Flexibility in Organic Systems
Remarks on Mechanisms for Adapting System Goals at Runtime

Christian Becker[1], Jörg Hähner[2] and Sven Tomforde[2]
[1]*System and Computer Architecture Group, Leibniz University Hannover, Appelstr. 4, 30167 Hannover, Germany*
[2]*Lehrstuhl für Organic Computing, Augsburg University, Eichleitnerstr. 30, 86159 Augsburg, Germany*

Keywords: Organic Computing, Flexibility, Machine Learning, Goal Exchange, Network Protocol Configuration.

Abstract: Within the last decade, technical systems that are capable of self-adaptation at runtime emerged as challenging approach to cope with the increasing complexity and interconnectedness in today's development and management processes. One major aspect of these systems is their ability to learn appropriate responses for all kinds of possibly occurring situations. Learning requires a goal function given by the user – which is subject to modifications at runtime. In order to allow for flexible manipulations of goals within the system's operation period, the learning component must be able to keep knowledge in order to respond to varying goals quickly. This paper describes attempts to implementing flexible learning in rule-based systems. First results show that efficient approaches are possible even in real-world applications.

1 INTRODUCTION

In recent years, technical systems capable of adapting themselves to changing environmental conditions have gained increasing attention in industry and academia. Driven by the insight that current design approaches and the corresponding systems reach their limits, a new paradigm for design processes has been proposed. The main concept of this paradigm postulates to move parts of the control authority (e.g. for configuration aspects or the decision about most-promising responses) from design time to runtime. Hence, systems themselves become responsible for finding appropriate reactions for occurring situations, although these situations have not been anticipated by an engineer in the first place, cf. initiatives like Autonomic (AC, (Kephart and Chess, 2003)) or Organic Computing (OC, (Schmeck, 2005b)). Typically, this increasing degree of autonomy is achieved by introducing self-* properties (Schmeck, 2005a) and by enabling automated learning capabilities – which results in an increased adaptivity and a higher robustness of the system compared to standard solutions.

In previous work (Tomforde, 2012), we introduced a basic system design and framework to achieve these desired *organic* capabilities. The approach is based on using population-based machine learning techniques that limit the trial-and-error parts of automated learning and thereby match safety-restrictions of real-world systems. Usually, machine learning systems are configured at design time by providing a certain goal (e.g. by applying a mathematical function) which is to be approximated over time. Upcoming OC systems face the demand of providing a possibility to modify such goals at runtime as reaction to changing user needs. We refer to this concept as *flexibility*, cf. (Schmeck et al., 2010).

Rule-based learning typically keeps track of a performance estimation for each rule – the *fitness*. This fitness is calculated and updated in response to observed system behaviour and depends on the given goal. After defining the term *flexibility* (section 2), this paper discusses the need of novel mechanisms for flexibility in rule-based machine learning techniques and introduces three different concepts (section 3). Afterwards, these concepts are evaluated within an exemplary application from the OC domain (section 4). Section 5 summarised the paper and gives an outlook to future work.

2 TERM DEFINITION: FLEXIBILITY

Similar to various terms used in the context of self-organised and organic systems, *flexibility* has manifold meanings in different research domains. Corre-

Becker C., Hähner J. and Tomforde S..
Flexibility in Organic Systems - Remarks on Mechanisms for Adapting System Goals at Runtime.
DOI: 10.5220/0004121002870292
In *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2012), pages 287-292
ISBN: 978-989-8565-21-1
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

spondingly, it is measured according to the aspects that are specific to the particular (application) domain. For instance, various metrics are proposed to measure the flexibility of manufacturing systems (Benjaafar and Ramakrishnan, 1996; Hassanzadeh and Maier-Speredelozzi, 2007; Shuiabi et al., 2005), programming paradigms, architecture styles, and design patterns (Eden and Mens, 2006) or different reconfigurable hardware architectures (Compton, 2004). There is no common definition, but the meaning of the term as "adapting the system appropriately when the goal is changed" is commonly agreed.

At a technical perspective and in the context of OC, we typically rely on using Observer/Controller patterns as basic system design (Schmeck, 2005b). The most prominent variant is the *Multi-level Observer/Controller* (MLOC) framework (Tomforde, 2012) which relies on using Learning Classifier Systems (LCS) for online learning tasks. Even when initially testing LCS' in artificial scenarios like the *animat* exploring its environment (i.e. the *Woods* scenario (Wilson, 1994)), the possibility of different kinds of targets (here: artificial food) is already envisioned. In the context of this paper we will rely on the OC terminology which was initially explained in (Schmeck et al., 2010). Thereby, two concepts are distinguished:

1. A system is characterised by its state $z(t)$. If $z(t)$ changes due to a change of the system (e.g. broken components) or a change in the environmental conditions (disturbance δ) and the system continues to show an acceptable behaviour, this system is called a *robust system*.
2. In case of changes of the evaluation and acceptance criteria, the system's state spaces that define the targeted and accepted behaviour would be modified. A system that is able to cope with such changes in its behavioural specification is called a *flexible system*.

The first aspect is needed for most systems and especially focused by diverse initiatives and research domains like OC, AC, or Proactive Computing (Tennenhouse, 2000). In contrast, the second aspect is mostly left unregarded.

3 ORGANIC SYSTEMS AND FLEXIBILITY

This section discusses the basic system design for organic systems according to the *Multi-level Observer/Controller* (MLOC) framework. Based on this framework and the goal-related tasks within this framework, we discuss technical issues and the corresponding research problem for achieving flexibility.

3.1 System Design

The MLOC framework for learning and self-optimising systems as depicted in Figure 1 – first introduced in (Tomforde, 2012) – provides a unified approach to automatically adapt technical systems to changing environments, to learn the best adaptation strategy, and to explore new behaviours autonomously. Figure 1 illustrates the encapsulation of different tasks by separate layers. Layer 0 encapsulates the system's productive logic (the *System under Observation and Control* – SuOC). Layer 1 establishes a control loop with safety-based on-line learning capabilities, while Layer 2 evolves the most-promising reactions to previously unknown situations. In addition, Layer 3 provides interfaces to the user and to neighbouring systems. Details on the design approach, technical applications, and related concepts can be found in (Tomforde, 2012).

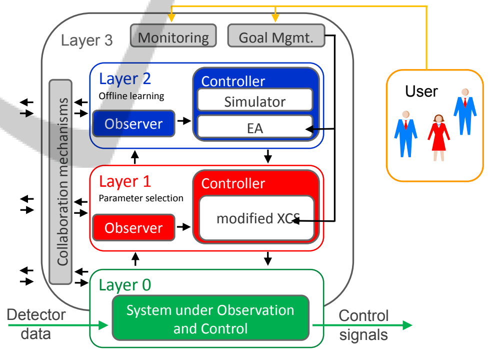

Figure 1: System Design.

In the context of this paper, we confine our regard to those components that have to cover changes in the user's goals at runtime – in particular at Layer 1 and Layer 2 of the architecture. At Layer 1 a modified LCS serves as controller within the regulatory control loop: it is responsible for adapting the SuOC's parameter configuration according to observed changes in the environmental conditions. Due to safety-restrictions in real-world systems, a standard LCS is not applicable to the learning problem as it would operate with too much freedom and relies strongly on trial-and-error. Therefore, we modified the *eXtended Classifier System* (XCS) (Wilson, 1995) by removing the rule-generation parts and restricted the set of possibly selectable rules. Details on these modifications are given in (Tomforde, 2012) – they don't influence the powerfulness and operability of the XCS algorithm. For understanding the technical impact of allowing the user to change the overall system goal at runtime, it is important to know that an XCS (and so the modified variant) stores its knowledge and experiences as rules in form of a basic 5-tuple. This tuple contains the attributes *condition* (in which situation is the rule applicable?), *action* (what to do if the rule is chosen?), *prediction* (what reward is expected if the rule is chosen?), *error* (how reliable is the prediction?), and *fitness* (what is the quality of the rule?). Learning in an XCS is realised by modifying the last three attributes using temporal difference algorithms according to an observed reward – which determines how well the user's goals have been achieved within the last evaluation cycle (typically referred to as receiving a *reward*). If the user changes the goals and thereby the reward function at runtime, the experiences and knowledge stored within these evaluation attributes don't reflect the correct reward function anymore. Hence, the XCS has to be further adapted in order to allow for keeping its learning behaviour and knowledge while simultaneously enabling an optimisation process towards the new system goal. The different possibilities to deal with this problem are discussed in section 3.2.

While Layer 1 is responsible for learning online from the observed system performance, Layer 2 realises the OC concept of moving the trial-and-error parts of learning to a sandbox. Similar to approaches like *Anytime Learning* (Grefenstette and Ramsey, 1992), a simulation-coupled optimisation heuristic evolves new rules in case of insufficient knowledge at Layer 1 and adds these novel rules to the rule-base of the XCS. Again, this exploration of novel behaviour in terms of rules needs a definition of good and bad system performance – the user's goals. First attempts to deal with the related flexibility problem are discussed in section 3.3.

3.2 Flexibility at Layer 1

Since Layer 1 of the architecture is concerned with automatically improving the selection strategy over time, its controller component is implemented using machine learning techniques – a modified XCS in case of the MLOC framework. Each machine learning technique uses a reward or fitness function $f:\ \mathbb{K} \to [0,1]$ to determine the strength of the used parameter setting $K \in \mathbb{K}$. Generalised versions with $f:\ \mathbb{K} \to \mathbb{B}$ can also be found in literature. Since it should be possible for the user to switch the system goal at runtime, we need to extend the XCS to hold a family of fitness functions f_α. The user is now capable of switching the system's goal by changing α. Starting from the simplest concept of "blowing up" the existing concept, we investigated three different approaches with increasing complexity to realise flexibility at Layer 1:

- **Selective Approach.** We extend the XCS to hold multiple populations – one disjoint set for each goal function. The XCS uses the population for the currently set goal function and switches over to another population when the user changes the goal function of the OC system.
- **Multi-dimensional Approach.** Extend each classifier of the XCS to hold an array of 5-tuples – one dimension for each goal function. The XCS selects the 5-tuple according to the α of the currently set fitness function. The difference to the previous approach relies on which rules are stored and the direct feedback for the evaluation criteria even if another α is chosen.
- **Cumulative Approach.** Accumulate all goal functions to a new one and use this new goal function to update the 5-tuple of the XCS.
  We've chosen the following function $f_{Cumulative}$ for accumulating the different goal functions
  $$f_{Cumulative}(K) = \mathbb{E}[f_\alpha(K)] - \left(\max_\alpha\{f_\alpha(K)\} - \min_\alpha\{f_\alpha(K)\}\right)$$
  since a maximisation of this function results in a minimisation of the difference between the maximum and the minimum while maximising the first moment. Consequently, all considered values are in a small corridor around the first moment.

3.3 Flexibility at Layer 2

In order to investigate flexibility at Layer 2, we analysed three different approaches to modify the existing optimisation heuristic to be flexible in terms of OC:

- **Multi-criterial Approach.** Use a standard Multi-criterial optimisation algorithm like the *$\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm* (SMS-EMOA) (Naujoks and Beume, 2005) and add the resulting set of rules to the population.
- **Selective Approach.** Minimise the amount of evaluation calls and start optimisations for each goal function with these minimised evaluation calls.
- **Cumulative Approach.** Accumulate all goal functions to a new one and optimise that new goal function. We have chosen the same cumulative function $f_{Cumulative}(K)$ as in Layer 1 (see 3.2).

Table 1: Variable parameters of the R-BCast protocol.

Parameter	*Standard configuration*
Delay	0.1 *s*
AllowedHelloLoss	3 *messages*
HelloInterval	2.0 *s*
δHelloInterval	0.5 *s*
Packet count	30 *messages*
Minimum difference	0.7 *s*
NACK timeout	0.2 *s*
NACK retries	3 *retries*

In contrast to the approach at Layer 1 of the architecture, the basic flexibility concepts at Layer 2 can be implemented using standard techniques in the first step. Therefore, the focus of this paper is confined to the Layer 1 aspects.

4 EVALUATION

The following section discusses the experimental investigation of the different flexibility concepts for Layer 1. The content is based on initial work as published in (Becker, 2011).

4.1 Example Application

The basic design approach as presented in section 3.1 has been applied to various application scenarios, including vehicular traffic control, production, and mainframe systems (Tomforde, 2012). In the context of this paper, we turn our attention to the *Organic Network Control* (ONC) (Tomforde et al., 2011) system as example application. ONC has been developed to dynamically adapt parameters of data communication protocols (i.e. buffer sizes, delays, or counters) in response to changing environmental conditions. It learns the best mapping between an observed description of the environment and the most promising response in terms of a parameter configuration.

Especially when applying ONC to broadcast algorithms in mobile ad-hoc networks (MANET), we demonstrated the significant benefit of the additional ONC control. For instance, when applying ONC to the R-BCast protocol (Kunz, 2003) ONC increased the algorithm's reliability and decreased the arising overhead. Within this paper, we reuse the implementation of ONC controlling this R-BCast protocol to demonstrate different flexibility strategies. Table 1 lists the variable parameters of R-BCast and thereby defines the configuration space $\mathbb{K}$. Details about the parameters' functionality and impact on the protocol's logic as well as an explanation of the algorithm can be found in (Kunz, 2003).

The online learning system relies on a situation description as conditional part of each rule. In this scenario, the distribution of other nodes in the direct neighbourhood according to a sector-based model as depicted in figure 2 serves as condition part, while the parameter configuration according to table 1 defines the action part.

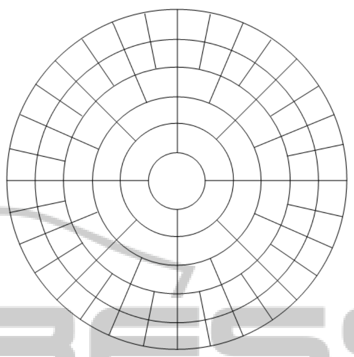

Figure 2: Environment representation.

4.2 Experimental Setup

We define the set of goal functions for the parameter setting $K \in \mathbb{K}$ at the environment situation S by the following family $f_{\alpha,S} : \mathbb{K} \to [0,1]$ with $0 < \alpha < 1$

$$f_{\alpha,S}(K) = \alpha \cdot f_{PDR,S}(K) + (1-\alpha) \cdot f_{IOH,S}(K) \quad (1)$$

where α is the weight for the sub goal functions Packet-Delivery-Ratio (PDR) and Inverse-Overhead (IOH). These goal functions comprise the most prominent aspects when evaluating MANET-based broadcast protocols – see e.g. (Williams and Camp, 2002) – and are defined as follows:

- Packet-Delivery-Ratio $f_{PDR,S}(K)$:

$$\frac{AgentReceived}{MaxSentMessages \cdot \#Nodes \cdot \#Senders}$$

- Inverse-Overhead $f_{IOH,S}(K)$: $\frac{AgentSent}{MacSent}$

We use the inverse function of the Overhead since our optimisation problem should maximise its function. This structure of the goal functions allows to change them at runtime by simply manipulating the weight factor α. An important reason for the selection of those sub goal functions is the fact that a maximisation of the PDR is assumed to result in a minimisation of the IOH and vice versa. We therefore have contrary oriented goal functions which also increases the learning problem's complexity as a simple coupling is not possible with contrary aspects.

The evaluation of the three different approaches uses the settings as depicted in table 2 for the simulation. The first eight hours rely on the goal function

Table 2: Setting of the ONC simulation used for the evaluation.

Goal function by α	0.5	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	Σ
Duration in hours	8	2	2	2	2	2	2	2	2	2	26

$f_{0.5,S}$ to build up the XCS' population and the corresponding 5-tuples for each classifier. After this build-up time the XCS uses time slots of two hours for the goal functions starting with $f_{0.9,S}$ and a decreasing α. We used this setting to let the XCS find a constant population before changes in the goal function occur.

4.3 Results of the Evaluation

The figures 3 - 6 contain the evaluation results of the simulation as explained in table 2 for the three different approaches as presented in 3.2. They show the Packet-Delivery-Ratio (green line) and the Inverse-Overhead (blue line) in addition to the progress of the weight factor α of the goal function $f_{\alpha,S}(K)$ (red line). Table 3 contains the averaged results. All approaches outperform the "ground-truth" of using the protocol's default values.

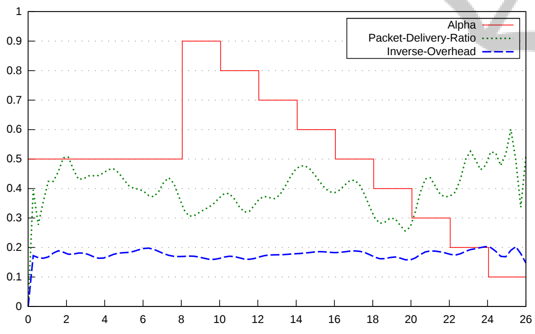

Figure 3: Default values.

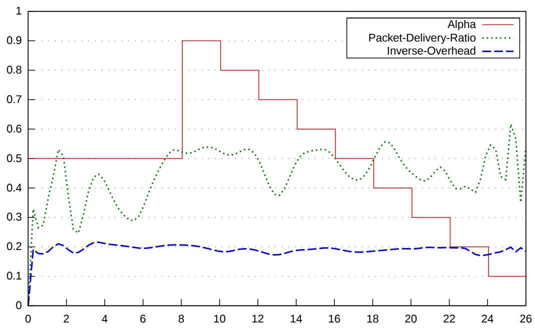

Figure 4: Selective approach.

The cumulative approach performs worst compared to the selective and the multi-dimensional approach. It was *not* possible to accumulate the contrary character of the different goal functions to a meaningful single value. Instead, the cumulative function

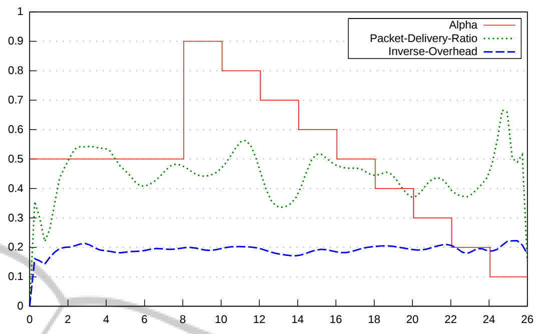

Figure 5: Multi-dimensional approach.

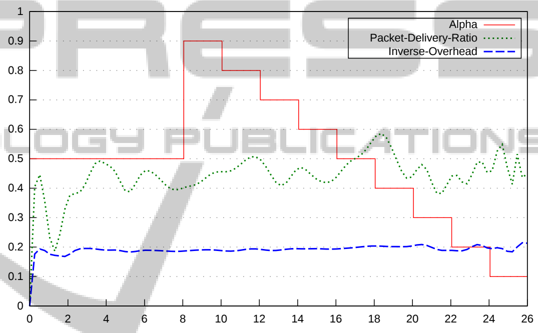

Figure 6: Cumulative approach.

$f_{Cumulative}$ results in a good rating of those classifiers that a) perform better than the default values for all goal functions but b) not optimal in any goal function.

The selective and multi-dimensional approaches are similar in terms of the two goal functions. The selective approach performs better for the Packet-Delivery-Ratio while the multi-dimensional approach results in a higher averaged value for the Inverse-Overhead. In contrast, the averaged number of classifiers needed to achieve this behaviour is different: the multi-dimensional approach outperforms the selective approach due to the disjunctive character of the latter one's population setup. In particular, the former approach does not rely on evolving a novel rule in each occurring situation as it has broader knowledge from other goal-aspects kept in the same population.

5 CONCLUSIONS

This paper discussed the need of novel mechanisms to allow for flexibility in organic systems. We named different approaches from various domains where the

Table 3: Averaged rewards of the different approaches.

Approach	*Packet-Delivery-Ratio*	*Inverse-Overhead*
Default values	0.4006	0.1772
Selective	0.4509 (+12.6%)	0.1923 (+8.5%)
Multi-dimensional	0.4495 (+12.2%)	0.1928 (+8.8%)
Cumulative	0.4438 (+10.8%)	0.1921 (+8.4%)

term *flexibility* is used and defined what we want to achieve in technical systems. Afterwards, we explained the need for novel techniques and mechanisms to allow for a flexible system behaviour in case of learning and organic systems. Therefore, the basic Observer/Controller pattern and its technical implementation have been mentioned.

The evaluation part introduced three different approaches to achieve flexibility for the rule-based online learning activities. According to the Organic Network Control system as example application, we compared the different concepts in a realistic environment. Based on these first insights, we will continue to find solutions that keep the existing experience in case of changing goals and allow for an efficient learning. The paper showed that flexibility is an important issue and needs further research activities.

In upcoming OC systems, flexibility will gain increasing attention. Therefore, future work will explicitly have to cope with related mechanisms and strategies. One possibility to investigate the problem separated from the limitations and noisy effects of realistic applications is the previously mentioned animat scenario (see section 2). This scenario will serve as one example and basis for more abstract investigations of the flexibility problem.

REFERENCES

Becker, C. (2011). Untersuchung von Mechanismen zur technischen Umsetzbarkeit von Flexibilitaet in Organischen Systemen. Diploma thesis, Leibniz Universität Hannover.

Benjaafar, S. and Ramakrishnan, R. (1996). Modeling, Measurement and Evaluation of Sequencing Flexibility in Manufacturing Systems. *Int. J. of Production Research*, 34:1195 – 1220.

Compton, K. (2004). Flexibility Measurement of Domain-Specific Reconfigurable. In *ACM/SIGDA Symp. on Field-Programmable Gate Arrays*, pages 155 – 161.

Eden, A. H. and Mens, T. (2006). Measuring software flexibility. *IEE Proceedings - Software*, 153(3):113–125.

Grefenstette, J. J. and Ramsey, C. L. (1992). An Approach to Anytime Learning. In *Proc. of the 9th Int. Workshop on Machine Learning*, pages 189–195.

Hassanzadeh, P. and Maier-Speredelozzi, V. (2007). Dynamic flexibility metrics for capability and capacity. *Int. J. of Flexible Manufacturing Systems*, 19:195 – 216.

Kephart, J. O. and Chess, D. M. (2003). The Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50.

Kunz, T. (2003). *Reliable Multicasting in MANETs*. PhD thesis, Carleton University.

Naujoks, B. and Beume, N. (2005). Multi-objective optimisation using S-metric selection: Application to three-dimensional solution spaces. In *Proc. of CEC'05*, pages 1282 – 1289. IEEE.

Schmeck, H. (2005a). Organic Computing. *Künstliche Intelligenz*, 3:68 – 69.

Schmeck, H. (2005b). Organic Computing – A new vision for distributed embedded systems. In *Proc. of the 8th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing*, pages 201 – 203.

Schmeck, H., Müller-Schloer, C., Çakar, E., Mnif, M., and Richter, U. (2010). Adaptivity and self-organization in organic computing systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 5(3):1–32.

Shuiabi, E., Thomson, V., and Bhuiyan, N. (2005). Entropy as a measure of operational flexibility. *European Journal of Operational Research*, 165(3):696 – 707.

Tennenhouse, D. (2000). Proactive computing. *Communications of the ACM*, 43(5):43–50.

Tomforde, S. (2012). *Runtime adaptation of technical systems: An architectural framework for self-configuration and self-improvement at runtime*. Südwestdeutscher Verlag für Hochschulschriften. ISBN: 978-3838131337.

Tomforde, S., Hurling, B., and Hähner, J. (2011). Distributed Network Protocol Parameter Adaptation in Mobile Ad-Hoc Networks. In *Informatics in Control, Automation and Robotics*, volume 89 of *LNEE*, pages 91 – 104. Springer.

Williams, B. and Camp, T. (2002). Comparison of broadcasting techniques for mobile ad hoc networks. In *Proc. of the 3rd ACM int. symp. on Mobile ad hoc networking & computing*, pages 194–205. ACM.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.