Organic Computing and an Approach using Multi-Agent Systems

Benjamin Dittes
School of Computer Science and Engineering
University of New South Wales

14th November 2005

COMP4416

Intelligent Agents

Course Essay

Contents

1 Introduction

I/T systems are growing. In fact, they are growing so fast that soon it will not be the hardware that limits its growth but the increasing amount of work it takes to keep the software consistent, secure, up-to-date and optimal. In 2001, IBM expressed this development, stating that "at current rates of expansion, there will not be enough skilled I/T people to keep the world's computing systems running"(IBM Research 2001).

IBM proposed what they called 'Autonomic Computing': Integrating automatic system management into the software components themselves. The term's similiarity with the biological counterpart, the 'autonomic nervous system', is not coincidental. Like the autonomic nervous system is subconciously regulating all basic body functions, autonomic systems are supposed to have self-x-properties: the main four being self-configuring, self-optimizing, self-healing and self-protecting.(Kephart & Chess 2003) Even beyond that, the claim is that like in biological systems comprised of vast amounts of cells "it is the self-governing operation of the entire system, and not just parts of it, that delivers the ultimate benefit"(IBM Research 2001).

Based on this idea of emergence[1], the AI branch of 'Organic Computing' has formed to analyze the potential, the problems and most importantly the challenges faced in building those systems. In an overview paper, the German Association for Informatics clearly seperates organic computing from artificial life research with its more biological and less application-driven focus and acknowledges 'Soft Computing'[2] as a research strand with similiar goals but confined to using very specific natural mechanisms.(Müller-Schloer et al. 2004)

I will introduch Organic Computing and its claims and issues in section 2 and then focus on a recently proposed multi-agent methodology for building Organic Systems(Kasinger & Bauer 2005) in section 3, including a personal critique. Section 4 will finish with some concluding remarks.

[1] Emergence here means that a large number of elements without central control form coherent and not predefined patterns (best example: the 'game of life').

[2] Soft computing includes evolutionary and genetic algorithms, simulated annealing and artificial neural nets.

2 Organic Computing

2.1 Overview

Biological systems easily achieve what today's software systems so painfully lack: they adapt and integrate seamlessly and automatically into the given environment. We can observe self-management at every level: dead cells are automatically replaced and the new cells integrate into the context, the human body heals and fights off infections on its own and individuals integrate into their society – and at every level, new complexity is gained beyond the function of the single parts.(Müller-Schloer et al. 2004)

While artificial neural nets or genetic algorithms try to emulate this behaviour by simulating large artificial populations, Organic Computing merely aims at developing enough intelligence at every (hardware) node to produce the desired emergence (adaptive, controllable and self-managing) on the global level.(Schmeck 2005)

Today, the internet is already an environment with enough real devices to make simulating smaller entities superfluous. Within years, the development of pervasive and ubiquitous computing will surround us with myriads of small, wireless and integrated devices. Only if those systems are able to configure and manage themselves, there is a chance for a breakthough of this technology.

2.2 Claims and Issues

Organic Computing is a very young research branch and as such is at the moment mainly dealing with identifying the central problems that have to be overcome. Nonetheless, the claims of Organic Computing are quite clear cut(Schmeck 2005):

- The network of organic-enabled devices will, on a global scale, emit self-x-properties, as proposed by IBM for Autonomic Computing(Kephart & Chess 2003). These include self-configuring (i.e. discovering and

connecting to other devices/networks, advertising the global functionality), self-optimizing (i.e. constantly observing and tuning local and global parameters), self-healing (i.e. identifying faulty sub-elements and correcting or excluding them without compromising the element's functionality) and self-protecting (i.e. defending against large-scale attacks as well as anticipating failure by evaluating sensor reports).

- On top of that, Organic Computing aims at device networks massively involving humans, as opposed to the very server-oriented Autonomic Computing. In the era of ubiquitous computing, the massive number of devices in our homes or cars will have to be controllable by the owner. Therefore, Organic Computing Systems will emit high-level managability with a focus on man-machine interaction (see section 2.3).

- Partly as an effect of the self-x-properties and partly through specific design, Organic Computing Systems will be more robust against fault and more reliable, protect the owner's privacy and thus be more trusted than current computer systems.

It is important not to forget that these points are the vision of Organic Computing. To get a better understanding of the way we have to go to achieve them, these are the most important issues:

- A problem with emergent behaviour is that it is not pre-programmed. Therefore the system designer's control on the final behaviour is rather limited, leading to a high risk of unintended system properties. The first concern is how we can make sure that the final system will really have all the desired self-x-properties, and has them correctly and completely. The second concern is how we can prevent any unwanted deliberation or mistakes. Clearly, it is neccessary here to find suitable controller mechanisms, such as the 'observer controller architecture', described in Kephart & Chess (2003).

- On the field of man-machine interaction it is important to not only optimize the interfaces (as begun by von der Malsburg (2004)) but also to gain the trust of the user. I think that this is one of most crucial obstacles that need to be overcome because nowadays not many people

are willing to give up control over the computers in their everyday lives to an artificial system.

- And finally: How can we be sure to think about all these concerns when building a Organic System? What we need is a methodology for building these systems, as described in section 3.

2.3 Organic Computing in Image Recognition

In an introductory paper, von der Malsburg (2004) proposed that image recognition, after "four initial decades of sheer frustration", has to accept that the central problems (such as edge finding, motion extraction or object recognition) are not conquerable by complex, single and – most importantly – hand-written algorithms. Rather, the presented solution is based on minute elements of processing and simple data structures which are then able to combine into complex dynamic descriptions of the scene.

They built a first system to detect heads walking by a camera. It is based on the dynamic integration of a set of heterogenous subsystems for "pixel change (static background), skin color, a head shape template, a local contrast histogram and motion continuity". Each produced a likelyhood for each pixel that it contained a head. The weights of each system in the global distribution were dynamically adjusted based on sucess and each sub-system adjusted its parameters to match the global consensus. Even without providing any initial information to the system (e.g. the head template was initially empty), it works "quite reliably".

This example and a few others in schema-based learning and figure-ground separation show that self-managing systems, even on this rudimentary level deliver the emergence of surprising image analysis capabilities. Hopefully, this will provide a basis for man-machine interaction that can support the massive demand by ubiquitous systems.

3 A Mulit-Agent Approach

In a recent paper, Kasinger & Bauer (2005) described a coarse-grained concept of a multi-agent architecture that was designed as an Organic Computing System and shows the potential to support its self-management properties. Just as important, they designed a methodology for populating it, based on the model driven architecture (MDA). I will go through the details of the system one by one and discuss the arising issues as well as the potential on the way.

3.1 Discussion of the Agent Architecture

The first main idea of Kasinger and Bauer's paper is the rough description of the multi-agent architecture. The following figure shows the high-level model of necessary components and their relations:

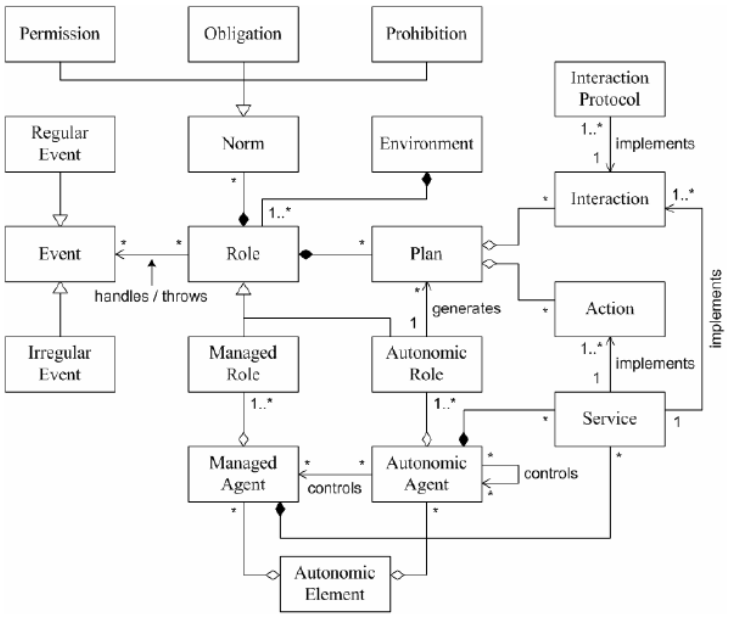

Figure 1: High-level model of the multi-agent approach to an Organic Computing System. There are two kinds of agents, Managed Agents and Autonomic Agents, and a set of events, norms, (inter-)actions and plans for each agent. From Kasinger & Bauer (2005).

The central element of the structure is the role. It acts as an 'agent template', containing a list of plans, a list of events the role can throw and handle and a list of norms to answer to those events. Regular events are known to the role whereas irregular events apear in case of failure or attack and might have to be handled by the supervising autonomic role (see below). The general idea is that the agent assuming a role when receiving a certain event adds a specific norm to the list of active norms (this connection between event and norm is defined by the role). A norm can be a permission or prohibition – thus influencing the agent's decission making – or an obligation – e.g. a goal.

From a classical point of view(Bratman 1990), I believe the norm takes on the place of an *intention*: it poses problems (obligation), constrains reasoning (permission / prohibition) and – in connection with a plan to respond to it – leads to actions. Although the ability to emit proactive behaviour necessary for this interpretation is never mentioned in the paper, it would be ignorant to overlook the design's obvious potential for it.

Each role contains a plan library to respond to the arising goals, although the paper does not discuss how plans are selected or maintained. A plan consists of actions (domain specific or throwing an event) and interactions, the latter being implementations of interaction protocols. The obvious example for a direct interaction protocol is an auction in which case the whole auction process would be one step in the plan. Another very intriguing option mentioned for possible interaction protocols are indirect communications based on stigmergy[3], as used by ants to communicate food sources or shortest paths.
To prevent tight coupling between agent and domain logic, both actions and interactions are not implemented directly but provided by services.

At this point, the design is split into managed and autonomic roles. Both have the same structure, with the exception that autonomic roles are able to construct plans – again, the paper does not elaborate on how and when this should happen but merely mentions that this possibility is necessary for self-management. Consistently, managed roles are assumed by managed agents and autonomic roles are assumed by autonomic agents. So far, the

[3]Stigmergy is the act of indirect message passing by modifying an active environment. The message can be intended for the agent itself – like leaving a note on the fridge – as well as for multiple other agents. An excellent application can be found in Nachibi & Akbarzadeh (2004).

terms ‘managed’ and ‘autonomic’ are arbitrary as they are both built on the same agent structure. What makes a managed agent ‘managed’ is the fact that it is controlled by autonomic agents. For lack of explanations in the paper, I assume here that this control includes both throwing events for the managed agent as well as directly modifying parameters of the agent itself and the (domain specific) platform the agent is running on (for instance a router’s buffer size or a workbench’s production speed). In two words: The managed agents work on the domain level while the autonomic agents work on the meta-level.

Finally, a group of related managed and autonomic agents are deployed inside an autonomic element and it is at this point at the latest that we see the Organic Computing concept emerge from this design: an autonomic element is one cell in the organic environment. It has limited potential to manage itself and can communicate (both on the domain and the meta-level) with other autonomic elements to produce the desired emergence of global self-management.

In my opinion, if judged by traditional standards, for instance as described by (Bratman et al. 1988, Georgeff & Ingrand 1989), this model can definitely not be called a description of an agent architecture. It is neither complete – there is no mention of any kind of belief – nor matching – not being a team approach the term ‘role’ seems misplaced and typical issues such as plan management are not discussed.

However, I believe that when approached with an open mind and good faith in the possibility that the ‘eternal issues’ of agent systems – most importantly belief representation and plan selection and management – can find an adequate solution for this case, the model describes an architecture that is versatile and complex enough as well as itself obviously built on the basic ideas of Organic Computing: within each autonomic element the domain specific (managed) and autonomic logic is seperatly represented but tightly coupled.

3.2 Discussion of the Methodology

To systematically create all the components mentioned above, Kasinger & Bauer designed a methodology based on the model driven architecture (MDA). MDA was chosen because it allows complex (mainly UML-based) designs to be semi-automated by providing transformation rules from higher-level to lower-level design steps. A short overview can be found in Kleppe & Warmer (2005), the official home-page is 'http://www.omg.org/mda'.

The following image shows the 19 design activities on the computational independent level (CIM) – usually called analysis phase – and the platform independent level (PIM) – usually called design phase. The links stand for possibly automatic MDA transformations.

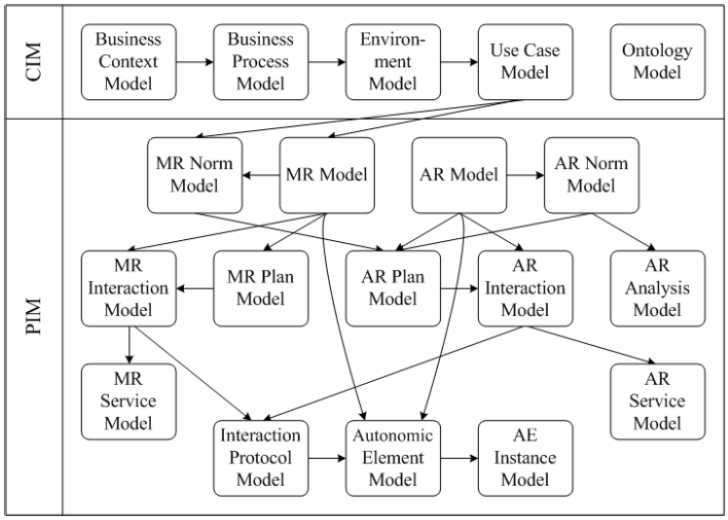

Figure 2: MDA model of the design methodology. There are 19 design activities, from the computational independent model (CIM) of the domain specification down to the platform independent model (PIM) of autonomic element deployment. The links depict possibly automatic MDA transformations. From Kasinger & Bauer (2005).

The process starts by defining the domain business model and its processes. Based on that the enviroment is defined through the needs of those processes and common use cases for chains of business processes are specified. Finally, the common 'vocabulary' – e.g. the objects and values the processes deal

with – is combined into an ontology model for the entire system. This step is very important since without a common ontology agent communication is impossible. So far, these steps are neither agent nor Organic Computing specific but very common and required steps in setting up any domain specific system.

The agent aspect is introduced when the use cases are transformed into managed roles, including norms, plans and interactions. This is a very time-consuming step, but if we exclude higher level reasoning – as the paper seems to do – a rather straightforward one.

The more interesting design process is the development of the autonomic roles. Clearly, as can be seen by the lack of ingoing links in the graph, this is an independent activity that is not much concerned with the domain. Together with the norms of the automatic roles, those two models form the basis for the autonomic behaviour of the agent. It seems that not only does this seperation allow a more focused analysis of the issues discussed in section 2.2 without having to worry too much about the domain but also endorses a reusability of the autonomic roles and norms (not the plans, of course!).

Back to the domain world, what makes the whole autonomic role system work is the autonomic analysis model. It is responsible for detecting faulty, sub-optimal or malicious situations and throw events to force the autonomic agent to react to them. Although the diagram does not show a link between the use case model and the autonomic analysis model, there is a clear influence and I think the reason the link is missing is that there is no *automatic* transformation.

With the autonomic role and analysis model in place, the autonomic plan and interaction models follow as an implementation of the processes needed to control the managed agents. Plan generation is not mentioned in the design process and it would be very hard to do so without a clearer definition of the belief representation and reasoning systems. On the same level of thought, defining the services for both managed and autonomic roles seems at this point very easy since the model does not yet deal with the actual implementation concerns.

Finally, the managed and autonomic interaction models are combined into

an interaction protocol model and the autonomic elements are defined and deployed. Creating the interaction protocol *after* the interactions have been defined and used (in the respective plans) seems a little untidy but is of no importance since there is no actual need for the protocols (as abstract definitions of the interaction) in a homogenous system where the interactions themselves already exist.

On the other hand, the fact that the model uses the extra layer of interaction protocols is the only hint I found that it might be possible to use heterogenous agents (or at least autonomic elements developed by different parties). In every other aspect the system design is completely monolithic and there is no mentioning of different agent vendors in the paper.

This methodology can of course only be the first part of a development model that leads to a final system. Although it seems plausible, an evaluation is impossible until it is used for a real system implementation – and even then there are many more things to be done beyond the proposed model, most importantly the completion of the described agent structure to form a consistent agent model including belief and reasoning, to make it run.

4 Conclusion

As professional I/T systems get more powerful and complex and ubiquitous computing is about to bring many devices into every home the need for self-management is very real. Without it, systems will be un-maintainable, un-protectable and constantly sub-optimal. The development of Autonomic Computing Systems that are able to control that complexity by themselves will, given the necessary standardization, go a long way in solving this problem for professional server systems.

In the home device sector, the need for more adaptivity, context-awareness and proactivity adds to the challenge. Sparked by the idea of emergence through autonomic control, Organic Computing Systems show the potential to deal with these issues and have even been successfully extended to new areas, such as image recognition. Admittedly, in many of these areas Organic Computing is not alone and there is a fair bit of overlapping[4] but the ideas of emergence of self-management and high-level control are quite unique.

However, until such systems are really ready there is still a lot to do: at the moment, there is no good theory about the nature of emergence and how to influence it at the cell level. As a first step, the presented agent architecture and methodology give a good, if coarse-grained, idea of what these cells could look like and is an excellent place to start refining – and eventually of course implementing – Organic Computing Systems.

What makes this young research branch so exciting is that agent technology may have finally found a target that requires – or better: allows – such a massive multi-agent involvement that we should see the benefits in both areas very soon.

[4] Indirect message passing by stigmergy, for instance, is also an important topic in swarm intelligence.

References

Bratman, M. (1990), 'What is Intention?'. In Cohen, P.R., Morgan, J. & Pollack, M.E. (Eds) *Intentions in Communication*. MIT Press, Cambridge, MA.

Bratman, M. E., Israel, D. J. & Pollack, M. E. (1988), 'Plans and Resource-Bounded Practical Reasoning', *Computational Intelligence, 4, 349-355* .

Georgeff, M. & Ingrand, F. (1989), 'Decision-Making in an Embedded Reasoning System', *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 972-978* .

IBM Research (2001), 'Autonomic Computing Manifesto'. http://www-03.ibm.com/industries/government/doc/content/bin/auto.pdf.

Kasinger, H. & Bauer, B. (2005), 'Combining Multi-Agent-System Methodologies for Organic Computing Systems', *Proceedings of the 3rd International Workshop on Self-Adaptive and Autonomic Computing Systems (SAACS 05), Copenhagen, Denmark* .

Kephart, J. O. & Chess, D. M. (2003), 'The Vision of Autonomic Computing', *IEEE Computer, Jan. 2003, pp. 41-50* .

Kleppe, A. & Warmer, J. (2005), 'Explore Model Driven Architecture and Aspect-oriented Programming, Birds of a Feather'. http://www.devx.com/enterprise/Article/27703.

Müller-Schloer, C., von der Malsburg, C. & Würtz, R. P. (2004), 'Organic Computing – Beherrschung der Komplexität'. http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/58/.

Nachibi, M.-B. & Akbarzadeh, M.-R. (2004), 'Stigmergy for hunter prey problem', *World Automation Congress, 2004. Proceedings Volume 16* .

Schmeck, H. (2005), 'Organic Computing – A New Vision for Distributed Embedded Systems', *Proc. of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)* .

von der Malsburg, C. (2004), 'Vision as an Exercise in Organic Computing'. http://www.neuroinformatik.ruhr-uni-bochum.de/VDM/PUBLIST/2004/index.html.