2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems

# Experience report in developing and applying a method for self-organisation to agile manufacturing

Giovanna Di Marzo Serugendo and Regina Frei
*School of Computer Science and Information Systems*
*Birkbeck College, University of London*
*London, UK*
*dimarzo@dcs.bbk.ac.uk, work@reginafrei.ch*

***Abstract*—The design and implementation of distributed, self-organising and self-adaptive systems are challenging. This article details our experience gained during the development of self-organising assembly systems, which provide solutions for user-friendly agile manufacturing systems. More specifically, we describe how both a development method for self-organising systems, called MetaSelf, and the above particular application were progressively shaped, each influencing the other.**

## I. Introduction

This poster describes our experience with the creation of Self-Organising Assembly Systems (SOAS) [5] where we added self-organising and self-managing capabilities to industrial assembly systems. We identified a suitable architecture and developed SOAS accordingly. As the research work progressed, experience gained SOAS provided feedback and insight into the actual development of self-organising systems. As a result, an initial development method, called MetaSelf, for self-organising and self-managing systems was devised, mainly based on the original architecture. Gradually, improvements in the development method caused changes in the design of SOAS, which were revisited and in turn provided additional feedback for improving the method.

## II. Manufacturing scenario

Manufacturing systems of the future have to be agile, distributed, user-friendly and increasingly autonomous. They need to cope with frequently changing requirements, low production volumes, many product variants, as well as perturbations and failures. Mechanical system reconfigurations are facilitated by modular hardware, but (re-)programming remains a work-intensive and error-prone procedure.

**Evolvable Assembly Systems (EAS)** [6] consist of robotic modules of varying granularity. A module is either an entire industrial robot with several skills (i.e. screwing, rotating and linearly moving) or a simpler module such as a robotic axis, a gripper, a feeder, or a conveyor having a single skill only. Every module is an embodied agent with thorough self-knowledge (about its skills and physical characteristics) as well as social abilities (to coordinate its work with other modules). Modules engage in coalitions to provide composite skills necessary to assemble the product. For instance, a gripper able to seize and release parts forms a coalition with a rotating robot to provide a screwing skill.

**Self-Organising Assembly Systems (SOAS)** [4] extend EAS with two additional features: 1) modules *self-organise* to produce a suitable layout for the assembly of the ordered product and 2) the assembly system as a whole *self-adapts* to production conditions and *self-manages* its behaviour.

## III. Experience

The stepwise design of the SOAS architecture went in conjunction with the evolution of the design method. We revisited the design of SOAS several times.

**Initiation - 1 to 2:** We had only a vague idea of the functionality provided by SOAS. This stage included a literature review to gain theoretical knowledge about self-organisation in natural systems, specific self-organising mechanisms, to identify and understand complexity concepts. Various forms of self-organisation were considered; see Figure 1 (1). We introduced (Figure 1 (2)) *dynamic coalitions* formed and modified by the agents themselves.

**Shaping - 2 to 3:** We decided to use the MetaSelf architecture [3] for its internal and external control capabilities, and for its focus on both self-organising and self-managing issues. We gained a better understanding of SOAS, identified the self-* requirements, and defined four SOAS life cycle phases (Figure 1 (3)). A development method was added to the MetaSelf architecture. It proposes a development process [2] in three phases (Figure 1 (3)): requirement and analysis, design (including the definition of patterns and self-* mechanisms as well as system design) and implementation.

**Refining - 3 to 4:** Firstly, the self-organisation mechanism and the architectural pattern are chosen. To obtain a truly bottom-up approach where modules spontaneously assemble to fulfill the product order given in input, we decided to follow the *Chemical Abstract Machine (CHAM)* paradigm [1]. Secondly, models including agents, metadata and policies are developed. Our approach was now based on the Metaself development method as illustrated in Figure 1 (3). A simulation phase within the design complements the method (Figure 1 (4)).

978-0-7695-4232-4/10 $26.00 © 2010 IEEE
DOI 10.1109/SASO.2010.24

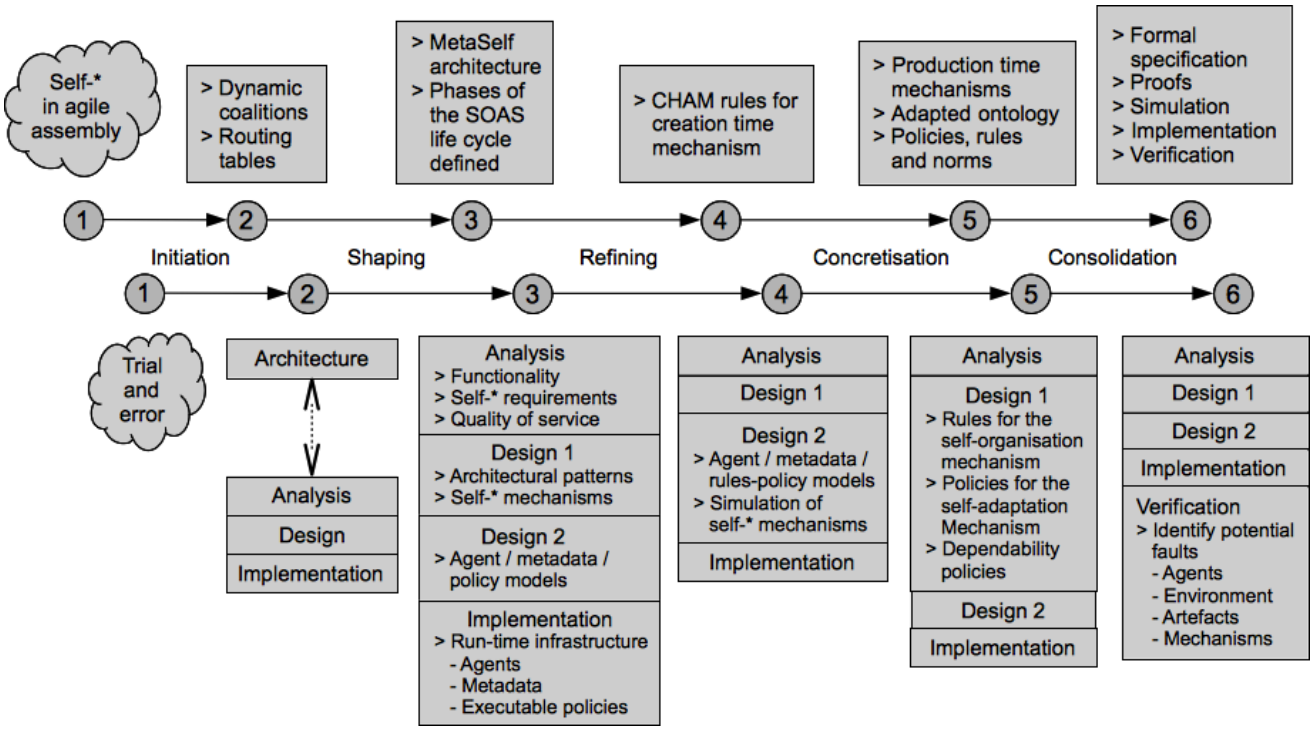


Figure 1. Refinement of the method in conjunction with the refinement of the application

**Concretisation - 4 to 5:** Self-adaptation to production conditions is obtained through a series of `if-then-else` policies applying to individual modules, dynamic coalitions or the whole system. Open issues: priority and conflicts among policies (Figure 1 (4)). The policies related to self-organisation and applying continuously are now called rules, while those related to self-management, applying punctually (to recover from faults) are called policies (Figure 1 (5)).

**Consolidation - 5 to 6:** A complete prototype will include the above described elements (Figure 1 (6)). A self-organising or self-adaptive system is now considered to be composed of active autonomous *agents* evolving into an *environment*, handling passive *artefacts*, and working according to some *self-* mechanism*. To determine potential faults in the system, the verification step consists in identifying faults that can arise in each of these elements (e.g. an error in the design of the self-organising rules, a malicious agent, or a faulty environment) and determine their impact on the system as a whole and how the system overcomes (or not) these faults.

## IV. Conclusion

Our experience helped gain insight on three levels: better understanding of the application we want to develop and of different design possibilities; better understanding of a development method for self-organising and self-managing systems; and finally, trade-off between design and implementation issues. While developing SOAS and applying the MetaSelf architecture, an accompanying design method was elaborated and step-by-step further refined.

**Acknowledgments:** This work was started while Regina Frei received a PhD grant from the Portuguese FCT; she currently receives a post-doc grant from the Swiss NSF. We also thank the EU-funded coordination action PerAda for financially supporting travel exchange.

## References

[1] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1998.

[2] G. Di Marzo Serugendo, J. Fitzgerald, and A. Romanovsky. Metaself - an architecture and development method for dependable self-* systems. In *Symp. on Applied Computing (SAC)*, pages 457–461, Sion, Switzerland, 2010.

[3] G. Di Marzo Serugendo, J. Fitzgerald, A. Romanovsky, and N. Guelfi. A metadata-based architectural model for dynamically resilient systems. In *ACM Symposium on Applied Computing (SAC)*, pages 566–573, Seoul, Korea, 2007. ACM.

[4] R. Frei. *Self-organisation in Evolvable Assembly Systems*. PhD thesis, Department of Electrical Engineering, Faculty of Science and Technology, Universidade Nova de Lisboa, Portugal, 2010.

[5] R. Frei, G. Di Marzo Serugendo, and J. Barata. Designing self-organization for evolvable assembly systems. In *IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 97–106, Venice, Italy, 2008.

[6] M. Onori. Evolvable assembly systems - a new paradigm? In *33rd Int. Symposium on Robotics (ISR)*, pages 617–621, Stockholm, Sweden, 2002.