A Universal Self-Organization Mechanism for Role-Based Organic Computing Systems

Florian Nafz, Frank Ortmeier, Hella Seebach, Jan-Philipp Steghöfer, and Wolfgang Reif

Lehrstuhl für Softwaretechnik und Programmiersprachen, Universität Augsburg, D-86135 Augsburg
{nafz,ortmeier,seebach,steghoefer,reif}@informatik.uni-augsburg.de

Abstract. An Organic Computing system has the ability to autonomously (re-)organize and adapt itself. Such a system exhibits so called self-x properties (e.g. self-healing) and is therefore more dependable as e.g. some failures can be compensated. Furthermore, it is easier to maintain as it automatically configures itself and more convenient to use because of its automatic adaptation to new situations. On the other hand, design and construction of Organic Computing systems is a challenging task. The Organic Design Pattern (ODP) is a design guideline to aid engineers in this task.

This paper describes a universal reconfiguration mechanism for role-based Organic Computing systems. If a system is modeled in accordance with the ODP guideline, reconfiguration can be implemented generically on the basis of an of-the-shelf constraint solver. The paper shows how Kodkod can be used for this and illustrates the approach on an example from production automation.

1 Introduction

Increasing complexity and steadily rising requirements more and more often become dominant problems during system development and maintenance. Organic Computing (OC) [9] and Autonomic Computing (AC) [6] are trying to tackle these challenging aspects. The basic idea is to build systems, such that they can autonomously adapt to changing requirements, optimize themselves at runtime for better performance or compensate failures by smart counter measures. These abilities are often referred to as self-adapting, self-optimizing and self-healing.

Although such self-x properties are highly desirable, it is still a challenging task to design and construct systems with such abilities. The Organic Design Pattern (ODP) is a design guideline for a specific-class of self-x systems. It provides efficient support for design and specification of an Organic Computing system. However it is still a challenging task to refine this design to an actual implementation. This paper shows how the problem of implementing a self-reconfiguration...
algorithm according to a given specification can be solved in a generic way. This is very valuable during system construction as there exist many, very elaborate construction processes for the functional parts of an OC system while implementing the organic behavior is often a very problem-specific and creative task. Technically, this is achieved by translating the corresponding design artifacts and OCL constraints which describe the behavioral and structural properties of a system into a model for a generic constraint solver (in this case: Kodkod).

The paper is organized as follows: Section 2 gives a brief overview of the Organic Design Pattern and the system class this paper focuses on. In Section 3 an introduction to Kodkod and the translation of necessary ODP artifacts into Kodkod’s (relational) modeling language are given. An illustration on a real world example from production automation is shown in Section 4. Finally, Section 5 concludes the paper with some related approaches and a brief outlook on future work.

2 Organic Design Pattern

The Organic Design Pattern (ODP) [12] is a design principle for a broad class of self-x systems, namely those which consist of a set of independent components interacting with each other and where reconfiguration and adaptation respectively can be expressed as a reallocation of roles. The components of such a system have to provide redundancy with regard to their functionality to enable such reallocations at run time. This means the components must have several capabilities they can use to fulfill different roles. The computation of a correct new role allocation then takes into account the possible interactions between the components, the capabilities of the components and the task that has to be achieved by the whole system.

The systems regarded in this paper are distinguished by changing tasks during runtime and the possibility to process several resources with different tasks at the same time. Examples for such systems are sensor networks, distributed smart devices which provide context sensitive services, or adaptive production automation systems. Furthermore, the systems can run in a degraded mode in which a task is only partially fulfilled, thus compensating for failure as long as possible. The core of the pattern which allows for modeling such systems is an elaborate role concept. The model has been based on a precise semantics which allows to define and measure self-x properties [3]. Additionally, the reconfiguration process can be described on an abstract level very intuitively.

The following paragraphs give a very brief introduction to important concepts of ODP and the “Restore-Invariant-Approach” [4] for specification of reconfiguration algorithms. An application of this design concepts to a real-world case study will be shown in Section 4.

2.1 Static View

An ODP system consists of Agents which process Resources with one or more of the agents’ Capabilities according to a given Task. A Task describes how a given