Using Building Blocks for Pattern-Based Simulation of Self-organising Systems

Christopher Haubeck, Ante Vilenica, and Winfried Lamersdorf

Abstract. The constantly rising complexity of distributed systems and an increasing demand for non-functional requirements lead to approaches featuring *self-organising* characteristics. Developing these systems is challenged by their hardly predictable dynamics and emergent phenomena and requires therefore the incorporation of *simulation* techniques. In doing so, not all needed development activities can be realised by just one software application because self-organisation often implies unique settings, goals, and development methods as well as the use of individual code sections. In order to handle such unique environments, this contribution presents a pattern-based concept that incorporates reusable patterns for different development issues of self-organising systems by encapsulating various methods, algorithms, and applications in so called *building blocks* and combining them in a coherent and hierarchical process.

1 Introduction

Old-fashioned concepts of developing distributed computer systems top-down with centralised control are not suitable for many new application domains which are characterised by high complexity, diversity, and heterogeneity of their components. Such applications as, e.g., in mobile / pervasive computing, typically exhibit a dynamic behaviour that is hardly predictable and requires fast adaption to changing application contexts that may even be unknown a priori. For such applications *Self-Organisation* (SO) has proven to be a promising approach that can deal with these challenges autonomously by (self-)adapting their respective structure and behaviour without any centralised or external control. However, there are three facts that challenge the purposeful development of self-organising systems: (I) the inherent bottom-up development process, (II) local interactions among components

Christopher Haubeck · Ante Vilenica · Winfried Lamersdorf
Distributed Systems, Informatics Department, University of Hamburg
e-mail: `{haubeck,vilenica,lamersdorf}@informatik.uni-hamburg.de`

that cause effects on global level and (III) the presence of emergent phenomena. To overcome these challenges [2] propose a simulation-based development process for SO, i.e. simulation becomes an inherent step during an iterative development as it evaluates whether the designed components produce the designated behaviour.

In terms of the considered system each simulation can be very distinct, e.g., settings, goals, used simulation tools and languages. For this reason the process of performing simulation studies can be considered as a unique task, which in most cases cannot be handled by just one software system. As a result the development of self-organising systems takes place in a heterogeneous and potentially distributed environment of various systems. At the same time the task of *coordinating* different software frameworks to achieve the designated goal is quite demanding and requires a lot of *manual* effort by the developer [6]. In this regard this novel concept of *building blocks* offers the possibility of developing and using patterns that guide the development process of self-organising systems. Thereby, this approach combines the flexibility of individual development processes with the exchangeability and reusability of standardised processes and services.

The remainder of this paper is organised as follows. Section 2 discusses related work and Section 3 introduces the concept of building blocks and the usage of hierarchical processes for a pattern-based development of self-organising systems. Finally, the last section draws a conclusion and mentions some future research goals.

2 Related Work

Self-Organisation is well known and widely spread in natural systems, e.g., swarm behaviour and neural networks, and various successful utilisations of the SO paradigm in computer systems, e.g automatically guided vehicles or ant optimisation, have been reported [7]. However, the purposeful development of self-organising systems is still a challenge since SO makes primarily use of the following concepts: autonomy, dynamics, adaptivity and decentralised organisation [3] which obviously challenges traditional top-down development approaches. Consequently, [3, 4] propose new approaches that target SO problems in which *simulation* is an inherent part of the development process. Here simulation ensures proper system behaviour and is used in *early* development stages to continuously monitor the system dynamics.

Despite these new approaches, there is still a lack of concepts that systematically support the practical usage of simulation in self-organising systems. In this regard, existing simulation support can be broadly categorised into three classes: First, there are approaches as *Swarm* or *DesmoJ* that offer simulation capabilities by utilizing standard programming languages. These frameworks are highly flexible, but often lack support for non-experienced users and do not feature a wide range of tool support. Second, there are approaches as *NetLogo* or *Arena*. These systems offer a user interface and focus more on usability and simulation support. But most of these systems are hardly customisable and applicable for different applications. Furthermore, the user of such systems is often limited to the provided functions by the system, because an integration of new methods and algorithms for the development of modern self-organising systems is not supported. Third, there are approaches on a conceptual level that focus on specific aspects as, e.g., optimisation or validation. These approaches are often used by the self-organising community and can be seen as *process patterns* for the development of self-organising systems [3, 4, 6]. Moreover, these approaches focus on specific goals and do neither consider the generic perspective of simulation support nor do they provide general simulation-based development patterns.

Furthermore, projects as COSMOS [1] propose extensive process models to engineer from real-world systems to simulation results by suggesting different meta models to describe and analyse the system under study. In contrast to this, the concept of building blocks and hierarchical process does not focus on a specific approach to develop complex systems from scratch but rather tries to support the development of SO in general by providing a well-grounded concept to combine individual development processes. So this more functional approach can be used in such a comprehensive process model to provide the developer with reusable best-practise patterns.

3 Building Blocks for Simulation Patterns in SO

It is assumed that all tasks of the development of self-organising systems can be seen as *activities* of a coherent development process which can, according to the analysis of related work, be realised by already existing systems and concepts. Therefore, the presented approach rather tries to define a general concept that mediates between already existing solutions than providing new solutions for single development activities. The long-term objective is to allow for integrating activities of heterogeneous solutions in order to enable the development and execution of customisable, expendable and reusable patterns of any set of specified activities. These patterns can then be used to answer important questions of engineering self-organising systems, like identify macroscopic properties and variables or define steady scenarios and analysis algorithms [9]. To achieve this overall goal the concept follows three essential guidelines: (I) the set of functionalities that can rely on a external system or manually procedures should not be limited, (II) developed patterns and desired activities should be customisable and expandable and (III) all activities and processes should be reusable and interchangeable if they provide the same functionality.

Because of such a diverse domain of interest, a strict separation of activity scopes is required. In order to handle the arising heterogeneity of such separated activities the presented concept offers *building blocks* which act like an adapter for the accomplished activity and provide a distinct execution scope. These blocks operate as so called *active components* which represent a novel approach to build complex distributed applications [5]. Active components can be understood as distinct components which are managed by an infrastructure but remain autonomous in regard to their execution. The logic is realised by different internal architectures which can be based on various execution logics, e.g., agent-oriented architectures or process-based architectures. Consequently, building blocks offer the concurrent execution

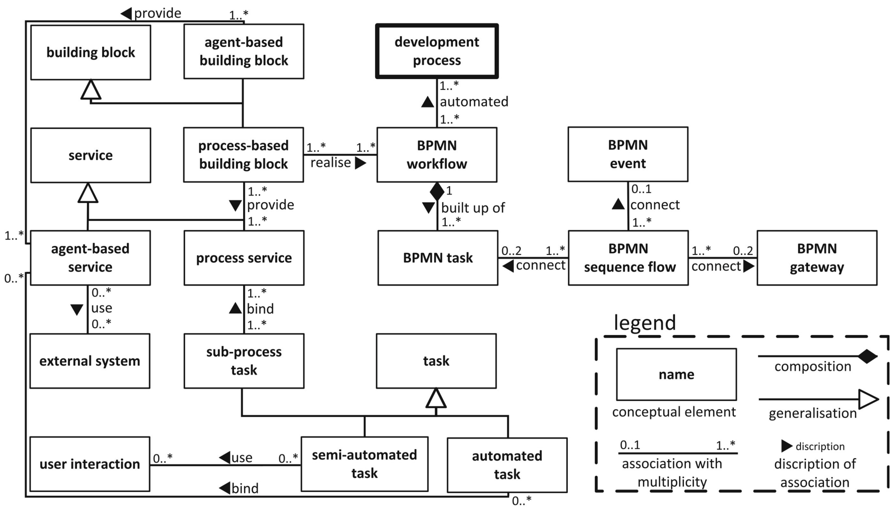

Fig. 1 Meta model for SO processes with building blocks

of self-contained actions that allows a fully independent execution, whereby any occurring dependencies of their actions remain within their building block scope.

In order to provide the functionalities and to ensure interchangeability and reusability of the activities, all functionalities of a building block are provided as one or more *services* in a service-oriented architecture which allows to publish, discover and utilise services locally or remotely. Thereby, the providing building block is responsible to interpret service requests and handle the corresponding actions in its scope which induces a loosely coupled system. For example, in case of a process-based architecture, an asynchrony service request can instantiate a new (sub-)process in which various tasks become active and perform their implemented activities which can consist of task specific program code or further service calls. The resulting system architecture of such consecutive service calls can be seen as a *Service Composition Architecture (SCA)*. This conjunction of both a process-based architecture and the use of any number of complemental services allows for complex process-based activities. In this way, a hierarchical structure of a building block, e.g., a development process, can be modelled by a service request to previously created building blocks, e.g., specialised development activities, that operate on a lower abstraction layer than the calling block. Thereby, the calling building block disguises characteristics and implementation details like, e.g., a concrete used method, by using services and, thus, enhances reusability, comprehension, and the further development of complex development processes.

Figure 1 illustrates our concept with a meta model for a combination of agent-based activities and BPMN processes. Here a development process is automated by a BPMN workflow which is built up of an arbitrary number of BPMN tasks that are connected within the modelled sequence flow. Each task can be semi-automated by

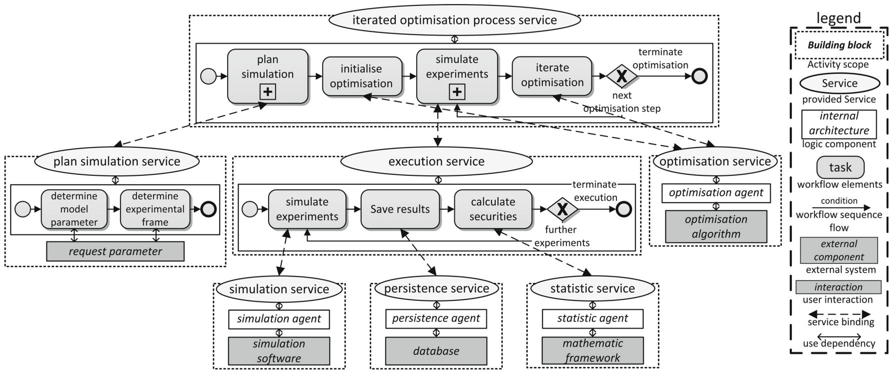

Fig. 2 A hierarchical process pattern for optimisation

a user interaction, automated by an agent-based service or can use a sub-process. Automated and sub-process activities are always provided by building blocks, that allow autonomous and independent actions and, in case of a process-based service, a hierarchical continuation by supplying a sub-process.

So on the one hand, external solutions, e.g., simulation systems or database support, can be used via service descriptions on a high abstraction level and on the other hand specialised demands can be realised by individual solutions on lower abstraction layers. As an example figure 2 shows a basic optimisation pattern. On the highest abstraction layer of this pattern a process-based building block coordinates between planing, simulation execution and an iterated optimisation algorithm. Simulation execution is realised by a simulation-based process cycle [6] which is realised within an *execution service* that executes and persists simulation runs until a defined confidence interval is reached. In doing so, single runs are performed via an agent that offers a simulation service for a specified simulation system that is integrated via a black box approach. Thereby, the presented concept also allows for specialised and encapsulated processes - rather than a strict black box model. Consequently, these processes can operate in their building block scope with knowledge about a specific system without affecting the generic usability of higher abstraction levels. This characteristic allows for model-specific runtime operation which is, for instance, rather useful in system benchmarking which is one of the most challenging tasks in SO [8]. To find an optimal solution the sequence flow of the pattern alternates between service requests to a direct optimisation method, e.g., a metaheuristic, which supplies different configurations of model parameters and the simulation execution which evaluates these configurations and provides performance indicators of the explored model. The presented example shows that prior manually performed procedures for self-organising systems can be decomposed in distinct, reusable and executable building blocks that can be merged in automatable patterns.

4 Conclusion and Future Work

This work has presented a concept that supports the systematic development of self-organising systems by reusable patterns that encapsulate activities, e.g., pure simulation, optimisation or evaluation, in *building blocks*. These encapsulated building units are addressed by a service description and can be hierarchically ordered and combined according to logical or temporal conditions. In conclusion, the presented concept can simplify the task of simulating complex systems since it allows for conveniently incorporating existing solutions by developing and executing best-practise process patterns as well as using state-of-the-art development technology. Therefore, the presented concept is highly flexible and customisable since it does not bound the developer to specific software systems and methods - a characteristic which is necessary to apply SO in real-world problems.

Future work shall, on the one hand, strive towards developing further patterns in order to support even more expedient application domains. This includes the shaping and categorisation of characteristic best-practise patterns as well as the integration of alternative development aspects. On the other hand, it is envisioned to implement an extensive simulation framework for SO that standardises and executes activities in order to allow a straightforward system design.

Acknowledgements. The authors would like to thank Deutsche Forschungsgemeinschaft for supporting this work through the project ”SO based on decentralised co-ordination in distributed systems” and also W. Renz from Hamburg University of Applied Sciences for inspiring discussions.

References

1. Andrews, P., et al.: Cosmos process, models, and metamodels. In: Stepney, S., et al. (eds.) Proc. of the 2011 Works. on Complex Systems Modelling and Simulation, pp. 1–14 (2011)
2. Edmonds, B.: Using the experimental method to produce reliable self-organised systems. In: Brueckner, S. (ed.) Engi. SO Systems: Methodologies and Applications, pp. 84–99 (2004)
3. Gardelli, L., et al.: Combining simulation and formal tools for developing self-organizing MAS. In: Uhrmacher, A.U., et al. (eds.) Multi-Agent Systems: Simulation and Applications (2009)
4. Gershenson, C.: Design and control of self-organizing systems. Ph.D. thesis, Vrije Univ. (2007)
5. Pokahr, A., et al.: Unifying Agent and Component Concepts - Jadex Active Components. In: Braubach, L., et al. (eds.) 7th Ger. Conf. on MAS Technologies (MATES), pp. 100–112 (2010)
6. Robinson, S.: Automated analysis of simulation output data. In: Kuhl, M., et al. (eds.) Proc. of the 37th Conf. on Winter Simulation (WSC), pp. 763–770 (2005)
7. Sauter, J., et al.: Performance of digital pheromones for swarming vehicle control. In: Proc. of the 4th Int. Conf. on Autonom Agents & Multiagent Syst. (AAMAS), pp. 903–910 (2005)

8. Vilenica, A., Lamersdorf, W.: Benchmarking and evaluation support for self-adaptive distributed systems. In: 6th Int. Conf. on Compl., Intel. & Softw. Intensive Syst. (CISIS), pp. 20–27 (2012)
9. Wolf, T., et al.: Engineering Self-Organising Emergent Systems with Simulation-based Scientific Analysis. In: Proc. of the 4th Int. Works. on Engi. SO Applications, pp. 146–160 (2005)