Cost of Control for Regulated Autonomy

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

Information Systems and Simulation
Johann Wolfgang Goethe-Universität Frankfurt am Main
Robert-Mayer-Str. 10, 60325 Frankfurt, Germany
{reschu,lattner,timm}@cs.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 our approach, 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. Giving up a locally optimal strategy for the collective with global objectives leads to some costs. In this paper, we investigate the measurement of such costs and present experimental results how often central control is used instead of autonomous decision taking.

Key words: balancing autonomy, multiagent simulation, manufacturing control, cost of control

1 Introduction

In the last years, a trend towards decentralized designs of decision support and decision taking systems can be observed. In contrast to the centralized systems, it is in practical cases not possible to compute optimal results according to a global objective function. Resulting plans and schedules will be suboptimal [16, 15]. Even so decentralized control systems are in the focus of actual discussion and research. The reason is that (cf. [7])

- systems with higher flexibility and reliability can be designed and
- decentralized control becomes part of actual company organization.

Additionally, it has to be stated that centralized decision taking is very complex and it becomes infeasible to compute effective decisions in a timely manner for big organizations like enterprises, or even significant parts of it.

A decentralized approach allows complexity reduction for decision taking, as the focus is limited to the scope of controlling the subsystem. So neither a purely centralized or decentralized system works satisfying in economic context.

Decentralization leads to more flexibility on the one hand 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 must be found. Therefore, we propose a regulation mechanism. 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 multi-agent approach to decentralized decision making in manufacturing control in the context of job-shop scheduling problems [15]. 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 as agents. In this paper, we present 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 shops' foreman. If the overall system tends to go in a state were 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. A first case study for this "regulated autonomy" can be found in Schumann et al. [14].

In this paper, we introduce the concept of "cost of control" into the setting. If the overall system's performance is below the regulation threshold and the shops are forced to follow the given (centralized) strategy, the performance of the single shops might become suboptimal from a local point of view. Having in mind that different shops with individual objectives optimize their performance, any modification will lead to identical or worse performance. In order to provide a measure for giving up on individual objectives for the collective, we introduce the "cost of control". The cost of control computes for each shop the resulting costs due to the replacement of the intended local strategy by a global one.

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

In the literature, there are discussions on different levels of autonomy [5, 12]. In early multiagent research, Castelfranchi and Conte [3] discuss a very high degree of autonomy, such as the influence of predefined norms, behavior patterns, or procedures is irrelevant, and the relevance is very low with respect to the action-selection-process within an agent, respectively. 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. Hentze et al. and Küpper [6, 8]). 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 [13]. This categorization is used to specify the levels of autonomy and to describe their respective adequacy in application domains. Table 1 summarizes the levels of autonomy with respect to the environment properties.

Level of autonomy	observable	deterministic	episodic	static	agents
Strong regulation	full	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

Table 1. Levels of autonomy and properties of environments

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 [11], 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 upon 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 autonomy 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 and 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 or 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 Timm and Woelk [19]. 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., if 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 [18] introduces an approach for strategic autonomy as an extension to conventional BDI. Here, the agents are enabled to dynamically compute interdependencies between desires and intentions with respect to the current state of the agent.

3 Regulated Autonomy

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. Timm [18]). In social science, the phenomena of microscopic-macroscopic interaction is widely researched e.g., Bohman [1]. 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 multi-agent systems [16]. 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"; [9]). 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 in 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.

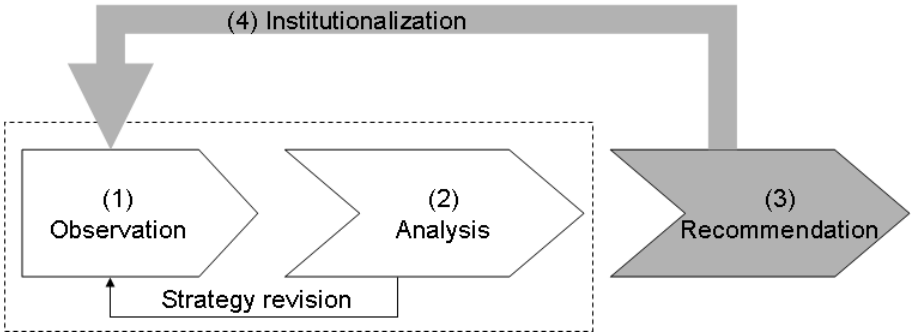


Fig. 1. Four stages of strategic management

In the observation stage, each agent reports its performance to a blackboard or central entity (group coordinator) within the group. The blackboard resp. 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 resp. group manager. 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 [16].

In 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. Whenever the strategy is changed, costs of the strategy adaptation is recorded.

4 Scenario Settings

As in our previous work we use the job shop scenario presented in [4] and [2]. This allows us to compare our actual results with our previous work. Therefore we sketch the scenario here briefly. Figure 2 presents only a schematic overview of the scenario. In the original paper by Cavalieri et al. [4] the processing entities were called machines. We call it 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 further work as well. Each shop has an input and an output

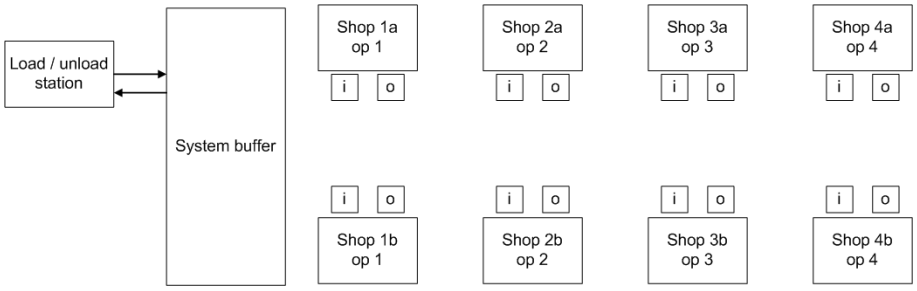


Fig. 2. Shop layout, according to [4]

buffer. It offers exactly one operation. For simplicity reasons transportation is not modeled explicitly. It is assumed that always enough transport capacity is available and transportation time is zero.

The job characteristics were taken from Brennan and O [2], were the scenario is used as well. In table 2 the duration and shop sequence are summarized. There exist five different job types, which differ in their processing time for each

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

Table 2. Process plan for different jobs, encoded as time/operation, according to Brennan and O [2]

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.

4.1 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 were taken from Pinedo [10]. They are presented in Table 3. All these strategies need no or only the duration and priority information of

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 SPT

Table 3. Dispatching strategies for shops

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]. 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 Schumann and Sauer [15], it was pointed out that in the given scenario 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. The strategies for the shops is assigned randomly using the strategies presented in 3.

4.2 Manufacturing Scenario and Performance Measurement

As we actually do not want to test the reactive scheduling abilities in our research it is assumed that no disturbing events occur and processing times are always met. 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 purely centralized control. But as already motivated it is not our goal to find the optimal solution but to investigate effects of regulated autonomy in a well known scenario.

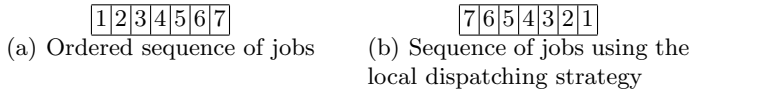
4.3 Regulation Strategy

As already mentioned jobs and shops are represented by agents. Moreover there exists a supervising agent, which is called the manager agent. If a job is finished, the corresponding job agent informs the manager and reports the flow time of this job. 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. Here this is the SPT strategy which is known to perform best in this scenario. If the actual mean flow time reaches an acceptable range again, it can allow the shops to work according to their locally preferred strategy.

4.4 Cost Model

The idea for the cost of control is that each shop dispatches the jobs in a sequence best for its local objective. If the manager orders the usage of a certain strategy, the resulting sequence is often not the best according to the local objective function. Thus, local costs are generated missing the local optima. These additional local costs are imposed by the centralized control, and are in focussed in this paper. As we use data from literature, it is not easy to come up with a practically motivated or realistic cost model. In consequence, we measure the difference between the schedules resulting from the usage of the different dispatching strategies. This is done by a comparison of the position of a dispatched job according to the different strategies. If a job is dispatched by an ordered strategy, this means that it is the first job given the actual set of jobs and the strategy. The position according to the originally used local dispatching strategy is computed for this job. The costs for this dispatching is defined as the difference of the positions in the different orderings. In other words the cost of control

are a metric for the deviation of the plans generated by the local best strategy and the ordered strategy. The following example should clarify the cost model. Assume the sequences of jobs is, as shown in figure 3. The number represents the different possible jobs that are stored in the queue waiting for processing by the shop.



(a) Ordered sequence of jobs

(b) Sequence of jobs using the local dispatching strategy

Fig. 3. Example: Cost of control computation, jobs are identified by numbers

In this case the difference for dispatching job no.7 is 6 as the absolute difference of the position of job no. 7 is 6 in both sequences. For each dispatching using the ordered strategy the costs are computed and summarized for each shop. The cost of control can be computed by summarize the cost incurred over all shops.

5 Evaluation

The results presented in this section were computed using a time driven simulation implemented as a multi-agent system based on the JAVA agent development framework JADE[1].

For evaluation purposes, we use three basic settings with different control cycles w.r.t. overall system's performance. In the `Con01` experiments, the current quality (mean flow time) is checked after each job. In the `Con03` and `Con10` settings, the control interval is set to three and ten, respectively. For each setting, ten different runs are performed where 100 jobs are generated and processed. Figure 4 footnoteAll statistical computations as well as plots have been generated with R Project for Statistical Computing 2.6.1, see `http://www.r-project.org/`. shows the average mean flow times for all three settings. The mean flow time is computed every time a job has been finished, i.e., the last value (job no. 100) indicates the mean flow time of all 100 jobs.

In each control cycle, it is tested if the current flow time does not exceed the threshold to switch to the strategy following the global objective. For each job processed during an interval of centralized control, the costs are computed as described above. In every run, the durations of centralized control and autonomy as well as the costs are computed. Figure 5 presents the Box-Whisker plots of the costs of the experiments of settings `Con01`, `Con03`, and `Con10`. Box-Whisker presents the upper and lower quantile and the median. Therefore they can be used to discuss the statistically spread of the data. As it can be seen, in our experiments the mean cost of the runs with setting `Con10` (control cycle only every ten jobs) the costs are lower than the costs in the other cases. The overall

[1] For the Java Agent DEvelopment Framework see `http://jade.tilab.com/`.

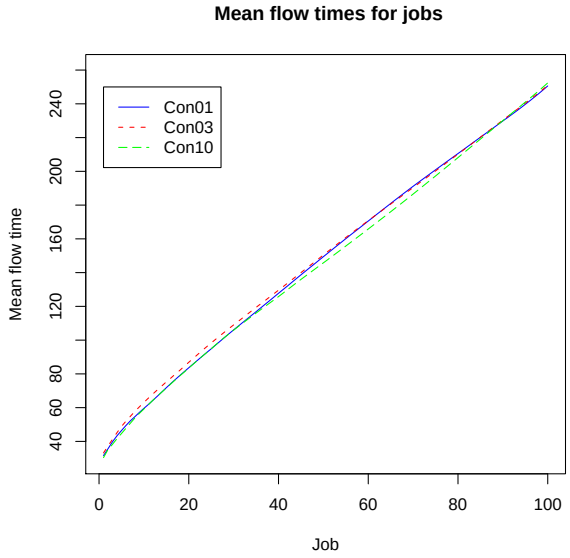


Fig. 4. Mean flow times

mean flow times of the three settings are 250.59, 250.946, and 252.378 for `Con01`, `Con03`, and `Con10`, respectively, i.e., the mean flow times are slightly lower for the shorter control cycles in the experiments (while having higher costs on average).

Figure 6 presents Box-Whisker plots of the relative central control time, i.e., the ratio of time interval lengths under central control divided by the total time. While the mean central control times of setting `Con01` and `Con03` do not differ a lot, the mean value of `Con10` is lower indicating that in our experiments central control is rather infrequent. Having in mind that these regulated strategies are capable to ensure an adequate level of the overall performance (see [14]) it can be stated, that this can be done restricting the local autonomy rarely. The median of th

In figure 7, the central control ratios of all three settings are related to the costs of the runs. The linear regression line is also shown in this figure. Pearson's product moment correlation[2] has shown significant correlations for all three settings (with $p < 0.001$ in all three cases) with positive correlation values of 0.877, 0.884, and 0.894.

[2] also known as the Bravais-Pearson correlation coefficient, which is commonly used to measure correlations

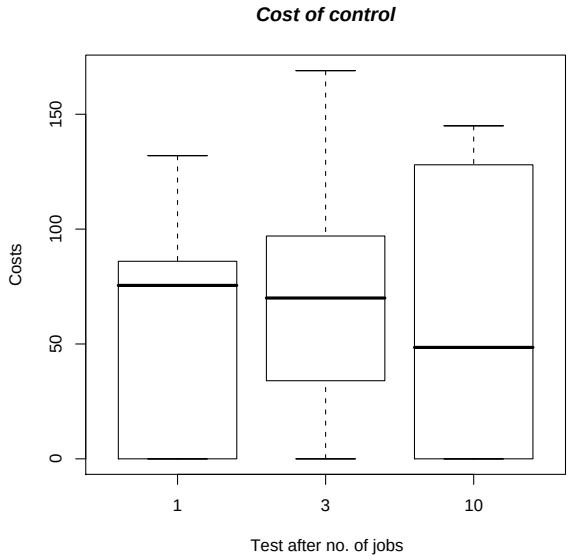


Fig. 5. Box-Whisker plot of costs

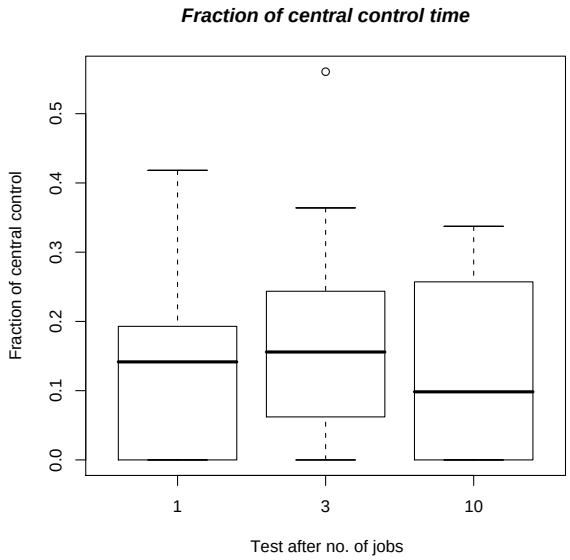


Fig. 6. Box-Whisker plot of relative central control times

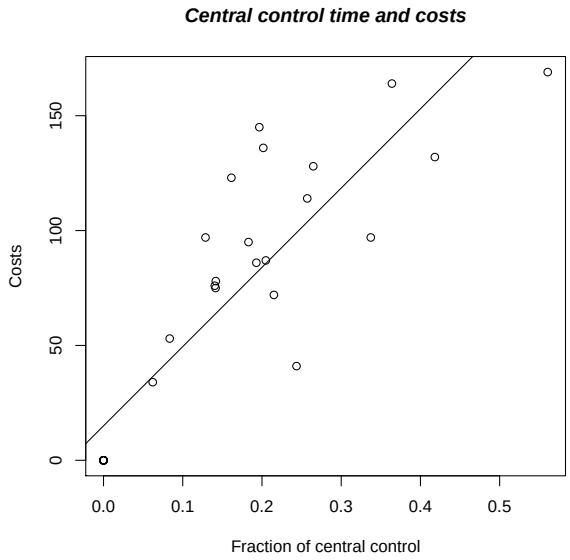


Fig. 7. Correlation of central control time and costs

6 Conclusion

In this paper we presented an approach to measure the cost of control in regulated autonomy. If different entities have to suppress local objectives for some time in order to help improving global performance, this usually affects their local performance. In this work, we have proposed to measure this cost of control. In our experimental evaluation, the difference in the order of jobs (centralized control strategy vs. individually preferred strategy) builds the basis for cost computation.

Interestingly, central control time and costs of control have been smaller with the third control cycle setting (`Con10`) in our experiments. This leads to the hypothesis that it can be actually disadvantageous to invoke control with too high frequency. Switching to centralized control immediately even if the threshold is only exceeded marginally can lead to costs while it seems that waiting for some more jobs can lead to some kind of "self-regulation" in the sense that the mean flow time will then be sufficient again. The data of the experiments shows a positive linear correlation between the ratio of the central control time and the emerging costs of a run.

In future work, we want to further investigate the influence of control frequencies to costs, and perform more experiments in order to have sufficient data for significance tests of different hypotheses. Additionally we are going to investigate different application scenarios for aspects of regulated autonomy. Typically

these scenarios emphasize local autonomy while it is an overall interest to avoid disastrous global states. One can think of application scenarios like the regulated electricity market, were decentralized producers can act autonomous as long as the power net stability is not affected. Another application scenario can be enterprise network, were no entity has interest that the networks breaks up, while is interested in maximize its own profit.

References

1. J. Bohman. *New Philosophy of Social Science - Problems of Indeterminancy*, chapter The Macro-Micro Relation, pages 146–185. The MIT Press, Cambridge, 1993.
2. R. W. Brennan and W. O. A simulation test-bed to evaluate multi-agent control of manufacturing systems. In *WSC '00: Proceedings of the 32nd Conference on Winter Simulation*, pages 1747–1756, Orlando, Florida, 2000. Society for Computer Simulation International.
3. C. Castelfranchi and R. Conte. Emergent functionality among intelligent systems: Cooperation within and without minds. *Journal on Artificial Intelligence and Society*, 6(1):78–87, 1992.
4. S. Cavalieri, L. Bongaerts, M. Macchi, M. Taisch, and J. Weyns. A benchmark framework for manufacturing control. In *Second International Workshop on Intelligent Manufacturing Systems*, pages 225 – 236, Leuven, Belgium, 1999.
5. R. Falcone and C. Castelfranchi. Grounding autonomy adjustment on delegation and trust theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(2):149–151, 2000.
6. J. Hentze, P. Brose, and A. Kammel. *Unternehmensplanung - Eine Einführung*. Bern, 2. edition, 1993.
7. S. Kirn, O. Herzog, P. Lockemann, and O. Spaniol. *Multiagent Engineering*. International Handbooks on Information Systems. Springer, Berlin et al, 2006.
8. H.-U. Küpper. *Controlling - Konzepte, Aufgaben und Instrumente*. Schäffer-Poeschel Verlag, 2. edition, 1997.
9. N. Luhmann. *Soziale Systeme, Grundrisse einer allgemeinen Theorie*. Suhrkamp, Frankfurt, 1984.
10. M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. Industrial and Systems Engineering. Prentice-Hall, 1995.
11. A. S. Rao and M. P. Georgeff. BDI Agents: From Theory to Practice. Technical Node 56, Australian Artificial Intelligence Institute, April 1995.
12. M. Rovatsos and G. Weiss. Autonomous software. In S. K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*, volume 3: Recent Advances, River Edge, New Jersey, 2005. World Scientific Publishing.
13. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey, 2003.
14. R. Schumann, A. D. Lattner, and I. J. Timm. Regulated autonomy: A case study. In *Proceedings of the Conference Track "Intelligente Systeme zur Entscheidungsunterstützung" at the Multikonferenz Wirtschaftsinformatik (MKWI 2008)*, Garching, Germany, 2008. SCS Publishing House. To appear.
15. R. Schumann and J. Sauer. Implications and consequences of mass customization on manufacturing control. In T. Blecker, K. Edwards, G. Friedrich, and F. Salvador, editors, *IMCM'07 + PETO'07*, volume 3 of *Series on Business Informatics and Application Systems*, pages 365 – 378, Hamburg, 2007. GITO.

16. I. Timm and F. Hillebrandt. Reflexion als sozialer Mechanismus zum strategischen Management autonomer Softwaresysteme. In M. Schmitt, F. Hillebrandt, and M. Florian, editors, *Reflexive soziale Mechanismen*. VS Verlag für Sozialwissenschaften, Wiesbaden, 2006.
17. I. J. Timm. Selbstlernprozesse in der Agentenkommunikation. In M. Florian and F. Hillebrandt, editors, *Adaption und Lernen von und in Organisationen*, pages 103–127. VS Verlag für Sozialwissenschaften, Wiesbaden, 2004.
18. I. J. Timm. Strategic management of autonomous software systems. Technical report, Universität Bremen, 2006. TZI Report No. 35.
19. I. J. Timm and P.-O. Woelk. Ontology-based capability management for distributed problem solving in the manufacturing domain. In M. Schillo, editor, *Multiagent System Technologies - Proceedings of the First German Conference, MATES 2003, September, Erfurt*, Lecture Notes in Artificial Intelligence (LNAI), pages 168–179, Berlin, 2003.