Regulated Autonomy: A Case Study

René Schumann, Andreas D. Lattner, Ingo J. Timm

Johann Wolfgang Goethe-Universität Frankfurt
Institute of Computer Science, Information Systems and Simulation
Robert-Mayer-Str. 10, 60325 Frankfurt am Main, Germany
[reschu|lattner|timm]@informatik.uni-frankfurt.de

Abstract. Actual control and decision support systems rely more and more on distributed software systems. A central aspect in the design of these systems is the selection of an appropriate level of local autonomy of these subsystems in the decision making process. Most of the research focuses on the regulation of these entities towards an integration of centralized control strategies or a strict local autonomy, so far. Both aspects are margins of a scale and it can be expected that for certain applications a balance between autonomy and regulations has to be found. In this article, we present first steps towards a dynamic regulation of autonomous decision taking in order to design these adaptive decision support systems that try to solve the conflict between local autonomy and global system performance.

Keywords: Balancing autonomy, multiagent simulation, manufacturing control

1 Introduction

In the last years, a trend towards decentralized designs of decision support and decision taking systems can be observed. This was motivated by the need for more flexible and reliable systems that can handle complex problems. Decentralization leads to more flexibility on the one hand (cf. [9]) but results in even higher complexity with regard to coordination of the different sub systems on the other hand.

The other extreme, a purely centralized decision making process or strictly regulated distributed entities performing a computational distributed but functionally centralized planning and control is nowadays usually not wanted due to organizational aspects. But on the other side it is well known that by only local decision taking the overall system's performance can become very bad.

Actually, what is practically needed is a strategy that on the one hand ensures an acceptable performance level of the overall system and on the other hand allows the decision taking entities a high degree of freedom. This results in an adjustable autonomy for the entities. That means that according to the actual situation the balance between regulations and local autonomy has to be found. Therefore, we propose a mechanism for the regulation of autonomy and show its potentials in this article. The main idea of our approach is that the actual system performance is constantly monitored by a supervision entity that is allowed to instruct the operating entities to perform according to global objectives. From a very abstract point of view this corresponds to a control process of the system performance.

As an evaluation example we use the domain of manufacturing control. Manufacturing control of complex products is a hard problem itself due to the huge combinatorial possibilities for setting up a job-shop schedule. Recent desires with respect to more flexible manufacturing, e.g., for mass customization or a more versatile range of products has led to decentralized control approaches.

In previous work, Schumann and Sauer presented a multiagent approach to decentralized decision making in manufacturing control in the context of job-shop scheduling problems [16]. Each shop follows its own strategy for processing the different jobs in its queue and each job makes its own independent choices for selecting the shops where it is processed. Both, shops and jobs are represented by agents.

In this paper, we introduce a technique to ensure an overall system performance. This can be seen as the definition of quality of service for a distributed system. Within the manufacturing control this function corresponds to the role of the shop's foreman. If the overall system tends to go in a state where it will show a bad overall performance, the foreman is allowed to ask the local decision taking entities, here the shops, to work according to an overall good strategy that might be evaluated bad according to the local objective function.

The paper is structured as follows: In section 2 we present related work - especially about properties and levels of autonomy. The basic theory for the strategic management is described in section 3. The settings of our job shop scenario and the experimental results are shown in sections 4 and 5. The paper closes with the conclusion in section 6.

2 Related Work

Considerable progress has been made in the interdisciplinary research on autonomy (cf. German Priority Research Program on "Socionics" funded by the DFG, [12]). Referring to Weiss et al., autonomy has been identified as "enabler for emerging information processing paradigms" which has lead to "a more general interest in autonomy as a software property" [21, p. 1].

2.1 Properties of Autonomy

There are multiple approaches to define autonomy in Distributed Artificial Intelligence (DAI). Looking from outside, a system is autonomous if it is acting non-deterministically, i.e., the system acts differently in two identical situations. The appearance of non-determinism arises from the limited view on the environmental state (situation). If the internal state of the system is included, an autonomous system might also be deterministic. A better approach to define autonomy (an autonomous system) is the consideration of properties. In this context, autonomy (an autonomous system) is best described by the three properties: pro-activity, interaction, and emergence (cf. [18]):

- Pro-activity: The decision of a subsystem (actor) is not only performed on the basis of hard-wired input-output schemes. Furthermore, the actor is capable of

interpreting the environment. Doing so, the system activates goals or initiates actions without specific external events, respectively. Therefore, the actors require the ability to reason about their goals and the current situation, i.e., an explicit representation of goals and environment is required.

- Interaction: The autonomous system is capable of interacting with its environment. This includes the perception of and the interaction with the environment as well as the communication with other autonomous systems. The actor should increase the individual utility as well as indirectly the utility of the overall system. In the latter case, the agent can get the information by interacting with other agents. A fundamental assumption for interaction is that there is some "slack" in the coordination or interaction of autonomous agents, respectively.
- Emergence: The elements introduced so far, pro-activity and interaction, are properties of autonomous systems, which have to be implemented within an actor. If there are more than one autonomous subsystems building a larger system, the larger system should contain properties, which emerge from the interaction of locally or pro-actively acting subsystems, respectively. The naive formulation of this fundamental assumption is that the system is more than the sum of these parts.

Emergent behavior became one of the most important criteria for the valuation of autonomous software systems like multiagent systems from the very beginning in agent research. Different views are provided by researchers in this area: Axelrod, for instance, emphasizes emergent properties as consequence of simple forms of interaction [1, p.4]. The organization within a multiagent systems is focused within the work of Ferber who introduced the term emergent organization [7, p. 144] and Wooldridge discusses emergence for enabling intelligent behavior [22, p. 49]. A closer look at all three aspects shows that interaction is needed for achieving global structures by local interaction, dynamic organization by simple (communication) rules, and intelligent behavior of an autonomous system as a whole. The emergent behavior of an autonomous system should result from interaction of the autonomous subsystems. On the one hand the cooperation and coordination follows local optimization criteria (goals), and on the other hand it has to take a joint optimization criterion (goals of autonomous systems) into account. This collaboration should lead to a global optimum of the system.

2.2 Levels of Autonomy

In the literature, there are discussions on different levels of autonomy [6, 14]. In early multiagent research, Castelfranchi and Conte [4] discuss a very high degree of autonomy, such as the influence of predefined norms, behavior patterns, or procedures is irrelevant or the relevance is very low with respect to the action-selection-process within an agent. Nevertheless, autonomy is a property which may lead to partially unwanted system states resulting from conflicting or inconsistent goal sets. The dynamic and complex interdependencies of autonomous subsystems can lead to systems, whose organization emerges at runtime. Thus, software engineers of autonomous systems may not consider any possible constellation of subsystems at design time.

Timm [18] introduces four levels of autonomy (LoA): strong regulation, operational autonomy, tactical autonomy, strategic autonomy following a systematic approach using the levels of decision making known from economics and system theory: operations, tactics, and strategies [cf. 8,10]. Autonomous systems are situated in an environment with the capability to perceive and interact with it. The complexity of the deliberation process within an agent is strongly related to different environmental properties as proposed by Russell and Norvig [15]. This categorization is used to specify the levels of autonomy and to describe their corresponding adequacy in application domains. Table 1 summarizes the levels of autonomy with respect to the environment properties.

Tab. 1. Levels of autonomy and properties of environments.

Level of autonomy	Observable	Deterministic	Episodic	Static	Agents
Strong regulation	Fully	Deterministic	Episodic	Static	Single
Operational autonomy	Partial	Deterministic	Episodic	Static	Multi
Tactical autonomy	Partial	Stochastic	Episodic	Semi	Multi
Strategic autonomy	Partial	Stochastic	Sequential	Dynamic	Multi

Strongly regulated (LoA 0) systems are the class of systems usually dealt with in "traditional" software engineering. There is no part of the system with autonomous capabilities. Any decision – regardless of the decision level – is predefined or determined by an external entity. Conventional monolithic systems are examples for this class of systems. They proved to be effective in environments with limited complexity characterized by full observability, determinism and episodic structure without further autonomous systems and with a mainly static behavior.

The first step of increasing autonomy is associated with the operational level of decision making (*operational autonomy*, LoA 1). Here, an autonomous software system gains the competence to decide on an operational level with a specific "slack" in the behavior. However, this decision still follows the tactical and strategic boundaries of the system. In agent technology, the operational autonomy may be implemented using reactive agent architectures. In the BDI approach, operational autonomy means that there is no flexibility in the desires or the intention selection mechanism, i.e., the desires and intentions cannot be modified by the agent. However, the agent is capable to reflect or refine plans. An approach to operational autonomy on the basis of adaptive action plans is presented by Timm [17].

Software systems implementing operational autonomy are effective in environments which are partially observable, deterministic, episodic, and static. Obviously, multiagent environments do not prevent operational autonomy. The benefits of operational autonomy are that the underlying algorithms allow for an efficient implementation and immediate response to (episodic) disturbances. However, operational auton-

omy is restricted to reactions on the basis of short term episodes and therefore does not consider mid- or long-term objectives.

Tactical autonomy (LoA 2) extends operational autonomy with respect to the tactical level. Tactical decision making in autonomous software systems enables the system to deliberate on different alternatives for operational behavior. While in operational autonomy plans are under consideration, e.g., linearization of partial plans, tactical autonomy is performed at the level of goals or intentions, respectively. Considering BDI agents, the planning and execution level is associated with the operational autonomy. The deliberation step of a BDI agent, where an agent selects resp. creates an intention with respect to its desires and its current state, is the corresponding level for tactical decision-making. An approach to enable tactical autonomy on the basis of capability management can be found in [20].

Software systems incorporating tactical autonomy are able to cope with stochastic and semi-dynamic task domains in addition to the capabilities of operational autonomous systems. The limitations of tactical autonomous systems arise if the environment contains a sequential problem structure, i.e., the dynamics of the environment become part of the task.

The highest degree of autonomy is the *strategic autonomy* (LoA 3). The strategic aspects represent the highest level of decision making within a software system and are conventionally determined by the system's designer in advance or by external influence, e.g., the user, during runtime. The architecture of each individual agent incorporates static strategies, e.g., by the definition of individual desires and specific algorithms. In conventional BDI, strategic autonomy by the agent itself is not feasible as the desires are determined statically. The selection of desires for the option selection process is based on an accessibility relation, i.e., the agent's beliefs are used for computing a relation identifying those desires, which are accessible by action sequences starting with the current state. Timm introduces an approach for strategic autonomy as an extension to conventional BDI [18]. Here, the agents are enabled to compute interdependencies between desires and intentions dynamically with respect to the current state of the agent.

3 Strategic Management of Multiagent Systems

Multiagent systems consist of autonomous agents, which interact on a local – microscopic – level. Optimal macroscopic behavior does not necessarily emerge from those interactions. Microscopic optimization can lead to local optimal situations. However, a key benefit of multiagent systems is assumed to be a positive emergence on a system level (cf. [18]). In social science, the phenomena of microscopic-macroscopic interaction is widely researched (e.g., [2]). Norms and regulations are introduced in a social system to establish a better system performance.

This paper is based on research by Timm and Hillebrandt who address strategic management in multiagent systems [19]. The basic assumption is that global system goals are known and that it is possible to evaluate a situation to what extent the target setting is achieved, i.e., a system has the ability to reflect about its overall performance explicitly within negotiations. It is distinguished between three color codes:

target accomplished (“green”), target slightly failed (“yellow”), and target failed (“red”). The multiagent conceptualization of reflection is based on the assumption that a multiagent system was chosen deliberately as a system design. In consequence, autonomy of the agents is not a side effect but one of the key features.

Timm and Hillebrandt propose a process with four different phases in order to introduce strategic management. The phases – observation, analysis, recommendation, and institutionalization (Fig. 1) – are inspired by social mechanisms of reflection as described in Luhmann’s general theory of social systems (Luhmann uses the terms “basal self reference”, “reflexivity”, “reflection”, and “planned reflection”; [11]).

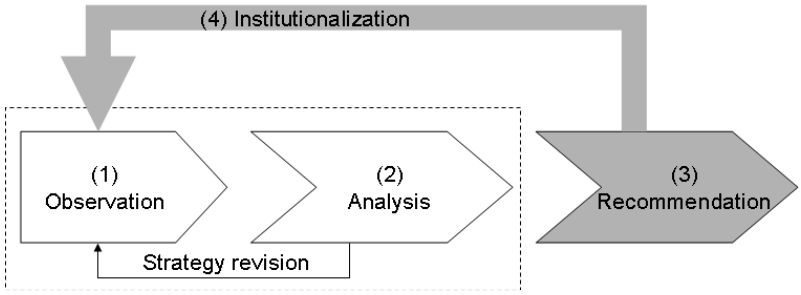

Fig. 1. Four stages of strategic management.

The approach is based on a group of agents with operational, tactical, and strategic autonomy as a core element. This group can be formed dynamically at runtime or specified at design time. In either way, it is assumed that the group of agents have some common goals and the fulfilment of this goal can be measured by the group of agents. Furthermore, for each goal, there are different levels of goal satisfaction, i.e., 0 implies that a goal is completely unsatisfied and 1 indicates that the goal has been satisfied. For utility-based goals a continuous scale is assumed while for logic-based goals the goal-satisfaction is a binary function. Furthermore, we assume that the consideration of global goals in every deliberation step would be inadequate with respect to computational or memory consumption.

In the observation stage, each agent reports its performance to a blackboard or central entity (group coordinator) within the group. The blackboard or the group coordinator computes the goal satisfaction on the basis of the individual results. Together with the concrete satisfaction level, the goal satisfaction is available for any agent of the group. If the goal-satisfaction is classified as deficient, the agents should adjust their operational autonomy. Doing so, the agents should plan the next action resp. action sequence under consideration of the global goal. E.g., assume that a BDI agent has instantiated an intention and associates this intention with a partial global plan. The agent would now choose a linearization of the plan, which is most suitable for supporting the global goal.

If the system performance with respect to a specific goal is critical (or cannot be achieved completely for some time, code “yellow”), the multiagent system's state changes to the analysis stage. In this stage, the agents have to communicate their currently pursued goals. The analysis is performed by the agents cooperatively or by a central entity or group manager, respectively. The group manager has to identify the

interdependencies of the goal selections of the individual agents and missing system performance on the group level. These interdependencies are then published. Under consideration of the autonomy of individual agents, the tactical autonomy has to be adjusted by the agents. Each agent should consider the effects of its goal instantiation, e.g., in our example the step of associating a plan to intentions, with respect to the group performance.

There are situations where an uncoordinated treatment of the mismatch of global goals by individual adaptations cannot lead to satisfying results. This can be the case especially if many agents adapt their behavior in a similar way which can lead to the over-achievement of one goal while the performance decreases w.r.t. other system goals. In the case of a severe system performance, the group of agents is transformed into the joint solution group. Here, the group manager mediates the negotiation about individual agents' goals. The agents are assumed to improve their strategic autonomy, i.e., the agents instantiate those goals which help the group performance.

The solution which has been negotiated in the group and which restored system's performance, is generalized as a social rule for later usage in severe situations (phase four), i.e., the experiences of phase three are made persistent for future situations and costly computation and communication can be avoided by handling similar situations in previous phases. For more details about this approach see [19].

In the case study of this paper, we focus on the first two phases. The basic idea is to provide each agent as much autonomy as possible, i.e., in default mode, each agent is free to select its behaviour as desired. Whenever the system performance reaches a critical state, phase two is initiated. In this phase, the manager agent instructs the shop agents to change their strategy in order to improve the system performance.

4 Scenario Settings

As mentioned before, we are looking here at scheduling scenarios in local production sites where several production units, transport and storage units have to be integrated. As a common, very simple, scenario we start with the scenario presented in [5].

4. 1 Shop Floor Layout

Numerous different job shop scenarios were presented in the literature. Since we do not want to increase this number, we chose an existing scenario from the literature. We decided to use the scenario presented in [5] as it is a quite simple scenario which eases the analysis. Figure 2 shows a schematic view of the scenario. We call the machines from the original example shops here because each machine can be seen as a local shop and therefore represented by its own agent. This allows us more general research in future work as well.

Each shop has an input and an output buffer. It offers exactly one operation. For simplicity reasons transportation is not modeled explicitly. It is assumed that always sufficient transport capacity is available and transportation time is zero. Other transportation services do not have to be considered, because they are out of scope for our investigation.

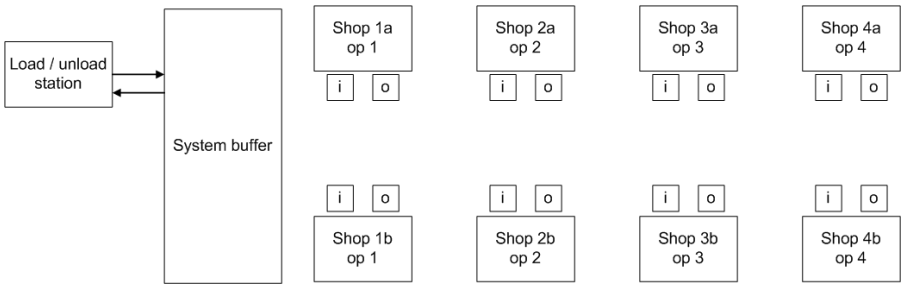

Fig. 2. Shop layout according to [5].

This scenario is used in [3] as well. The authors additionally describe the job characteristics they use. These job descriptions are presented in Table 2 and will be used here, too. By reusing the scenario data as far as possible we try to ground our research and try to focus on the implications of the regulation of autonomy as far as possible. This also provides us with the possibility to compare our results with previous ones.

Tab. 2. Process plan for different jobs, encoded as time/operation according to [3].

Step / job type	1	2	3	4
J1	6/1	8/2	13/3	5/4
J2	4/1	3/2	8/3	3/4
J3	3/4	6/2	15/1	4/3
J4	5/2	6/1	13/3	4/4
J5	5/1	3/2	8/4	4/3

There exist five different job types, which differ in their processing time for each operation and the sequence of operations needed to be done. By the design of the durations for each step it becomes clear that especially the resources Shop 3a and Shop 3b are bottlenecks.

4.2 Manufacturing Control Strategies

There exists a huge number of different dispatching strategies for shops. Here, only a subset is presented and used in the simulation studies. All these strategies are well known and are taken from [13]. They are presented in Table 3.

We decided to start with a representative subset of possible strategies. All these strategies need no or only the duration and priority information of the job. Thus, we restrict the scenario to use no generated release and due dates because we want to concentrate on the data presented in the literature. For the WSPT strategy priority information is needed. This priority is assigned to the job by the manager. In this strategy, the processing time is multiplied with the priority. The priority for a job is a uniformly distributed value in the interval [0,1].

Tab. 3. Dispatching strategies for shops.

Strategy Code	Description
SIRO	Service in random order
FIFO	First in first out
SPT	Shortest processing time first
LPT	Longest processing time first
WSPT	Weighted shortest processing time first

For the job routing different strategies can be found, as well. Examples of routing strategies might be shortest queue (SQ), shortest queue according to sum of durations (SQSD) or service in random order (SIRO). In [16] it could be found that in the given example the usage of different job routing strategies do not have significant effects. Moreover, as we actually want to study the regulated autonomy of the shops changing or different strategies for jobs could disturb the observations. Therefore, we fix the strategy for jobs in this research to the SQ strategy.

4.3 Manufacturing Scenario and Performance Measurement

The processing times are assumed to be deterministic and are always met. There exist no disturbing events, since we do not want to investigate the robustness of the used scheduling technique. That can be addressed in further research steps.

In a scenario with shops that can choose their dispatching strategy autonomously according to an internal objective, it is difficult to define global performance measures. Here, the global mean flow time of a job, from the starting of the first operation to the end of the last operation is used.

It has to be kept in mind that the shop floor system is not necessarily optimizing these objectives. This implies that the results may be worse than in a scenario with a centralized control. As already mentioned we did not aim to find the best solution, but to investigate a setting of regulated autonomy in a context that is well known.

4.4 Regulation Strategy

As already mentioned jobs and shops are represented by agents. Moreover there exists a supervising agent, which is called the manager agent. In previous work [16], this agent was only used for the initial strategy assignment to jobs and shops. For the purpose of autonomy adaption this agent has to be extended. If a job is finished, the corresponding job agent informs the manager and reports the flow time of this job. Actually, the manager agent can monitor the mean flow time according to the jobs finished so far. If this value falls below a specified threshold, the manager agent can order the shop agents to work to a specific strategy. If the actual mean flow time is in an acceptable range, it can allow the shops to work according to their locally preferred strategy.

4.5 Implementation Details

The simulation system is implemented as a multiagent system based on the java agent development framework JADE[1]. A time driven simulation was implemented to evaluate the different effects of local decision making in the manufacturing context. In this multiagent system three different agents exist.

- The manager agent assigns the initial strategy and does the runtime monitoring and can ask the shops to change their strategies.
- The job agent represents a single job. Once the strategy has been assigned the job routes itself autonomously through the shop floor.
- The shop agent represents a single shop in the shop floor. Each shop processes the jobs that have enqueued themselves to this shop. The sequence in which the jobs are processed is computed according to a dispatching strategy. Initially, this strategy is given by the manager.

In short, the autonomy of the single agent roles is as follows: The manager decides when the other agents use their own preferred or a centrally selected strategy. The job agent has the autonomy to select which shops to use for the different processing steps and the shop agent can select the strategy for handling the jobs.

5 Experimental Results

All tests were made using the above described scenario setting with initially 100 jobs (20 per job type) starting at the same time and all shops have an empty queue. All presented results are means computed by 10 runs of each experiment.

In our first experiments, all shops were fixed to one strategy. This allows comparison and gives an impression of the abilities of the existing approaches. The mean flow time can be computed after each job has finished. This temporal development of mean flow times is shown in figure 3. As expected the SPT strategy performs best, as it is known that this strategy is well suited to minimize the mean flow time. The FIFO strategy performs worst. This has already been observed in [16].

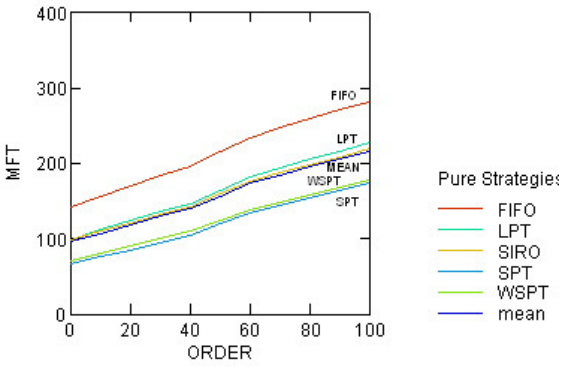

Fig. 3. Mean flow time during order completion

[1] http://jade.tilab.com/

In order to establish a control process as sketched above, threshold values have to be defined that allow to test if a mean value after a certain number of fixed jobs is acceptable or not. We decide to use the mean of all pure strategies.

Our further tests aim to investigate if an autonomy regulation strategy can ensure the overall performance to be kept in an acceptable range. Initially, in the next experiments the dispatching strategies are randomly assigned to the shops like in the random experiments. We compare this random assignment with and without adjustable autonomy. If autonomy can be adjusted the manager agents supervise the performance. If the actual performance drops below the average defined above, it orders the shop to use the SPT strategy. If the shops are following the central strategy and the actual performance is better than the adjustment threshold (i.e., the minimal acceptable one), the shops will be allowed to work according to their initially assigned strategy again. It can be seen in figure 4 that a random strategy assignment leads to mean flow times that are above (i.e. worse than) the threshold values. In our next experiments we examine two questions.

1. Can the adjustment of autonomy ensure that overall system performance remains acceptable?
2. What are good intervals for a test of system performance?

To evaluate the second question we design three experiments where we vary the interval when the actual performance is compared to the threshold values. We tested after a certain number of jobs informed the manager that they were done. We chose to test after 1, 3 and 10 jobs. These strategies are called CON_1, CON_3 and CON_10. If these strategies are compared combined to other strategies, these strategies are referred to as the reflexion strategies. The development of the mean flow times for 100 jobs is shown in figure 4. In figure 5 we show the mean values and the spread of the pure strategies and the reflexion strategies.

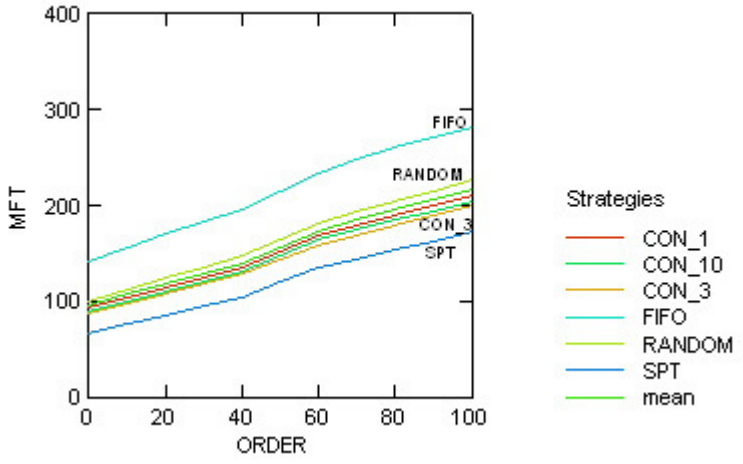

Fig. 4. Mean flow time (we show here the mean of 10 test runs).

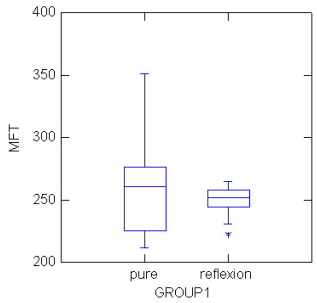

Fig. 5. Box Whistler diagrams for pure and reflexion strategies.

As very heterogeneously performing strategies are used, one can see in figure 5 that the spread of the pure strategies is very high. What is interesting about the pure values is the marked mean value that is the threshold for the reflexion strategies. The reflexion strategies can remain below this threshold with their entire interquartile range. Taking the mean flow time as an objective the reflexion strategies shows that they work very effective.

According to [16], in case of distributed decision taking the mean flow time is just one aspect that has to be observed. The stability of a solution or in other words the statistical spread becomes an important aspect. Therefore we perform a pairwise t-test between the SPT and reflexion strategies and between the reflexion strategies and the random assignment of strategies. We compare the random strategy assignment with the SPT as the best known solution technique, against the results of the reflected strategies. The results are shown in figure 6.

First we analyze the results of the t-test between SPT and reflexion strategies. The results are highly significant as probability of an error is close to 0% (p-value < 0.001). We have observed that the reflected solutions have a wider spread and perform not that good compared to the SPT strategy. The mean of the reflected strategies is 13.67% worse compared to the mean value of the SPT strategy. The statistical spread is obviously also not as good as the pure SPT strategy.

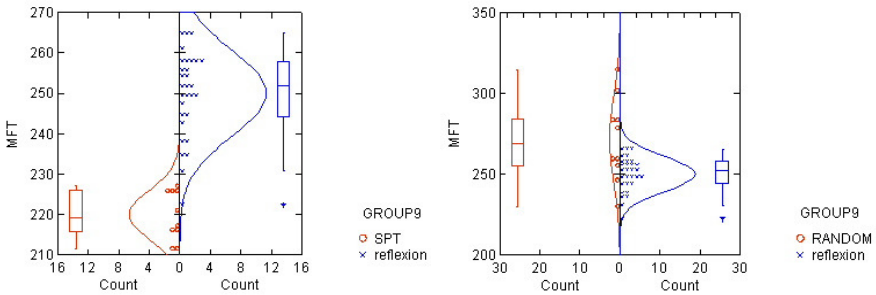

Fig. 6. Box Whistler diagrams for SPT and reflexion / random and reflexion strategies.

The second t-test between the random strategy assignment and the reflected strategies has a probability of error (p-value) below 3% that these results are significant. It can be shown that the reflected strategies show a better performance according to a lower mean flow time and in direct comparison the spread has been drastically reduced.

To investigate the potentials of the restriction of autonomy we run another experiment. Per default the FIFO strategy was assigned to all shops initially, so it can be expected that the overall performance is going to be drastically below the average. The manager agent controls the performance every third finished job. The results of these experiments are shown in figure 7.

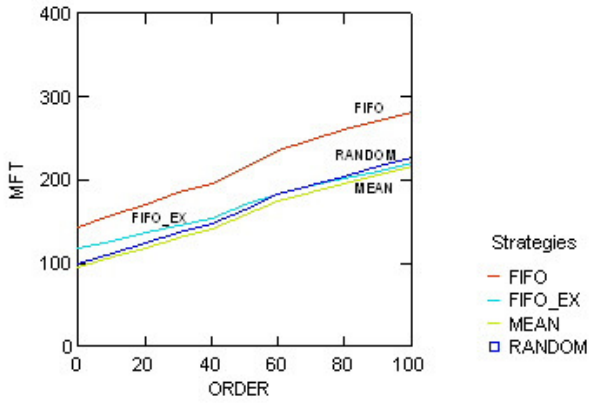

Fig. 7. Mean time for different strategies.

Without the adjustment of autonomy it was shown that the overall system performance is going far above the acceptable flow times. Actually, with the adjustment of autonomy it is possible to allow a performance slightly below a purely random assignment of strategies. But actually the flow times are below the threshold values. This shows that the given threshold values cannot work as a fixed bound for ensuring a quality level in a worst case scenario.

Summarizing these results it can be stated that the adjustment of autonomy can improve the overall system performance and allows producing results significantly better than the simple random assignment strategy on average. Our tests indicate that there can be intermediate states where the performance of a regulated test run can also be worse than a random assignment, but finally the performance is better. This affects the results because the local dispatching strategies are changed during the test runs.

It remains hard to find an answer to our second research question. If the three strategies CON_1, CON_3, and CON_10 are compared different expected and unexpected observations can be made. Our hypotheses are the following:

- Regularity of performance monitoring correlate positively with the mean flow time that means the more often tests against the threshold values are done the smaller mean flow times are expected.

- Regularity of performance monitoring correlate positively with the spread, that means the more often tests against the threshold values are done the smaller spreads of the mean flow time are expected.

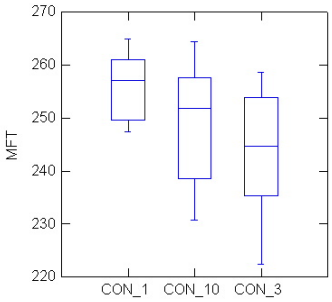

Fig. 8. Box Whistler diagram for the reflexion strategies.

Our results, presented in figure 8 show inconsistent aspects. What supports our hypothesis is that one can see a significantly smaller spread for CON_1 in comparison to the other strategies. The reduction of the mean flow time from CON_10 to CON_3 goes in line, too. But on the other side one can see that the mean flow time of CON_1 is higher than for CON_3 and that the spreads of CON_10 and CON_3 do not change significantly. Both observations contradict our hypothesis. This indicates that further research has to be done to find corresponding configuration settings for the adjustment autonomy mechanism.

6 Conclusion

In this paper, we have presented an approach to autonomy regulation. The basic theory behind this work is a strategic management inspired by mechanisms of reflection in social systems adapted to multiagent systems. The focus in this work has been set to the first two phases of strategic management. In a case study, we have shown that the adjustment of autonomy can lead to acceptable results retaining a certain level of overall system performance. This was done aiming at the full autonomy for the participating agents while the overall system performance is kept in acceptable boundaries.

Besides the consideration of the remaining phases of strategic management (recommendation and institutionalization) there are some aspects which have to be addressed in future work. It would be interesting to investigate the “cost of autonomy”, i.e., to compare the costs for the initially chosen (minimal cost) strategies with the costs of the revised strategy due to regulation at the local entities. As the hypothesis for the second research question has not been supported by some of the results, further investigation is needed in this regard. Another aspect that should be addressed by future work is the predictability of a guaranteed performance, i.e., to identify which thresholds lead to which performance values.

References

1. Axelrod, R. M.: The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton Studies in Complexity. Princeton University Press, Princeton, 1997
2. Bohman, J.: New Philosophy of Social Science – Problems of Indeterminancy, chapter The Macro-Micro Relation. The MIT Press, Cambridge 1993, pp. 146–185.
3. Brennan, R. W.; O, W.: A simulation test-bed to evaluate multi-agent control of manufacturing systems. In Proc. of the 32nd Conf. on Winter simulation, Orlando, Florida 2000, pp. 1747-1756.
4. Castelfranchi, C.; Conte, R.: Emergent functionality among intelligent systems: Cooperation within and without minds. Journal on Artificial Intelligence and Society, 6 (1992) :78–87.
5. Cavalieri, S.; Bongaerts, L.; Macchi, M.; Taisch, M.; Weyns, J.: A Benchmark Framework for Manufacturing Control. Paper presented at the Second Int.Workshop on Intelligent Manufacturing Systems, Leuven, Belgium 1999, pp. 225-236.
6. Falcone, R.; Castelfranchi, C.: Grounding autonomy adjustment on delegation and trust theory. Journal of Experimental and Theoretical Artificial Intelligence, 12(2000):149–151.
7. Ferber, J.: Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence. Addison-Wesley, Harlow, UK 1999.
8. Hentze, J.; Brose, P.; Kammel, A.: Unternehmensplanung. Bern, 2. Edition 1993.
9. Kirn, S.; Herzog, O.; Lockemann, P.C.; Spaniol, O. (Eds.): Multiagent Engineering. Theory and Applications in Enterprises. Springer: Berlin 2006.
10. Küpper, H.-U. : Controlling - Konzepte, Aufgaben und Instrumente. 2. Edition 1997.
11. Luhmann, N.: Soziale Systeme, Grundrisse einer allgemeinen Theorie. Suhrkamp, Frankfurt 1984.
12. Nickles, M.; Rovatsos, M.; Weiss, G.: A schema for specifying computational autonomy. In Proc. of the Third Int. Workshop "Engineering Societies in the Agents World" (ESAW 2002), Madrid, Spain 2002.
13. Pinedo, M.: Scheduling: Theory, Algorithms and Systems: Prentice-Hall 1995.
14. Rovatsos, M.; Weiss, G.: Autonomous software. In Chang, S. K., editor, Handbook of Software Engineering and Knowledge Engineering, volume 3: Recent Advances, River Edge, New Jersey. World Scientific Publishing 2005.
15. Russell, S. J.; Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, second edition, 2003.
16. Schumann, R.; Sauer, J.: Implications and Consequences of Mass Customization on Manufacturing Control. In Proc. of the 2007 Joint Conf. Mass Customization Meeting, Int. Conf. on Economic, Technical and Organisational Aspect of Production Configuration Systems, Hamburg, Germany, June, 2007, pp. 365- 378.
17. Timm, I. J.: Selbstlernprozesse in der Agentenkommunikation. In: Adaption und Lernen von und in Organisationen. VS Verlag für Sozialwissenschaften, Wiesbaden 2004, pp. 103–127.
18. Timm, I. J.: Strategic Management of Autonomous Software Systems. Technical Report. TZI-Bericht Nr. 35, Universität Bremen 2006.
19. Timm, I. J.; Hillebrandt, F.: Reflexion als sozialer Mechanismus zum strategischen Management autonomer Softwaresysteme. In: Reflexive soziale Mechanismen. VS Verlag für Sozialwissenschaften, Wiesbaden 2006.
20. Timm, I. J.; Woelk, P.-O.: Ontology-based capability management for distributed problem solving in the manufacturing domain. In Schillo, M., editor, Multiagent System Technologies - Proc. of the First German Conf., MATES 2003, pp. 168–179, Berlin 2003.
21. Weiss, G.; Fischer, F.; Nickles, M.; Rovatsos, M.: Operational modeling of agent autonomy: theoretical aspects and a formal language. In Proc. of the 6th Int. Workshop on Agent-Oriented Software Engineering (AOSE), Utrecht, The Netherlands 2005.
22. Wooldridge, M.: Intelligent agents. In Weiss, G. (Eds): Multiagent systems- a modern approach to distributed artificial intelligence, MIT Press, 1999, pages 27–78.