

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

Kenneth L. Ehresman
U.S. Navy
3030 Eddy Street
Marina, CA 93933
+1 831.384.8960

Klehresman@aol.com

Joey L. Frantzen
U.S. Navy
664 Bailey Court
Marina, CA 93933
+1 831.884.9713

Joeyf40804@aol.com

1. 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.
To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGAda 2001 09/01 Bloomington, MN, USA
© 2001 ACM 1-58113-392-8/01/0009...\$5.00

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 approach facilitates a platform-independent presentation of multiple views of the same data.

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.

1.1 Keywords

U.S. Navy, Maneuvering Board, Navigation, Contact Avoidance, GtkAda, GNAT, Model-View-Controller, Officer of the Deck Aid.

2. INTRODUCTION

Prior to Maneuvering Boards, the traditional mariner relied upon the seaman’s eye and the knowledge gained from many hours of standing watches on the bridge to help determine what was the right decision to make when confronted with other vessels. The evolution of radar allowed vessels to see contacts at great distances and measure the bearing and ranges of those contacts. The Maneuvering Board quickly followed the radar allowing ship drivers an alternate visual representation of the radar contacts based upon basic trigonometric fundamentals. This allowed Officer of the Decks and Commanding Officers a better way to frame the problem in more concrete terms.

The OOD decision-making process is designed to try and reduce uncertainty by gathering information, and transforming this information into knowledge and understanding. The utilization of radars and Maneuvering Boards aids a Commander/OOD in reducing the level of uncertainty. This process is known as the OODA process: Observation, Orientation, Decision, and Action. Whenever trying to establish Command and Control there are two fundamental factors that shape the environment: uncertainty and time. The Maneuvering Board model lies within the Orientation phase of the OODA Loop. See Figure 1 and Figure 2.

The Electronic Maneuvering Board and Dead Reckoning Tracer Decision Aid will reduce the level of uncertainty and amount of time inherent to the Maneuvering Board process.

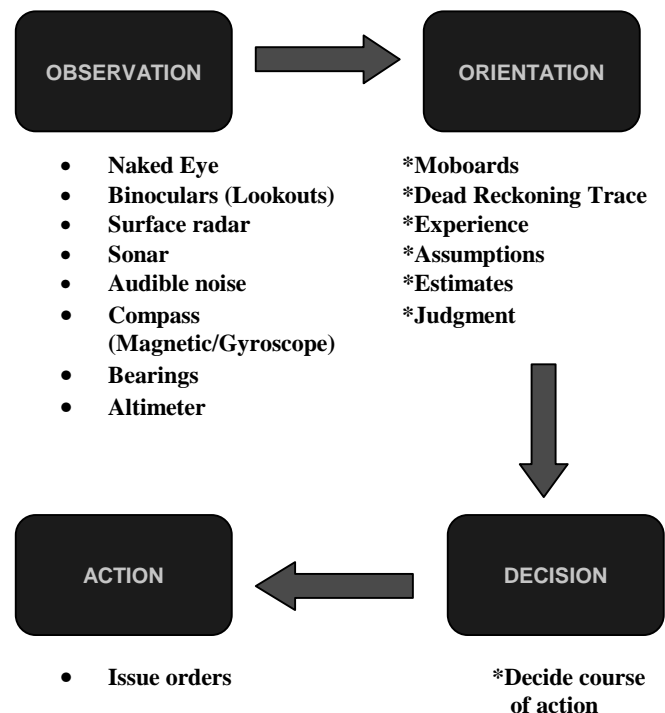


Figure 1

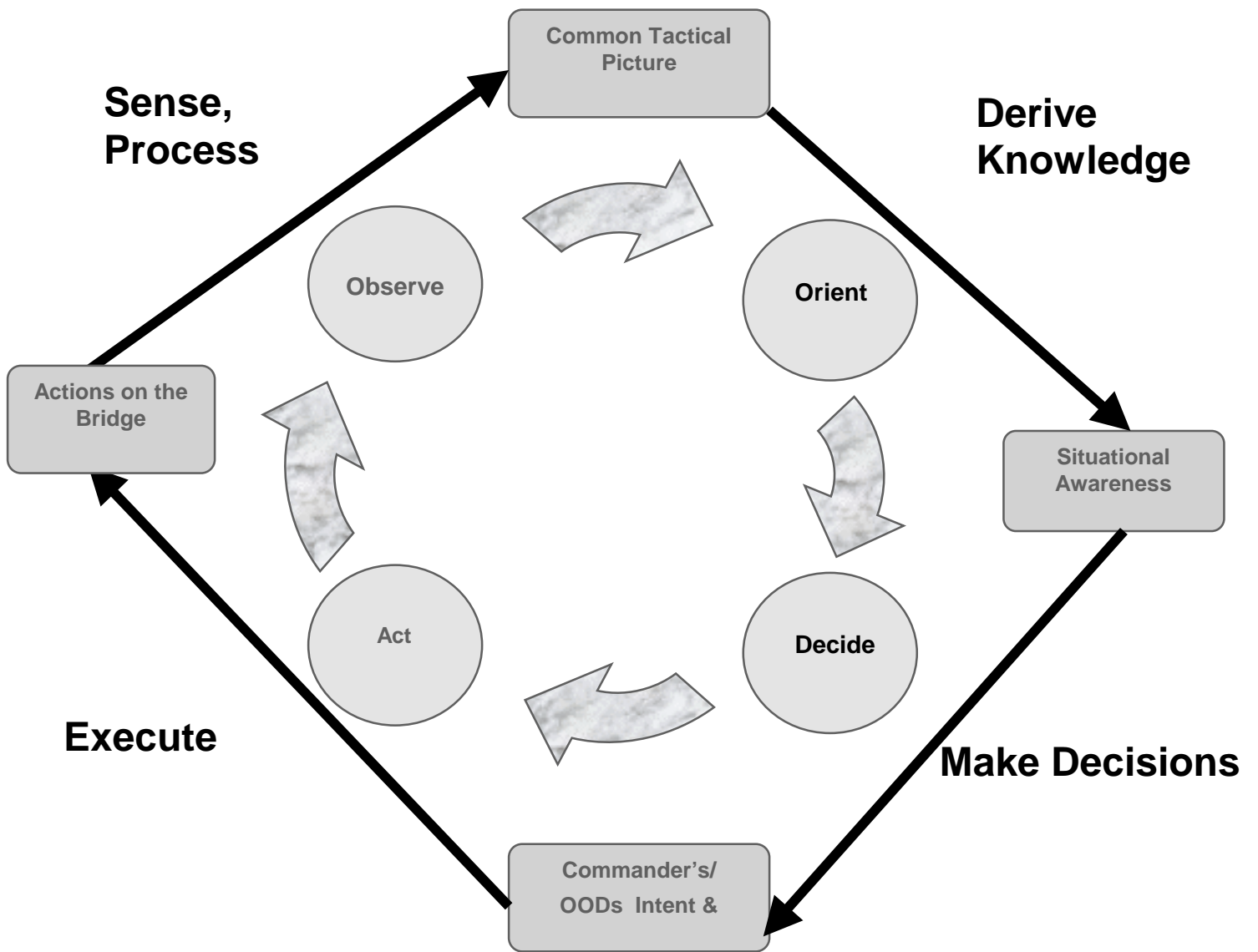


Figure 2

2.1 Observation

Observes the environment (using all sensors, information systems, and situational reports from his subordinates) to collect data about their surroundings and the status of contacts. This data is typically correlated, fused, and displayed in a common tactical picture—a representation or image of the contact space. A Commander or OOD had several methods of retrieving data via visual lookouts, surface radars, sonar, and/or his/her own eyes.

2.2 Orientation

A Commander/OOD orients themselves to the environment—that is, they form a mental picture of the situation—by converting sensor data and other information into estimates, assumptions, and judgments about what is happening. From this orientation a commander/OOD derives his understanding of the contact space, or situational awareness.

2.3 Decision

Based on the understanding derived from his/her Orientation, the commander/OOD then decides on a course of action and comes up with a plan.

2.4 Action

The commander/OOD sets forth his intent and issues orders to put that plan into action.

3. TRADITIONAL VS. DIGITAL

The argument for or against traditional paper-based Moboards versus Digital-based Moboards is based upon two simple factors. Digital-based Moboards are more accurate and take less time to calculate a maneuvering problem compared to paper-based Moboards.

Traditional paper-based Moboards are done with a pencil and straightedge. This process can be inaccurate and is often prone to human error. Even a very experienced sailor can make mistakes when doing a Moboard solution, especially in time critical situations and/or periods of rough seas.

Traditional paper-based Moboards can be slow and time consuming. It is not uncommon to see up to four junior officers working on Moboards during advanced maneuvering situations. Based upon their level of experience the resulting solution may or may not be an accurate or timely one.

Digital-based Moboards will speed this process up and eliminate some of the human-based errors inherent to the paper-based Moboard process. By decreasing the time required to produce a Moboard solution it in turn decreases the time required to complete the Orientation process and thus speeds up the overall OODA Loop decision process.

4. REDUCING MANNING REQUIREMENTS

With the evolution of “Smart Ship” and the DD-21 initiative the manning of Navy Ships has become a high profile issue. The future Navy will no longer have the luxury of 350 manned ships. The Navy of the future will require less men and woman who are more technically proficient and better trained. The bridge of the next generation will still rely upon good seamanship, experience, and a well-trained eye. But instead of using paper-based tools, the tasks and aides used to process contacts and information will be done in a digital-based medium.

The modern navy will have to depend on exceptional sensors and computer systems that are able to frame an abundance of information into a manageable and clear presentation. The Electronic Maneuvering Board and Dead Reckoning Tracer Decision Aid is designed to do just that. With this Computerized Decision Aid the requirements for multiple Junior Officers doing Moboards or several Operation Specialist in Combat maintaining a DRT contact picture will be greatly reduced. GPS will automatically be entered into the computer system, instantly giving the Commanding Officer and OOD Latitude and Longitude information on all contacts begin maintained in the database.

Additionally our computer program will have the ability to maintain a digital log, vice having to use a paper-based log currently in use by the U.S. Navy. This may be another avenue by which the U.S. Navy can reduce the manning requirements on the bridge. The Quartermaster will no longer be required to log each course and speed change, Officer of the Deck (OOD) watch changes, casualties, etc. This will all be maintained in the central database allowing for a visual playback of events for any post-operations analysis. This feature will allow the evaluator to view a list of events as well as display a visual contact picture chronologically corresponding with these logged events. Thus, the end result is a better post-operations analysis and understanding of the environment on the bridge during post-operation, post-mishap, or post-exercise proceedings.

4.1 Future Work

Future work will include a set of additional views(an extensibility built into our design). Additionally, our program will contain the functionality to alert the OOD and CO when a contact will be within in the set distance based upon the Commanding Officers Standing Orders. An alert will be generated signaling the OOD and/or CO that the designated contact will be within minimum distances set. The program will include functionality that gives the OOD recommendations based upon the Rules of the Road library. This library will alert the OOD/CO of important Rules of the Road and give recommended course and speed changes based upon those rules and the contacts course and speed. The default being, maneuver the ship to maintain minimum

set Closest Point of Approach (CPA). The computer program will also alert the OOD/CO if a course and speed change will affect the CPA of other contacts, or affect the safe navigation of ownship. These enhancements will be a guide/aid to the OOD/CO designed to promote safe navigation at sea and prevent collisions and human-errors that result in collisions at sea.

5. OVERALL SOFTWARE DESIGN

5.1 Unified Modeling Language (UML)

Our software design is based upon the Unified Modeling Language (UML). UML allows us to construct a software model that is supported by the ADA programming language. UML also provides significant benefits to us, as software engineers, by helping to build rigorous, traceable and maintainable models that will support the software development cycle. Our design is based upon these fundamental tenants: Non-Operating System dependent, Non-Hardware System dependent, Extensible and Modular design. ADA provides a certified compiler and environment, making our code robust and assuring the “buyer” that the program does what we advertise it to do. We also chose ADA because of the Re-usability inherent to the modular design structure. Our program does not use hardware specific libraries/architecture such as MFC.

Our design was based upon the following UML diagrams. Use-Case, Sequence, Class, Object, and Deployment Diagrams.

5.2 Use-Case Diagram

The Use-Case Diagram was based upon what an OOD or CO would require the computer system to do. Basic functions such as Plot Contact, Display Contact, Calculate Contact Course and Speed, and Calculate CPA were the primary Use-Cases for the basic computer program. Other actors such as Global Positioning System and Generic Radar System will be required for implementation at a later date. All of these actors interact with two systems, Dead Reckoning Trace and Maneuvering Board. These two systems are just separate views of the same data, framing our visual representation for the user.

5.3 Class Structure

To truly make a computer program reusable, modular, and maintainable the programmer must adhere to a strict Object Oriented Methodology (OOM). Understanding this key concept, the programmer must build his/her classes in an Object Oriented approach. With this in mind, our Electronic Moboard and DRT was structured to be modular, maintainable, and reusable.

5.3.1 The basic classes

5.3.1.1 Moboard Class

Description: The Moboard Class displays OwnShip and Track Class Data in a relative 0 to 360 degree coordinate system scale based upon the speeds of the vectors.

5.3.1.2 DRT Class

Description: The DRT Class is similar to the Moboard Class in that it displays information from the Ownship and Tracks Class. The main difference is the DRT Class presents a true picture vice a relative picture presented in the Moboard Class.

5.3.1.3 Date Class

Description: The Date Class contains a type Date (Day, Month, Year). The Class includes Functions and Procedures to Get a Date from the computer system or GPS, as well as, return all the Date values and set all the Date values.

5.3.1.4 Time Class

Description: The Time Class contains a type Tim (Hours, Minutes, Seconds). The Class includes Functions and Procedures to Get a Time from the computer system or GPS, as well as, return all the Time values and set all the Time values.

5.3.1.5 Latitude/Longitude Class

Description: The Lat_Long Class contains a type Latitude (Degrees, Minutes, Seconds, Sign) and type Longitude (Degrees, Minutes, Seconds, Sign). The Class includes functions to convert from Nautical Miles to Kilometers, Yards to Kilometers, Yards to Nautical Miles and the converse functions as well. Also included in this class are two unique procedures. The first, Calculates a Latitude and Longitude given a bearing and range from a known Lat/Long Position. This Procedure also returns the back bearing based on trigometric formulas that take into account the curvature of the earth. The other Procedure takes two known Lat/Long Positions and then calculates the distance between them and the forward and back bearings. The Class also contains the basic set and get procedures/ functions for each Latitude and Longitude type.

5.3.1.6 Hit Class

Description: The Hit class contains all of the data necessary when a hit is taken (a bearing and range to a contact from a radar scope). A hit type includes Bearing, Range, Latitude, Longitude, Date, Time, Ownship Course and Speed, Target Course and Speed, and Target Angle. Hit Class contains all the procedures necessary to set and get data from each element of the Hit type.

5.3.1.7 Track Class

Description: The Track Class contains all of the data necessary to track a contact. The track type includes Track Number, Track Ident, Track Course and Speed, CPA Bearing, CPA Range, CPA Time, a Hit Count and a List of Hits. Track includes functionality to set and get all of the elements of hit type. Track also maintains an array of 10,000 elements that are of type track. This is the memory allocation for all of the tracks prior to being saved onto a harddisk or storage device.

5.3.1.8 OwnShip Class

Description: OwnShip Class manages and maintains all information concerning OwnShip. This includes Number, Identification, Course and Speed, Latitude, and Longitude. The OwnShip Class contains all of the required Get and Set functions and procedures necessary to modify and retrieve type data. OwnShip Class also retrieves Latitude and Longitude Information from either manual keyboard entry or from a GPS port(ex. COM1). This is based upon the user selected mode.

5.3.1.9 GPS Class

Description: This Class manages the interface between the GPS System and the Computer. This Class retrieves GPS information from the designated port(ex. COM1) then parses and stores this information into usable formats, i.e. Latitude, Longitude, Date, Time, etc.

5.3.1.10 Speed Class

Description: Defines subtype Speed that is a Real type with digits 1 range 0.0 to 130.0 This maximum values was chosen based on the simple fact that our design of this system is intended to track surface vessels, not aircraft , etc.

5.3.1.11 Degree Class

Description: Defines subtype Degree that is a Real type with digits 1 range 0.0 to 359.9

5.3.1.12 Realnum Class

Description: Defines subtype Real that is digits 12. It also instantiates Generic_Elementary_Functions(Real).

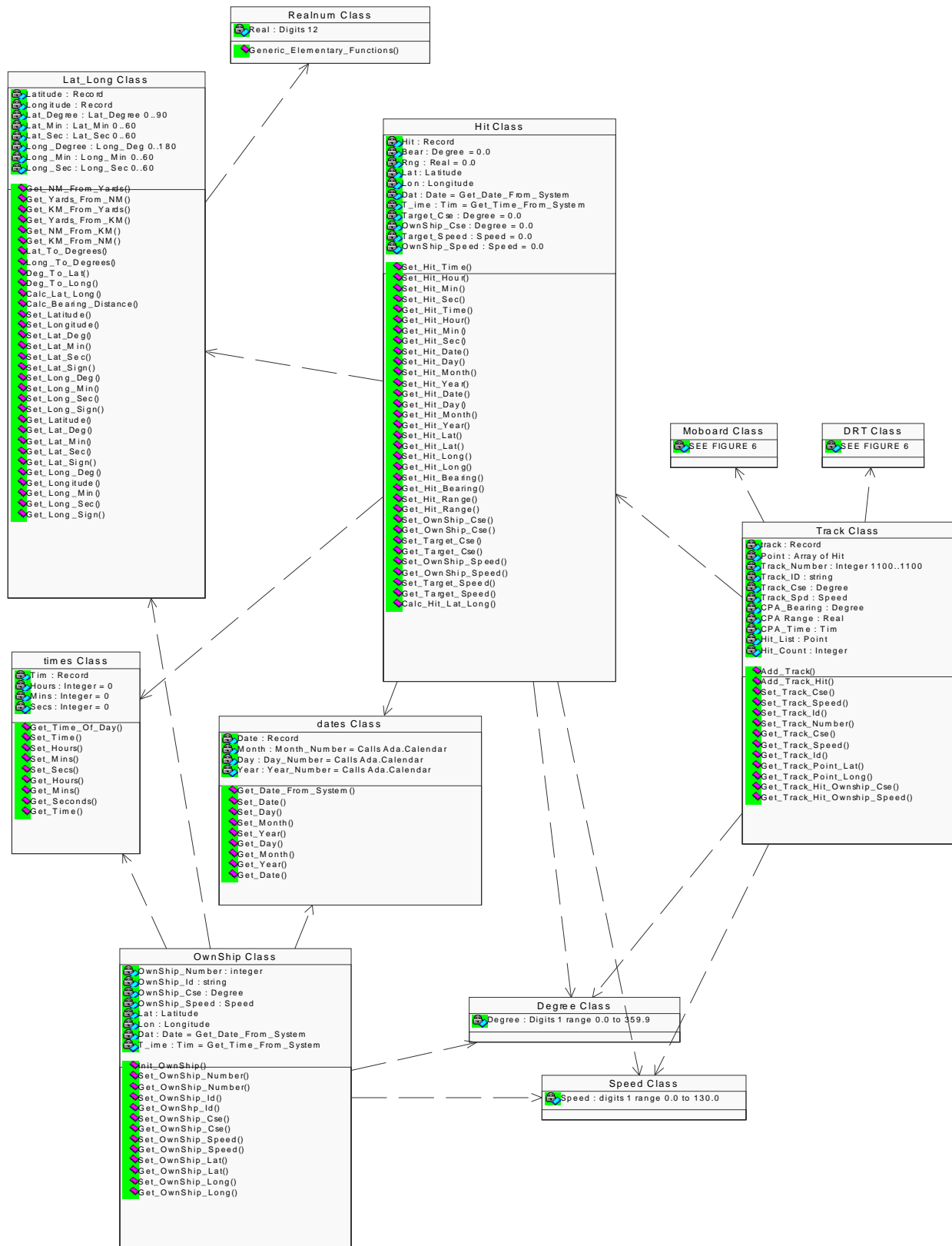
5.3.1.13 Radar Class

Description: This is a future Class that will be designed to handle Radar input of track information and manage the interface between the two systems.

5.3.1.14 Network Class

Description: Manages all of the network traffic and sequence of events required to pass data over a small network.

See Chart on next page.



6. GtkAda and GNAT

There are several reasons we chose to use GtkAda and GNAT Compilers. First off, as students with no funding, both of these compilers were free, a very attractive quality. Secondly, part of our research was to design a computer program that was hardware and software (Operating System) independent, thus portable. GtkAda is a high level portable graphical toolkit based on the gtk+ toolkit and one of the official GNU toolkits. Additionally GtkAda uses Ada95 features and supports Object Orientation. GtkAda supports OpenGL, another attractive feature. GtkAda toolkit had also been tested on the following platforms:

- Linux/x86
- Linux/sparc
- Linux/ppc
- Solaris/sparc
- Solaris/x86
- Dec Unix
- SGI IRIX 6.5
- HP/UX
- NT 4.0
- Windows 2000
- AiX 4.3.2
- SCO UnixWare 7.1
- FreeBSD 3.2

The wonderful feature of GtkAda is that it was designed to be portable from the start. Since our application only calls GtkAda it is completely portable. See Figure 3.

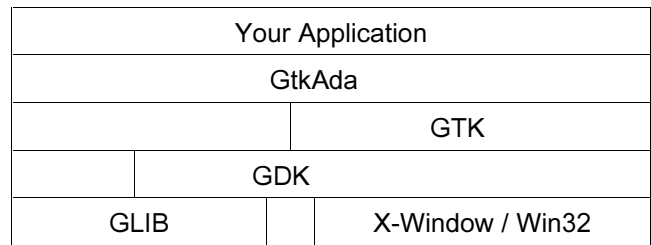
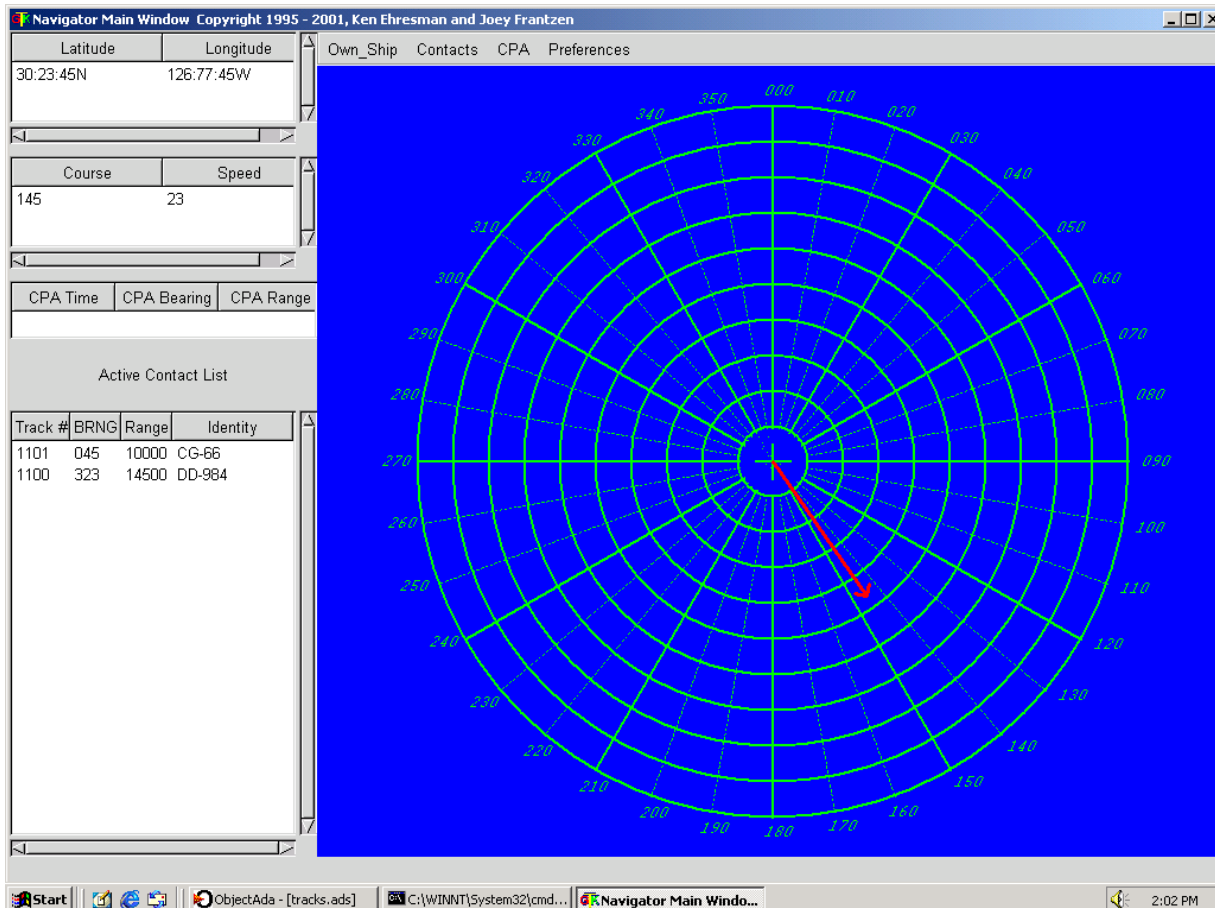


Figure 3

Programming concurrently in Windows 2000 and Linux we are able to demonstrate the portability of our code. Under Windows 2000 we used GtkAda 1.3.11 and GNAT 3.13p for Windows NT. Under Linux Redhat 7.0 we used GtkAda 1.2.10-1 and GNAT 3.13p. As the project progressed code written on the Windows 2000 platform would be transferred for inclusion with the Linux Redhat 7.0 platform written code, or vice versa.

7. SCREEN SHOT PICTURE

Picture of GUI Interface using GtkAda.



8. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

9. REFERENCES

- [1] Riehle, Richard, D. Ada and Object Technology (August 1999).
- [2] ACM SIG PROCEEDINGS template.
<http://www.acm.org/sigs/pubs/proceed/>
- [3] Feldman, M. B. and Koffman, E. B. Ada 95 Problem Solving and Program Design, 2nd Ed. Addison-Wesley Publishing Company, New York, October 1996.
- [4] Cohen, Norman, H. Ada as a second language, 2nd Ed. McGraw-Hill Companies, New York, 1996.
- [5] Schufeldt, H.H. USNR(Ret.) and Dunlap, G.D. Piloting and Dead Reckoning, 3rd Ed. Naval Institute Press, Annapolis, MD. 1991.
- [6] Etter, Delores, M. Structured Fortran 77 for Engineers and Scientists, 5th Ed. Addison-Wesley Publishing Company, New York, 1997.
- [7] Bowditch, Nathaniel. American Practical Navigator—An Epitome of Navigation, Volume 1, 1984 Edition. Defense Mapping Agency Hydrographic/Topographic Center, 1984.
- [8] Clynych, J. R. Naval Postgraduate School. Fortran Algorithm for GEODESIC GDS DIR and GDSINV from Geodesic Routines, 1999.
- [9] Brobecker, J. and Briot, E. and Charlet, A. GtkAda Reference Manual, Version 1.2.10, Document Revision level 1.12, November 20th, 2000.
- [10] Ada Core Technologies, Inc. GNAT User Guide, GNAT for Windows NT, GNAT GNU Ada 95 Compiler, Document Revision Level 1.316, GNAT Version 3.13p, May 5th, 2000.
- [11] Ada Core Technologies, Inc. GNAT Reference Manual, GNAT for Windows NT, GNAT GNU Ada 95 Compiler, Document Revision Level 1.135, GNAT Version 3.13p, May 5th, 2000.
- [12] Brukardt, R.(Editor), Ada 95 Reference Manual, Technical Corrigendum 1 (ISO/IEC 8652:1995/COR1:2000).

