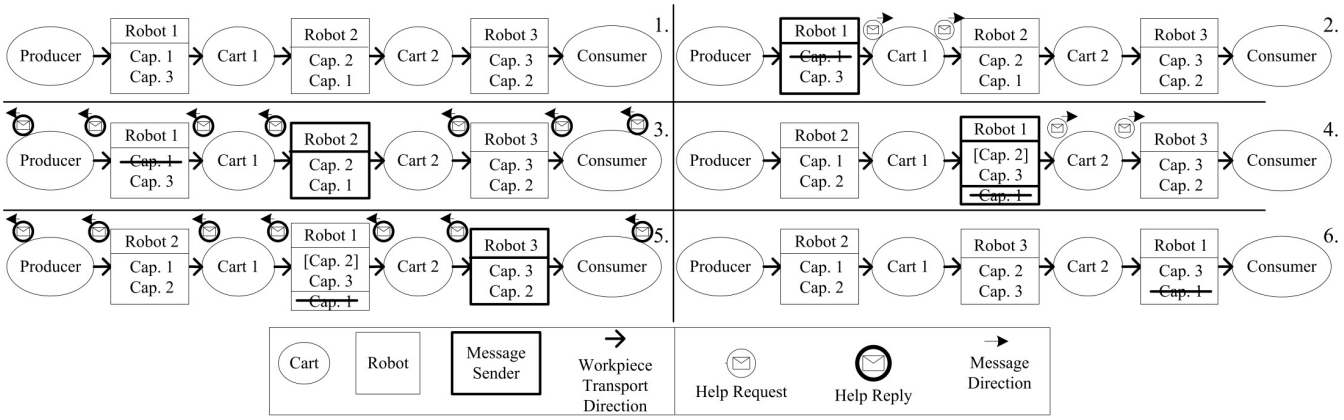


Fig. 2. Exemplification of the decentralized reconfiguration.

all cases is such a direct swap of roles possible to get the system back into a working state. In some cases the deficient robot is not able to apply the roles which the other robot has to swap with it. In this case a so called *transitively* swap is performed. The robot which receives a help request still gives up its roles and adopts the roles from the deficient robot, even if the deficient robot can not apply them. The deficient robot then stays in a deficient state but now tries to find an other robot with which it can swap its new but still deficient roles. The number of roles a robot is capable to exhibit depends on the so called *redundancy rate*. The redundancy rate provides information about how often a certain capability is represented in the system e.g. a redundancy rate of 10% means that a capability is present in 1 out of 10 robots. The redundancy rate for a system is the mean value of all the redundancy rates of all the capabilities present in the system. A higher redundancy rate means a higher chance of a successful reconfiguration in case of an error.

The operating principle is exemplified in Fig. 2. Three robots are connected in series (1). The set of applicable capabilities for each robot are listed and the topmost capability is used in the currently active role. *Robot 1* is currently configured to apply a role that involves *Cap. 1* but also can apply *Cap. 3*. The *Producer* and *Cart* carts represent the initial provision and final collection of the processed resources. Due to an error *Robot 1* can no longer apply *Cap 1* and sends a help request (2). The request is routed along the communication graph, which in this example equals the resource–flow with the exception that there is an extra communication link between the *Producer* and *Consumer* which is not shown in the figure, until it reaches a robot which is able to execute the needed capability (*Robot 2*) (3). The reply is routed to the requesting robot as well as all carts that are connected to the swapping robots. The robots and carts reconfigure their local roles and the resource flow is re–established by adjusting the ports in the pre- and postconditions in the affected roles. In (3) *Robot 2* is able to provide capability *Cap. 1* but *Robot 1* is not able to replace *Robot 2* as it does not have the required capability *Cap. 2*. So after the swap *Robot 1* remains in a problematic state and requests assistance (4). This request is propagated again until it reaches a robot with the required capability (*Robot 3*) and the swap proceeds as mentioned before (5). Since *Robot 1* is able to replace the currently active capability of *Robot 3* (*Cap. 3*) the resource–flow is finally re–established (6). This example is simplified for easier understanding. In principle robots can be part of several resource–flows. In this case robots only reconfigure for broken roles and continue applying roles that are not broken.

The here proposed decentralized reconfiguration process is the *Solution Dynamic* for this self-healing system. The *Problem* and *Solution Dynamic* for this ODP-based resource-flow system can also be described with an *Agent Causal Behavior Graph* (ACBG) [13], illustrated in Fig 3. Robots are initially *running*. Random errors can affect one or more roles in one ore more robot. These adoption of roles that can not be applied is controlled by a fluctuating rate (*RF interrupt*) that is positivity influenced by the availability of running but breakbale agents and the changing number of *Errors*. This rate describes the *resource-flows* (RF) that are interrupted due to the breaking of robots. These failures within robots influence the number of running robots (negative link), thus the problematic system behavior is dominated by a negative feedback loop (α). This dynamic would cause the number of robots which are not running to increase if not handled. So the design of an appropriate Solution Dynamic concerns the derivation of robot behavior that counteracts this effect. Robots with roles that they can no longer apply are *Waiting for Reconfiguration*. The rate of interrupts positively influences the increase of this variable. The system is equipped with a reconfiguration mechanism, and for each of the waiting robots a new configuration is determined. Thus the system shows a causal relation. In absence of waiting robots, no reconfigurations take place. Occurrences of waiting robots enforce subsequent reconfigurations (Reconfigure) to restore a set of executable roles. The reconfigurations thus increase the number of running robots by complementing a counteracting feedback loop (β).

After the anticipation of the affected system behavior this

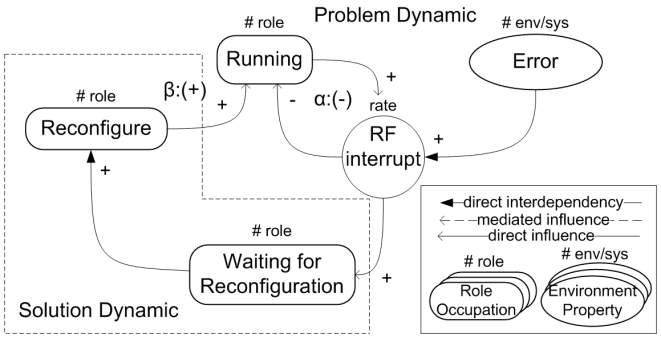


Fig. 3. The Problem and Solution Dynamic of the ODP-based resource-flow system [5].

reconfiguration strategy has been integrated into a MAS. The system implementation makes use of the freely available *Jadex*[1] agent framework. The robots and carts within the production lines are represented by Jadex–agents and the exchanged resources are mimicked by plain Java–objects that are exchanged via Jadex–services [14]. As an addition to the model described before the implemented system was extended with output buffers for the robots. These buffers were necessary for the modeling of loops in the resource–flow. By design for every role a robot owns at least one buffer place has to be reserved. So the robot can store the resources processed by this role before they are transported from a cart to the next robot in line. The ratio of roles robots own and the size of it's buffer is called (static) *workload*. The amount of roles a robot owns is not allowed to be larger than the robot's buffer size. So a workload of one is the maximum. Also differential to previous publications [3] the agents were implemented as micro agents and not as BDI[2] agents allowing simulations with a much higher count of agents (up to 200 micro agents as compared to up to 20 BDI agents).

The coordination is implemented using the Decentralized Coordination for Multi–Agent Systems (*DeCoMAS*) architecture [13]. In the current version this architecture makes use of *Coordination Spaces* [15] for the distribution of the coordination information (in this case the help request and reply messages). For this application a tailored medium realization is utilized that routes request and reply messages along a communication graph which spans a token ring over all participating agents. So conceptually all agents are aligned in a circle thus all agents can be reached independent of the location of the incapacitated agent. Endpoints encapsulate the coordination logic to reconfigure the agents and interact via the medium. Endpoints observe the agents and initiate the reconfiguration process by sending a help request if an agent becomes deficient. The help request is forwarded through the medium as described in the example (see Fig. 2). Each endpoint along the message path decides locally whether to adopt the deficient agent's role or continues forwarding the message. If the endpoint decides to adopt the role a reply is sent. The reply is sent backwards through the medium until all affected agents (robots and connected carts) are informed. Again each endpoint receiving the reply decides to change the agent's configuration. If an endpoint receives multiple coordination messages (requests and replies) these messages are queued and processed in the order of their arrival.

Based on this approach of a tailored coordination medium a reconfiguration strategy interface was implemented that allowed the use of different reconfiguration strategies. These strategies are responsible for the local decision the endpoints made during the reconfiguration process. Locally deciding which roles from a help request they adopt and which roles of their own they dispense to a deficient agent. One of the implemented reconfiguration strategies was the so called *Hybrid Strategy* which is a hybrid strategy consisting of multiple rounds, out of two other strategies which map the behavior of the strategy's first and second round. This strategy is based on the assumption that if an agent can not directly swap roles with an other agent it could lead to a better result if he forwards the help request further on instead of performing a swap that would lead to a transitive swap. So the help request is routed through the system in three rounds. In the first round agents only adopt roles if they can perform a direct swap of roles with the deficient agent. The second round makes use the of output buffers. So if no robot was able to perform a direct swap with the deficient robot in the first round the receiving robots check whether their buffer are big enough so they can adopt the included roles without the need to dispense roles of their own. This will lead to an increased workload in the adopting robot but it will avoid a possible transitively swap and ensure the system's functionality. Still the invariant that a workload greater than one is not allowed should not be violated. In the third and last round a transitively swap is performed.

IV. Stochastic Model

Depending on the system's parameters the stochastic model should give information over the probability of a role swap with a specific robot. From these probabilities it is possible to derive how far the distance to a swapping partner is. A help request passes stepwise through all the robots. If a robot is not able to process a help request according to reconfiguration strategy, the help request will be forwarded to the next robot and the distance stored in the help request is incremented. If a robot can process the help request then it will not be forwarded any further and the role swap is successful. If p is the probability with which a robot can process a received help request, then this consideration is based on a geometric distribution described as $P(X = n) = p(1 - p)^{n-1}$. With this geometrical distribution it is possible to calculate the probability with which a help request is processed after n steps.

Like described earlier before a help request runs through the system in multiple rounds because the robots are arranged as a ring. The hybrid strategy contains of three rounds with each showing a specific behavior. After every round the behavior

[1]http://jadex-agents.informatik.uni-hamburg.de/

[2]Belief, Desire, Intentions

is altered. Therefore the role swap distance is incremented continuously. Based on the assumption that the system contains of r robots, the help request will have passed r robots after the first round, $2r$ robots after the second and $3r$ robots after the third and final round when it reaches the deficient robot (sender). These number of steps are the quality attribute of the reconfiguration. The smaller this number is the better (faster) is the reconfiguration process. With regard to the stochastic model the number of steps is the random variable and called Ψ. Ψ is a discrete finite set of steps which can occur during the reconfiguration. Because a deficient robot can not process its own help request, Ψ can not take on the values r and $2r$. But Ψ contains the value $3r$ which marks a reconfiguration error when no robot could process the help request. Which means that after three rounds no swapping partner was found. Ψ can be described as follows: $\Psi = \{j \in \mathbb{N} | 1 \leq j \leq 3r \wedge j \neq r \wedge j \neq 2r\}$. The containing elements can be referenced as ψ_i. The index i describes the position of the element ψ in the set. The set can be indexed because it is totally ordered.

The strategy defines the probability with which a help request can be processed by a robot. According to the description above the hybrid strategy switches to another behavior after every round. Therefore the probability with which a help request can be processed changes after every round. The probability mass function of the geometrical distribution $P(X = n)$ needs to be defined stepwise. So the probabilities of success for the different rounds have to defined as p_1 to p_3. Resulting in the following probability function:

$$P(\psi = n) =$$

$$\begin{cases} p_1(1-p_1)^{n-1}, & n < r \\ (1-p_1)^{r-1}p_2(1-p_2)^{n-r-1}, & r < n < 2r \\ (1-p_1)^{r-1}(1-p_2)^{r-1}p_3(1-p_3)^{n-2r-1}, & 2r < n < 3r \\ (1-p_1)^{r-1}(1-p_2)^{r-1}(1-p_3)^{r-1}, & n = 3r \end{cases}$$

The first three cases map the geometrical distribution. Thereby the failure of the first round has to considered in the calculation of the probabilities for the second round. According to this the failure of the two previous rounds has to be considered for the third round. The fourth part of the function describes the failure probability of all three rounds. Because every event from Ψ has a assigned probability and the failure is defined at $3r$ steps the distribution is normalized. The sum of all probabilities out of Ψ is one.

According to the hybrid strategy the probabilities for p_1 to p_3 have to be defined for each round. In the hybrid strategy's first round a partner for a direct role swap is searched. Therefore a robot is searched which can apply the deficient capability. The probability for this is equivalent to the earlier before mentioned redundancy rate ρ. Additionally the condition, that the swapping partner has at least one capability which the deficient robot can apply has to be valid and the swapping partner has to own a role which applies this capability so he can dispense it to the deficient robot. To calculate this probability one divides the number of favorable event through the number of possible events. For this calculation the whole number of capabilities in the system is needed. In the following this amount is called c. According to the definition of the redundancy rate ρ every robot owns $\rho \cdot c$ capabilities in middle. But because not only the capabilities have to match, but the swapping partner also has to own roles for theses capabilities, also the role-density δ is needed. For a specific system δ states the middle rate of capabilities for which a robot owns roles. A δ value of 0.5 means that in middle for half of the capabilities a robot has, it also owns roles. The role-density has to be calculated for every configuration depending on the role assignments. For the calculation of the possible events, the combinations of the possible capabilities for every robot has to be determined. Because the deficient robot owns $\rho \cdot c$ capabilities where one deficient capability has to be subtracted $k = (\rho \cdot c) - 1$ elements out of the universal set with c capabilities could be chosen. Because a random distribution of capabilities is excepted in this system, $\binom{c}{k}$ possibilities exist for the deficient robot. A similar consideration has to be done for the swapping partner. It also owns $\rho \cdot c$ capabilities. Because it can adopt the deficient capability it also has to be subtracted in this case. Since the swapping partner has to own roles which it can dispense to the deficient robot the range lessens according to the role-density δ. So the swapping partner chooses $t = \delta((\rho \cdot c) - 1)$ from the c capabilities, which results in $\binom{c}{t}$ possibilities. Therefore the amount of possible events is $\binom{c}{k} \cdot \binom{c}{t}$.

Out of this set of events the ones which have at least one capability in common in $\binom{c}{k}$ and $\binom{c}{t}$ have to be chosen. The amount of possibilities with nothing in common can be calculated with $\binom{c}{k} \cdot \binom{c-k}{c-k-t}$. To calculate the favorable, the unfavorable have to be subtracted from all the possibilities. Hence the swapping probability for the first round is:

$$p_1 = \rho \cdot \frac{\binom{c}{k}\binom{c}{t} - \binom{c}{k}\binom{c-k}{c-k-t}}{\binom{c}{k}\binom{c}{t}} = \rho \cdot \frac{\binom{c}{t} - \binom{c-k}{c-k-t}}{\binom{c}{t}}$$

In the second round the hybrid strategy tries to find a robot which can apply the deficient capability. Therefore the according roles should be taken additional to the robots previous roles and no roles should be dispensed. Therefore the deficient robot does not need to be considered. Like in the first round a robot has to be found which owns the deficient capability. The probability for this is ρ. If a robot can apply further roles depends on whether it has enough space in its buffer. This is modeled with the mean system workload ω. The probability to find a robot which can apply the roles for the deficient capability is $p_2 = \rho \cdot (1 - \omega)$. ω does not model the dynamic processes in the system but is a static value. However ω is used in the model as a calculated simplification. It is possible that a role can not be applied during the second round although the according robot has more buffer spaces then roles. This happens if the spaces are blocked by work–pieces that are not yet transported to the next robot. Therefore the stochastic model has a higher probability for a swap in the second round than the implemented system.

For the third round of the hybrid strategy the probability is $p_3 = \rho$. In this round the strategy just tries to find a robot which can apply the deficient capability for transitive role swap.

With this probabilities it is possible to use the geometrical distribution function $P(\psi)$ to calculate concrete probabilities for specific distances. With the distribution function $F(n)$ it is possible to calculate the probability that a role swap would be successful in the first n steps. Therefore the function sums up all the probabilities from Ψ:

$$F(n) = P(\Psi \leq n) = \sum_{\psi \in \Psi \wedge \psi \leq n} P(\psi)$$

The expectation value for Ψ can also be calculated with $P(\psi)$:

$$E(\Psi) = \sum_{\psi \in \Psi} \psi \cdot P(\psi)$$

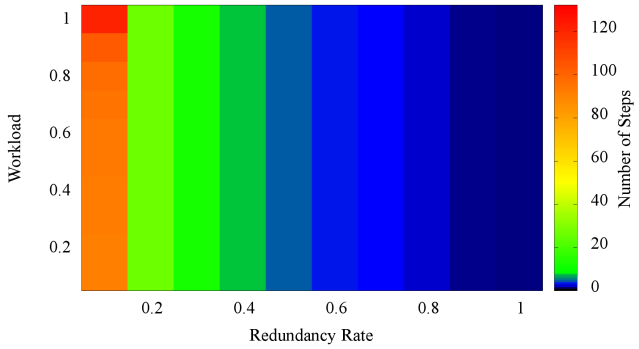
The expectation values returns a concrete value for the needed number of steps for the reconfiguration process depending on ρ and ω. This number is excepted as the mean value for a high number of reconfigurations.

The here described model offers a mesoscopic view [16] on the reconfiguration process. Mesoscopic since it approximates transition probabilities for the microscopic characteristics of an agent calculated from an average behavior of the rest of the system under an ergodicity assumption. The system's behavior is described with probabilities, which are modeled with the geometrical distribution function. The geometrical distribution function predicates the probability of a successful reconfiguration of the system after a specific number of steps. So the expectation value is a mesoscopic parameter which describes the system's behavior.

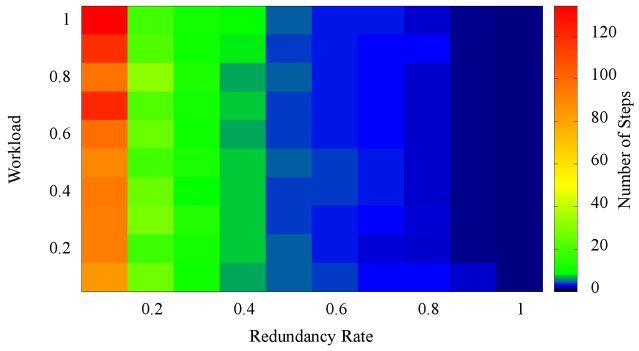
V. Results

To analyze the scalability and robustness of the developed MAS several simulation scenarios were performed. These scenarios consists of different numbers of robots and different resource–flow layouts. The system sizes ranged from 20, 30, 40, 50, 70 and 100 up to 200 robots. For each of these seven different scenarios different redundancy rate and workload values each in the range of 10% to 100% in 10% steps were used. For each value pair of redundancy rate and workload 50 simulations were performed each starting the reconfiguration process due to a different random failure. That resulted in a total of 5000 simulation runs for each scenario and an overall total of 35000 runs for the whole analysis.

The first step in the process of analyzing the scalability and robustness of the developed system was to validate the stochastic model against the actual system. In order to use the stochastic model for scalability analysis it had to be verified that the stochastic model maps the behavior of the actual system credible. Therefore, the mean value for the needed number of help request steps for a reconfiguration was measured for each simulation scenario, characterized by the number of robots, the redundancy rate and workload, based on the mentioned 50 runs. This mean value then was compared with the calculated expectation value based on the stochastic model (see Section IV). The results for 200 robots are illustrated in Fig. 4. Fig. 4(a) shows the results of the stochastic model and Fig. 4(b) shows the simulation results. Both figures showed the same trends in numbers but further analysis was required to verify the stochastic model. Although the figures for the other scenarios showed the same trends.



(a) Stochastic Model Results.



(b) Simulation Results.

Fig. 4. Results for 200 Robots.

The next step in the verification process was to calculate the relative difference between the expectation value and the simulation result. Therefore the simulation result value was subtracted from the expectation value. The difference then was divided by the expectation value to get the normalized result. Fig. 5 shows the results for 200 robots. A positive value means that the simulated system found a reconfiguration faster than the stochastic model. Vice versa a negative value means that according to the stochastic model the simulated system needed more number of steps to find a reconfiguration than expected. The relative difference maps did not show any trends in the data which could lead to any systematic deviations between the simulated system and the stochastic model.

To analyze the robustness of the developed system it had to be shown that the reconfiguration algorithm is actually able to reconfigure the system. Therefore the number of runs that could not be reconfigured were counted and divided by the number of all runs. Fig. 6 illustrates the results with a logarithmic y–axis, showing that there is an exponential decay

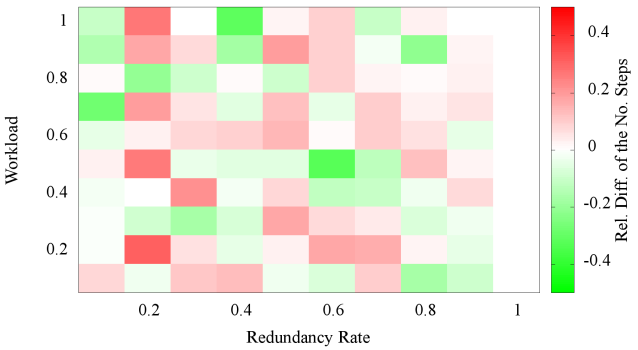


Fig. 5. Relative Difference of the Number of Steps for 200 Robots.

in the reconfiguration errors as the data produces a straight line with negative slope. As shown by the fit function. If the number of robots increases, the number of reconfiguration errors decreases exponential, showing that the algorithm is optimal and robust for system with a high amount of robots. The robustness mostly depends on the redundancy rate and the system size. A large system with a high redundancy rate results in a high number of redundant capabilities present in the system. Therefore the chance to find a partner for a direct swap is also high. As defined by the second round of the described reconfiguration algorithm the influence of the workload is negligible in this cases because the algorithm will find a new configuration in the first round most of the times.

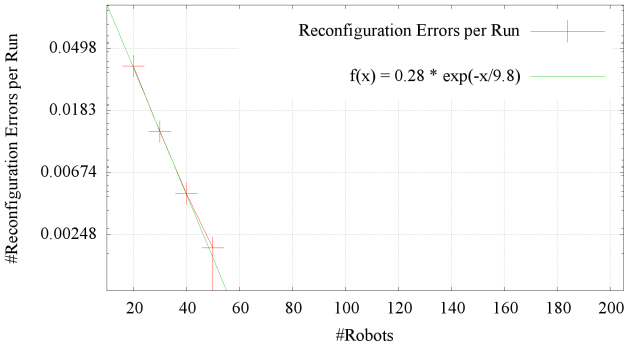


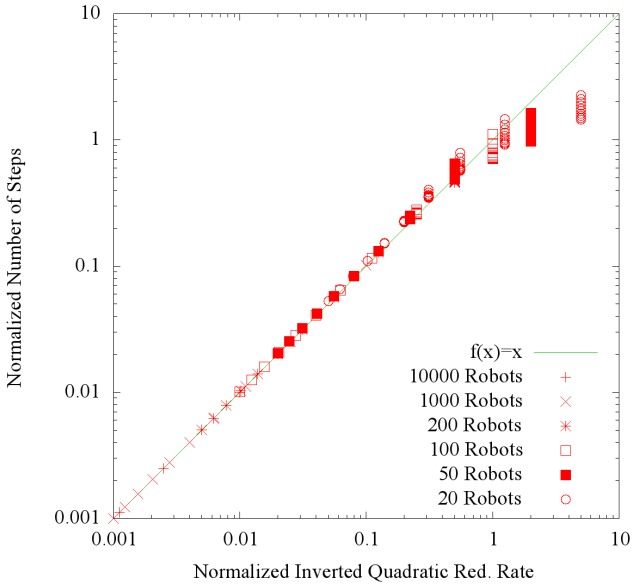
Fig. 6. Robustness analysis: Amount of reconfiguration errors.

To show that the reconfiguration strategy proposed in section III scales well Fig. 7 was created. The y–axis of this plot shows the number of required help request steps for the reconfiguration normalized to the system size (number of robots). On the x–axis the inverted quadratic redundancy rate also normalized to the system size is plotted, so values for a low redundancy rate are plotted at higher x–values. The quadratic redundancy rate maps the probability of a successful role–swap in the reconfiguration strategy's first round. This is a simplification to the stochastic model as described in section IV because the influence of the mentioned *role-density* δ was neglect-able. Both axis are plotted with a logarithmic scale. The fit function $f(x) = x$ shows the asymptotic scalability behavior. For a decreasing redundancy rate the required help request steps for a reconfiguration increase proportional to the system size. Fig. 7(a) illustrates the results for the stochastic model, showing that $f(x)$ fits perfectly for large systems. Only for smaller system with 20 and 50 robots and also 100 robots if the redundancy rate is low, the values differ from the fit function. The variation in the values for the same redundancy rate displays the influence of the workload on the reconfiguration. Showing that the workload only influences the reconfiguration for small systems with a low redundancy rate. Fig. 7(b) illustrates the simulation results for the actual system showing the same asymptotic behavior. The higher variation in the values for the same redundancy rate indicates that the actual influence of the workload on the reconfiguration process is bigger than mapped by the stochastic model. Although the redundancy rate still has a much bigger influence and the influence of the workload still decreases for larger systems. But both results show that the proposed reconfiguration strategy scales very well for large system.

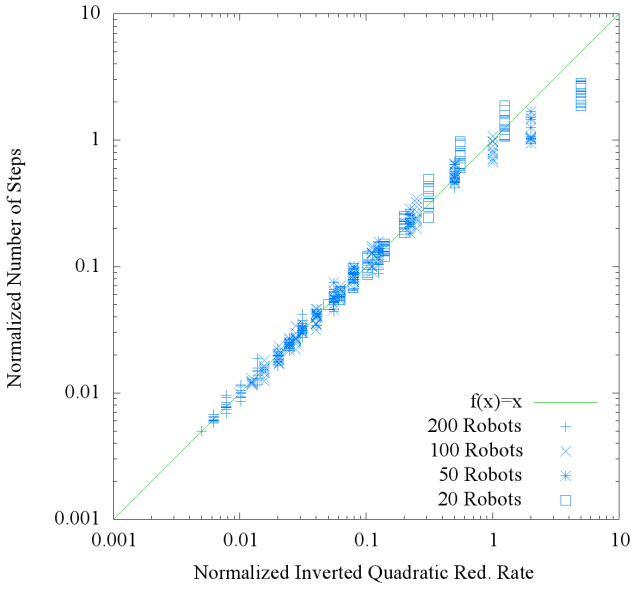
VI. Conclusion

In this paper a class of self–organizing resource–flow system was introduced. These systems profit enormously from self–organization like self–healing as they become more robust against failures. In order to guarantee this robustness as a self–healing property, reconfiguration algorithms are required. These algorithms can be centralized or decentralized. While centralized algorithms are often more robust then their decentralized counterparts. The decentralized algorithms often scale better [3]. In this paper a decentralized algorithm that is both robust and scales well was presented. Therefore a resource–flow system in the application of a production line where robots and AGVs process and transport resources was introduced. In order to maintain the system's functionality after a failure a decentralized coordination process which reconfigures the system was developed. Based on a large number of simulation runs it was shown that the proposed decentralized solution is optimal and robust for systems with a high amount of robots.

To show the scalability of the proposed reconfiguration process not only based on a multi–agent based simulation it was also mapped as a stochastic model. Before the stochastic model could be used to prove that the designed decentralized algorithm scales well, it had to be verified that the stochastic model maps the designed algorithm sufficient. Therefore the results of the stochastic model were compared with the ones from the implemented system. As no systematic deviations between the simulated system and the stochastic model were found, it was declared suitable for the scalability analysis. Based on the stochastic model it was shown that the proposed algorithm scales well for large systems. With the stochastic model this conclusion could be based on systems up to 50 times larger then the simulated ones. So by verifying the stochastic model against the actual system, not only could be shown that the algorithm scales well for larger systems but also that it is practicable to use such a stochastic model to



(a) Stochastic Model Results.



(b) Simulation Results.

Fig. 7. Scalability analysis: Asymptotic behavior.

make predictions over a system's behavior in cases where the simulation costs would be very high.

ACKNOWLEDGMENT

We would like to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting this work in a project on "Self–organisation based on decentralized Coordination in Distributed Systems" (SodekoVS).

REFERENCES

[1] A. Hoffmann, F. Nafz, A. Schierl, H. Seebach, and W. Reif, "Developing self-organizing robotic cells using organic computing principles," in *Bio-Inspired Self-Organizing Robotic Systems*, ser. Studies in Computational Intelligence, Y. Meng and Y. Jin, Eds. Springer Berlin / Heidelberg, 2011, vol. 355, pp. 253–273.

[2] M. Güdemann, F. Nafz, F. Ortmeier, H. Seebach, and W. Reif, "A specification and construction paradigm for organic computing systems," in *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO '08*, Oct. 2008, pp. 233 –242.

[3] J. Sudeikat, J.-P. Steghöfer, H. Seebach, W. Reif, W. Renz, T. Preisler, and P. Salchow, "On the combination of top-down and bottom-up methodologies for the design of coordination mechanisms in self-organising systems," *Information and Software Technology*, Sep. 2011.

[4] J. Sudeikat, L. Braubach, A. Pokahr, W. Renz, and W. Lamersdorf, "Systematically Engineering Self-Organizing Systems : The SodekoVS Approach," *Electronic Communications of the EASST*, vol. 17, 2009.

[5] J. Sudeikat and W. Renz, "On the modeling, refinement and integration of decentralized agent coordination," in *Self-Organizing Architectures*, ser. Lecture Notes in Computer Science, D. Weyns, S. Malek, R. de Lemos, and J. Andersson, Eds. Springer Berlin / Heidelberg, 2010, vol. 6090, pp. 251–274.

[6] P. Davidsson, "Multi agent based simulation: Beyond social simulation," in *Multi-Agent-Based Simulation*, ser. Lecture Notes in Computer Science, S. Moss and P. Davidsson, Eds. Springer Berlin / Heidelberg, 2001, vol. 1979, pp. 141–155.

[7] M. A. De Cerqueira Gatti and C. J. P. De Lucena, "A multi-environment multi-agent simulation framework for self-organizing systems," in *Multi-Agent-Based Simulation X*, ser. Lecture Notes in Computer Science, G. Di Tosto and H. Van Dyke Parunak, Eds., vol. 5683. Springer Berlin / Heidelberg, 2010, pp. 61–72.

[8] M. Cossentino, C. Lodato, S. Lopes, and P. Ribino, "Multi agent simulation for decision making in warehouse management," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, sept. 2011, pp. 611 –618.

[9] L. Henesey, P. Davidsson, and J. A. Persson, "Evaluation of automated guided vehicle systems for container terminals using multi agent based simulation," in *Multi-Agent-Based Simulation IX*, ser. Lecture Notes in Computer Science, N. David and J. Sichman, Eds. Springer Berlin Heidelberg, 2009, vol. 5269, pp. 85–96.

[10] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever, "Decentralized control of e'gv transportation systems," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '05. New York, NY, USA: ACM, 2005, pp. 67–74.

[11] S. Bussmann and K. Schild, "An agent-based approach to the control of flexible production systems," in *Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation*, vol. 2, 2001, pp. 481–488.

[12] O. Gurcan, O. Dikenelli, and C. Bernon, "Towards a generic testing framework for agent-based simulation models," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, sept. 2011, pp. 635 –642.

[13] J. Sudeikat and W. Renz, "Masdynamics: Toward systemic modeling of decentralized agent coordination," in *Proceedings of KIVS 2009 - Kommunikation in Verteilten Systemen*, K. G. K. David, Ed. Springer, 2009, pp. 79–90.

[14] A. Pokahr and L. Braubach, "From a research to an industrial-strength agent platform: Jadex v2," in *Business Services: Konzepte, Technologien, Anwendungen - 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, H.-G. F. Hans Robert Hansen, Dimitris Karagiannis, Ed. Österreichische Computer Gesellschaft, 2009, pp. 769–778.

[15] A. Vilenica, J. Sudeikat, W. Lamersdorf, L. Braubach, and A. Pokahr, "Coordination in multi-agent systems: A declarative approach using coordination spaces," in *Proceedings of the 20th European Meeting on Cybernetics and Systems Research (EMCSR 2010) - International Workshop From Agent Theory to Agent Implementation (AT2AI-7)*, R. Trappl, Ed. Austrian Society for Cybernetic Studies, 2010, pp. 441–446.

[16] W. Renz and J. Sudeikat, "Mesoscopic modeling of emergent behavior - a self-organizing deliberative minority game." in *Engineering Self-Organising Systems*, 2005, pp. 167–181.