Dependable risk-aware efficiency improvement for self-organizing emergent systems

Jonathan Hudson, Jorg Denzinger, Holger Kasinger, Bernhard Bauer

Angaben zur Veröffentlichung / Publication details:

Hudson, Jonathan, Jorg Denzinger, Holger Kasinger, and Bernhard Bauer. 2011. "Dependable risk-aware efficiency improvement for self-organizing emergent systems." In *2011 IEEE Fifth International Conference on Self-Adaptive and Self-Organizing Systems, 3-7 October 2011, Ann Arbor, MI, USA*, edited by Patrick Kellenberger, 11–20. Los Alamitos, CA: IEEE. https://doi.org/10.1109/saso.2011.12.

2011 Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems

Dependable Risk-Aware Efficiency Improvement for Self-Organizing Emergent Systems

Jonathan Hudson, Jörg Denzinger
Department of Computer Science
University of Calgary, Calgary, Canada T2N 1N4
Email: jwhudson@ucalgary.ca, denzinge@cpsc.ucalgary.ca

Holger Kasinger, Bernhard Bauer
Department of Computer Science
University of Augsburg, 86135 Augsburg, Germany
Email: {kasinger, bauer}@informatik.uni-augsburg.de

***Abstract*—An efficiency improvement advisor agent acts as a consultation service for a self-organizing multi-agent system that improves operational efficiency. It identifies recurrent tasks in past problems that allow the creation of so-called exception rules for individual agents to limit future inefficient behavior. There exists the danger that introduced rules could possibly infringe on the flexibility and therefore reliability of the system. In this paper, we present a dependable risk-aware efficiency improvement advisor that uses Monte Carlo simulation techniques in strategic analysis assessing the long-term potential and risks of prospective rules. Our experimental evaluation, for the domain of dynamic pickup and delivery problems, shows that the result is a minimal, yet effective, set of risk-averse exception rules. These rules can be provided to individual agents to reliably achieve an overall long-term improvement in efficiency while maintaining flexibility.**

***Keywords*-risk management; control; dependability; self-organization; software architecture;**

I. Introduction

Operational efficiency in self-organizing emergent systems [25] is usually a secondary concern. System architects are primarily concerned with developing distributed systems exhibiting the desirable properties of scalability, flexibility, adaptability, and dependability. This focus is justified since many application areas represent dynamic problems that require these attributes. Furthermore, for some situations, achieving optimal operational efficiency is simply not possible without perfect prediction of future events. Despite this, the deployment of self-organizing emergent systems, requiring significant fiscal or resource investment, necessitates the consideration of operational efficiency (e.g. agents required, solution costs, ...). Introducing operational efficiency generally requires a trade-off between maintaining the purity of the primarily local self-organizing system properties and introducing modifications that address the unavoidably global concern of efficiency.

In [26], we described a concept that improves the operational efficiency of self-organizing emergent systems while almost completely preserving their earlier described beneficial properties. This concept is called an *efficiency improvement advisor* (EIA). The EIA agent acts as a consultant, analyzing collected agent histories offline from the base system's operation and only communicating with the agents when they are otherwise unoccupied and/or in range, for example after returning to a central depot at the end of the day. The EIA uses data mining techniques to independently identify previously completed recurring tasks in the collected history of the system's execution. A nearly optimal solution is determined for this subset of recurring tasks using conventional optimization techniques. From this solution the EIA derives advice in the form of minimalistic exception rules focused on limiting future inefficiencies related to these recurring tasks. In this manner, a slight trade-off is admittedly made between the base system's flexibility in return for a substantial improvement to efficiency.

The initial form of exception rules (i.e. the type of advice), presented in [26], were termed *ignore rules*, which suggest to an agent that it should ignore a particular task. *Ignore rules* act as a minor influence on agent behavior, as demonstrated in [13], where they showed low potential for exterior abuse (i.e. reduced performance due to inflexibility). On the other hand, [16] calls attention to unaddressed operational inefficiencies in self-organizing systems where *ignore rules* were ineffective. To address this, in [27], the concept of *pro-active rules*, that suggest to an agent to start working on the first step of an expected forthcoming recurring task after the occurrence of a trigger, was introduced. Results demonstrated that these *pro-active rules* successfully improved efficiency when *ignore rules* were ineffective, but were easier to exploit as they represent a more invasive influence on flexibility. The results in [27], using the automated testing method introduced in [13], exposed the danger that the good-intentioned but short-sighted application of rules by the original EIA can negatively impact the dependability of the system.

In this paper, we introduce the concept of a dependable *risk-aware efficiency improvement advisor* (RA-EIA) capable of applying strategic analysis when assessing prospective advice (i.e. exception rules). The goal of this analysis is to determine if, in the long-term, prospective rules are risk-averse with respect to the system's history and future expectations. Risk-averse rules provide gains by limiting inefficiencies related to completing the recurring tasks. At

the same time, risk-averse rules avoid the potential of long-term losses resulting from introduced inflexibilities that are exploited by the dynamic nature of non-recurring tasks.

Through long-term risk assessment, a single consultation increases the period of time for which the advised system can dependably operate. By screening the multitude of prospective rules for those that are the least risky, yet still effective, the RA-EIA minimizes the quantity of advice and is capable of better maintaining the base system's flexibility. The RA-EIA provides a limited-in-quantity, yet comprehensive, set of long-term beneficial risk-averse rules so that the underlying system is capable of extended periods of reliable independent online operation.

Similar to previous work in [26], [13], and [27], we use *pickup and delivery problems* (PDP) [23] as the application domain for the experimental evaluation of this concept. Our results show that the RA-EIA, by taking advantage of the time available through independent offline execution, is able to provide minimal risk-averse advice achieving improved performance compared to the original EIA. The results confirm that, although it is not possible to avoid all inefficiencies due to the unpredictable and dynamic nature of the problem, in general risks can be minimized such that the overall gains are significantly positive. The complete RA-EIA assisted system represents a solution that gains the improved efficiency provided by the EIA concept while dependably ensuring minimal long-term impact to the flexibility of the base self-organizing emergent system.

II. Related Work

In regard to related work, we are primarily concerned with the distribution of system control and how to achieve reliable operational efficiency. There exists a spectrum when considering the distribution of control. At one end exists centrally controlled systems that continuously observe system state and guide individual behavior, such as [10]. With central control, operational efficiency is the easiest to attain, especially given a static problem, as there can be complete knowledge of the system. However, central control is not always feasible. For example, when the problem is dynamic, and messaging has to be kept to a minimum, distributed control represents a well-suited alternative [3]. At the distributed end of the spectrum, each component acts with limited local knowledge, and operational efficiency cannot be guaranteed to emerge from system behavior.

The unpredictability found in many distributed systems prevents them from being widely adopted in industry [21]. When attempting to achieve coherency in a distributed system, inherent difficulties exist in accommodating the effect of local decisions on global operational efficiency [2]. The more dynamic the problem, and the higher the requirement is for short system response times, the more desirable a flexible distributed self-organizing solution becomes.

Control theory represents the most developed centralized approach to achieving operational efficiency in which numerous approaches attempt to learn from past behavior to predict the future. For example, Model Predictive Control (MPC) [4] attempts to use a stochastic linear model created from the system's history to derive a control action. Limited lookahead control (LLC), based on a fixed stochastic model, is used in [1] to optimize the forecast behavior of the system. Contrary to our purposes, these stochastic models are static, and the controller completely controls the system removing all flexibility. Additionally, linear models for self-organizing emergent systems are generally not available. In general, the majority of traditional control theory approaches represent a tightly coupled and functional relationship between the controller and the base system leaving little flexibility.

There are a number of top-down self-managing approaches in which control theory methods are applied to manage system behavior. The observer/controller (O/C) from Organic Computing [22] requires observation and control of the underlying system at all times. Similarly, the multi-agent system management-by-exception approach [24] involves a high-level agent taking total control over other agents. In Autonomic Computing [14], an autonomic manager acts as a central control, monitoring sensors and adjusting effectors. Autonomic Communication's autonomic control loop [7] is essentially an abstraction of autonomic managers. The Autonomic Computing paradigm does allow for planning and the Autonomic Communication method for risk analysis in the control loop, but neither makes the important distinction that the underlying system remains completely independent during online execution.

There also exist completely bottom-up self-organized distributed approaches to managing system behavior. Sometimes these systems are known as Decentralized Autonomic Computing (DAC) systems. The ant colony path optimization technique [8] may be used to find good solutions for PDP but it is not designed to actively solve a dynamic PDP problem as it is announced over a period of time. Numerous design patterns have been created based on natural paradigms to design self-organizing emergent systems. The base system used in the experimental evaluation of this paper is one example of such a system and is described in [15]. In completely distributed systems, the ability to centrally enforce operational efficiency is generally sacrificed for more scalable and flexible distribution of control.

Hybrid solutions, such as the EIA concept, exist to combine the beneficial properties of self-organizing distributed systems with those of self-adapting centralized systems. The distributed hybrid approach presented in [28] includes both top-down exogenous self-management through self-healing central control as well as a bottom-up endogenous self-management through self-adaptation of organizations. Hybrid multi-agent systems for PDP are generally based on predefined stochastic models [17] which either completely

control the system or do not incorporate learning from the past, such as [19]. The EIA method of exception rules, and even more-so the risk-aware RA-EIA, is intentionally designed to learn from the past to predict the future and avoid central control as it infringes on flexibility.

The complexity of systems and availability of monitored data have motivated the application of machine learning and other statistical techniques to system design. In [6], statistical approaches are applied to characterize performance problems in middle-ware-based systems. Monte Carlo methods and simulation are used in many fields such as well drilling [5], project scheduling [18], and finance [9] to assess risk. Monte Carlo simulation is a process that uses the random variables associated with components of a problem to generate probable future problem instances. The performance of the system, over probable future instances, is sampled to form a statistical distribution, the properties of which are used to assess the system. Monte Carlo simulation is commonly used to quantify risk where there is the possibility of long-term costly consequences to decisions.

The RA-EIA combines statistical Monte Carlo methods with the EIA's limited machine learning to prepare for the future. This is an examination of the variability of risk associated with the ruleset. The exploration of the extent of the severity of risk associated with rules is currently being explored, but is not included in this paper. The RA-EIA operates offline from the base system and thus is unable to influence the system during active operation. However, unlike online approaches, the RA-EIA is able to utilize the computational intensive method of Monte Carlo simulation. This allows more global assessment of risk during the consideration of the collected histories. The rules provided after consultation advise the system so that it can reliably achieve independent long-term operational efficiency. Moreover, because the EIA control loop is decoupled with regard to execution time, scalability remains largely independent and the majority of flexibility is maintained.

III. Basic Definitions

A multi-agent system (MAS) is a set of agents $A = \{Ag_1, Ag_2, \ldots, Ag_n\}$ that are acting in a shared environment Env. The class of problems we are interested in solving consists of a selection of tasks from a set T that are announced in Env during a time interval $Time$. This announcement is generally not simultaneous, resulting in a dynamic task fulfillment problem.

We will call this sequence of announced tasks a run instance, which may be defined as a sequence of task-time pairs $ri = ((ta_1, t_1), (ta_2, t_2), \ldots, (ta_m, t_m))$ with $ta_i \in T, t_i \in Time$, and $t_i \leq t_{i+1}$. Typically, a problem consists of a sequence of run instances. A sequence of run instances of length k is then $(ri_1, ri_2, \ldots, ri_k)$. Within this sequence, nearly similar tasks may occur in multiple individual run instances.

A solution sol for a run instance compiled from the emergent responses of a self-organizing multi-agent system A is then $sol = (as_1, as_2, \ldots, as_m)$ where $as_i = (ta'_i, Ag'_i, t'_i)$ with $ta'_i \in \{ta_1, ..., ta_m\}, ta'_i \neq ta'_j$ for all $i \neq j$, $Ag'_i \in A, t'_i \leq t'_{i+1}$, $t'_i \in Time$. A tuple (ta'_1, Ag'_1, t'_1) means task ta'_i was started by agent Ag'_i at time t'_i. In this paper, fulfilling at task requires at least two actions to be taken by agent Ag'_i. We identify the first (or preparation) action of a task ta by $prep(ta)$.

In order to address the efficiency of a multi-agent system A, when solving a run instance, it is necessary to associate with each solution sol a quality measure $qual(sol)$. The nearer the solution of A to the optimal quality, the more efficient A is. Quality measures are naturally dependent on both the multi-agent system A and the application area. Additionally, multiple quality measures often exist for a chosen system and application area. For dynamic problems, where the complete problem is not apparent at one time, it is generally not possible for A to produce the optimal solution unless the static variant of the problem is trivial.

IV. Efficiency Improvement Advisor

The EIA is an additional consultation agent Ag_{EIA} that is added to A. Note that the original system is not dependent on the EIA during online operation. Outside of limited communication with agents, when they return to a central depot, the actions of the Ag_{EIA} occur offline from system execution. Several requirements that must be met to use the EIA are described below.

1) The agents in A require both the opportunity and capability to transmit their local histories to the Ag_{EIA}.
2) The problem must contain recurring tasks, allowing the Ag_{EIA} a limited opportunity to "predict" the future.
3) Since the derivation of advice takes place offline, possibly over a period of several run instances, the set of recurring tasks must remain stable long enough that advice remains valid when communicated.
4) The agents in A are modifiable by exception rules.

The actions of the EIA are depicted in the functional architecture of an EIA, Figure 1, and described below:

1) **receive**(Ag_i, H_i): collects the local history H_i of agent Ag_i when it is non-disruptive to communicate.
2) **transform**$(H_1, \ldots, H_n) = GHist$: creates the global history $GHist$ from the collected local histories of A. Essentially, $GHist$ consists of a sequence of run instances $(ri_1, ri_2, \ldots, ri_k)$ A has solved so far and A's emergent solution sol_{emg} for each run instance.
3) **extract**$(GHist) = ri^{rec}, NR$: data mines from this history a common set of recurring tasks ri^{rec} and disjoint set NR containing the remaining non-recurring tasks.
4) **optimize**$(ri^{rec}) = sol^{rec}_{opt}$: computes the optimal solution sol^{rec}_{opt} to the sequence of recurring tasks. This

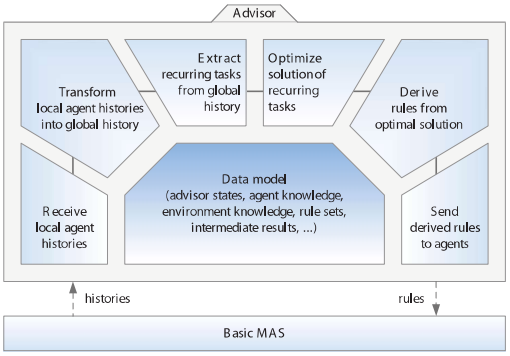

Figure 1. Efficiency Improvement Advisor

optimal solution to the recurring set of tasks sol^{rec}_{opt} is compared with A's emergent solution to the recurring set of tasks sol^{rec}_{emg}. If $\frac{qual(sol^{rec}_{emg})}{qual(sol^{rec}_{opt})} < qual_{threshold}$, then no rules need to be derived and the work of Ag_{EIA} is complete.

5) **derive**$(sol^{rec}_{opt}, sol^{rec}_{emg}, ri^{rec}, NR) = rules$: creates for each agent Ag_i a set R_i of exception rules, if applicable, where R_i can also be empty.
6) **send**(Ag_i, R_i): communicates the set R_i of exception rules to the agent Ag_i, the next time communication with Ag_i is non-disruptive.

These actions are abstractions and obviously the concrete realizations of them will be application dependent. The details of the EIA are discussed in [26] along with an implementation for PDP. The remaining discussion will primarily be focused on the important aspects of each action.

The actions **receive**, **send**, and **transform** operate as previously described. The global history is important to the operation of the EIA and will be even more important for the risk-aware RA-EIA. The global history represents the only reliable source of global context that the Ag_{EIA} has on which to base long-term risk-aware rule decisions. It is important to note that the Ag_{EIA} is only ever aware of tasks that have been previously observed by agents. Unobserved, resultantly uncompleted tasks, will not be captured in the global history and furthermore not appear in the sequence of run instances produced by **transform**.

Determining recurring tasks can be completed in many ways. An important requirement in many applications is that "recurring" is considered within a fuzzy definition. That is, variations of a task between run instances of a sequence should be allowed, within a certain similarity measure of a general abstraction. This is possible through an application dependent similarity measure sim and having a threshold value $minocc$ for how often the abstract task has to occur to be recurring. Using a limited period k allows for the recurring tasks to change. In [26], as in this paper, the **extract** action determines the recurring set of tasks by data mining the last k run instances with a clustering algorithm, precisely, *Sequential Leader Clustering* [12]. The clustering algorithm also provides, for each task, an identifier that can be used by exception rules.

The action **optimize** should generally have little influence on the derivation of reliable risk-aware advice. Nevertheless, the accuracy of the determination of the optimal solution to the recurring set of tasks must be considered. Unfortunately, the determination of the optimum for even the static variant of many interesting (computationally hard) problems is infeasible, as was the case in [26]. Therefore, algorithms that approximate the solution, such as genetic algorithms, simulated annealing, or tabu search are required. A cache of solutions, obtained for run instances, can be maintained for a single execution of the system, but between separate executions the determined optimums can and will be different. This, at times, can make the deterministic repetition of a specific EIA experimental result challenging.

For this paper, we are concerned primarily with the **derive** action. This action encompasses the determination of exception rules for individual agents. We will discuss the original EIA's variant of this action in the following Section IV-A and its weaknesses in Section IV-B.

A. EIA Exception Rule Derivation

The original **derive** action works as follows. The EIA contains a set of rule derivators, each associated with a rule variant. A rule derivator rd creates rules, $rd(sol^{rec}_{opt}, sol^{rec}_{emg}, x) = rules$, by comparing the optimal solution sol^{rec}_{opt} with the computed emergent solution sol^{rec}_{emg} for the recurring tasks. A rule derivator returns rules that attempt to address a single deviation between the emergent and optimal solution, that is a single location j with either $ta^{opt}_j \neq ta^{emg}_j$ or $Ag^{opt}_j \neq Ag^{emg}_j$. The variable x tells the rule derivator to returns rules for the x^{th} (first, second, third, ...) of these locations. This allows a rule derivator to search for rules to address a later deviation even if no rules could be found to fix the previous deviation. If there exists no x^{th} location, then an empty set is returned.

The EIA's rule derivators are ordered by priority. The **derive** action queries each derivator for the first difference in order of their priority. The EIA tries to improve the emergent solution using the rules from the first derivator. If there is no change in the emergent solution, then it removes them and tries the next derivator. So far there have been two types of rules, and corresponding rule derivators, developed.

The first is an *ignore exception rule*, described in [26], that instructs the agent not to perform a particular abstract task (defined within variability allowed by sim), for at least a period of time. The hope is that, due to such a detraction, another agent will perform this particular task. This results in a more efficient emergent solution, at least for the set of recurring tasks. More precisely, an *ignore rule* is created for an agent from the emergent solution Ag^{emg}_j instructing

it to ignore a task ta_j^{emg} it originally completed, hopefully making the optimal agent perform it.

In contrast *pro-active exception rules*, described in [27], influence an agent to begin work on a particular abstract task (defined within variability allowed by sim), before this task usually appears. The purpose of this variant is two-fold. First, the agent will be able to complete preparation for the specified task before it appears, resulting in quicker satisfaction of the task's requirements. Second, an agent occupied with a *pro-active rule* will not be distracted by other tasks, hopefully leaving them for more suitable agents. *Pro-active rules* are also designed to create a more efficient emergent solution, at least for the set of recurring tasks. More precisely, a *pro-active rule* is created for the optimal agent Ag_j^{opt} instructing it to perform the preparation for an optimal task $prep(ta_j^{opt})$ on the occurrence of a trigger.

There are two general categories for triggering *pro-active rules*: *task-triggered pro-active rules* that are triggered by the occurrence of the task the specific agent should have completed before the indicated task in the optimal solution, and *time-triggered pro-active rules* that are triggered by a specific execution time. A timeout is included, for the rule being active, to avoid having the agent being blocked if a task never appears.

B. EIA Weaknesses

There are some important weaknesses to the original EIA concept that cannot be ignored, especially when considering the reliability of the system.

The EIA is rather short-sighted, creating rules one at a time and using the measured response of the system to judge if the rule was actually a good idea or not. If a rule is not successful, then it is deleted and a new rule is attempted. Altogether, this means the EIA takes a long time to change the system's behavior and is reliant on interaction after each run instance to do so. This, although less intrusive than central control methods, still infringes on the independence of the underlying system and has the possibility that a badly advised rule will cause decreased performance until it can be deleted on the next interaction.

The EIA will continue to advise rules as long as there remains a difference between the underlying system's emergent solution and the optimal solution to the recurring set of tasks. The EIA works under the naive assumption that the rules it generates will, at least eventually, have a positive impact on improving the system's performance. When there is the guarantee that all the tasks are recurring and will remain the same this assumption is generally true, as the EIA's advice simply forces the underlying system solution to be strictly the optimal response. Unfortunately, in the majority of application areas, the recurring tasks will change over time and there will be some non-trivial amount of non-recurring tasks that rules will impact. Resultantly, the EIA can eventually produce a large set of rules. Many of these may be largely ineffective and represent an irrefutably dangerous infringement on the base system's flexibility.

Overall, if the derivation of exception rules is inadequate, short-sighted, and/or restrictive, then the risk exists that the overall performance of the system will be decreased. Experimental results in [27] demonstrated both the high potential and corresponding risk that exception rules represent. Therefore, it is desirable that (1) the risks associated with rules are assessed before communication to agents, (2) the frequency of interaction is decreased, and (3) a minimal comprehensive set of rules is provided. It is undesirable, in fact contrary to the EIA's purpose, that efficiency and flexibility are negatively impacted.

V. Risk-Aware Efficiency Improvement Advisor

A *risk-aware efficiency improvement advisor* (RA-EIA), is an enhancement of the existing EIA concept. The original EIA concept is already capable of collecting local histories to form a global history from which it is able to determine and communicate exception rules designed to improve the system's response to recurring tasks. However, this derivation is undertaken in a naive and short-termed manner. The RA-EIA replaces the **derive** action of the original EIA concept with its own **derive**$_{RA-EIA}$ action which uses Monte Carlo simulation to assess the overall long-term impact of a minimum set of effective rules before communication.

To accomplish this there are two requirements:

1) The RA-EIA must be capable of determining some form of statistical distribution of the expectation of future events, which enables the use of Monte Carlo simulation.
2) It must be possible to simulate, or at least predict, the performance of the system when encountering one of these future problem instances.

The actions that form the **derive**$_{RA-EIA}$ action are described below:

- **generate**$(ri^{rec}, NR) = ri^{gen}$: creates a run instance ri^{gen} that is likely to occur in the future using the recurring run instance ri^{rec} and the set of non-recurring tasks NR extracted from the global history.
- **assess**$(ri^{rec}, NR, rules) = \overline{x}$: uses Monte Carlo simulation and determines a sample mean $\overline{x}$ for the statistical distribution of the simulated future performance for the given ruleset $rules$ compared to the base systems default performance.
- **derive**$_{RA-EIA}(sol_{opt}^{rec}, sol_{emg}^{rec}, ri^{rec}, NR) = rules$: creates for each agent Ag_i a set R_i of exception rules, where R_i can also be empty.

These actions are abstractions and obviously the concrete realizations of them will be application dependent.

The **generate** action creates a possible future run instance. Its size is chosen from the distribution of recent sizes seen

in the last k run instances of the global history. The new run instance ri^{gen} begins as the existing set of recurring tasks ri^{rec}, and additional task-time pairs are added from the set of non-recurring task-time pairs NR. These pairs are chosen so that they are not duplicates of those already in the generated run instance.

The **assess** action performs Monte Carlo simulation using the **generate** action to create possible future run instances. These run instances are generated until a statistically acceptable amount n_{req} has been created such that the sample mean $\overline{x}$ is acceptable or a maximum n_{max} is reached. After a minimum count n_{min} is passed, this acceptable sample size is determined using the equation

$$n_{req} \geq \frac{z_\alpha^2 \cdot \sigma_S^2}{\delta^2} \tag{1}$$

where z_α is the z-score associate with the chosen confidence level α (95% for this paper), σ_S is an estimated sample variance ($range(S)/4$ for this paper), and δ is the absolute error that we desired to estimate the mean within (1% for this paper). Details for the statistical methods are available at [20].

To form a sample, each generated run instance is simulated and evaluated for a measurement:

$$\frac{qual(sol^{gen}_{emg,rules}) - qual(sol^{gen}_{emg,default})}{qual(sol^{gen}_{opt})} \tag{2}$$

This measurement compares the quality of the system's solution with the ruleset $qual(sol^{gen}_{emg,rules})$ to the quality of the solution of the system without any rules $qual(sol^{gen}_{emg,default})$. This measurement is normalized by the optimal $qual(sol^{gen}_{opt})$. If a measurement is negative, then the ruleset improved the system's solution compared to the default base system. If the final sample mean $\overline{x}$ is negative, then we know that the ruleset $rules$, on average, represents an improvement compared to the base system (with a certain confidence within a given absolute error).

The RA-EIA contains a set of rule derivators, each associated with a rule variant. The **derive**$_{RA-EIA}$ action consists of a loop that continues until, either the derivators no longer provide unique prospective rules, or the time given for deriving rules expires. The loop begins with an empty set of accumulated rules $rules_{acc}$, a set of rules currently advised $rules_{exist}$, and with the rule derivators considering the first difference ($x = 1$) between the existing emergent solution and the optimal solution to the recurring set of tasks.

For each set of prospective rules $rules_i$, returned by a derivator rd_i, a sample mean $\overline{x_i}$ is assessed. This mean is assessed using the **assess** action for the ruleset $rules_{exist,acc,i}$ which consists of the currently advised, the accumulated, and the prospective rules. Additionally the sample mean $\overline{x_{acc}}$ is assessed for only the accumulated set of rules using the **assess** action for the ruleset $rules_{exist,acc}$ consisting of the currently advised and the accumulated rules.

If one of the prospective rulesets produces a negative sample mean, then the prospective rules $rules_{min}$ associated with the most negative sample mean $\overline{x_{min}}$ are examined. If one of the rule derivators created prospective rules that improve on the accumulated set of rules (i.e. $\overline{x_{min}} < \overline{x_{acc}}$), then the associated rules $rules_{min}$ are added to the accumulated set $rules_{acc}$. The emergent solution of the system to the last run instance encountered sol^{rec}_{emg} is simulated again, and the rule derivators are reset to look at the first difference $x = 1$. Otherwise, none of the prospective rulesets provided a further improvement, so the rule derivators are incremented to look at the next difference $x = x + 1$.

The RA-EIA assesses the long-term overall risk (i.e. the statistical distribution of performance) of prospective rulesets. The ultimate result of the **derive**$_{RA-EIA}$ action is a quality set of accumulated rules that has been determined to provide on average the greatest performance increase relative to the default base system. Quality rules are well targeted rules that increase performance potential with limited risk of performance loss.

Prospective rules are created to address a single inefficiency at a time, and only the most effective rules are accumulated, thus limiting the quantity of communicated rules. The larger the quantity of rules that exist, the higher the likelihood that dynamic changes in the problem will result in one of the rules producing performance loss. Rather than providing a large set of rules that may provide complete coverage for the recurring set of tasks, but no flexibility for change, a minimal in quantity, yet targeted, set of rules is advised. This allows the system to retain flexibility and increases long-term reliability with respect to the dynamic nature of the problem.

Together the **derive**$_{RA-EIA}$, **assess**, and **generate** actions replace the original **derive** action. These actions perform a possibly significant amount of computation. However, this computation is done offline from the underlying system's execution, and the advice it provides from a single consultation is designed to be sufficient for a long period of time. The desire is that the RA-EIA acts as a consultation service for the underlying system, which only needs to be periodically queried for a reliable improvement in overall performance.

VI. Instantiation for PDP

This section describes the instantiation of the *risk-aware efficiency improvement advisor* (RA-EIA) for an application domain. We will first introduce the application domain of *pickup and delivery problems* (PDP) as an instance of dynamic task fulfillment. The underlying advised system solves the PDP problem online using *digital infochemical coordination* (DIC) [15] as its basic decentralized coordination model. Next, we describe the same instantiation for the original *efficiency improvement advisor* (EIA) with *ignore rules* from [26] and *pro-active rules* described in [27].

Finally, we address the additions required for a *risk-aware efficiency improvement advisor* for this application domain.

A. Pickup and Delivery Problems

The general problem class of *pickup and delivery problems* (PDP) [23] is well known. Formally, a task ta of a dynamic PDP consists of a pickup location l_{pickup}, a delivery location $l_{delivery}$, and the needed transportation capacity $ncap$. Each agent has a corresponding limit of transportation capacity cap_{Ag}, which dictates the quantity of tasks that can be undertaken at a time.

The majority of real-world PDP problems are in practice dynamic problems [11]. There are agents tasked with moving goods from a pickup location to a delivery location. Often, there is a certain subset of these tasks that occur regularly allowing the opportunity for an EIA to improve the operation of the system. *Ignore exception rules* advise individual agents to ignore a task identifier, while *pro-active exception rules* advise individual agents to begin the action of moving to a pickup location in expectation of a task's appearance.

The measure of system operational efficiency (i.e. $qual$) can be chosen from a number of possibilities. For this paper, $qual$ is defined as the sum of the total distance traveled by all agents in the system during the given run instance. This definition of $qual$ is the same as in [13] and [27], which allows us to use the testing approach they developed for the experimental evaluations.

B. Using EIA for DIC for PDP

The underlying self-organizing system uses *digital infochemical coordination* (DIC) implemented within a simulation tool. Decentralized coordination is achieved for agents in A using digital infochemicals that are propagated through Env. An agent Ag_i accesses all the infochemicals at its current location and bases its decisions purely on this local view of the environment. An agent Ag_i determines an action by choosing the highest utility computed for task synomones it perceives. The utility is influenced by the intensity of an infochemical as well as by the agent's current status (i.e. current undertaken tasks and advised rules). When an Ag_i selects a task it follows the infochemical gradient to find it.

Adding an EIA advisor to this base system is not very difficult. By collecting each agent's local history of infochemicals using **receive**, the EIA can compile the global history (the past 20 run instances) using **transform**, and extract the recurring set of tasks using **extract**. This extraction uses a similarity measure sim where tasks occurring at the same location and time are similar, with a threshold value $minocc = 70\%$, and the limited period $k = 20$. The **optimize** action uses an or-tree-based search to make sure that the genetic operators only create solutions that fulfill the hard constraints of the PDP (this is the same method used in [13] and [27]).

The original instantiation of **derive** for PDP makes use of the fact that the utility evaluation performed by an agent is very similar to a rule-based system. Each infochemical comes with a "rule" describing its contribution to the utility of the task it was created for. For *ignore rules*, ignoring a particular task can be achieved by setting its utility to 0. This also automatically achieves that the agent will go after the next-best task. For this paper, we only consider *task-triggered pro-active rules*. Such a rule is triggered by a determined previous recurring task ta_{trig} and directs an agent to the pickup location for a target task ta_{targ} that the EIA wants it to perform. The rule derivator priorities for the EIA's **derive** action first consider the *ignore rule* derivator and then the *pro-active rule* derivator.

C. Using RA-EIA for PDP

The $\textbf{derive}_{RA-EIA}$ action uses the same two rule derivators, one for *ignore rules* and one for *pro-active rules*. To determine the performance of the system in the **assess** action, the RA-EIA translates the internal representation of a run instance into a configuration and simulates the performance of the system. For the **assess** action, performance is simulated without any rules and with the rules under consideration. The simulation of the new recurring emergent solution sol^{rec}_{emg} is done with the accumulated set of rules $rules_{acc}$.

VII. Experimental Evaluation

To compare and contrast the new risk-aware RA-EIA concept with the original EIA concept, we report the results of several experiments. The first set of experiments uses the exploratory testing method introduced in [13] and the fitness functions from [27]. This exploratory testing method will be described in the following Section VII-A. This testing approach produces fitness function biased run instance examples, examined in Section VII-B, by which the two advisor variants may be differentiated. The second set of experiments, in Section VII-C, is based on randomly created run instances that provide an unbiased average performance comparison.

A. Automated Testing of Self-Adaptive Systems

The task of exploring the possibilities associated with a particular self-organizing emergent system is challenging. Naturally, it is common to design such a system with the assistance of tests created by developers interacting with field experts. The difficulty is that such systems are purposefully designed to be flexible and therefore capable of solving any problem within their defined application, including those not anticipated by the designers. Relying on a system architect to anticipate and formulate a set of tests, which can be trusted to fully explore both the risks and potential of a self-organizing system, is asking a lot. [13] presented an automated exploratory testing method based on machine learning to discover such tests.

The general idea is evolutionary learning of event sequences that fulfill a user specified fitness function. This fitness function quantifies the quality of behavior found in the self-organizing emergent system. An initial set of randomly created run instances (25 in our experimental evaluation) are evolved over a number of generations (100 in our experiments). A run instance is evaluated by translating the genetic algorithm's internal representation into a configuration for the simulation tool, simulating the self-organizing emergent system's response and applying the fitness function. In general, the fitness function compares the quality of the emergent (actual) system's solution before $qual(sol_{emg}^{bef})$ and after $qual(sol_{emg}^{aft})$ the advisor has adapted the system. This value is then normalized against the quality of the optimal solution $qual(sol_{opt})$ to the run instance. The manner in which this is done is specific to each fitness function.

Evolutionary operators are applied to create subsequent generations from the previous. The operators consist of: single-point-mutation which changes either the task or time of a single task-time pair in a parent run instance and crossover choosing task-time pairs randomly from two parent run instances to form a child. These parents are selected using tournament selection, which is biased towards more fit run instances.

In the first experimental set, we produced run instances to compare and contrast the potential and risks of the RA-EIA and EIA concept. Described below are the different fitness functions (first seen in [27]) required for each type of run instance desired. Note, all components of the fitness functions (*theo*, *pract*, *adapt*) are limited to positive values with negative values simply treated as 0.

Advisor's Potential Fitness Function fit_{pot}

$$\frac{5 \cdot theo + 500 \cdot adapt^{+}}{opt} \tag{3}$$

where $theo = qual(sol_{emg}^{bef}) - qual(sol_{opt})$ indicates how much theoretical improvement is possible, $adapt^{+} = qual(sol_{emg}^{bef}) - qual(sol_{emg}^{aft})$ indicates how much improvement the advisor achieved, and $opt = qual(sol_{opt})$ normalizes the result by the optimal. By maximizing this fitness measure we get individuals that show the potential improvement offered by each advisor variant.

Advisor's Risk Fitness Function fit_{risk}

$$\frac{5 \cdot theo + 10 \cdot pract + 500 \cdot adapt^{-}}{opt} \tag{4}$$

where $theo$ and opt are as above, $pract = qual(sol_{emg}^{aft}) - qual(sol_{opt})$ indicates how much improvement remained after adaptation, and $adapt^{+} = qual(sol_{emg}^{aft}) - qual(sol_{emg}^{bef})$ indicates how much of a negative impact the advisor had. By maximizing this fitness measure, we get individuals that show the risks that exist with each advisor variant.

B. Evaluating Potential and Risks

The experimental results in this section represent the comparison of run instance examples produced by a biased exploratory search. These run instances are designed to reveal the potential and risks of the tested system. The environment for the agents was a 10×10 grid with the depot in the middle. These experiments are based on 2 agents encountering problems consisting of 4 recurring tasks. The advised system is capable of higher complexities then this but the genetic algorithm exploratory search method must simulate the advised system's execution once for each individual. Consequently, to perform multiple exploratory test runs the complexity of the search space was limited. However, the produced results are sufficient to expose the differences and similarities between the two advisors.

We produced 5 experimental runs for each advisor with each fitness function. The results reported in Table I compare the potential and Table II the risks of the advised system. The top 5 rows are produced by searching for results for the EIA and then simulated after for the RA-EIA. The bottom 5 are produced from the reverse application of this process. The three columns Baseline, EIA, RA-EIA report the distance traveled without advice, with EIA advice, and with RA-EIA advice. Additionally, the number of rules created by each of the EIA and RA-EIA is reported.

Table I reveals that there exists run instances where each advisor has high potential for a large improvement in performance. It also shows that, for each advisor, the opposing variant is relatively as capable at improving performance. This is expected since the same tools (*exception rules*) are shared between the two variants. In the examples produced, both advisors in fact change the system's emergent solution into the optimal through exception rule advice. The rule columns show that the EIA generally requires more rules to achieve the same result as the RA-EIA.

The EIA, by providing a more restrictive set of rules, is able to gain a slightly better increase in performance through unintentionally preventing partial agent responses to tasks they do not complete. These partial responses are emergent behaviors that are not anticipated in the $qual$ measure. The original EIA concept is able to prevent more unanticipated behavior through a stricter set of rules at the cost of flexibility. The results in the random experimental set in Section VII-C will demonstrate that this slight advantage disappears when non-recurring tasks are a factor. On the other hand, the RA-EIA provides a limited set of effective rules that improve the system performance to relatively the same level.

Table II demonstrates the importance of the risk-aware nature of the RA-EIA. The run instance examples for the EIA show that it is more than capable of negatively impacting the base system's existing performance even when all the tasks are recurring. *In contrast, the testing method was unable*

Table I
RESULTS FOR EXPLORATION OF ADVISOR POTENTIAL

Test		Total Distance Traveled			Rules	
Advisor	Series	Baseline	EIA	RA-EIA	EIA	RA-EIA
EIA	1	61.36	21.31	21.31	5	4
	2	97.78	30.38	33.21	5	4
	3	71.15	21.56	24.14	6	4
	4	76.91	25.56	28.38	5	4
	5	104.95	30.97	44.28	4	2
RA-EIA	1	91.40	32.97	35.80	5	4
	2	82.91	31.90	33.90	5	4
	3	83.98	28.97	30.97	4	3
	4	65.84	24.38	24.38	5	5
	5	59.36	17.07	19.90	4	3

Table II
RESULTS FOR EXPLORATION OF ADVISOR RISK

Test		Total Distance Traveled			Rules	
Advisor	Series	Baseline	EIA	RA-EIA	EIA	RA-EIA
EIA	1	33.31	49.90	33.31	5	0
	2	33.46	50.43	33.46	3	0
	3	43.80	74.77	43.80	4	0
	4	44.63	68.43	44.63	1	0
	5	29.56	56.18	27.56	3	1
RA-EIA	1	46.63	17.07	41.80	6	1
	2	64.38	52.63	55.56	5	1
	3	71.74	71.74	71.74	0	0
	4	48.28	46.63	48.28	5	0
	5	57.70	27.31	55.46	4	1

to produce a single example, given the limitation of only recurring tasks, in which the RA-EIA reduced the system's performance. Resultantly, the RA-EIA examples exhibit no bias towards any specific behavior.

The original EIA is short-sighted and naive by providing rules, without assessing their potential or risk, until it can force the system's emergent solution into the optimal. This means it is capable of unintentionally decreasing the system's efficiency. The new risk-aware RA-EIA first assesses the long-term risks of rules it provides, eventually advising an effective, yet risk-averse, set of rules. By providing fewer rules the RA-EIA is sometimes unable to prevent unanticipated emergent behaviors but generally provides the same improvement. The risk-aware RA-EIA retains more flexibility than the EIA and, through strategic risk analysis, manages the dangers of non-recurring tasks.

C. Evaluating Random Instances

The experiments in this section address the evaluation of average performance for the two advisor variants. Similar to [27], we now consider run sequences that consist of both recurring and non-recurring tasks. Table III and IV report the average, min and max efficiency improvement of 50 advised randomly created run sequences compared to the base system. Each run sequence consists of 30 run instances. The different columns represent different numbers of tasks per run instance. A column title $xRyN$ means that we have x recurring tasks and y non-recurring tasks in each run instance of the run sequences for that column. As in the previous section, there are 2 agents on a 10×10 grid with a depot in the middle. The $(+/-)$ allows a quick relative comparison between the EIA and RA-EIA values. In Table III, the advisor can expect future non-recurring tasks since they are contained in the global history. In Table IV, non-recurring tasks are completely random so the advisor has no chance to anticipate the future beyond the recurring tasks.

In both experimental series, the EIA and RA-EIA show that they are quite capable of improving the average base system performance. Additionally, the RA-EIA is consistently the better of the two (apart from the $4R0N$ series) at improving the average overall performance of the base system's response. This gain is primarily due to the RA-EIA's ability to avoid rules that result in the loss of performance while providing rules that increase performance and its ability to advise a complete set of rules after a single consultation rather than incrementally deriving them. *In fact, the risk-aware RA-EIA avoids any overall decrease in performance over a run sequence.* While for the original EIA concept, almost every experimental series produced an example of a run sequence where the overall performance was decreased compared to without advice.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the concept of a dependable *risk-aware efficiency improvement advisor* for self-organizing emergent systems. Through offline Monte Carlo simulation, the long-term potential and risks of prospective rules are assessed. Our experimental evaluation, for the domain of dynamic *pickup and delivery problems*, showed that the result is a minimal, yet effective, set of risk-averse exception rules. These rules can be provided to individual agents so that the underlying system is capable of extended periods of reliable independent online operation.

The experiments reveal, if the recurring tasks remain stable, then the risk-aware advisor is capable of providing a reliable overall improvement in performance. Currently, the possibility of utilizing the exploratory testing method internally by the RA-EIA to assess the severity of the extent of risk represented by a prospective ruleset is being examined. Another area of interest is determining how to manage advice in order to achieve graceful degradation if/when the advisor is removed. Another direction of future work, as suggested in [16], will be examining the application of the different types of exception rules to other coordination methods.

REFERENCES

[1] S. Abdelwahed and N. Kandasamy, "A Control-Based Approach to Autonomic Performance Management in Computing Systems", in *Autonomic Computing: Concepts, Infrastructure, and Applications*, M. Parashar and S. Hariri, Eds. CRC Press, 2006, pp. 149–167.

[2] A.H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1988.

[3] A. Brintrup et al., "Distributed Control of Emergence: Local and Global Anti-Component Strategies in Particle Swarms and Ant Colonies", in *Proc. SASO'09*, San Francisco, 2009, pp. 216–222.

Table III
RESULTS FOR EVALUATION OF RANDOM WITH EXPECTED NON-RECURRING

		Test Series											
Advisor	Metric	4R0N	4R1N	4R2N	6R0N	6R1N	6R2N	6R3N	8R0N	8R1N	8R2N	8R3N	8R4N
EIA	avg	19.22%	12.47%	10.50%	16.24%	12.69%	7.07%	6.71%	12.04%	9.72%	5.49%	4.77%	3.27%
	max	35.17%	28.06%	23.10%	35.71%	31.80%	19.39%	20.48%	25.69%	30.96%	14.65%	14.22%	11.21%
	min	−14.31%	−1.68%	0.95%	−16.55%	−5.37%	−0.54%	−1.11%	−3.68%	−3.71%	−5.09%	−1.77%	−1.42%
RA-EIA	avg	16.63%	15.59%	13.97%	19.03%	17.78%	13.16%	11.24%	15.88%	16.52%	11.82%	9.87%	8.35%
	max	36.47%	33.82%	30.09%	39.78%	38.66%	23.28%	24.80%	29.88%	40.88%	21.23%	22.33%	20.27%
	min	0.00%	2.26%	0.30%	0.00%	5.05%	2.43%	0.95%	0.00%	2.23%	1.35%	1.01%	1.61%
(+/-)	avg	−2.59%	+3.13%	+3.47%	+2.79%	+5.09%	+6.09%	+4.54%	+3.84%	+6.80%	+6.34%	+5.11%	+5.08%
	max	+1.30%	+5.76%	+6.99%	+4.07%	+6.86%	+3.88%	+4.32%	+4.19%	+9.92%	+6.58%	+8.11%	+9.06%
	min	+14.31%	+3.94%	−0.65%	+16.55%	+10.42%	+2.97%	+2.06%	+3.68%	+5.94%	+6.44%	+2.78%	+3.03%

Table IV
RESULTS FOR EVALUATION OF RANDOM WITH UNEXPECTED NON-RECURRING

		Test Series											
Advisor	Metric	4R0N	4R1N	4R2N	6R0N	6R1N	6R2N	6R3N	8R0N	8R1N	8R2N	8R3N	8R4N
EIA	avg	18.26%	13.40%	12.57%	13.24%	12.97%	10.37%	10.14%	12.74%	9.88%	7.87%	7.67%	6.01%
	max	39.25%	30.45%	22.43%	39.77%	29.15%	21.29%	19.77%	29.32%	29.30%	17.51%	16.91%	14.39%
	min	−10.91%	0.33%	1.93%	−6.20%	−1.75%	−0.36%	2.13%	−6.47%	−0.91%	−3.86%	−7.59%	−1.55%
RA-EIA	avg	16.81%	15.99%	13.93%	17.46%	17.26%	13.94%	12.86%	17.34%	15.75%	12.73%	11.39%	8.89%
	max	41.91%	34.37%	25.09%	43.80%	32.78%	28.69%	23.61%	32.93%	33.56%	21.80%	24.74%	20.63%
	min	0.00%	4.22%	2.84%	0.00%	3.94%	3.43%	2.35%	0.00%	0.45%	4.09%	0.70%	1.98%
(+/-)	avg	−1.45%	+2.59%	+1.36%	+4.22%	+4.28%	+3.57%	+2.72%	+4.60%	+5.87%	+4.86%	+3.71%	+2.88%
	max	+2.66%	+3.92%	+2.65%	+4.03%	+3.63%	+7.40%	+3.84%	+3.61%	+4.26%	+4.29%	+7.83%	+6.24%
	min	+10.91%	+3.89%	+0.91%	+6.20%	+5.69%	+3.80%	+0.22%	+6.47%	+1.36%	+7.95%	+8.29%	+3.53%

[4] E. Camacho and C. Bordons, *Model Predictive Control*, Springer, 2004.

[5] D. Coelho, M. Roisenberg, P. Filho, and C. Jacinto, "Risk Assessment of Drilling and Completion Operations in Petroleum Wells using a Monte Carlo and a Neural Network Approach", in *Proc. WSC'05*, Orlando, 2005, pp. 1892–1897.

[6] S. Correa and R. Cerqueira, "Statistical Approaches to Predicting and Diagnosing Performance Problems in Component-Based Distributed Systems: An Experimental Evaluation", in *Proc. SASO'10*, Budapest, 2010, pp. 21–30.

[7] S. Dobson et al., "A Survey of Autonomic Communications", *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 2, pp. 223–259, 2006.

[8] M. Dorigo, V. Maniezzo, and A .Colorni. "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[9] G.E. Evans and B. Jones, "The Application of Monte Carlo Simulation in Finance, Economics and Operations Management", *Computer Science and Information Engineering, WRI World Congress on*, vol. 4, pp. 379–383, 2009.

[10] I.A. Ferguson, "Integrated Control and Coordinated Behavior: A Case for Agent Models", in *Proc. ECIA'94*, Amsterdam, 1995, pp. 203–218.

[11] T. Flatberg, G. Hasle, O .Kloster, E.J. Nilssen, and A. Riise, "Dynamic and Stochastic Vehicle Routing in Practice", in *Dynamic Fleet Management*, Springer US, 2007, pp. 41–63.

[12] J.A. Hartigan, *Clustering Algorithms*, Wiley, 1975.

[13] J. Hudson, J. Denzinger, H. Kasinger, and B. Bauer, "Efficiency Testing of Self-adapting Systems by Learning of Event Sequences", in *Proc. ADAPTIVE'10*, Lisbon, 2010, pp. 200–205.

[14] IBM, "Autonomic Computing Whitepaper: An Architectural Blueprint for Autonomic Computing", June 2006.

[15] H. Kasinger, B. Bauer, and J. Denzinger, "Design Pattern for Self-Organizing Emergent Systems Based on Digital Infochemicals", in *Proc. EASe'09*, San Francisco, 2009, pp. 45–55.

[16] H. Kasinger, B. Bauer, J. Denzinger, and T. Holvoet, "Adapting Environment-Mediated Self-Organizing Emergent Systems by Exception Rules", in *Proc. SOAR'10*, Washington, 2010, pp. 35–42.

[17] J. Koźlak, J.-C. Créput, V. Hilaire, and A. Koukam, "Multi-agent Approach to Dynamic Pick-up and Delivery Problem with Uncertain Knowledge about Future Transport Demands", *Fundamenta Informaticae*, vol. 71, no. 1, pp. 27–36, 2006.

[18] B. McCabe, "Monte Carlo Simulation for Schedule Risks", in *Proc. WSC'03*, New Orleans, 2003, pp. 1561–1565.

[19] M. Mes, M. Heijden, and A. Harten, "Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems", *European Journal of Operational Research*, vol. 181, no. 1, pp. 59–75, 2007.

[20] NIST/SEMATECH, "e-handbook of statistical methods", May 2011; http://www.itl.nist.gov/ div898/handbook/.

[21] M. Pechoucek and V. Marik,, "Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies", *Autonomous Agents and Multi-Agent Systems*, vol. 7, no. 3, pp. 1387–2532, 2008.

[22] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck, "Towards a generic observer/controller architecture for Organic Computing", in *Informatik'06*, Dresden, 2006, pp.112–119.

[23] M.W.P. Savelsbergh and M. Sol, "The General Pickup and Delivery Problem", *Transporation Science*, vol. 29, no. 1, pp. 17–29, 1995.

[24] R. Schumann, A.D. Lattner, and I.J. Timm, "Management-by-Exception - A Modern Approach to Managing Self-Organizing Systems", *In Communications of SIWN*, vol. 4, pp. 168–172, 2008.

[25] G.D.M. Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-Organisation and Emergence in MAS: An Overview", *Informatica*, vol. 30, no. 1, pp. 45–54, 2006.

[26] J.-P. Steghöfer, J. Denzinger, H. Kasinger, and B. Bauer, "Improving the Efficiency of Self-Organizing Emergent Systems by an Advisor", in *Proc. EASe'10*, Oxford, 2010, pp. 63–72.

[27] T. Steiner, J. Denzinger, H. Kasinger, and B. Bauer, "Proactive Advice to Improve the Efficiency of Self-Organizing Emergent System", in *Proc. EASe'11*, Las Vegas, 2011, pp. 97–106.

[28] D. Weyns et al., "Endogenous Versus Exogenous Self-Management", in *SEAMS'08*, Leipzig, 2008, pp. 41–48.