USING ADA SOURCE CODE GENERATORS IN A LARGE PROJECT

Rob Duell, Hugo J. Sebel, Franklin C. A. de Wit

Royal Netherlands Navy
Center for Automation of
Weapon and Command Systems

Abstract

In large software projects, especially for heterogeneous systems, good support for parallel Application Function development is essential. For this reason the RNLN/CAWCS has developed a concept in which there is a clear split in system-wide software infrastructure (related to ISO layers 2-6) and multiple, domain specific Application Functions.

The approach taken comprises building information models for the complete hardware architecture, the software architecture, and the system level information (emanating from the Application Function Information Modeling activities). Major standards such as the OSI stack have been adopted. The second step in the process was to build one large database application from these models. Code generators then were produced for the different end products, such as ADA source code and \LaTeX{} source files for documentation.

The adopted strategy has already proven itself for CAWCS-built systems, especially in the areas of a rapidly maturing process for the code generator, high quality code and superior flexibility in a System Integration phase. The produced software components are specific in the sense that only the required infrastructure is generated for each Application Function in each computer, instead of one generic general purpose package.

1 Introduction

The area of Naval Combat Systems is no exception to the phenomenon of the ever increasing demand for a higher level of automation. Taking into account the decreasing personnel and equipment budgets, it is obvious that this calls for new strategies and software engineering concepts.

The production of more lines of source code (SLOCs), reflecting the increase in demand for such a higher level of automation, with the same or even smaller staff necessitates the use of previously developed and tested parts of software (reuse) and a considerable increase of the overall quality of newly produced parts. The same fault density, expressed in undiscovered bugs per SLOC, will lead to a larger absolute number of undiscovered bugs when more SLOCs are
produced. This problem can be alleviated by introducing new system approaches and software production techniques.

In this paper we present the approach taken by the development team of the software for the Royal Netherlands Navy M-Frigate and focus on the use of Ada source code generators as a means of both increasing the productivity per capita and the overall quality of the produced code.

# 2 M-Frigate Software Architecture

## 2.1 Overview

Figure 1 depicts the M-Frigate Software Architecture as a three dimensional model consisting of layers.

![M-Frigate Software Architecture Diagram](image)

The basic layer encompasses the hardware units used, e.g. computers, processor boards and networks. On top of this layer we find the software units such as operating systems and the M-Frigate software. The software in itself is layered. There exists a layering from application or system component (slice in front), through the operating system independent interface to the operating system itself. The front slice is also layered and relates to the ISO/OSI Reference Model.