Software Tool for Naval Surface Warfare Simulation and Training

Sergiu Dascalu* Sermsak Buntha* Daniela Saru Narayan Debnath*****

* University of Nevada, Reno, USA
Department of Computer Science and Engineering
1664 N. Virginia St., MS 171
Reno, NV, 89523, USA
E-mail: dascalus@cse.unr.edu
Phone: +1-775-784 4613
Fax: +1-775-784-1877

** University "Politehnica" of Bucharest, Romania
Department of Control and Industrial Informatics
Spl. Independentei 313
Bucharest, Romania
E-mail: saru@aii.pub.ro

*** Winona State University, MN, USA
Department of Computer Science
P.O. Box 5838, 103 Watkins Hall
Winona, MN 55987, USA
E-mail: ndebnath@winona.edu
Phone: +1-507-457-5261

Corresponding Author: Sergiu M. Dascalu, University of Nevada, Reno,
Department of Computer Science and Engineering
1664 N. Virginia St., MS 171, Reno, NV, 89523, USA
E-mail: dascalus@cse.unr.edu, Phone: +1-775-784 4613
Fax: +1-775-784-1877

Software Tool for Naval Surface Warfare Simulation and Training

Sergiu Dascalu* Sermsak Buntha* Daniela Saru** Narayan Debnath***

* University of Nevada, Reno, USA
** University "Politehnica" of Bucharest, Romania
*** Winona State University, MN, USA

Abstract

Determining optimal operations involves performing complex tasks by navy commanders. In this paper we present a software tool with four main functions that facilitate commanders' tasks in naval surface warfare. First, a human-computer interaction function presents battlefield information using information-rich graphical symbols. Second, a command and control function allows commanders to achieve centralized control of naval warfare operation in battlefield situations. Third, an interception function is available to determine the optimum course for intercepting a particular enemy target ship. Fourth, an escape route planner can be invoked to automatically calculate routes that minimize detection by enemy platforms. The paper presents background information on naval surface warfare, excerpts of the proposed software tool's model, and details of the tool's user interface. A brief survey of related projects as well as several pointers to future work are also included.

Keyworkds: naval surface warfare, command and control, simulation, training, UML

1. Introduction

Of paramount importance for naval commanders is to gain advantage over hostile forces. This means the commanders have the major responsibility of clarifying the puzzle of information collected from various sources and transforming it into a comprehensible image. Frequently, they have to take optimal decisions based upon limited information available. Typically, the tasks that support decision making by commanders require numerous staff as well as significant amounts of time. Large paper navigation maps as well as acetate overlays and tactical boards for pencil annotations have traditionally been used to display the summary image of battlefields. Unfortunately, this traditional approach is often inconvenient and can lead to late results, which are no longer useful for commanders.

The work presented in this paper intends to provide an effective software solution for command and control in naval task force for surface warfare. Using a systematic software engineering process [[13]] and the Unified Modeling Language (UML) [1] as major development devices, we have designed and implemented a prototype of Command, Control and Intelligence (C^2I) software tool that runs on Linux. We have enhanced this system with Human-Computer Interaction (HCI) features that facilitate software usage and allow rapid access to intelligence functions that efficiently support commanders in their decision making process. This software tool is designed to be cost-effective by using Commercial Off-the-Shelf (COTS) technology such as personal computers. The intended users of the proposed software are fleet commanders, commanding officers (ship captains), and high-level navy leaders in charge of commanders and commanding officers. In its current stage, the prototype software presented in this paper can be used for training and simulation purposes.

1.1 Naval Surface Warfare

Naval surface warfare is an area of naval warfare which involves surface fleet's ships for deploying navy companies, controlling sea voyages, and providing support power for on-shore troops. Naval activities include the typical tasks of naval forces, primarily maintenance, deployment, attack, and defense. Naval surface warfare encompasses four dimensions of warfare: anti-air warfare, anti-surface warfare, anti-submarine warfare, and amphibious warfare.

Because they are very important activities in a surface warship, the operations are always commanded and controlled by the Combat Information Center (CIC). The sequence procedure of CIC starts from *collecting* as much data as possible, based on information available from various kinds of sources. Such sources include surveillance and detection sensors, intelligence agents, and so forth. Then, the data is *processed* (analyzed) and *displayed* to commanders. The commanders *evaluate* the situation and then *distribute* information and *dispatch* commands. These kinds of tasks require significant resources in terms of operational personnel and processing time. In particular, for fast processing, a rather large number of staff is required. However, in more complex situations, the staff who work to provide information for commanders have difficulty keeping pace with battlefield dynamics using only conventional methods. Currently, advanced computer technology can significantly improve traditional methods of surface warfare. The present paper proposes a solution in this direction.

1.2 Command and Control, and Intelligence (C^2I)

Command and Control (C^2) are vital, complex tasks involved in military operations. Out of necessity, C^2 has long ago become a significant part of the Navy. C^2 plays a critical role in operating a warship. If we compare a warship to a human body, the C^2 system is the equivalent of the human brain. To gain advantage over enemies, efficient decision-making is essential and hence sophisticated processes (tasks) are required. These tasks necessitate a considerable amount of staff support. Therefore, to have an effective "brain" in the scope of naval power, Command Control and Intelligence (C^2I) is needed.

1.3 Paper Structure

The remaining of this paper provides background information on work related to C^2I systems and presents details of developing a C^2I support software by using UML as modeling notation. Also, the paper demonstrates the software prototype (tool) via snapshots of the tool "in action" and descriptions of its user interface.
More specifically, Section 2 provides background information on several key aspects pertaining to the proposed software solution. Section 3 describes the mechanisms used for enhancing the prototype tool from an HCI point of view. Details of the algorithms that support intelligent functions included in the proposed prototype are also provided in this section. Section 4 presents excerpts of the UML-based software model of the prototype. Section 5 demonstrates the prototype built via snapshots of its user-interface. Section 6 contains pointers to future work and a summary of the paper's contributions.

2. Background

In this section, background information is provided on: Naval Tactical Data System (NTDS), Genetic Algorithms (GAs), and the Unified Modeling Language (UML). Work related to our approach is also briefly surveyed.

2.1 Naval Tactical Data System

Naval Tactical Data System (NTDS) is a shipboard device used in battlefield and training situations for displaying tactical data [[15]]. With NTDS, combat data obtained from various kinds of sources is collected. Then, the data is processed by discarding redundant data, correcting erroneous information, and completing deficient data. The data thus processed is transformed into a battlefield picture. This picture is important because it allows the commander to evaluate the situation and decide on tactical actions. Table I presents several symbols used in NTDS.

Table I: *Symbols used in NTDS*

Identify / Type	Unknown	Friendly	Hostile	Neutral	Our ship
Aircraft					
Surface ship					
Submarine					

2.2 Genetic Algorithms

Genetic Algorithms (GAs) constitute a branch of Artificial Intelligence (AI). GAs are search algorithms based on biological mechanisms of natural evolution and selection. They work with a set of input binaries (*chromosomes*) to maximize the value (*fitness value*) of an unknown function (*fitness function*). With three basic operations—*reproduction*, *crossover*, and *mutation*—genetic algorithms manipulate a set of chromosomes (population) with the goal of generating an individual chromosome that is characterized by higher fitness [[8]].Figure 1 shows the general steps of regular GAs. At the initial point, the parents are randomly created. After the parents' chromosomes pass the process of reproduction, crossover, and mutation, the children are produced. After that, the children are evaluated and selected by using the fitness function. In this operation, the survival or death of each child is decided. Those who survive will be the parents of the next generation. This loop continues until a satisfactory value of a child's fitness is obtained or the number of generations reaches the pre-set limitation of the algorithm.

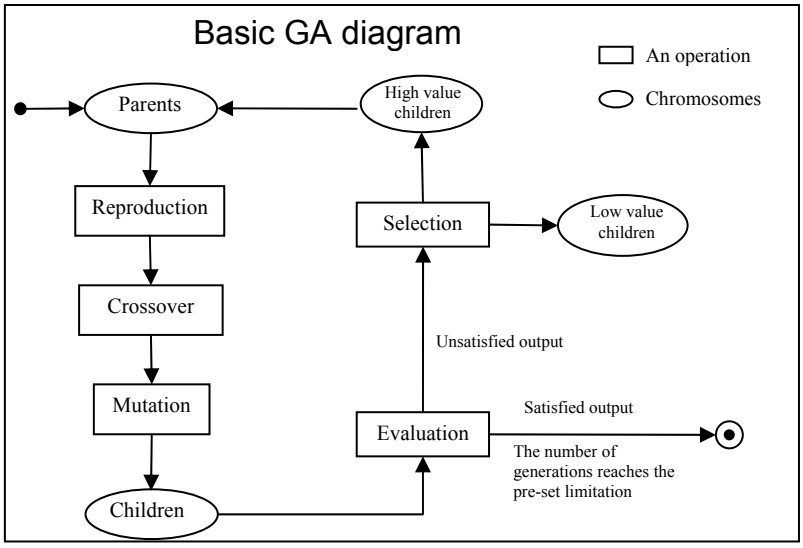

Figure 1: The basic procedure of GAs

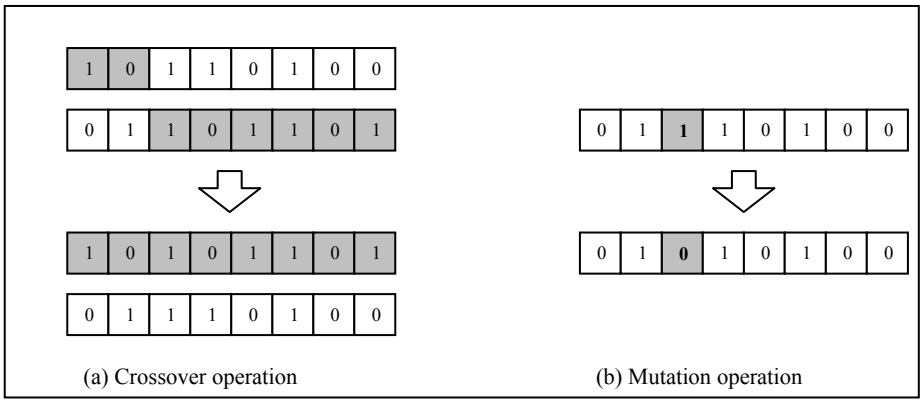

Figure 2: Crossover and mutation operations

The reproduction operator selects some good individual chromosomes from the population to be the parent chromosomes for the next generation. The crossover operator crosses over among selected populations to create new offspring chromosomes. Finally, the mutation operator allows traveling around some areas that have not been explored. Figure 2 illustrates (a) the crossover, and (b) the mutation operators.

There are many applications that are based on GAs. For example, uses of GAs for military purposes are described in [11].

2.3 The Unified Modeling Language

Originally introduced in 1997 by Grady Booch, James Rumbaugh, and Ivar Jacobson, the Unified Modeling Language (UML) has been rapidly and widely accepted by the software engineering community as "the standard graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive systems" [[1]]. In our work, we have used diagrams in the UML notation to design the proposed software tool for naval surface warfare. The nine most important diagrams of UML are [1]: class diagrams, object diagrams, use case diagrams, sequence diagrams, collaboration diagrams, statechart diagrams, activity diagrams, component diagrams, and deployment diagrams. Samples of these are shown in Section 4 of the paper.

2.4 Related Work

For more than a decade, there have been many researchers in the area we also have endeavored to explore. The vast majority of them have been employed by the Department of Defense (DoD), and have worked on various projects [2, 3, 4, 6, 7, 9, 10]. The following describes several examples of related work developed by the DoD or with DoD support.

In 1983, a prototype for naval Battle Group Simulation (BGS) was developed by Jerry Golub and Willis A. Soper [9]. A simulation prototype was created for evaluating software design, user interfaces, and other features and details of various models. The BGS prototype was important in that it offered a development baseline which can be characterized as a single radar and missile system.

In 1995, Curtis L Blais presented the Marine Air Ground Task Force (MAGTF) Tactical Warfare Simulation (MTWS) [2, 3]. The purpose of this simulation system was to support the training and the preparation of the U.S. Marine Corps (USMC) commanders and their staff. Blais proposed the main mechanisms of the system and indicated ways for improving the simulation [2]. In later work, published in 1998, Blais presented a prototype system for advanced warfare gaming

capabilities, aimed to be used in the 21st century warfighter [3]. This prototype system, entitled the Joint Simulation System (JSIMS), was designed for the training of commanders and other DoD staffs. The main goals of the training with this system were to enhance commander and staff interaction with the system and to support the development of intelligent military tactics. Remarkably, the system enabled a smaller number of exercise control personnel to prepare, organize, conduct, and assess the results of more complex exercises. The system was intended to support the procedures of the Command Control Communications Computers and Intelligence (C^4I) system, a system that is used in practice by commanding staff in almost all US military units.

In 2001, George F. Stone and Georgy A. McIntyre published their work on the Joint Warfare System (JWARS) [14], a system that proposes a campaign-level model of military operations. This system has been developed by the US Office of the Secretary of Defense (OSD) with the purpose that the results of this project will be used by the OSD, the Joint Staff, and the war fighting Commands. The JWARS is intended to provide users with a complex representation of joint warfare, including elements of operational planning and execution. There are three functions included in the system, dealing respectively with: C^4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance) systems and processes, the impact of logistics in combat areas, and operational-level aspects of maneuver warfare. However, [14] emphasizes only top-level concepts and highlights some of the significant problems and limitations of existing solutions for these types of functions. The JWARS consists of three software domains, namely *problem, simulation,* and *platform*. All three domains are integrated into a single executable package. Firstly, in the JWARS problem domain, the Battle Space Entities (BSEs) are assigned as battalion-level for maneuver units, air mission elements for air operations, ships for maritime assets, and individual platforms for critical ISR systems. One of the most significant advantages of JWARS is to allow the simulation of forces' activities before the battle starts. Secondly, the simulation domain provides a comprehensive range of synthetic theater environment details such as terrain roughness, weather, movement of infrastructure, wind conditions, and so forth. Lastly, the platform domain provides details on the JWARS hardware as well on the system's human-computer interface.

Because of insufficient technologies and high cost of existing military computers, it becomes practical and efficient to replace them with COTS. Indeed, there have been several research attempts to employ COTS computers instead of military-type computers in command, control and intelligence systems [7, 12].

In 2002, Sushil Louis, a faculty member of the University of Nevada, Reno, in collaboration with John McDonnell and Nick Gizzi from Space and Naval Warfare (SPAWAR) Systems Center introduced a dynamic strike force asset allocation using genetic search and case-based reasoning [11]. The system commands and controls air strike fighters to go through enemy areas so that they can be close enough to launch missiles at hostile targets. The system supports destruction of enemy air defenses in real time. The authors applied an advanced genetic search algorithm, called *case injection to bias genetic algorithm*, to provide a plan. The basic idea of this genetic search algorithm is that the system initially searches for pre-selected points instead of random points.

3. Proposed Software Tool

The main functions of the system can be divided into four categories: human-computer interaction display, command and control center, interception, and escape route planner.

3.1 Human-Computer Interaction Display

Of paramount importance for efficient command and control is to provide valuable information to commanders who have been invested with central authorities. In warfare conditions, it is necessary that commanders understand the details of their own forces as well as of the enemy forces. Furthermore, the information provided to them must be continuously, constantly, and rapidly updated.

The ideal supporting software tool shall depict the curent events of a battlefield on a computer screen or a projector. In our solution, we have adopted NTDS symbols (which most US Navy operational personnel are familiar with) as symbols to be displayed on the screen. However, with modern computer technology available, we have extended the traditional set of NTDS symbols with several new symbols that allow for real time display and manipulation. Furthermore, in order to keep the Combat Information Center (CIC) room dark we use a black background screen instead of a white one. As such, the relevant information and alarm signals that always display with colorful symbols such as red, white, blue, and green can more easily be noticed. To make the screen look uncomplicated, the tool usually displays only the platforms' symbols as well as their tracks and speed vectors to show current positions, previous positions, and movements. However, the minute representation details can be accessed with ease via mouse clicks. In addition, zoom-in and zoom-out options are available to adjust the level of details shown on screen.

3.2 Command & Control Center

One major goal of the software tool we developed has been to centralize combat command and control. The tool allows commanders to centralize control tactical operations and thus it represents a command & control center. The commanders can maneuver their own platform and they can control sensors, weapons, and operation activities by providing fire control to engage weapons or to operate active sensors. For example, without the fire signal from the system controlled by the commander, the gunnery officer cannot operate his or her weapons. In addition, the commander monitoring overall combat conditions can arrange movement of platforms in his or her control to get advantage over the enemy via the system's communications.

3.3 Interception

In this section, we illustrate how to determine the optimum course to interception. Interception is an regular task of both the Officer Of Deck (OOD) and the Tactical Action Officer (TAO) [15]. However, it requires a certain computational time and is sometimes insufficiently accurate. Also, whenever the course and speed of a target change, recalculation is required. With our system, this calculation can be done in real time. For the interception algorithm, we use simple vector and trigonometric operations. Figure 3 illustrates the drawing procedures of vector combination for interception.

In Figure 3 the following are known:

- $|T|$ = scalar value of target vector T
- $|O|$ = scalar value of real interception vector O
- α, the direction of relative interception vector R
- β, the direction of target vector T

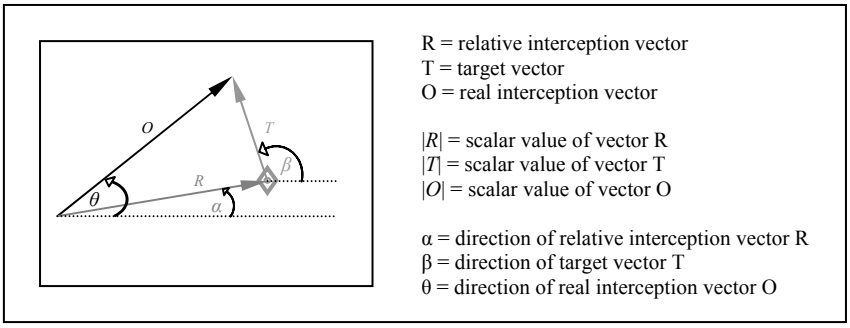

Figure 3: Geometrical representation of interception

The objective is to find θ, the direction of the real interception vector O. However, we do not know $|R|$, the scalar value of relative interception vector R. To determine θ, we need to determine $|R|$ first.

From the combination of vectors shown in Figure 3:

$$0 = R + T \tag{1}$$

Separate vectors on the vertical and horizontal directions are:

$$|O|\sin\theta = |R|\sin\alpha + |T|\sin\beta$$

$$|O|\cos\theta = |R|\cos\alpha + |T|\cos\beta \tag{2}$$

This yields the scalar equation:

$$|O|^2 = |R|^2 + 2|R||T|\cos(\alpha - \beta) + |T|^2 \tag{3}$$

To determine $|R|$, we rewrite the equation in the form

$$|R|^2 + \{2|T|\cos(\alpha - \beta)\}\,|R| + \{|T|^2 - |O|^2\} = 0 \tag{4}$$

By using the quadratic formula, we determine the positive roots. $|R|$ is the smaller value of two positive root values, or the positive one if the other is negative or an imaginary number. If both root values are negative or imaginary numbers, it indicates that interception is impossible.

To determine θ, we substitute $|R|$ in equations (2) and obtain:

$$\theta = \tan^{-1}\left(\frac{|R|\sin\alpha + |T|\sin\beta}{|R|\cos\alpha + |T|\cos\beta}\right) \tag{5}$$

From the above, the angle of interception θ is determined.

3.4 Escape Route Planner

Escape, meaning to move our platform through all hostile forces without detection by enemy sensors, requires complex plans and calculations. Our intelligent system shall provide an optimum trajectory for escape function. As in the interception function, whenever the course and speed of any hostile platforms change, a new plan and recalculation are needed. With current position, course, and speed of all hostile targets exposed to our sensors, the intelligent system must determine the shortest and safest escape route and recalculate it in real time when any hostile target changes its course, speed, or both. Moreover, the escape route is created with the condition that our platform has to maintain its current speed. Therefore, whenever our platform changes its speed, the intelligent function must reroute the escape automatically. In Figure 4 an example of escape trajectory is shown.

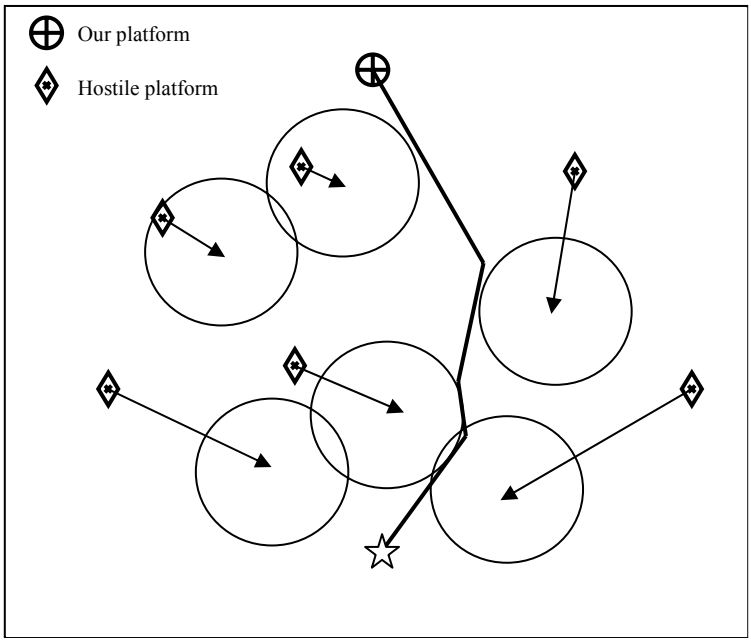

Figure 4: Example of escape trajectory

Similar to the method of a robot's movement to its target destination, the problem here is to avoid obstacles. The robot's route is the shortest path avoiding any obstacles. In the case of escape route planning, there is an analogy between the obstacle and the detection area of hostile sensors. Instead of obstacles, our platform needs the shortest route to the destination without moving into any sensor surveillance area, which is assumed circular. However, if it is unavoidable to go through the hazard area, our platform will minimize its stay in that area to maximize its escape chances. Unfortunately, the surveillance areas of the enemy sensors are not static. The movement of hostile platforms with sensors means that enemy surveillance areas move as well. Obviously, this problem is more complicated than the static obstacle avoidance problem. To solve this problem, complex mathematics, a good model and a good optimum search method are needed. Fortunately, we found a search genetic algorithm that does not involve complex mathematics and solves the problem reasonably well.

We discussed the genetic search methods in Section 2. At this point, we explain how we apply a GA to solve the problem of escape planning. We show the representation and evaluation aspects of this particular problem. Due to search optimizations in GA, we need to represent the search area of this problem using binary numbers. Because our platform is always a surface ship, the problem is within on 2D space. Therefore, the trajectory of the escape route can be drawn from the initial point to the destination. With our design, the trajectory consists of 11 partial straight lines. Each straight line extends from a minimum of zero to a maximum of the distance between the initial point and the destination. With this design, the trajectory is constrained with a maximum of 11 turns, which we assume are enough to reach the destination. The maximum number of segments, 11, has resulted from considerations related to the computational power of the computer we use to develop our prototype software. The number can be larger if a faster processor is used or parallel processing is performed. The representation of the escape route is shown in Figure 5.

As illustrated in Figure 5, we dedicate the entry combination of 11 segments to the escape route. We need to indicate 10 information lines, each with course C and distance D shown in the rectangles and associated with arrows that point to the beginning of each line (for the first 10 segments). For the last segment, we do not indicate this information because the end point of the

10^{th} segment is connected directly with the destination. Therefore, with these characteristics, we determine the course and distance of the first 10 segments. To transform these linear values into binary numbers, we represent them with 8 bits for each course and 8 bits for each distance, as shown in Figure 6.

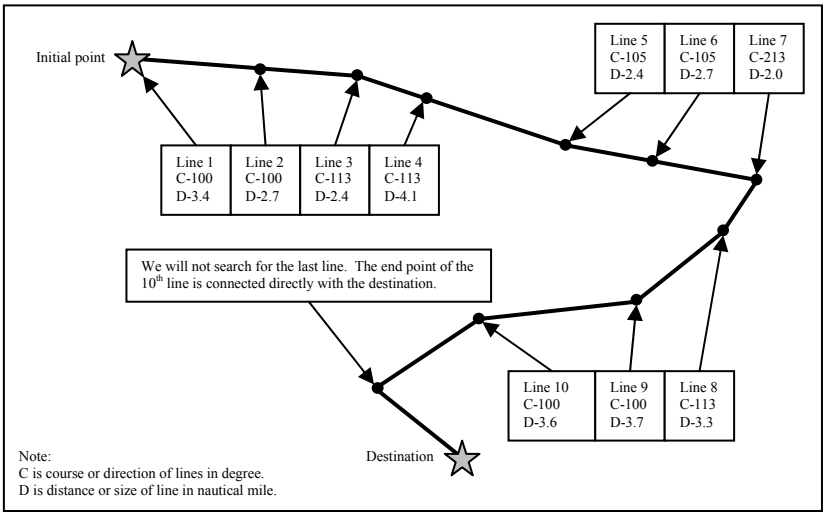

Figure 5: The representation of the escape route

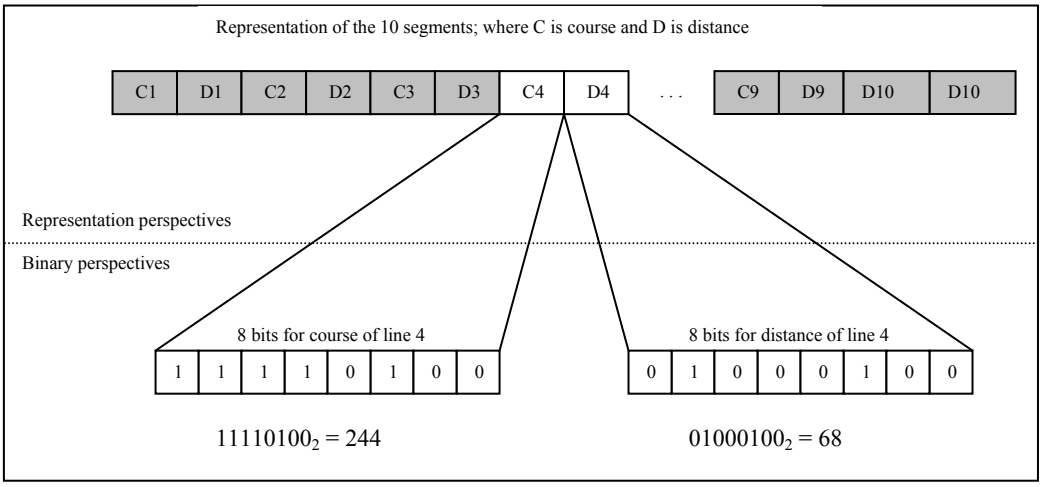

Figure 6: Transformation of course and distance to binary

In Figure 6, the course and distance of segment 4 are transformed into two 8-bit representations. The binary representation of C_4, in this case 11110100_2, corresponds to 244 in decimal numbers. The decimal value of C_4 is determined by using the following formula:

$$\alpha_{c,i} = \beta_{c,i} \times \frac{360}{256} \tag{6}$$

where $\alpha_{c,i}$ is the decimal course value of segment i [0, 360] and $\beta_{c,i}$ is the corresponding value on 8 bits (range 0..256). Therefore, in Figure 6, the exact value of C_4 is $244 \times \frac{360}{256} \approx 343$. Next, to determine the exact value of D_4, the following equation is used:

$$\alpha_{d,i} = \beta_{d,i} \times \frac{\lambda}{256} \quad (7)$$

where $\alpha_{d,i}$ is distance value of segment i, in decimal, $\beta_{d,i}$ is the corresponding value on 8 bits (range 0..255), and λ is the direct distance between the starting and the destination points of the escape trajectory.

From the above equation, it is obvious that the maximum distance of each straight line is the direct distance between the start and end points of the escape trajectory. In Figure 6, by assuming λ is 32 nautical miles, then the exact value of D_4 is $68 \times \frac{32}{256} = 8.5$ nautical miles.

Next, we shift the description to the evaluation aspect of the solution. As previously mentioned, this evaluation is key to the process of survival in GA [8]. We determine how good the escape route reproduced is by using a fitness function. There are two variables involved: the total distance of the escape route and the total amount of time during which our platform is exposed to hostile sensors. The objective is to minimize the value of both these variables. However, sometimes the two variables are conflicting, i.e. the shortest route may lead to longer exposure time to enemy sensors. We need to weigh which one is of higher importance in a particular situation. The following is the fitness function of the algorithm used:

$$F = \mathrm{K} - \sum_{t=0}^{T} \left\lfloor \mu \sum_{i=1}^{N} \left(30 - x_{t,i}\right)^3 \right\rfloor - \nu T \quad (8)$$

where F = fitness,
K = a large constant value, for example 3,000,000
μ = gain of exposed value,
ν = gain of travel time from starting to ending points,
$x_{t,i}$ = distance between our platform and hostile platform i at time t;
(if $x_{t,i} < 30$; otherwise $x_{t,i} = 30$),
T = time our platform travels from the starting to the destination points,
N = number of hostile platforms.

The genetic search maximizes fitness F but our objective is to minimize the value of the two variables. Therefore, the fitness involves subtraction of the two parameters from a large constant value (K). Usually, the longest effective detection range of surface radar is around 30 nautical miles, so we assume that our platform will be safe when the range is more than 30 nautical miles. Otherwise, the exposed value will be to the exponential of 3 for the distance that is less than 30 nautical mines. In addition, a user can assign gains of both variables; μ for exposed value and ν for time of traveling. By default, we assign μ a significant value in order to eliminate the exposure to enemy sensors.

4. Software Model

The prototype software presented in this paper has been developed using a systematic software engineering approach [1, 13] and UML as modeling notation. In this section several excerpts of the main elements of the software's specification and design are presented. Specifically, examples of the following are provided: functional and non-functional requirements, use case diagram, and sequence diagrams. For more details, the reader is referred to [5].

4.1 Requirements specification

The proposed software's requirements can be classified as functional and non-functional. The former describe the intended behavior of the system, while the latter specify constraints on its implementation. In the following, some of the tool's functional requirements are presented.

Functional requirements

R1. The system shall display on a black screen the current position of our ship as well as the positions of all detected platforms, represented using tactical symbols and colors that conform to NTDS.

R2. The system shall continuously calculate the next position of our ship as well as the next positions of all detected platforms.

R3. The system shall display the time elapsed from the start of the program's execution.

R4. The system shall allow the user to adjust the time speed for faster than real time representation.

R5. The system shall provide a function to manually create a target.

R6. The system shall provide a function to manually modify a target.

R7. The system shall provide a function to manually remove a target.

R8. The system shall allow the user to reinitialize the starting position of our ship.

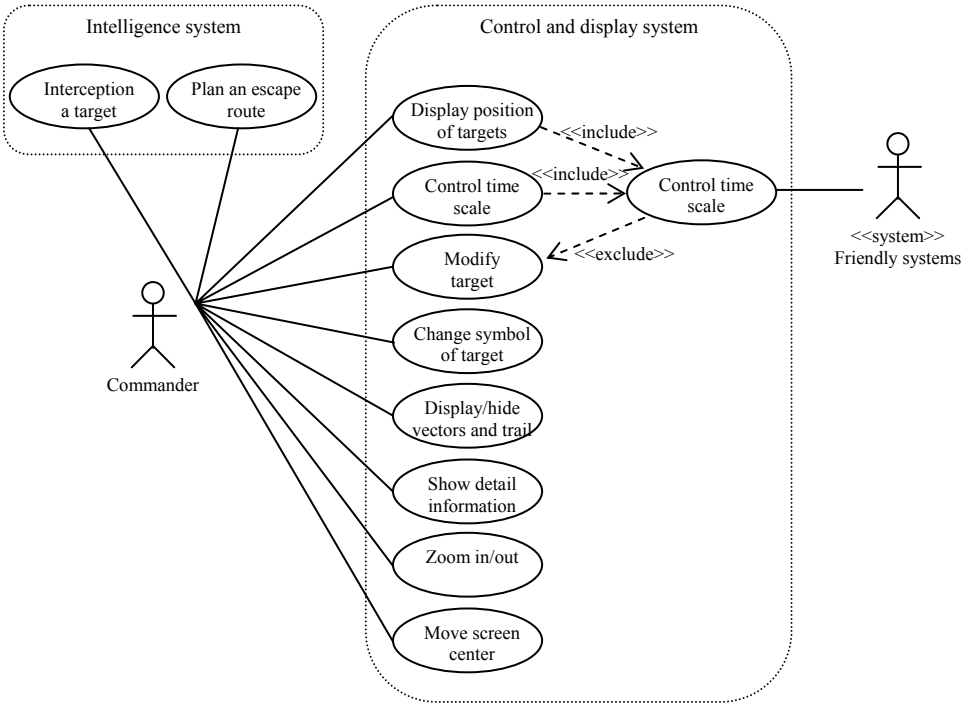

Figure 7: Use case diagram of the proposed software tool

4.2 Use case diagram

Figure 7 presents the use case diagram of the proposed software tool. This diagram shows the relationships among actors involved and the use cases that describe the system's functionality. There are two actors and 11 use cases in the software model that we have created.

First, the `Commander` actor represents the broad kind of tactical commanders that could use the system, including fleet commanders and commanding officers. This actor plays the most prominent role in the system. Most of the use cases are created to describe the commander's interaction with the system. The commander is in charge of monitoring, understanding all battle field circumstances, and providing command to his task force.

Second, the `Clock` actor is included in the system because it needs to provide accurate time for real-time calculations and displaying of platform positions.

4.3 Sequence Diagrams

Figure 8 shows an example of sequence diagram, namely the sequence diagram for calculating the positions of the targets. The targets' information stored in the `ListOfTargets` can be manually modified by calling `UpdateEvent`. The automatically updating procedure starts when the `SystemClock` sends a `TimeToUpdate` signal. The `Calculate` component fetches information of the first target from the `ListOfTargets` and gets current time from `System Clock`. The `Calculate` component determines the new position of the first target and writes over to the `ListOfTargets`. The procedure is repeated until the last target in `the ListOfTargets` is reached. Then all targets in the `ListOfTargets` are displayed on screen. Finally, this process sleeps and waits for the next signals from `SystemClock`.

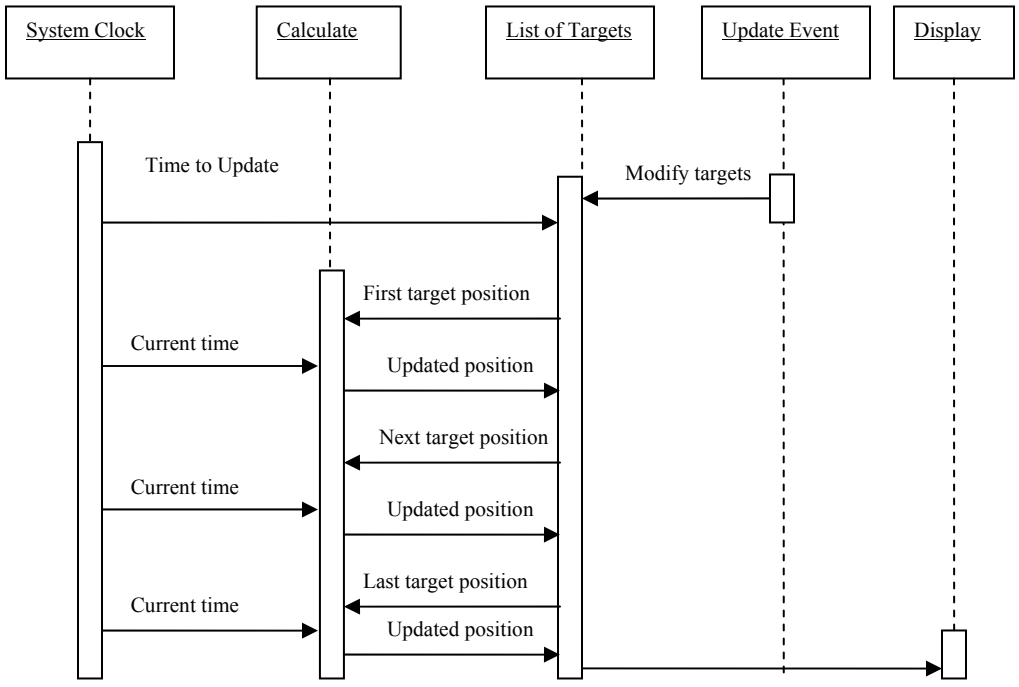

Figure 8: Sequence diagram for calculating the positions of the targets

5 Prototype Details

In this section we present the prototype of the software tool we created on the Linux distribution of Fedora Core 2, running on a COTS computer (Dell 8052). For illustration purposes, we set the screen resolution of a 17-inch monitor to 1024x768. This allowed for both satisfactory speed performance and clarity of information display. The prototype is shown via several snapshots taken during its operation. More details can be found in [5].

The main screen of the software tool is shown in Figure 9. From the beginning, it is useful to note that in this system we can run multiple processes that display various areas of the battlefield, each area in its associated window. We chose a windows-based solution because windows can be manipulated using a variety of operations: moving, resizing, hiding, and displaying on full or partial screen. Throughout its entire operation, the prototype retains the same center of the screen, regardless of window resizing. In addition, resizing does not affect the size of the text and the symbols shown on the screen.

This tool displays NTDS symbols on a black background. Although we have utilized NTDS symbols, we also have modified some presentation details. For example, instead of using a white background, we have a black one to keep the room dark and thus provide better visualization contrast. More specifically, using colored symbols on black background makes easier the noticing of alarms and changes. Consequently, the symbols for `unknown` platforms and for our own platform have been changed from black to white. The trails of targets are displayed with combinations of continuous lines. The colors of the lines correspond to their identification status in real time (for example, one trail can change from `unknown` to `hostile`). The speed vector of each symbol is displayed with continuous dots. Its size and direction correspond to `speed` and `course`, respectively. Its color is the color of the symbol it is associated with.

Also, we deliberately have placed detailed information at the corners of the screen. On the top-left corner of the screen the points of the compass and a scale are shown. The top of the screen is always North. The scale information consists of a text and a bar. The text shows the real world distance that is scaled to display on the screen as the length of the bar. For example, the text "5 NM" indicates that the length of the yellow bar behind the text stands for 5 nautical miles. The length of the bar is equal to the length of the background grid square.

On the top-right corner of the screen, the *tactical time* (the time elapsed from the beginning of the C^4I prototype's operation session) and the *time scale* (simulated operation speed) are indicated. Note that the time scale has been included for simulation purposes only. By default, the time scale is 1:1, which corresponds to the real-time. From this value, it can be increased up to 1:50, which means that simulation speed is 50 times faster than the speed of real-time events.

On the bottom-left corner of the screen, detailed pieces of information about our `own platform` and about the `selected target` are located. The first information group shows the current course of our `own platform` in the navigation system, its speed in knots, and its position in nautical miles. The second information group is associated with the `selected target`, a target that is marked by a small yellow square placed inside the target's symbol. In this group, besides the course, the speed, and the position of the selected target, the bearing and the range from our platform to the selected target are shown as well.

To meet its command and control purpose, the proposed the software tool provides functions for maneuvering our own platform. After the `Maneuver our ship` option is selected, a dialog window (panel) is shown on the screen. The dialog panel consists of four editable entries (parameters): x-position, y-position, course, and speed. The current values of these four parameters are provided. To maneuver own platform, the user can enter only the value(s) of those

parameter(s) that he or she would like to update. Figure 10 shows the structure of the dialog for maneuvering our own platform.

As shown in Figure 11, on the left-hand side of the screen, below the scale bar, the relevant data involving the target to be intercepted and the user's own platform are displayed. When the distance between the user's own platform and the target is less than three nautical miles, the remaining distance and remaining time to interception is displayed using red text. To cancel the interception function, the user needs to click on the `Disable` submenu of the `Interception` option.

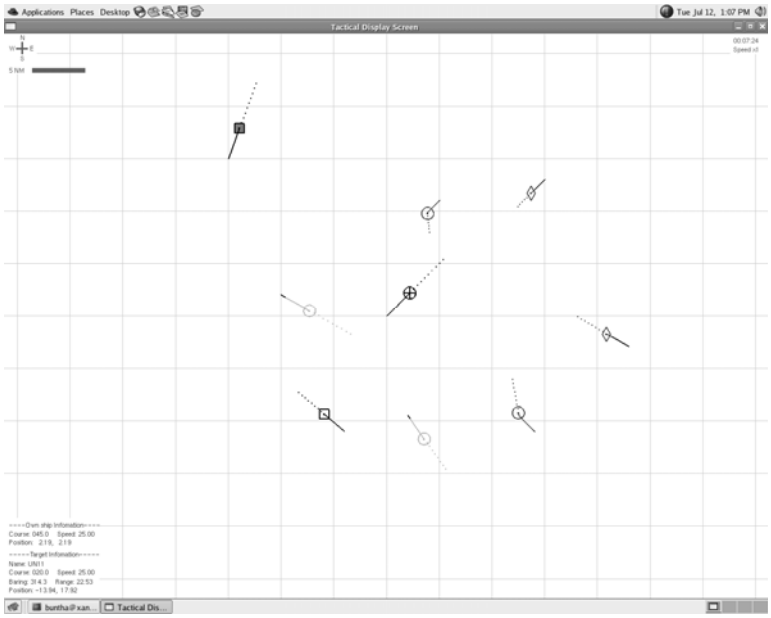

Figure 9: The main screen of the prototype software tool

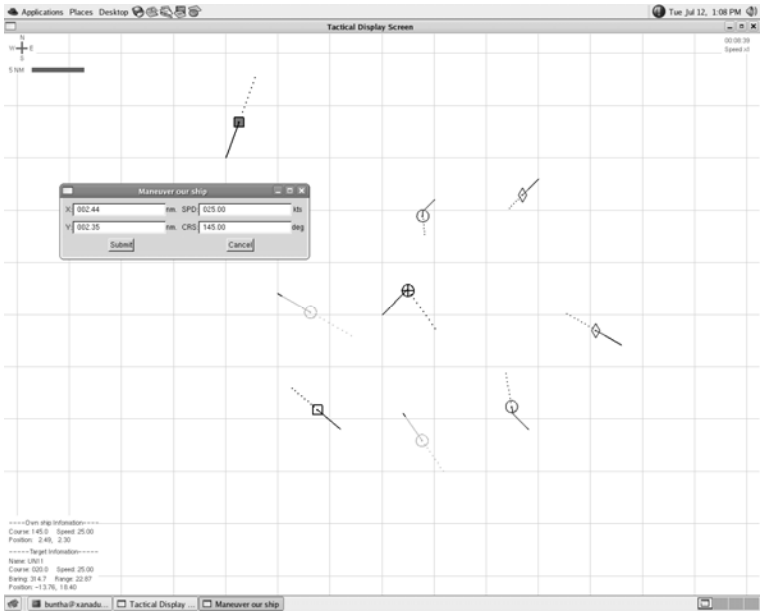

Figure 10: The dialog panel of the 'maneuver our ship' option

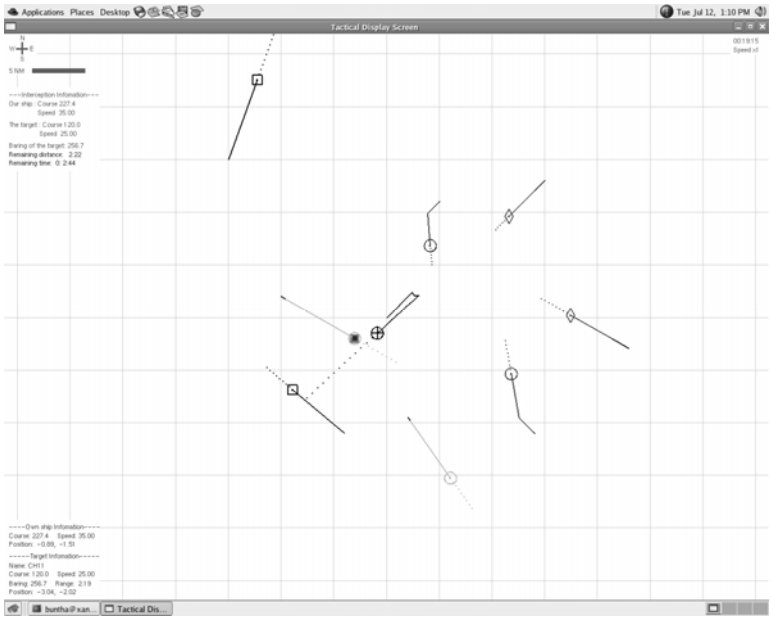

Figure 11: Example of target interception

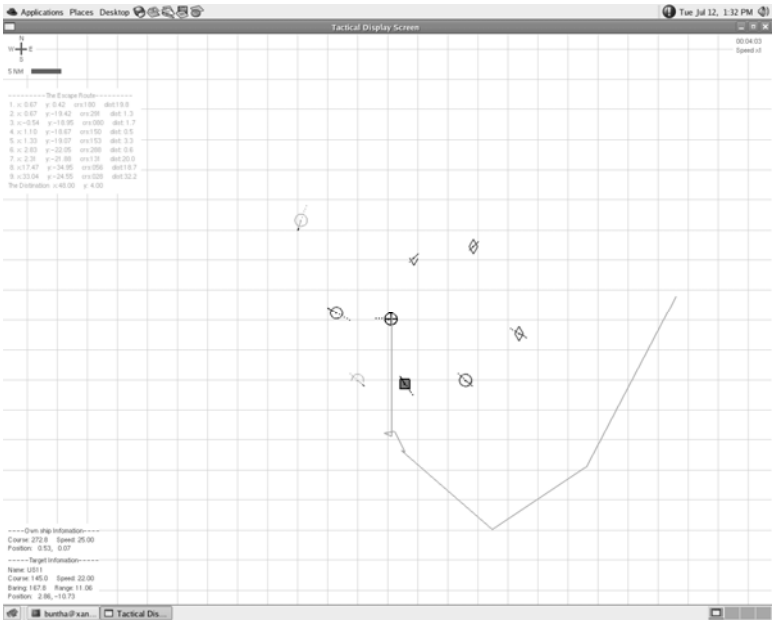

Figure 12: Example of escape route planning

Also, when the user activates the `Escape route planner` function, the interception function is automatically disabled. Figure 12 shows the escape route plan drawn on the screen as a sequence of yellow line segments and detailed by descriptive text presented on the left-hand side of the window, (just below the scale bar). In both figures, the user's own platform and a hostile ship are represented. The user's own ship's objective is to reach a chosen destination with minimum chance of detection by the enemies. The user starts the escape procedure by clicking on the `Escape route's Operate` function and then clicking on the screen to specify the desired destination. Consequently, the software tool computes and provides a plan that brings the user's own ship to the destination. The user's own ship will follow this plan automatically. The escape

route is displayed on the screen in green-yellow color. For instance, we can notice that in the particular example shown in Figure 12 the route goes down first in order to move away from hostile platforms.

Whenever any hostile target changes its course, speed, or both, the software tool changes the escape plan accordingly by providing a new route. To stop following the plan provided by the software tool, the user has to click on the `Disable` function of the `Escape route` option.

6 Future Work and Conclusions

6.1 Future Work

The proposed software tool has been developed in terms of comprehensive software specification, detailed architectural and GUI design, and operational implementation in C. While the prototype serves well as a proof of concept and can be used in practice for naval surface warfare training and simulation purposes, the work presented in this paper can be further expanded in several directions.

First, the software needs to be completed and enhanced with more functionality and improved graphical user-interface.

Second, the algorithms underlying the support for command and control could be further elaborated and optimized. For example, better genetic search algorithms need to be investigated, evaluated, and applied.

Third, the software solution presented involves the use of a single computer. A networked version of the program could increase significantly the efficiency of training and simulation via real-time, multi-user involvement of trainers and trainees.

Fourth, an ambitious goal for this project would be to evolve in a multi-platform version of naval surface warfare training and simulation (multiple task force system). At this time, only one user's own platform can be controlled by the software, but larger and more realistic situations should be modeled by involving a fleet of user's own platforms.

Finally, the ideal accomplishment of this project would be to be used in real-world navy applications, not as a training or simulation tool, but as an effective, real-time solution for supporting actual combat operations. For this, in addition to the enhancements suggested above, interfacing with navy detection sensors and maneuvering equipment is necessary.

6.2 Concluding Remarks

The main goal of the work presented in this paper has been to develop a prototype software tool that provides support for centralized command and control in naval surface warfare. To build this prototype, we have defined the modes of operation for the tool and have designed algorithms needed for its implementation. These algorithms include intercepting an enemy target platform and calculating an escape route. A detailed GUI interface with easy-to-use options facilitates the interaction with the computer by navy commanders and tactical action officers. Currently, the system is operational for training and simulation in naval surface warfare. Also, it provides a basis for more complex research and development.

The software was designed and developed using a systematic engineering approach, supported by the modern modeling notation UML. Four main functions have been incorporated in the prototype

of the software tool. First, an HCI function depicts the battlefield information using NTDS graphical symbols and enhances the use of computers by commanders via detailed graphical interfaces. Second, a command and control function provides support to commanders for centralized control of the battlefield situation. Third, an interception function calculates an optimum course for intercepting a particular enemy target. Fourth, an escape route planner provides a route characterized by minimum chance of detection by hostile platforms.

Furthermore, we have used a personal computer as hardware equipment on which our software tool executes. This is important, because with this approach both the development costs and the expenses associated with the tool's use are substantially reduced.

References

[1] ARLOW, J. and NEUSTADT, I., UML and the Unified Process: Practical Object-Oriented Analysis and Design, ADDISON-WESLEY, 2002.

[2] BLAIS, C.L., Scalability Issues in Enhancement of the MAGTF Tactical Warfare Simulation, Procs. of the 1995 Winter Simulation Conference, Dec. 1995, pp. 1280-1287.

[3] BLAIS, C.L., Prototyping Advanced Warfare Gaming Capabilities for the 21st Century Warfighter, Procs. of the 1998 Winter Simulation Conference, Dec. 1998, vol.1, pp. 841-848.

[4] BOWDEN, F.D.J., GABRISCH, C, and DAVIES, M., C3I Systems Analysis Using the Distributed Interactive C3I Effectiveness (DICE) Simulation, Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation, Oct 1997, vol.5, pp. 4326 – 4331.

[5] BUNTHA, S., Command Control Communications Computers and Intelligence Software for Naval Surface Warfare, Master thesis, University of Nevada, Reno, December 2004.

[6] CERUTI, M.G., Challenges in Data Management for the United States Department of Defense (DoD) Command, Control, Communications, Computers, and Intelligence (C^4I) Systems, Procs. of the Computer Software and Applications Conference (COMPSAC '98), Aug. 1998, pp. 622 – 629.

[7] GOLD, H., and SUGGS, C., Implementing Commercial off-the-shelf (COTS) Technologies into a Navy Tactical Display Communication System, Proceedings of the IEEE Military Communications Conference, Oct. 1998, vol.3, pp. 929-933.

[8] GOLDBERG, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, ADDISON-WESLEY, 1989.

[9] GOLUB, J. and SOPER, W.A., Prototyping for Naval Battle Group Simulation Development, Proceedings of the 16th Annual Symposium on Simulation, March 1983.

[10] JEDRYSIK, P.A., MOORE, J.A., STEDMAN, T.A., and SWEED, R.H., Interactive Display for Command and Control, Proceedings of the IEEE 2000 Aerospace Conference, Mar 2000, vol.2, pp. 341 – 351.

[11] LOUIS, S. J., MCDONNELL J., and GIZZI N., Dynamic Strike Force Asset Allocation Using Genetic Search and Case-based Reasoning, Proceedings of the Sixth Conference on Systemics, Cybernetics, and Informatics, Orlando, Jul 2002.

[12] MCKENNA, L.H. and LITTLE, S., Developing Tactics Using Low Cost, Accessible Simulations, Proceedings of the 2000 Winter Simulation Conference, Dec 2000, Vol.1, pp. 991-1000.

[13] SOMMERVILLE, I., Software Engineering, ADDISON-WESLEY, 2004.

[14] STONE, G.F. and MCINTYRE, G.A., The Joint Warfare System (JWARS): A Modeling and Analysis Tool for the Defense Department, Proceedings of the 2001 Winter Simulation Conference, Dec. 2001, vol.1, pp. 691-696.

[15] SURFACE WARFARE OFFICERS SCHOOL COMMAND, Useful Information, accessed March 2006 at: http://www.swos.navy.mil