Grammars for problems solved by dynamic software systems

David Martinek

October 21, 2011

Abstract

Contemporary, commonly used software engineering techniques do not consider the possibility of development of a system during its goal exercise. A transition from a model to its software implementation is only one-way in such systems. The resulting software system deterministically processes exactly described problem.

Dynamic software architectures allow to solve even problems whose specifications change during the run-time. Specifications of the problem can be modified by external influences that were not known during programming of the software system. Specifications of the problem can also be modified as a result of activity of the software system itself. Dynamic software architecture allows programming by models that are created and simulated during the run-time. Design of such systems accounts for such technologies like for example technology of software agents, reflective systems, or genetic algorithms.

Remarkable way of use of dynamic software systems is solving tasks with too large or not known in-the-time-of-design state space. The important parts of the state space are mapped and described during the execution of the software system solving of the task. The description can be subsequently used for more effective implementation of the software system.

Tasks of dynamic planning and organising are examples of problems that are solvable by dynamic software systems. The goals of these tasks involve creating and deriving of effective plans of different, usually parallel activities that use shared resources. The number of shared resources and boundary conditions of their use can change during the run-time and these changes need not be known before. So, the program must react dynamically to these changes.

The common software systems solve problems, that are deterministic by its nature. These problems can be described by languages generated by context-free grammars. Dynamic software architectures allow creating of programs that can solve even nondeterministic problems. The usual price for that is computational time because such systems are less effective than "hard wired" programs. The languages that describe the class of problems solvable by dynamic software systems can be generated by context-sensitive grammars. Specifications of some problems are changed or particularised during solving of these problems, so complete languages of these specifications or their grammars are not known in advance but they are also particularised during the run-time. Such improved grammars can be used for reprogramming of software systems or at least theirs parts.

The goal of this work is to create dynamic software system capable of reactions to the task formulation changes during the run-time and of particularisation of the grammar of solved task specifications at the same time.