Domain-Specific Modeling Language for Self-Adaptive Software System

Filip Krikava, Romain Rouvoy, Lionel Seinturier, INRIA Lille / Université Lille 1 - LIFL, France

The growing complexity and operational costs of contemporary software systems points to an inevitable need for making them autonomously adaptable at runtime\(^1\). A common approach for engineering such self-adaptive software systems is to use Feedback Control Loops (FCLs)\(^2\). Using measurements of a system outputs (e.g., response times, utilizations), a FCL adjusts the system control inputs (e.g., scheduling, concurrency policies) to achieve some externally specified goals\(^3\). From the different techniques that can be used to build such a FCL, Control Theory (CT) provides solid foundations and tools for synthesizing a reliable feedback control with some formal guarantees\(^3\). However, the integration of CT controllers is usually left to an extensive handcrafting of a non-trivial implementation code. This is a challenging task, in particular when considering the variety and complexity of contemporary distributed computing systems.

In our work, we address this by providing flexible abstractions of feedback control. We propose a domain-specific modeling language called FCDL for integrating adaptation mechanisms into software systems through external FCL. The key advantage in the domain-specific modeling approach is the possibility to raise the level of abstraction on which the FCLs, their processes and interactions are described. This makes the resulting architecture amenable to automated analysis and implementation code synthesis. FCDL defines FCL architectures as hierarchically organized networks of adaptive elements. Adaptive elements are actor-like entities that represent the corresponding FCL processes such as monitoring, decision-making and reconfiguration. They interact with one another through a mixture of explicit push-pull communication channels. The model is statically typed, handles composition, supports element distribution via location transparency and is reflective thereby enabling coordination of multiple control loops using different control schemes. It is adaptation domain and technology agnostic making it applicable to a wide range of software systems and adaptation properties. The language semantics is described using a model of computation that defines operational rules governing interactions among the feedback processes. The model of computation is further complemented with a notion of interaction contracts. They constrain the allowed interactions using formal descriptions enabling various verifications such as architecture consistency, determinacy and completeness. As a result, this should allow researchers and engineers to experiment and to put easily in practice different self-adaptation mechanisms and policies.

---