Autonomous Self-Integration in Interwoven Systems

Jörg Hähner, Sebastian von Mammen, and Sven Tomforde

Universität Augsburg, Organic Computing Group, Augsburg, Germany,
joerg.haehner@informatik.uni-augsburg.de

1 Introduction

Nowadays, the vision of *Ubiquitous Computing* as formulated by Marc Weiser in 1991 [4] becomes increasingly realistic. Information and communication technology (ICT) has become a fundamental part of human lives and supports us embedded in the environments that we encounter on a daily basis – it already pervades every aspect of our daily lives. This inclusion fundamentally changes our way to control and manage technical systems; but it also dramatically changes the way we design and integrate these utilised systems – which resulted in a tremendous complexity that evolves over long periods of time.

Compared to the status of technical equipment one or two decades back in time, we can observe a large number of distributed and tightly coupled systems that we are using routinely today – partly without even noticing that these systems are there and serving us in a ubiquitous manner. From household appliances or cars to logistics and public infrastructures: Everything works on the foundation of being connected to distributed communication services, such as the Internet. These "hidden" technical helpers are working in addition to the obvious traditional IT-driven devices (i.e. web-technology driven platforms). As a result, we have build up a technical network of *interwoven* structures that are no longer isolated entities but more and more interfere with each other (see [2] for a definition of "Interwoven Systems"). We can already observe that this networked structure is far too complex for being able to predict its behaviour and the (transitive and indirect) mutual influences between the coupled entities.

2 Idea: System Self-Integration

Obviously, it is not an option to reverse this trend to counter the resulting complexity issues – e.g. by trying to build isolated systems again. Hence, we have to accept the fact that we are facing novel challenges; and that these challenges have to be addressed appropriately. We argue that current approaches to design, develop, and engineer systems are not applicable any more - or at least that they come with intrinsic drawbacks preventing them from solving the corresponding issues and consequently entailing failures and outages resulting from the interwoven system structure.

The general idea to counter these negative effects is to move design-time decisions to runtime. As a result, systems themselves take over the responsibility for their integration decisions. Thereby, *self-integration* consists of several aspects:

1. **Self-adaptation of behaviour**: The internal control mechanism of the system alters the configuration of the productive strategy according to observed changes in the environmental and internal conditions. Here, e.g. concepts from the Organic Computing domain are utilised [3].
2. **Self-management of relationships**: The system itself decides about its cooperation partners – by adding and removing relations to other systems, the overall system is re-organised. As a result, the structure of the system is adapted at runtime.
3. **Quantification of success**: Besides performance-related metrics, subsystems need a quantification method to estimate the success of the integration status. Further influences have to be taken into account, e.g. reliability of interaction partners, availability of resources, or redundancy to avoid outages.
4. **Self-modelling**: As basis for the decision process, subsystems have to generate and update models of themselves, their surroundings, and their (possible) interaction partners, including dependencies and transitive processes among them (e.g. extending the Models@Runtime concept [1]).
5. **Technical trust**: Closely related to modelling is the capability to establish (technical) trust relationships. Based on observations of historical behaviour, estimations are derived how interaction partners will behave on future situations. This is especially important in open, heterogeneous systems.
6. **Flexibility and goal adaptation**: Subsystems need the freedom to reflect about their current goal and strategy – also allowing them to accept non-optimal states for a certain period.

3 Advantages and Challenges

Large-scale distributed systems consisting of self-integrating subsystems will be characterised by a high degree of adaptivity and robustness against disturbances. Implicit and explicit dependencies will be detected autonomously by the subsystems themselves using e.g. concept such as self-modelling and dependency detection. Besides the advantage of enabling autonomous and more accurate reactions, malfunctions and oscillating effects can be prevented. This poster demonstrates the challenges concerned with developing fully self-integrating subsystems.

References

1. G. Blair, N. Bencomo, and R. B. France. Models@Runtime. *IEEE Computer*, 42(10):22 – 27, 2009.
2. S. Tomforde, J. Hähner, and B. Sick. Interwoven Systems. *Informatik-Spektrum*, 37(5):483–487, 2014. Aktuelles Schlagwort.
3. Sven Tomforde, Holger Prothmann, Jürgen Branke, Jörg Hähner, Moez Mnif, Christian Müller-Schloer, Urban Richter, and Hartmut Schmeck. Observation and Control of Organic Systems. In *Organic Computing - A Paradigm Shift for Complex Systems*, pages 325 – 338. Birkhäuser Verlag, 2011.
4. Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.