Experience Report

Reengineering an Ada95-programmed Command and Control Information System by Using UML

Heinz Faßbender
Research Establishment for Applied Sciences
Research Institute for Communication, Information Processing, and Ergonomics
Neuenahrer Straße 20
D-53343 Wachtberg-Werthhoven
Germany
Telephone: (+49) 228 9435 640
Fax: (+49) 228 9435 685
E-mail address: fassbender@fgan.de

Abstract

We describe the concepts and experiences we have made in an ongoing project by modeling and reengineering an experimental command and control information system which is nearly completely implemented in Ada95. For this purpose, we use the UML tool Software through Pictures from Aonix which offers a reengineering component that produces class diagrams from the source code. These diagrams serve as a starting point of a model of the complete system which itself serves as a base for modeling the most important requirements of the system. Furthermore, by refining the model through state-transition diagrams and redesigning the system the UML tool allows an automatic code generation for components which will be implemented in an other language like Java. Hence, the result of the reengineering process should be a more structured system, where proprietary solutions are replaced by commercial products (e.g. an application server) as much as possible.
1. The Integration Platform for Command and Control Information Systems INFIS

In our department Command and Control Information Systems of our institute an integration platform for command and control information systems called INFIS, has been developed which serves as a testbed in international tests for interoperability of command and control information systems. The system is structured as a three tier architecture as illustrated in Figure 1.

![Figure 1. Structure of INFIS](image)

We use the ORACLE 8 system as the database system DBS. It is connected by a proprietary implementation to the CONTROLLER which is completely implemented in Ada95. The CONTROLLER implements the application logic and the connection to other systems. The graphical user interfaces GUIs are implemented as a Java-applet and they are connected to the CONTROLLER by a CORBA-Interface. This realizes the platform independent access to the system.

The CONTROLLER consists of nearly 300,000 lines of code in 1,100 Ada95 units. By historic reasons this system has been developed without developing a detailed model of it. Furthermore, a lot of mechanisms which are now applicable in commercial of the shelf products, like application servers, are implemented by proprietary solutions. These are the reasons why we have started to reengineer the system by the process which will be described in the following section.

2. Reengineering Process for INFIS

We start from the actual system and use the reengineering component of the UML tool Software through Pictures from Aonix. This tool produces a lot of class diagrams and class tables. These diagrams are combined with manually developed diagrams of INFIS’ high-level structures. The resulting diagrams serve as the basic model. Because of the syntactical structure of Ada95 programs which is not pure object oriented compared to Java programs, the classes in the basic model only consist of attributes. Thus, the methods have to be manually added to the classes of the models. Nevertheless, the automatic generation of the models saves a lot of time for modeling and reduces the number of mistakes in the models.

By adding the methods to the models’ classes, some requirements to the system which have not been explicitly expressed before, are fixed and illustrated by use case, sequence, and activity diagrams. Starting from this analysis of the system, the redesigning process will be started.
Therefore, it will be decided which components of the system are replaced by commercial ones, e.g. by an application server and which components are restructured and reimplemented in a more modern way, e.g. as Enterprise Java Beans.

The reimplementation will be supported by the UML tool *Software through Pictures*, too. Aonix has developed a new code generation technique called *Architecture Centric Design* that generates code from class models, as well as from state-transition diagrams. This should increase the amount of code that can be automatically generated, from 10% to 50%. Furthermore, Architecture Centric Design enables the programmer to define the structure of the produced code by defining templates in the *Transformation Description Language*.

The reengineering process is illustrated as a state-transition diagram in Figure 2.

![Figure 2. Reengineering Process for INFIS](image-url)