Organic Computing

Oleksandr Pavlichenko
Computer Engineering Group
ZITI, University of Heidelberg
Heidelberg, Germany
pavlichenko(at)stud.uni-heidelberg.de

Prof. Dr. Holger Fröning
Computer Engineering Group
ZITI, University of Heidelberg
Heidelberg, Germany
holger.froening(at)ziti.uni-heidelberg.de

***Abstract*—The paper presents a short overview of the state of the art in the field of organic computing, explaining its most vital components and techniques, as well as providing some examples to applications of organic computing systems.**

Keywords—organic computing; self-x; machine learning; genetic optimization; autonomous systems

I. INTRODUCTION

A. Motivation

Nowadays we are faced with rapidly increasing requirements to computer systems. Most of the systems have to function in complex and dynamically changing environments. We have reached the point when hardware improvements are mostly quantitative, e.g. putting more cores on a single die or increasing cache size. The situation is a cause to the fact that new systems are more complex, incorporate more components and/or modules.

Conventional computer or software system reactions are limited to what developers have foreseen and implemented, i.e. have a number of static reactions which will not change over time. It would have been much more efficient to develop a system that can adapt itself to a environment or, in case of pure software solution, to a context, where it works, to enhance performance, resource usage and stability. One of the most advanced requirements to a system is the best possible human interaction, which is hard to achieve using predefined settings.

B. What is Organic Computing?

OC is a new and rapidly developing research field in computer science, operating on the edge between life sciences, hardware engineering, software development and potentially even more research areas. Organic computing (OC) systems are not related to so called bio-computers, and operate using conventional silicon chips, not organic chips, as one may think. The name is chosen because of inspiration that OC takes from the live sciences (especially biology).

OC system consists of many intelligent subsystems, which communicate with each other, ensuring good collaboration between components and overall better system functionality.

C. Goals

Main goal of OC is to develop a robust and flexible system which can adapt itself to different situations even if system reaction was not directly implemented beforehand. This can be achieved with addition of life-like properties to the system, which is exactly a more specified goal of organic computing. Other aim is to create a goal-driven system, which means we don't have to define explicit simple tasks, but a more abstract goal to achieve, which greatly improves the human interaction possibilities of a system.

As an extension of the main goal we defined, there are more options which improve the system functionality, but are not obligatory to include in the system. Such additions are for example learning techniques for intelligent systems, and evolutionary algorithms, which in conjunction with learning may drastically improve system stability and performance.

D. EXTOLL Flow Control

An automatic network traffic flow control system (such as used in EXTOLL project) is a simple example of OC inspired system, despite the paper doesn't mention anything about OC. The system is designed to optimize the usage of available hardware resources, and relies only on its internal data (no outside control), so we may say that the system is self-optimizing itself for a given workload.

II. SELF-X PROPERTIES

In previous chapter we outlined that the main goal of OC is to add life-like properties to an intelligent system (IS). This properties aim to replicate the behavior of life organisms or colonies of organisms. The set of properties is called self-x properties, with "x" naming behaviors, e.g. organization, optimization, healing, protection, configuration etc. "Self-" means that the system or given subsystem incorporates the feature in itself and no external control is used to ensure the correct functionality of a property. Thus, the system which has self-x properties can be considered autonomous and flexible to some extent, with current development searching for ways to push the limit further.

A. *Self-organization*

Most important self-x feature is self-organization. It incorporates all other self-x in itself, because we can derive any other life-like property from organization, e.g. self-optimization is equal to "self-(re)organize for better performance".

Uncontrolled self-organization may be dangerous since we cannot predict the behavior of the system. That is the reason why OC uses controlled self-organization. As an architecture of choice comes Observer/Controller architecture, adapted to needs of certain application. A simple example for the architecture is shown in fig. II-1. Observer/Controller is a part of the system, so for the external observer system will be self-organized.

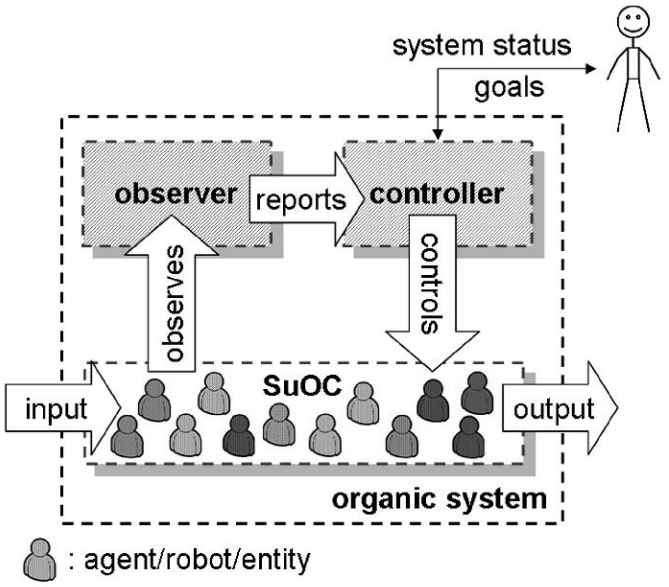

Figure II-1. Simple Observer/Controller architecture[1]

There are three main approaches to the implementation of Observer/Controller loop (control module or CM). First possibility is a central control, which corresponds to fig. II-1. This type includes only one CM that watches over the whole system. This type doesn't make the system very flexible, and is simplest way to go with. A more complex and flexible alternative is a decentralized control schema. This approach incorporates CM inside of every module, which is shown on the left side of following figure:

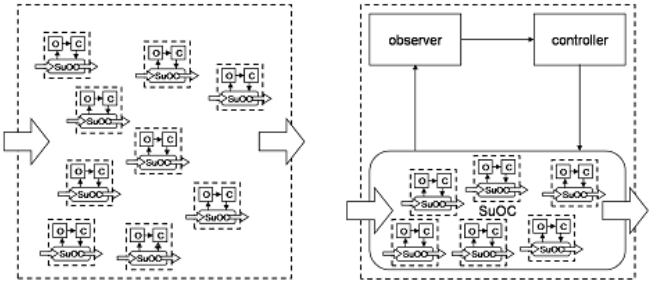

Figure II-2. Observer/Controller implementations[1]

Last variant is a multilevel approach, which is most complex and rewarding. It's a mixture of first two possibilities with addition of (multiple) control levels. The basic schema for the approach is shown on the right side of fig. II-2. The choice of exact implementation to use depends on many factors such as system scale, timing requirements, available hardware resources, task to be performed etc.

B. *Degree of Self-organization*

From three implementations of Observer/Controller architecture that we discussed in previous subsection, we can derive a quantitative measure of system self-organization called degree of organization. The degree depends on the relation between number of CM's within the system to number of agents (subsystems). If we assume that number of CM's is equal to *n*, and number of agents *m*, we can define three degrees of self-organization:

- *n = 1*, the system is weakly self-organized, as we have only one central control unit.
- *n > 1*, the system is self-organized, as some elements of the system organize itself, and we possibly have a global CM in addition to local ones.
- n > m, the system is strongly organized, as we can assume that most (or each of) elements organize itself and we have a multilevel regional and/or global CM.

The degrees of organization can be used to classify a system as self-organized OC system.

C. *Summary and Overview of other Properties*

We already mentioned that all self-x properties may be derived from self-organization property. This does not mean that every self-organized system automatically possess all of self-x features, it is just opens a possibility to add such feature as a part of self-organization. On the other hand, if we know that system possesses any of self-x properties, we can assume that it is self-organizing. One additional feature, which doesn't meet the self-x format is context-awareness. This means that system tries and is able to get the environment information and reacts accordingly to data it becomes.

III. EVOLUTION

We know that every OC system consists of many subsystems that communicate and work together to achieve some goal. If we assume that each subsystem is a chromosome, we can build our complete system as a combination of the chromosomes, the same way it's done in nature. The idea of evolutionary optimization is than next generation will be better than predecessor and after some number of generations we become a system that is fit for our purpose.

As a preparation to evolutionary optimization we must develop a set of basic tasks and construct agents that can fulfill the task. The whole process takes many iteration, each iteration consisting of the following steps. Firstly we randomly generate many systems from subsystems (number of subsystems may vary, so as a repeatability of a single subsystem within a

system). Then we test if generated systems satisfy our requirements. Fittest systems are then exchange their properties by copying or replacing some subsystems and thus the next generation is created. This process continues until some system satisfy all of the requirements. An addition of mutations can speed-up the process drastically, but determination of "good" mutations may be very hard and apply to big of an overhead. The drawbacks of the method are that it could potentially never or take a very long time to converge because of its randomness.

Another possibility to utilize genetic concept is a organic model where a system evolves to a new generation with added or altered functionality, based on information it accumulated during current generation. This technique is closely coupled with learning.

IV. Learning

(Machine) Learning is a capability of a system or an algorithm to achieve better results over time as a result of an accumulation and analysis of data. This feature is very important to every system that must operate in autonomous mode at times.

Learning process consists of many phases, such as:

- data acquisition, when we collect the data;
- data preprocessing when we filter and sometimes reduce the working set to improve performance of the learning algorithm;
- actual learning or prediction based on the data we have;
- storage to a knowledge base or output of the results, depending on previous operation.

OC systems should be able to operate efficiently in dynamic environments, which makes learning a perfect addition to the set of features. Learning allows for much better self-optimization, because leaning feature mean that system has memory (not in terms of hardware) and can "recall" or "predict" (to a some degree) the outcome of given operation, if this operation was already performed by the system.

The optimization and configuration potential are evident. For example running many tests with slightly different parameters and in different environments will allow a self-optimizing system with learning capabilities to "remember" good configurations for each environment and use it as the basis for further optimization.

There is a huge number of learning algorithms and techniques, each with its advantages and drawbacks. Most interesting for the self-organizing systems is so called online-learning. This means that a system learns and extends its knowledge, at the same time actively processing the main task. Online learning algorithms must be lightweight and efficient, with minimum possible overhead.

As we have many subsystems, each of can potentially learn independent from other components, a collaborative learning approach is very interesting technique to use. This approach uses the knowledge exchange function, to distribute new data between the components. If done correctly, this can parallelize the learning process. Very important during knowledge exchange is the "weight" of information, which can be determined using following properties:

- uniqueness
- importance
- representativity

V. Applications of Organic Computing

OC systems can be used in wide variety of areas, ranging from traffic control systems to a system on a chip (SoC) solutions. In the following we discuss some of current and past projects that use OC systems as their basis

A. Organic Traffic Control[6]

The project aims to optimize the regulation of traffic flow in modern cities using OC-based system. The static system will not operate well in such environment, as the dynamic changes to the traffic patterns are impossible to foresee at design time. This forces that optimization and coordination decisions are made by the system at the runtime.

A core of the system is an Observer/Controller architecture that allows for adaptive learning. The system uses two-level learning and optimization mechanism. The whole algorithm operates on-line, i.e. optimizations and learning are done during the workload. First level of the mechanism selects suitable signal patterns to use from the knowledge base, while the second level mechanism optimizes those if necessary.

Additionally, an optimization in form of self-organized coordination was used. During the tests, there was confirmed that completely decentralized control is inferior to hierarchical which should be the next step in the development.

B. Organic Computing in Off-Highway Machines[8]

This projects aims to enhance the control system of off-highway vehicles such as tractors with the concepts of OC, which should greatly improve reliability and robustness of such vehicles.

The project uses generic Observer/Controller architecture as a basis, then focusing on improving the Observer side of the CM. The system uses machine learning techniques to extract usual patterns for a vehicle and optimizes the control accordingly. As in most cases the online learning algorithm was used, allowing for on-the-fly optimizations of the behavior.

C. Organic Self-organized Bus-Based Communication Systems[7]

This project tries to overcome a drawbacks of usual off-line design of CAN protocol with the aim to improve the communication scheduling model.

In the work, authors present three types of streams that are used in the CAN protocol and try to optimize it introducing organic communication stream. To ensure fairness and the fact that all hard deadlines will be met (concerning real-time streams), the project uses dynamic offset adaptation algorithm.

On the other hand, high bandwidth streams also receive a fair control using the enhanced priority based medium access game algorithm. To learn the communication patterns, the penalty learning algorithm is used.

The experimental results showed the feasibility of a dynamic approach to a CAN protocol control, but many optimizations are still to be done.

VI. Summary and Outlook

Organic Computing is a promising field of research with vast possibilities. The basis for OC are self-x properties which are inspired by the biological systems. The properties allow to design more safe and robust systems, which show better human interaction behavior. Features such as genetic optimization and machine learning are important aspects of OC, and add even more flexibility and performance of applications.

OC can be used in almost every field of computer science, has projects in automotive area and is always expanding. There is still loads of research and optimization room left in the field. Many projects start every year, with new ideas and solutions.

REFERENCES

[1] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 5-37, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[2] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 95-109, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[3] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 111-125, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[4] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 237-251, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[5] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 253-265, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[6] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 431-446, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[7] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 489-501, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[8] M. Wünsche, S. Mostaghim, H. Schmeck, T. Kautzmann, M. Geimer, Organic Computing in Off-Highway Machines

[9] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 267-280, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011

[10] http://en.wikipedia.org/wiki/Machine_learning, as of 27.01.2013

[11] C. Müller-Schloer et al. (eds.), Organic Computing – A Paradigm Shift for Complex Systems, 615-627, Autonomic Systems, DOI 10.1007/978-3-0348-0130-0_1 Springer Basel AG 2011.