Model-driven Development of Self-organizing Control Applications

Hans-Ulrich Heiß, Gero Mühl, Helge Parzyjegla, Jan Schönherr
Berlin Institute of Technology

Torben Weis, Arno Wacker, Sebastian Schuster
University Duisburg Essen

Application model.

Scenario

Actuator and sensor networks (AS-Nets) will become an integral part of our living and working environment. AS-Nets are formed by modern end-user devices (ranging from PCs over PDAs and smartphones to service robots) that communicate wirelessly (e.g., by using WLAN, Bluetooth, or IrDA) and may cooperatively provide services in e-Home and e-Office scenarios. Developing applications for AS-Nets is challenging. The heterogeneity of devices as well as frequent communication faults vastly increase the complexity. Both issues state a problem for the developer since one can neither completely predetermine the configuration nor anticipate all potential faults that might occur at runtime. Hence, devices and applications must be able to work as autonomously as possible, requiring only little to no manual intervention.

MODOC Project

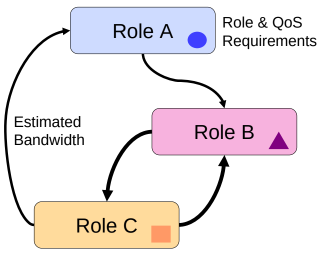
The MODOC project follows a model-driven approach to the development of autonomous control applications. The goal is to empower the application developer to create self-organizing and robust applications for AS-Nets with only minimal expert knowledge. Therefore, we do not only use the model for the design and deployment of applications but also to dynamically adapt at runtime. We aim at providing a tool chain that comprises an easy-to-learn modeling language, a graphical development environment, and a model transformation process that encapsulates necessary expert knowledge to deal with heterogeneity and self-organization at design time and runtime.

Modeling Language

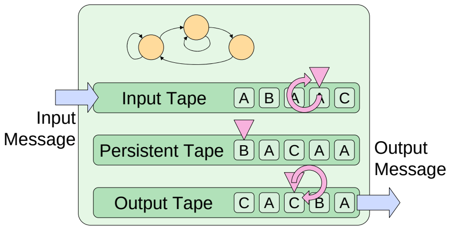
We are developing a modeling language that describes the behavior of an AS-Net application at a high level of abstraction. Developers using this language do not have to care about heterogeneity, network structures, self-stabilization, or even self-organization. A model transformer splits the model in separate roles and determines how these roles cooperate, i.e. which ones are strongly coupled and which cooperate rarely. The extraction of this meta-information is the key reason for using a model-driven approach. Based on this meta-information, the system can self-organize at runtime. Furthermore, the model transformer can generate an executable implementation from the model which is at the same time guaranteed to be self-stabilizing.

Roles

Roles are extracted from the application model and realize a certain part of the application's workflow. Each role presumes a set of capabilities that describe the minimum requirements for a device to be able to execute that application part. Additionally a role specification may be enriched with information about the desired quality of service, the induced load, or the expected bandwidth usage. This additional knowledge can be kept as meta-information during the rest of the transformation process and exploited at runtime to guide the self-organization process. Since roles do only address other roles and not a concrete node, they decouple the distributed application from the actual device it runs on and, thus, facilitate the development of flexible and adaptive applications.



Enriched role model.



Self-stabilizing, energy-aware Turing machine.

Self-Stabilization

In our approach, self-stabilization is a key concept to achieve robustness. A self-stabilizing application is guaranteed to recover from any transient fault within a bounded number of steps provided that no further fault occurs until the system is stable again. Transient faults include temporary network link failures resulting in message duplication, loss, corruption, or insertion, arbitrary sequences of process crashes with subsequent recoveries, and arbitrary perturbations of data structures.

To face these problems, we developed a self-stabilizing Turing Machine which features an energy concept which is based on the laws of thermodynamics. Every program that executes correctly on this machine in the absence of errors is guaranteed to be self-stabilizing in the presence of errors. This greatly simplifies the development process since no manual proof of the self-stabilization property is required. The key to the self-stabilization property is that derived information has always a lower energy than the information it has been derived from. Together with the inability to amplify the energy of information we get the desired self-stabilization property.

Self-Organization

One interesting feature of self-stabilizing algorithms is that they do not need any initialization and, hence, are perfectly suited for systems that must organize themselves. We employ a self-stabilizing role assignment algorithm which takes care of assigning all roles that are needed to nodes which are capable of serving them. Additionally, the assignment must be adapted to network changes and faults afterwards without external intervention. The more details of the enriched role model are available at runtime, the more adequately the role assignment can be carried out. If role requirements are additionally equipped with meta-information about the desired quality of service, devices are able to advertise how good they are in performing a particular role. Using these advertisements it is possible to determine the most convenient assignment for the user in the present context. However, the assignment currently does not take network limitations into account. It can happen that two heavily communicating roles are assigned to different nodes that only share a small-bandwidth link. Therefore, we use meta-information to optimize the role placement at runtime. We exploit knowledge derived from the application model as well as from the fragmentation process of an application into roles to mark these roles that are presumed to have a high communication demand.

Conclusions

The MODOC project aims at encapsulating necessary expert knowledge in the model transformation process in order to free application developers from having to care about distribution, heterogeneity, deployment, and self-organization. In order to facilitate applications to organize themselves, we build on a flexible role concept and the usage of self-stabilizing algorithms that do not require initialization. Knowledge about behavioral aspects of the application as specified in the model and embedded in the role specification is a very valuable resource that can be exploited in the self-organization process at runtime. As an example, we use meta-information derived from the application model and the role model to find optimized role assignments regarding the quality of service and communication costs.

Optimized dynamic role assignment.

Web: http://kbs.cs.tu-berlin.de/projects/modoc E-mail: modoc@kbs.cs.tu-berlin.de