Implementing UML hierarchical state machines using the State-oriented programming design pattern

Written by Tamás Rehák (rehak@sch.bme.hu) 2002.
Consultant: Dr. András Pataricza, Gergely Pintér

The task for this semester was to examine the State-oriented programming design pattern, the hierarchical state machines, basic code-generation disciplines and write a code-generation tool using this design pattern.

There aren't much real working solutions for transforming hierarchical state machines into source code, expect the State design pattern and the above mentioned State-oriented programming pattern.
This pattern was developed by Miro Samek, and the aim of this pattern is to provide a small, efficient, framework-based solution for this problem. The pattern was developed to be used in embedded, hard real-time environment where both high speed and small memory footage were crucial.

In this form of the design pattern, it supports only a subset of UML statechart's features: nested states with proper handling and execution of entry/exit actions upon entering/exiting states and the handling of guards and history states can be also implemented. But developing the pattern itself, maybe we can include much more from the standard for example orthogonal regions.

During the development it was important for me to implements this design pattern in an expandable way. First I created all the class hierarchy that is written down in the UML MOF standard than specified the two interfaces the incoming and the outgoing. Their implementation can be easily expanded or replaced. The program uses the standard XMI description of UML statecharts, and generates C or C++ codes.
The generated C or C++ code can be commented (extracted from the XMI) and easily expanded with the "useful" program code. The guard conditions are not evaluated or checked as it is not important from our view.

Using code generation for writing programs makes the work of the programmer much easier: he or she models the problem in a high-level UML tool, than exports it to the standard XMI format and the code can be created with a small, but maybe powerful tool. In the future these code-generation tools will be much more integrated with the modelling tools and the code synthesis will be easier.

During this semester I got to know the UML statemachine model, its operational semantic, the UML metamodel, its data exchange format. Now I have some knowledge about code generation, XML-processing, and I learned the C# programming language in it I implemented this program.