Code Generation from UML models [4]

Ákos Frohner, Zoltán Porkoláb and Dr. László Varga

Creating a generic, object-oriented, component-based, transactional business system, which covers the whole lifecycle, is possible only with integration of commercial tools, component technologies, newly developed class libraries and using code generators. Most of the recently used tools for development techniques are focusing only of one the layers of the model from the code generation point of view. As a consequence the inter-layer connections are lost in the generated code.

In this article we describe a code generator technique, which uses a UML model as a starting point and generates the following layers directly:

- Presentation layer (clients), which provides the user interface.
- Business logic layer. This can be a physically and geographically distributed N-tier architecture:
  - Business objects: correspondents of business entities. They implement the basic behaviour of the entities including persistency.
  - Business processes: a series of operations on the business objects.
- Storage layer: either relational or object-oriented databases, special storage systems and legacy systems.

Meanwhile generating the code it preserves the original inter- layer relationships originated in the model.

Based on the experiences with 4GL systems it is obvious the need to provide customisation in the generated code. We offer a multi-paradigm approach [1] to let the developer to choose the appropriate solution for her or his implementation:

- Polymorph code sections: the calling part requests a general behaviour, the called code section substitutes a specialised version according to the context. [2]
- Run-time parameterisation of the generated code: the calling part requests a specialised behaviour using a metadata parameter in the routine call.
- Aspect weaving: the modifications (new aspects) of the code are placed in a separated source file, which is woven into the generated code before the compilation. [3]

References

[1] Jim Coplien: Multi Paradigm Design for C++, 1998, Addison- Wesley, ISBN: 0-201-82467-1

[2] Barbara Liskov: Data Abstraction and Hierarchy, SIGPLAN Notices 23, 5. May 1988

[3] Gregor Kiczales: Aspect Oriented Programming, AOP Computing surveys 28(es), 1996 p.154

[4] This work has been supported by the Grants OMFB ALK-00229/98