GENERATING JAVA CODE FROM OBJECT-PROCESS LANGUAGE SCRIPT - PROJECT MSc. THESIS

Iris Reinhartz-Berger

Supervisor: Dr. Dov Dori

Abstract

The life cycle of developing a system in general and an informational system in particular includes three phases: analysis, design and implementation. The implementation phase transforms the analysis and design results into database schema and executable code. Many case tools provide automatic implementation generators in general and code generators in particular. The benefits of automatic code generation includes increasing productivity and quality, enabling mechanical and repetitive operations to be done quickly, reliably and uniformly, relieving designers from mundane tasks so they can focus on essence and enforcing programmers to write structural legible code. In addition, industrial experience indicates that most of the complexity is at the schematic level and not in the detailed code writing.

There are several approaches for automatic code generation in case tools.

In the 100% code generation approach the code of the implementation is derived completely from the analysis and design products. This approach frees the user from having to know several programming languages, several database systems, writing structural code, etc.

The skeleton generation approach does not expect the software to replace the programmers, only to help them. Therefore the generated code is only a skeleton, which helps the programmers to write structural code.

The concept generation approach divides the concepts into two groups: analysis and design concepts on one-hand and implementation concepts on the other hand. The code generator generates only the concepts, which belong to both groups. Some case tools claim that using this approach can yield 70% of the application code. This approach is the most commonly used and therefore is used throughout this work.

The purpose of this work is to define and develop an automatic code generator for systems, which are analyzed and designed using Object-Process Methodology (OPM).

OPM unifies the system's structure and behavior throughout the analysis, design and implementation of the system within one frame of reference.

The inputs for the code generator are OPL script - a description of the system in an English-like language (Object-Process Language – OPL) and a relational database schema corresponding to the same OPL script.

The output of the code generator is Java executable code, which runs the system. Java was selected as a target language of the code generator since it is simple, object-oriented, network savvy, robust, secure, architecture neutral, portable, multithreaded, dynamic and database connective.

The work consists of three main parts. The first part describes OPM using the meta-modeling technique, i.e. OPM is modeled through diagrams (Object-Process Diagrams – OPDs) and descriptions in OPL. The second part specifies the OPM-to-Java Library, which includes a Java class for each key concept in OPM (object, process, event, state, etc.). The last part defines rules for converting each OPL sentence to a piece of Java executable code, using an existing relational database of the system.

A comprehensive case study demonstrates the application of the code generator.