"Electronic Maneuvering Board and Dead Reckoning Tracer Decision Aid for the Officer of the Deck"

LT Kenneth L. Ehresman, United States Navy
B.S., University of Maryland
Masters of Science in Computer Science-September 2001
LT Joey L. Frantzen, United States Navy
B.S., United States Naval Academy, 1994
Master of Science in Computer Science-September 2001
Advisor: Richard Riehle, Naval Postgraduate School

Problem Statement and Abstract

The U.S. Navy currently bases the majority of our contact management decisions around a time and manning intensive paper-based Maneuvering Board (MOBOARD) process. The use of Maneuvering Boards is a perishable skill that has a steep learning curve. In order to overcome inherent human error, it is not uncommon to have up to four people simultaneously involved in solving just one maneuvering problem. Additional manning requirements are involved on many Naval Ships in order to accurately convey the information to the Officer of the Deck (OOD) and/or the Commanding Officer. When given situations where there exist multiple contacts, the current system is quickly overwhelmed and may not provide Commanding Officers and OODs a complete and accurate picture in a timely manner.

The purpose of this research is to implement a stand-alone system that will provide timely and accurate contact information for U.S. Navy Commanding Officers, OODs, and CIC watch teams. By creating a reliable, automated system in a format that is familiar to all Surface Warfare Officers we will provide the Navy with a valuable decision-making tool, while increasing ease of data exchange and reducing current redundancies and manning inefficient practices.

Our software design is diagramed using the Unified Modeling Language (UML) with the underlying purpose of implementing an Ada-based system that is neither operating system nor hardware dependent. This approach allows us to develop and implement a design for a wide

range of platforms, and will ultimately result in an extremely modular and portable system.

Model-View-Controller

Additionally, we approached our project using the Model View Controller (MVC). This approach has allowed for flexibility in the current and future use of our core model. Not only does the model meet today's current needs, it is highly extensible and will meet emerging needs for many years to come.

In order to meet all of our criterion (i.e. operating system independent and hardware independent) we have implemented our project using GtkAda libraries and a GNAT compiler. Although GNAT makes it possible to link to other languages, we have avoided the use of other languages. This approach has resulted in a robust program that compiles and runs on a wide variety of platforms. The ultimate goal of our thesis is to produce an automated navigation system for the U.S. Navy that will help the Navy to meet reduced manning requirements while increasing the accuracy and reliability of its navigation and contact management systems.

The MOBOARD software takes advantage of Ada's capability to separate the system specification from its implementation. In this design, the specification is designed to represent only those features necessary for the required features. That is, designing the context clauses at the specification level, we moved them to the package body so there would be a minimum of dependency on the graphics packages and other platform-specific library units. This has facilitates a platform-independent presentation of multiple views of the same data in

Of particular interest to some Ada practitioners will be our use of the floating-point facilities of Ada 95. We have made extensive use of the floating-point attributes as well as the generic elementary functions package. This was necessary because we are doing a lot of calculations

that involve global coordinates, the movement of targets relative to a reference point, navigation calculations, prediction of position, and timing.

Ada has proven useful in this project because of its underlying language structure. This structure has allowed us to decompose the software architecture along:

1) A coherent conceptual view
2) A platform-independent module view
3) A maintainable code view
4) A platform-targeted execution view.

Each of these views evolves naturally as we progress from the UML use cases through the class and interaction diagrams, and onward to the Ada specifications.

Future Work

We expect to see this project evolve further into a set of additional views (an extensibility built in to our design). We also expect that new computations will be added to enhance the sophistication of our initial design.