NPS-CS-99-004

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Re-engineering the Janus(A) Combat Simulation System

by

V. Berzins
Luqi
M. Shing
M. Saluto
J. Williams

January 1999

Approved for public release; distribution is unlimited.

Prepared for: U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM Robert C. Chaplin
Superintendent

Richard S. Elster
Provost

This report was prepared for Naval Postgraduate School
and funded in part by the U.S. Army Research Office and TRAC-Monterey

This report was prepared by:

Luqi
Professor, Computer Science

Reviewed by:

Dan Boger
Chairman, Computer Science

Released by:

D. W. Netzer
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1999	Technical Report

4. TITLE AND SUBTITLE
Re-engineering the Janus(A) Combat Simulation System

5. FUNDING NUMBERS
MIPR No. 8FNPSARO36

6. AUTHOR(S)
V. Berzins, Luqi, M. Shing , M. Saluto, J. Williams

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Group, Department of Computer Science, Naval Postgraduate School, Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NPSCS-99-004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709-2211

U.S. Army TRAC-Monterey, P.O.Box 8692, Monterey, CA 93943

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this report are those of the authors and do not reflect the official policy or position of the Department of Defence or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (Maximum 200 words.)

This report describes a case study to determine whether computer-aided prototyping techniques provide a cost-effective means for re-engineering legacy software. The case study consists of developing a high-level modular architecture for the existing US Army Janus combat simulation system, and validating the architecture via an executable prototype using the Computer Aided Prototype System (CAPS), a research tool developed at the Naval Postgraduate School.

The case study showed that prototyping can be a valuable aid in re-engineering of legacy systems, particularly in cases where radical changes to system conceptualization and software structure are needed. The CAPS system enabled us to do this with a minimal amount of coding effort.

14. SUBJECT TERMS
Combat Simulation, Janus(A), Computer aided prototyping, Object-Oriented Design, Software evolution, Software architecture, Re-engineering

15. NUMBER OF PAGES
160

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Re-engineering the Janus(A) Combat Simulation System

V. Berzins, Luqi, M. Shing, M. Saluto, J. Williams

Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5118

Abstract

This report describes a case study to determine whether computer-aided prototyping techniques provide a cost-effective means for re-engineering legacy software. The case study consists of developing a high-level modular architecture for the existing US Army Janus(A) combat simulation system, and validating the architecture via an executable prototype using the Computer Aided Prototype System (CAPS), a research tool developed at the Naval Postgraduate School.

The case study showed that prototyping can be a valuable aid in re-engineering of legacy systems, particularly in cases where radical changes to system conceptualization and software structure are needed. The CAPS system enabled us to do this with a minimal amount of coding effort.

Table of Contents

E. THE OBJECT MODELS FOR THE JANUS CORE ELEMENTS

I. PROBLEM STATEMENT

1. OVERVIEW

We address the need to modernize the software of the Janus(A) systems into a maintainable and evolvable structure. This research develops

- a high-level modular architecture for the existing Janus(A) systems using CAPS,
- an implementation of the design using the Ada95 programming language. The higher level goal of this research is to evaluate the effectiveness of computer-aided prototyping and software evolution tools when applied to legacy software, as opposed to prototype software that is initially developed in the context of the CAPS system and its prototyping language PSDL.

A good modular design is an essential feature of every maintainable and evolvable software system. Such designs partition large systems into sets of highly cohesive and loosely coupled components and minimize the effect of changes to individual components. The high-level architecture, in its executable form, enables designers of the Janus(A) systems to incrementally re-engineer the existing software system and to easily add new functionalities to the Janus(A) systems. The Computer Aided Prototyping System (CAPS) provides automation support for future evolution of the Janus(A) systems while reusing existing code to the maximum degree possible.

This experiment also serves as a reality check for computer-aided prototyping. CAPS is a research tool that has been developed to show the feasibility of providing significant computer aid for software evolution when that software has been developed using a prototyping language PSDL that has been specifically designed to support prototyping and evolution. This research evaluates the effectiveness of these same tools and techniques for re-engineering a piece of legacy code that was *not* designed to support software evolution. The experiment helps determine whether computer-aided prototyping techniques provide a cost effective means for re-engineering legacy software.

The experiment is based on the premise that the re-engineering problem for the Janus(A) system is typical of the re-engineering problems facing many older software systems used by DoD, and that the capability for rapid construction, evaluation, and modification of software architectures will help to speed up the process of recovering high-level design information from old code and restructuring the code so that old functions can be used in new ways and can be extended to include new functionality. We summarize the lessons learned from this experiment to suggest several useful improvements to the CAPS system as well as uncover critical problems that should be addressed to enable more effective and less expensive re-engineering of legacy software systems.

2. BACKGROUND

Janus(A) is a software-based war game which simulates ground battles between two adversaries. It is an interactive two-sided, closed, stochastic, ground combat simulation that features precise color graphics. Janus is "interactive" in that command and control functions are entered by military analysts who decide what to do in crucial situations during simulated combat. The current version of Janus consists of a large number of FORTRAN modules, organized as a flat structure and interconnected with one another via FORTRAN COMMON declarations, resulting in a software structure that makes modification to Janus very costly and error-prone.

As explained in Janus' Software Design Manual, Janus manipulates a large number of files to generate data structures and to draw input parameters for its simulation algorithms. Data structures in Janus are FORTRAN arrays shared by relevant subroutines via COMMON statements. As such, and following the tradition of FORTRAN data structure definition, a weapon system's operational parameters are spread across a number of arrays and are identified by an array index value. Information regarding a particular system is therefore not encapsulated within a single construct as would be the case with more modern programming techniques and languages such as ADA. Furthermore, Janus algorithms are parameterized. That is, the behavior of a weapon system's simulation algorithm relies on data drawn from that weapon system's array entries. This strongly suggests that, to change the behavior of a weapon system, one must be able to edit that weapon's corresponding array entries. This is currently supported by the Janus interface module through a complex menu-driven editing process. However, altering the nature of the simulation algorithm for a given weapon system is not currently supported by Janus. Sophisticated programming skills and a clear understanding of Janus' data representation and software is needed. A modular hierarchical software architecture is needed to ease the understanding and reduce the risk and cost in modifying the Janus system. We try to extract the communication structures from the old code, and to embody those structures in a clean high level PSDL description that can drive the automated program generation capabilities of CAPS, and can enable easy arrangement and enhancement of those communication structures.

3. GUIDING PRINCIPLES

Our high-level modular software architecture encapsulates the major functions of Janus into loosely connected subsystems, with a well-defined interface between the subsystems. The architecture is the result of a conceptual reformulation based on the principles of data abstraction and generalization. Data abstraction decouples the software interfaces from the detail of data representations, thus limiting the impact of future representation changes. Generalization factors out the properties common to many different but similar data types, thus simplifying code and eliminating redundancy via polymorphic programming. The modular structure allows engineers to re-engineer and extend the Janus system in a piece-by-piece fashion.

The capability to test and evaluate early models of such architectures is essential to mitigate the risks of error inherent in any major problem reformulation. Our thesis is that

early prototyping is a valuable tool in software re-engineering because it explores conceptual inconsistencies and missing pieces early in the process. Such errors are easy to correct at this stage because there are fewer implementation details to be undone and then redone. Moreover, the availability of an executable software architecture enables engineers to test and evaluate modified components for correctness and efficiency.

The modular architecture also promotes the use of reusable components in Janus evolution. A representation for the software components must exist in order to reuse software components in a repository. Under the 3C model, a software component can be represented in terms of its *concept, content* and *context.*

- The concept of a software component describes the purpose and behavior of the component. It is usually represented by a brief component abstraction in the form of a formal/informal specification, or a software wrapper with identifiable attributes and keywords.

- The content of a software component consists of the data structures and algorithms needed to implement the software. It may contain several alternatives for different combinations of time/space requirements.

- The context of a software component describes where and how a component is used and what it relies on in its environment to work properly. The context is particularly important in large scale software reuse because successful software reuse tends to be domain specific.

The availability of an executable architecture enables testing components in their proper contexts, thus reducing the risk of incompatibility and mitigating the risk of improperly formulated concepts for reusable components.

II. SUMMARY OF THE MOST IMPORTANT RESULTS

This report summarizes the work done on the Modular Software Architecture of Janus (A)" project for the period from April 1998 to December 1998. The research of this project is aimed at developing a software architecture to support the re-engineering of the Janus Combat Simulation System. In these past seven months, we have analyzed the Janus Fortran source code, interviewed Janus domain experts, developed an Object-Oriented architecture for the Janus Combat Simulation subsystem, and validated the architecture with an executable prototype.

1. THE LEGACY SYSTEM

Janus is a software-based war game that simulates ground battles between up to six adversaries. It is an interactive, closed, stochastic, ground combat simulation that features precise color graphics. Janus is "interactive" in that command and control functions are entered by military analysts who decide what to do in crucial situations during simulated combat. The current version of Janus operates on a Hewlett Packard workstation and consists of a large number of FORTRAN modules, organized as a flat structure and interconnected with one another via FORTRAN COMMON blocks, resulting in a software structure that makes modification to Janus very costly and error-prone. There is a need to modernize the Janus software into a maintainable and evolvable system and to take advantage of modern Personal Computers to make Janus more accessible to the Army. TRAC-Monterey is re-engineering Janus into an object-oriented software system that is written in the C++ programming language and operates on Personal Computers. Prior to rewriting Janus in C++, the Software Engineering group at the Naval Postgraduate School is asked to extract the existing functionality through reverse engineering and to produce an object-oriented architecture that supports existing and required enhancements to Janus functionality.

2. THE RE-ENGINEERING PROCESS

The research team consists of four professors (Luqi, V. Berzins, M. Shing, and J. Guo) and two CS students (MAJ J. Williams and CPT M. Saluto). In addition, two other CS students (CPT. M. Merrell and Lt.Jg. Ilker Duranlioglu) also participated in the initial phase of the project.

2.1 SYSTEM UNDERSTANDING

The first step in our process, system understanding, took the form of a series of brief meetings with the client, TRAC-Monterey, which also included a short demonstration of the current software system. We asked questions and made notes on the system's operation and it's current functionality. We paid particular attention to the client's view of the system to gather their ideas on its strengths, weaknesses, and desired and undesired functionality. Additionally we collected copies of the Janus User's Tutorial manual, Janus User Manual, the Software Design Manual from a previous version of Janus (3.X/UNIX), and the Janus Version 6.88 Release Notes. Our goal was to gather as much information as we could about the currently existing system to aid in gaining a clearer understanding of its present functionality. The intent of this procedure was to ensure that the systems' current functionality was not lost nor misrepresented in the transformation into a more abstract, modular format.

2.2 REVERSE-ENGINEERING

The focus of this step was to abstractly capture the system's functionality and then produce system models that would most accurately represent that functionality.

Armed with the Janus source code, we proceeded to divide the code by directories amongst the team members. Each team member was assigned roughly six to seven directories to explore, examine and gather information. Using strictly manual techniques with UNIX commands and review procedures, we were able to get a fairly good idea of what each subroutine was designed to do. We also used the Software Programmers' Manual to aid in understanding each subroutine's function. In doing so we were able to group the subroutines by functionality to get a better understanding of the major data flows between programs. Using that knowledge, we developed functional models from

the data flows. We used an automated tool known as CAPS, Computer-Aided Prototyping Systems, version 2.0, developed by Professor Luqi and the Software Engineering group at the Naval Postgraduate School, to assist in developing the abstract models. CAPS allowed us to rapidly graph the gathered data and transform it into a more readable and usable format. Additionally, CAPS enabled us to develop our diagrams separately with the associated information flows and stream definitions, and then join them together still under the CAPS environment, where they can be used to generate an executable model of the architecture.

Figure 1 shows the resultant top-level structure of the existing Janus system. It consists of five subsystems - *cs_data_mgmt*, *scenario_db*, *janus*, *jaaws*, and *postp*. The *cs_data_mgmt* subsystem manages combat system databases. The *scenario_db* subsystem manages the different scenarios and simulation runs in the system. The *janus* subsystem simulates the ground battles. The *jaaws* subsystem allows analyst to perform post-simulation analysis and the *postp* subsystem allows Janus users to view simulation reports.

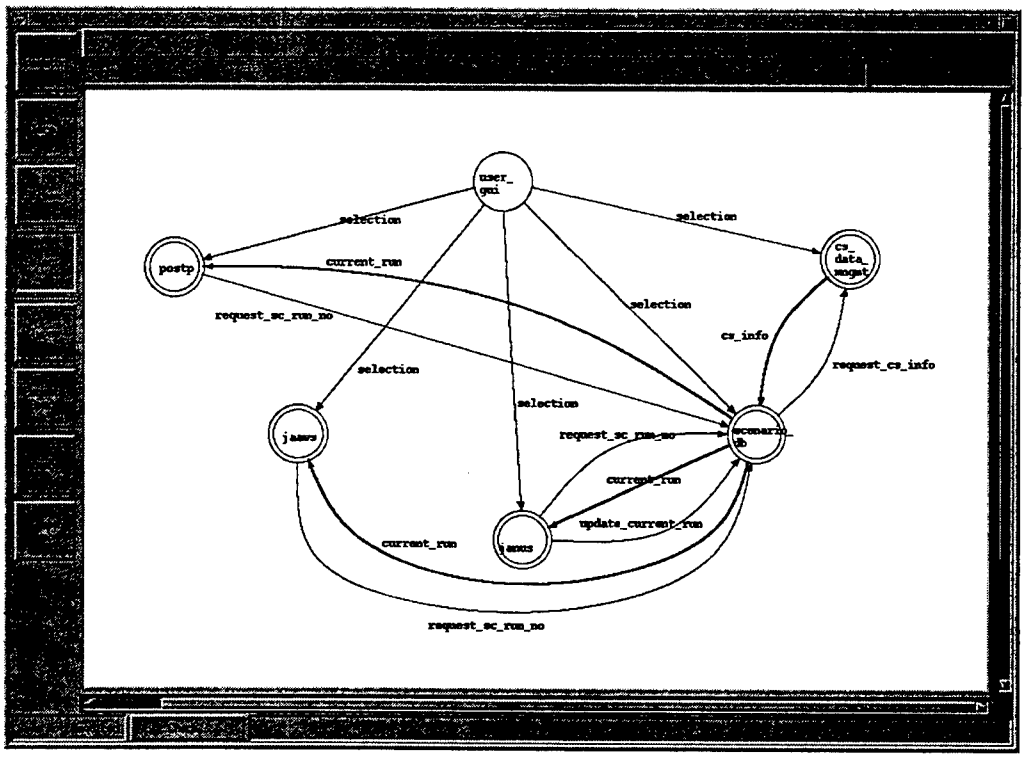


Figure 1. Top-level communication structure of the existing Janus software

2.3 TRANSFORMATION OF FUNCTIONAL MODELS TO OBJECT MODELS

Next, we proceeded to develop object models of the Janus System using the aforementioned materials and products, to create the modules and associations amongst them. This was probably the most difficult and most important step. It required a great deal of analysis and focus to mentally transform the currently scattered sets of data and functions into small, coherent and realizable objects, each with its own attributes and operations. In performing this step, we used our knowledge of object-oriented analysis and applied the OMT techniques and the UML notations to create the classes and associated attributes and operations. This was a crucial step because we had to ensure that the classes we created accurately represented the functions and procedures currently in the software. We used the HP-UNIX systems at the TRAC-Monterey facility to run the Janus simulation software to aid in verifying and/or supplementing the information we obtained from reviewing the source code and documentation. This step enabled us to better analyze the simulation system, gaining insight into its functionality and further concentrate on module definition and refinement.

2.4 REFINEMENT OF THE OBJECT MODELS AND THE DEVELOPMENT OF THE OBJECT ORIENTED ARCHITECTURE

During this phase of the project, the re-engineering team met several times each week for a period of two and a half months to discuss the object models for the Janus core elements and the object-oriented architecture for the Janus System. They presented the findings to the Janus domain experts from TRAC-MTRY and Rolands & Associates at least once per week to get feedback on the models and architectures being constructed. In addition, the re-engineering team also presented the findings to members of the OneSAF project, the Combat21 project, and the National Simulation Center. Based on the feedback from the domain experts, the re-engineering team revised the object models for the Janus core elements (Appendix E) and developed a 3-tier object-oriented architecture for the Janus System (Appendix A).

3. SOFTWARE ARCHITECTURE FOR THE JANUS COMBAT SIMULATION SUBSYSTEM

3.1 ARCHITECTURE OF THE EXISTING JANUS CODE

Central to the existing Janus Combat Simulation Subsystem is the program RUNJAN, which is the main event scheduler for the Janus simulation. RUNJAN determines the next scheduled event (called "process" in the Janus manual) and executes that event. If the next scheduled event is a simulation event, RUNJAN will advanced the game clock to the scheduled time of the event and perform that event. RUNJAN categorizes all events into 17 events to be handled by the following event handlers:

1. DOPLAN - interactive and CAC planning activity
2. MOVEMENT - simulation event to update unit positions which have been changed due to movement
3. DOCLOUD - simulation event to create and update status of smoke and dust clouds, and redraw them if necessary
4. STATEWT - periodic activity to write unit status data to disk
5. RELOAD - simulation event to plan and execute the direct fire events
6. INTACT - activity to update the graphics displays
7. CNTRBAT - simulation event to detect artillery fires
8. SEARCH - simulation event to update target acquisitions, choose weapons against potential targets, and schedule potential direct fire events
9. DOCHEM - simulation activity to create chemical clouds and to transition units to different chemical states
10. FIRING - simulation event to evaluate direct fire impact and to execute an indirect fire mission
11. IMPACT - simulation event to evaluate and update the results of a round impacting
12. RADAR - simulation event to update an air defense radar state, and to schedule a direct fire event for "normal" radar
13. COPTER - simulation event to update a helicopter state
14. DOARTY - simulation event to schedule an indirect fire mission

15. DOHEAT - simulation event to update units' heat stress

16. DOCKPT - activity to perform automatic checkpoints

17. ENDJAN - housekeeping activity to end the simulation

3.2 PROBLEMS WITH THE EXISTING EVENT SCHEDULER

The existing event scheduler uses global arrays and matrices to maintain the attributes of the objects in the simulation. Hence, one of the major tasks in designing an object-oriented architecture for the Janus combat simulation subsystem is to distribute the event handling functions to individual objects. Moreover, it is necessary to redefine some event categories in order to provide a uniform framework to eliminate redundant coding of the same or similar functions and to take advantage of dynamic dispatching of event handling functions in the object-oriented architecture. For example, the process for a unit to search for targets, select weapons, prepare direct fire, and execute direct fire differs depending on whether a unit has a normal radar, special radar, or no radar at all. Existing Janus uses the RADAR event handler to carry out the whole process if the unit has normal radar. It uses SEARCH, RADAR and RELOAD event handlers to carry out the process if the unit has special radar, and uses the SEARCH and RELOAD event handlers to carry out the process if the unit has no radar at all.

Interactions between the simulation engine and the world modeler are performed implicitly within the various event handlers in the existing Janus. Such interactions are made explicit in the new architecture in order to provide a uniform framework to update World Model objects during the simulation.

In order to allow the Janus implementation to take advantage of the reusable components provided by Tapestry, details of the event handlers for interactive and CAC planning activities, writing of unit status data to disk, updates of the graphics displays, and automatic checkpoints are not provided because of all these activities involve either the graphical user interface or the persistent storage of the objects. However, we do have recommendations regarding the design of the history mechanism (see Section 5).

3.3 THE PROPOSED OBJECT-ORIENTED ARCHITECTURE FOR THE JANUS COMBAT SIMULATION SUBSYSTEM

The new architecture uses an explicit priority queue of event objects to schedule the simulation events. Each event object has an associated simulation object, which is the target of the event. There are 14 event groups, which correspond to the 14 event subclasses shown in Appendix B. Note that an object oriented approach enable us to reduce the number of event types needed in the simulation. Depending on the subclass that an event object belongs to, the "execute" method will invoke the corresponding event handler of the associated simulation object to handle the event (Appendix C). The simulation object superclass defines the interface of the event handlers for the event groups, and provides an empty body as the default implementation for the event handlers. The methods are overridden by the actual event handler code at the subclasses that have non-empty actions associated with the events, as shown in Appendix D.

The above architecture enables a very simple realization of the main simulation loop (see page G-31). The code for the event control loop follows:

```
initialization;
While not_empty(event_queue) loop
        e := remove_event(event_queue);
        e.execute( );
End loop;
finalization;
```

Note that this same code handles all kinds of events, including those for future extensions that are yet to be designed. Event objects are created and inserted into the event queue either by the initialization procedure at the beginning of the simulation, by the constructors of the objects, or by the actions of other event handlers. Depending on the actual implementation of when and how events are inserted into the priority event queue, it may be necessary to allow events to change their priorities while waiting in the queue.

World Model object subclasses (with names starting with the "WM" prefix) are created to provide specialized methods for the world modeler to update the objects from other simulators. Information concerning objects local to the Janus simulator can be broadcast over the simulation network either periodically by an active world modeler object, or by individual local objects whenever they update their own states.

4. DEVELOPMENT OF AN EXECUTABLE PROTOTYPE USING CAPS

In order to validate the proposed architecture and to refine the interfaces of the Janus subsystems, we developed an executable prototype using CAPS. Figure 2 shows the top-level structure of the prototype, which has four subsystems: Janus, GUI, JAAWS and the POST-PROCESSOR. Among these four subsystems, the Janus and the GUI subsystems (depicted as double circles) are made up of sub-modules shown in Figures 3 and 4, while the JAAWS and the POST-PROCESSOR subsystems (depicted as single circles) are mapped directly to objects in the target language. After we entered the prototype design using CAPS, we use the CAPS execution support system to generate the code that interconnects and controls these subsystems. (Refer to Appendix F for the PSDL specification of the prototype and Appendix G for the target Ada/C code for the implementation.)

Due to time and resource limitations, we only developed the prototype for a very small simulation run, which consists of a single object (a tank) moving on a two-dimensional plane, 3 event subclasses (move, do_plan, and end_simulation), and 1 kind of post-processing statistics (fuel consumption). In addition, a simple user interface was developed using CAPS/TAE (Figure 5). The resultant prototype has over 6000 lines of program source code and contains enough features to exercise all parts of the architecture. The code that handles the motion of a generic simulation object was very simple, but it was designed so that it would work in both two and three dimensions without modification (currently the initialization and the movement plan of the tank object never call for any vertical motion). The code was also designed to be polymorphic, just as was the main event loop. This means the same code will handle the motion of all kinds of simulation objects without any modifications, including even new types of simulation objects that are part of future enhancements to Janus and have not yet been designed or implemented (see page G-76 and G-79).

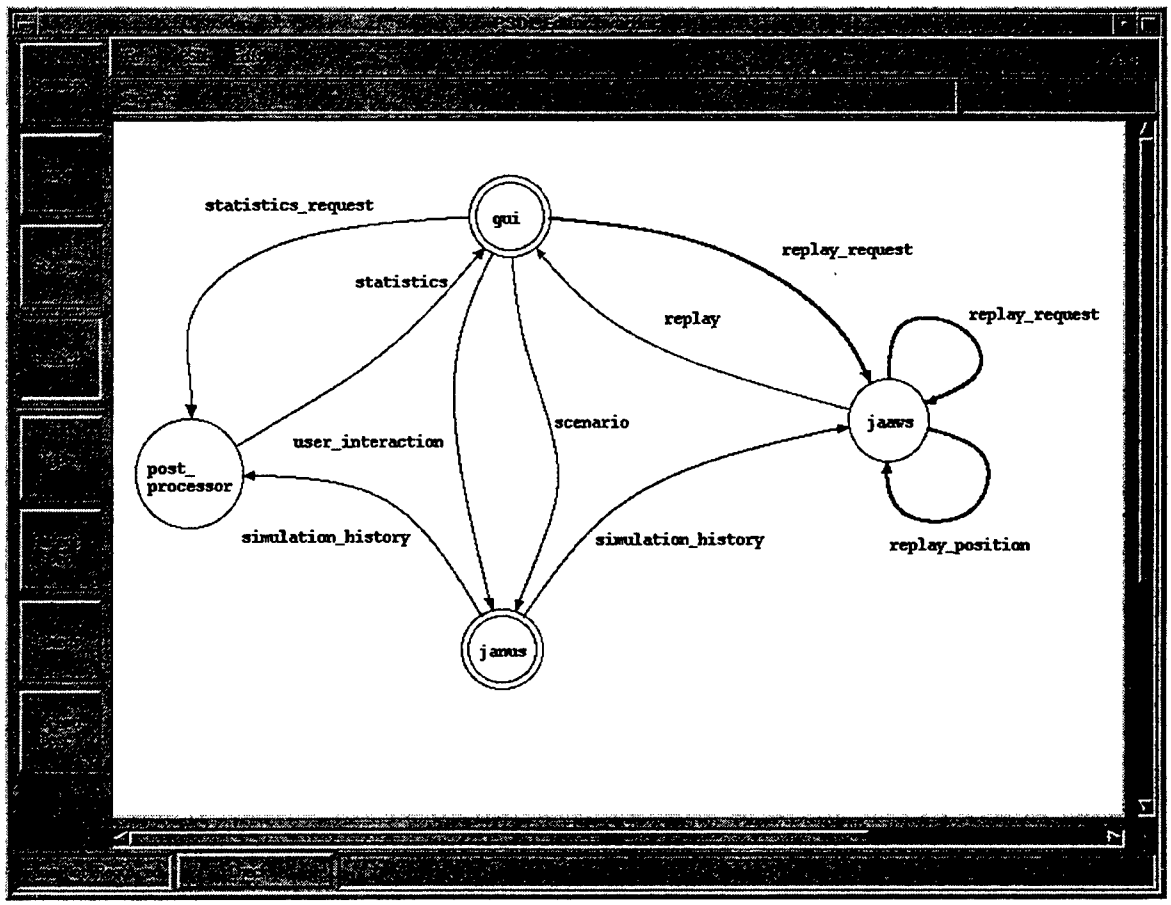


Figure 2. Top-level decomposition of the executable prototype

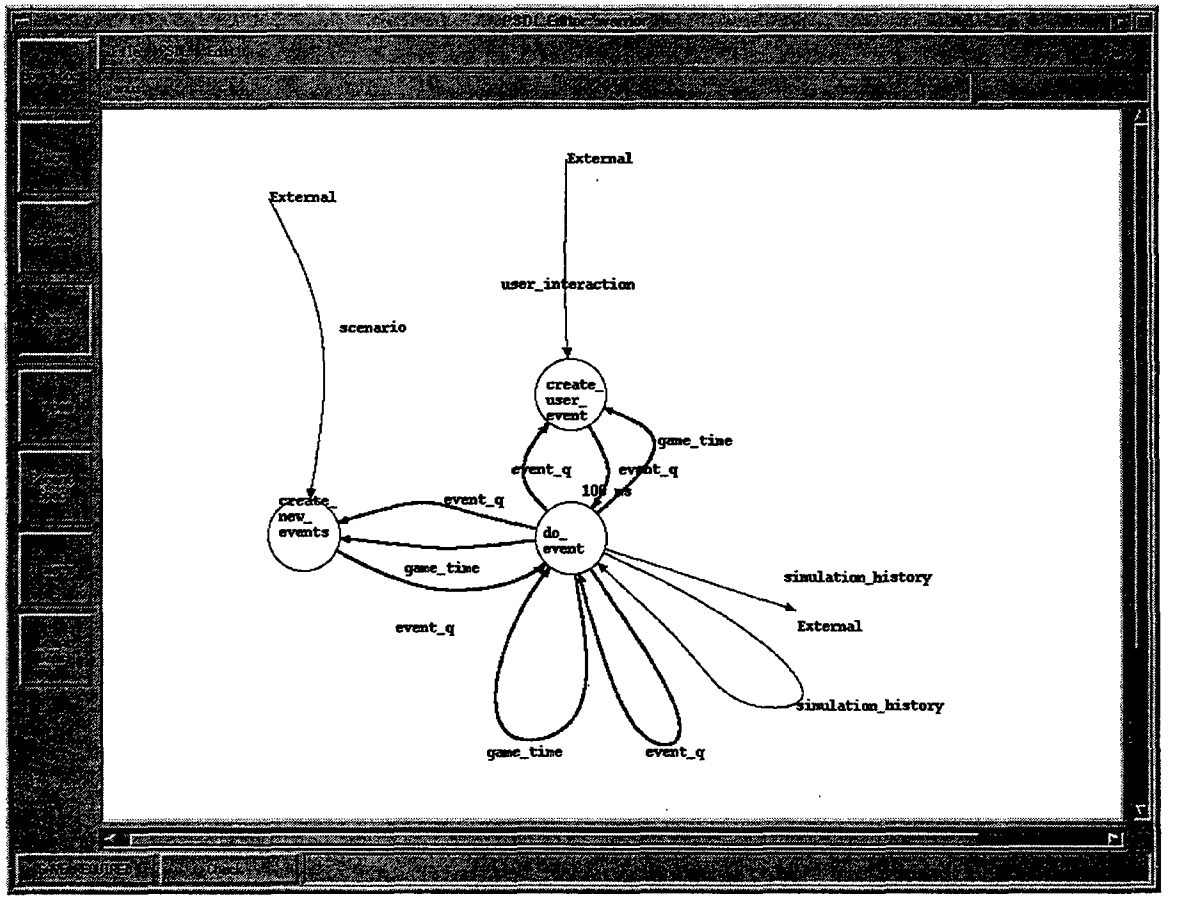


Figure 3. The JANUS subsystem of the executable prototype

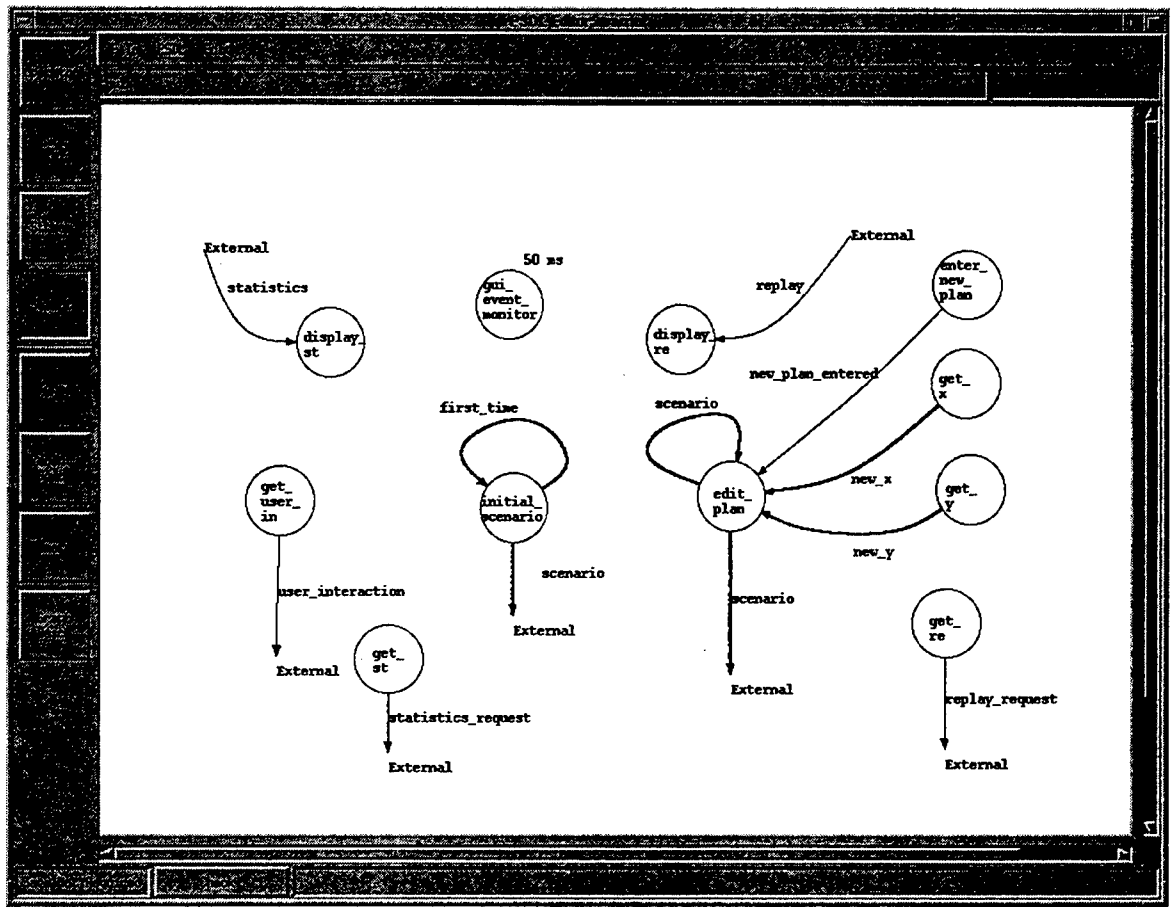


Figure 4. The GUI subsystem of the executable prototype

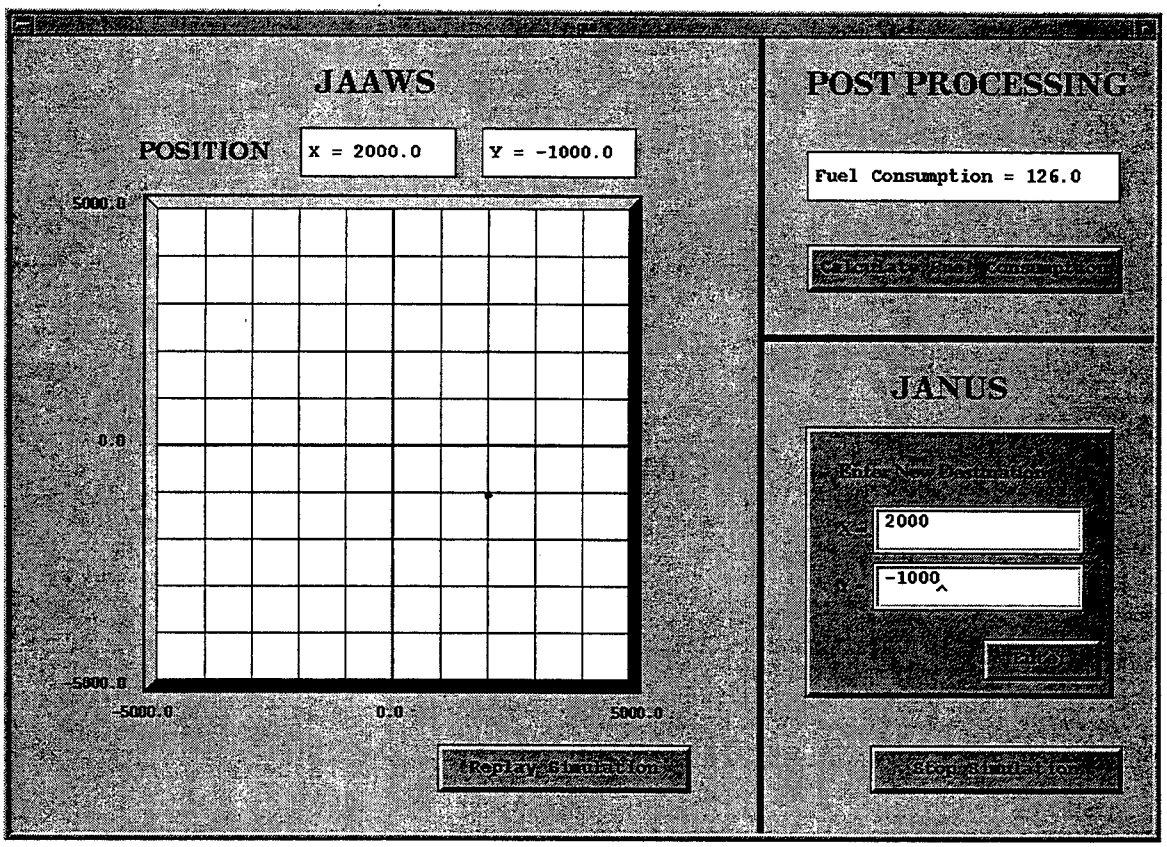


Figure 5. The Graphical User Interface of the executable prototype

5. LESSONS LEARNED

Our prototyping experiment showed that the proposed object-oriented architecture allows design issues to be localized and provides easy means for future extensions. We started out with a prototype consisting of only two events subclasses (move and end_simulation) and were able to add a third event subclass (do_plan) to the prototype without modifying the event control loop of the Janus combat simulator.

We also demonstrated the use of inheritance and polymorphism to efficiently extend/specialize the behavior of combat units. For example, to implement the move_update_object method of a tank subclass which uses the general-purpose method from its superclass to compute its distance traveled and a specialize algorithm to compute its fuel consumption, we simply include 1 statement to invoke the move_update_object method of its superclass followed by 3 lines of code to update its fuel consumption. Moreover, other combat unit subclasses can be added easily to the prototype without the need to modify the event scheduling/dispatching code.

The prototype also resulted in the following refinements to the proposed architecture:

(1) Instead of a procedure with no return value, change the Execute_Event operation to return the time at which the next event is to be scheduled for the same simulation object, and introduce a special time value "NEVER" to indicate the no next event is needed. The proposed change turns the communication between the event dispatcher and the simulation objects from a peer-to-peer communication into a client-server communication. This change eliminates the need for the simulation objects to know the details of the event queue and allows the event dispatcher to use a single statement to schedule all recurring events for all event types. It would also eliminate the need for the write-status event (see Appendix B, page B-1).

(2) Instead of recording the history of a simulation run in terms of set of data files, model the simulation history as a sequence of events. The proposed change provides a simple and uniform way to handle history records for all events, and allows the same modular architecture to be use for real-time simulation as well as post-simulation analysis. This also provides the greatest possible resolution for

the event histories, which implies that any quantity that could have been calculated during the simulation can also be calculated by a post-simulation analysis of the event history, without any loss of accuracy. The only constraint imposed by this design refinement is that the simulation objects in the events must be copied before being included in the simulation history, to protect them from further changes of state as the simulation proceeds. This constraint is easy to meet because the process of writing the contents of an event object to a history file will implicitly make the required copy.

The prototyping effort also exposed a design issue – should null events appear in the event queue? A null event is one that does not affect the state of the simulation, such as a move event for an object that is currently stationary. The prototype version adopted the position that such events should not be put in the event queue, since this corresponds to current scheduling policies in Janus, and appears at first glance to improve efficiency.

Our experience with the development of the prototype suggests that this decision complicates the logic and may not in fact improve efficiency. In particular, the process create_new_events (see Figure 3) could be eliminated if we allowed null events. This process scans all simulation objects once per simulation cycle to determine if any dormant objects have become active, and if so, schedules events to handle their new activity. The alternative is to have the constructor of each kind of simulation objects schedule all of its initial events, and to have each event handler specify the time of next instance of the same event even if there is nothing for it to do currently. Handlers might still set the time of its next event to NEVER in the case of a catastrophic kill; however this is reasonable only if it is impossible to repair or restore the operation of the units that have suffered a catastrophic kill.

The reasons why this design change may improve efficiency in addition to simplifying the code are that:

(1) the check for whether a dormant object has become active is done less often - once per activity of that object, rather than once per simulation cycle,

(2) executing a null event is every fast – a few instructions at most, so the "unnecessary" null events will not have much impact on execution time, and

(3) the computation to find and test all simulation objects periodically would be eliminated.

One recommendation is to allow null events in the event queue, and to explicitly schedule every kind of events for every object unless it is known there cannot be any non-empty events of that type in any possible state of the object. For example, under the proposed scheduling policy, immobile or irrecoverably damaged objects would not need to schedule future move events, but those that are currently at their planned positions would need to do so, because a change of plan would cause them to move again in the future, even though they are not currently moving.

6. CONCLUSION

Our experience in this case study suggests that prototyping can be a valuable aid in re-engineering of legacy systems, particularly in cases where radical changes to system conceptualization and software structure are needed.

In particular, we found that constructing even a very thin skeletal instance of the proposed new architecture raised many issues and enabled us to correct, complete, and optimize the architecture for both simplicity and performance.

The computer-aided prototyping tools in the CAPS system enabled us to do this with a minimal amount of coding effort. The bulk of the code was generated automatically, enabling us to concentrate on system structuring issue, to consider and evaluate various alternatives, and to improve the design while doing detailed manual implementation for only a few pages of critical code.

BIBLIOGRAPHY

1. *Janus 3.X/UNIX Software Design Manual.* Prepared for: Headquarters TRADOC Analysis Center, Ft. Leavenworth, Kansas. Prepared by: Titan, Inc. Applications Group, Leavenworth, Kansas. Nov. 93.

2. *Janus 3.X/UNIX Software Programmer's Manual.* Prepared for: Headquarters TRADOC Analysis Center, Ft. Leavenworth, Kansas. Prepared by: Titan, Inc. Applications Group, Leavenworth, Kansas. Nov. 93.

3. L.R. Larimer, *Building an Object Model of a Legacy Simulation*, MS Thesis, NPS, June 1997.

4. *Janus Version 6 User's Manual,* Simulation, Training & Instrumentation Command, Orlando, Florida, 1995.

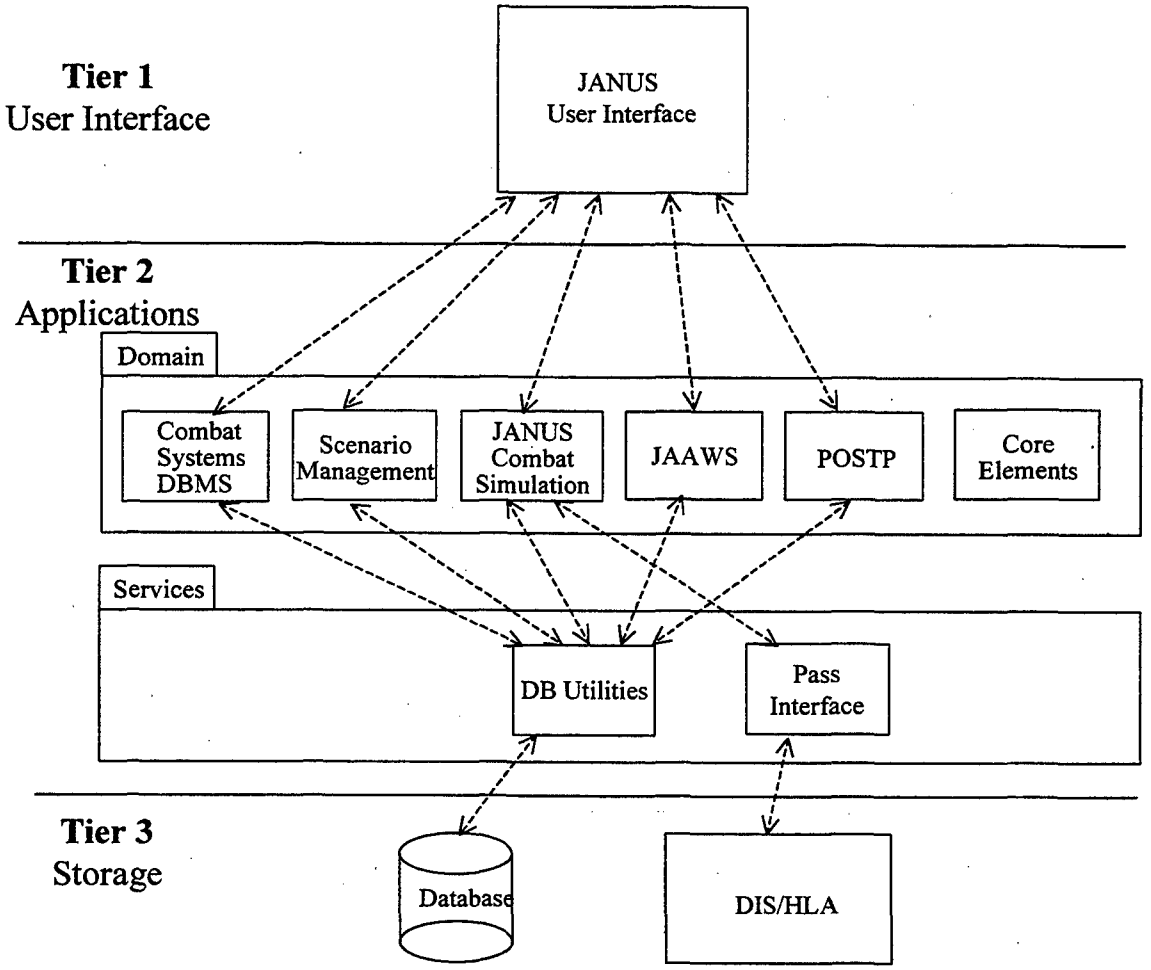
5. *Janus Version 6 Data Base Manager's Manual,* Simulation, Training & Instrumentation Command, Orlando, Florida, 1995.

6. J. Pimper and L. Dobbs, *Janus Algorithm Document*, Version 4.0, Lawrence Livermore National Laboratory, California, 1988.

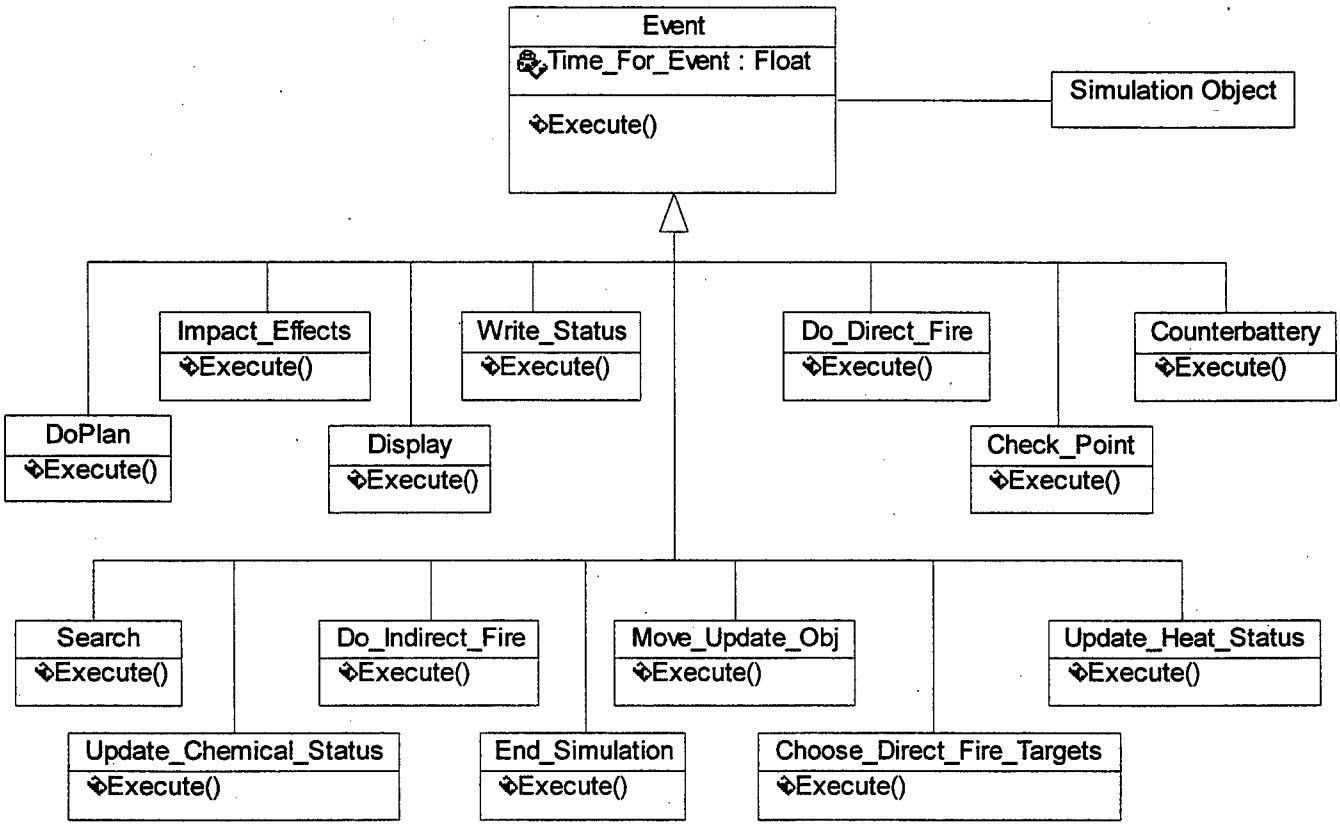
7. Luqi and M. Ketabchi, "A Computer-Aided Prototyping System", *IEEE Software*, 5(2), 1988, pp. 66-72.

8. Luqi, "System Engineering and Computer-Aided Prototyping", *Journal of Systems Integration – Special Issue on Computer Aided Prototyping, 6(1),* 1996, pp. 15-17.

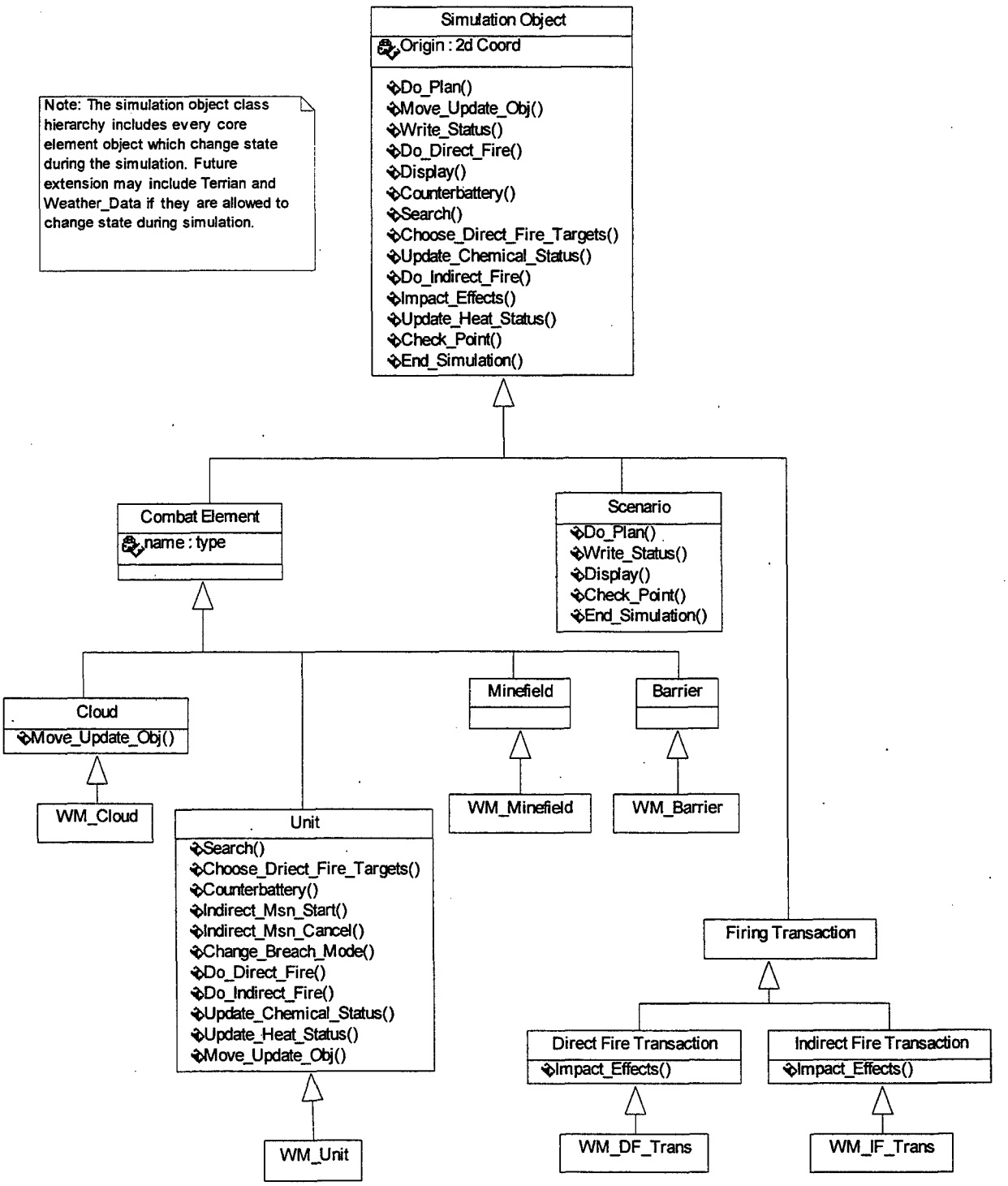
APPENDIX A. The proposed 3-tier object-oriented architecture.



APPENDIX B. The event class hierarchy showing the different event handling operations of the JANUS Combat Simulation Subsystem.



APPENDIX C. The simulation object class hierarchy showing the distribution of the event operations.



APPENDIX D. The event classes of the JANUS Combat Simulation Subsystem.

Name of Event	Objects responsible to handle the event	Remarks
Do_plan	scenario	May be initiated by the graphical user interface and it in turn may schedule other events. See Note 1.
Display	scenario	May be triggered at regular time interval. See Note 1.
Write_Status	scenario	May be triggered at regular time interval. See Note 1.
Check_Point	scenario	May be triggered at regular time interval. See Note 1.
Move_Update_Obj	unit	Updates unit position due to movement, and schedules the next Move_Update_Obj event for the object. Mapped to movement.f, copter.f, update.f updsldr.f, updflyer.f
	cloud	Updates shape, location (if needed), and expiration time. It schedules the next Move_Update_Obj event for the object. Mapped to part of docloud.f, cldupd.f, and chmcld.f.
Search	unit	Update potential target list. Use different methods depending on the kind of sensors the unit has. It also schedules the next search event for the object. Mapped to search.f, part of radar.f for normal and special radar.

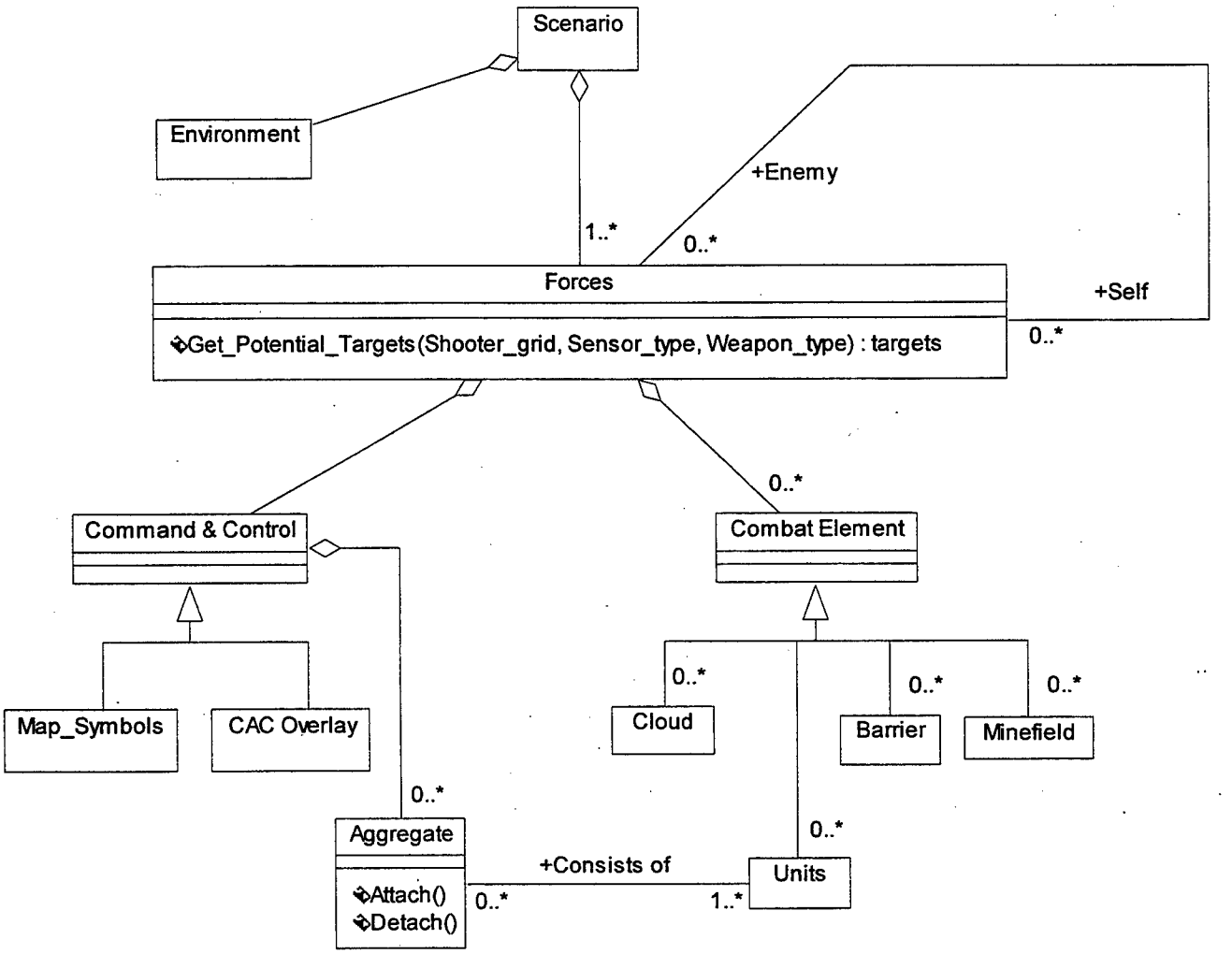
Choose_Direct_Fire_Targets	unit	Updates visibility level and performs IFF to produce confirmed target list. Selects weapons for the targets in the potential target list. Choose target from the confirmed target list and schedule a direct fire event. Use different methods depending on the kind of platform the unit belongs to and the kind of sensors the unit has. It also schedules next Choose_Direct_Fire_Targets event for the object. Mapped to dodetect.f, detect.f, flydetc.f, handoff.f, and reload, and part of radar.f (for normal radar).
Do_Direct_Fire	unit	Creates a Direct_Firing_Transaction object and schedules an Impact_Effects event for the object. Mapped to shoot.f , and adfire.f for normal radar.
Do_Indirect_Fire	unit	Execute an arty mission. Creates an Indirect_Firing_Transaction and schedule an Impact_Effects event for the object. Mapped to doarty.f, and part of firing.f. Event scheduled by Do_Plan event.
Impact_Effects	direct_firing _transaction	Evaluate the effect of the direct fire event and update the affected objects accordingly. Mapped to dfmpact.f.
	indirect _firing _transaction	Evaluate the effect of the indirect fire event and update the affected objects accordingly. May create cloud objects and schedule Move_Update_Obj event for the cloud objects. Mapped to impact.f.
Counterbattery	unit	Searches for projectiles of enemy fires and schedules next Counterbattery event for the object. The information about enemy fires can be displayed to the user via a Do_Plan event. Mapped to cntrbat.f.
Update_Chemical_Status	unit	Updates chemical effects on unit and schedules next Update_Chemical_Status for the object. Mapped to part of dochem.f

Update_Heat_Status	unit	Updates heat effects on unit and schedules next Update_Heat_Status for the object. Mapped to part of doheat.f
End_Simulation	scenario	Clears the priority event queue and performs housekeeping activities. See Note 1.

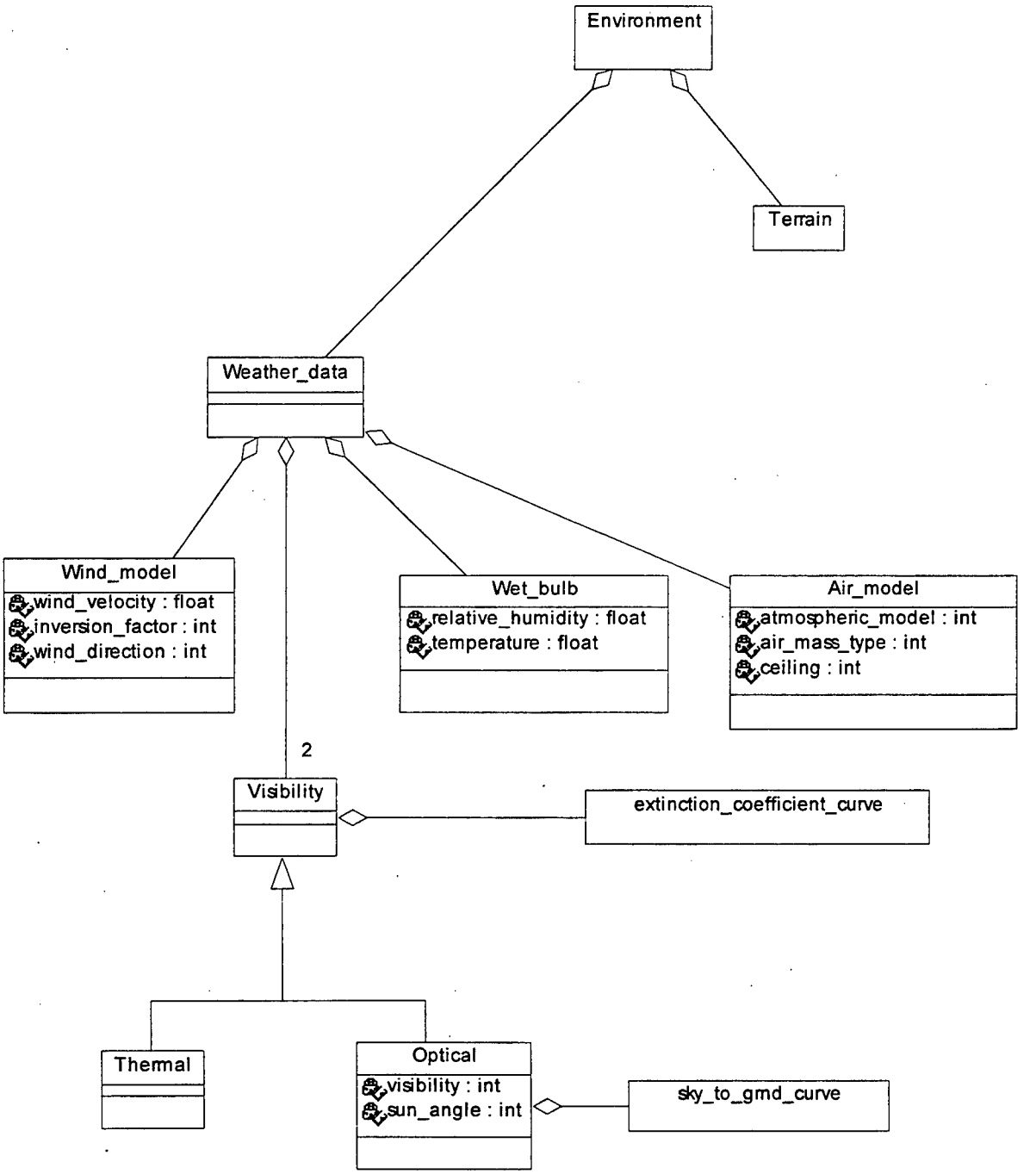
Note 1. Depending on the graphical user interface design, this event may be replaced by different events and assign the event handlers to the individual objects.

APPENDIX E. The object models for the JANUS core elements.

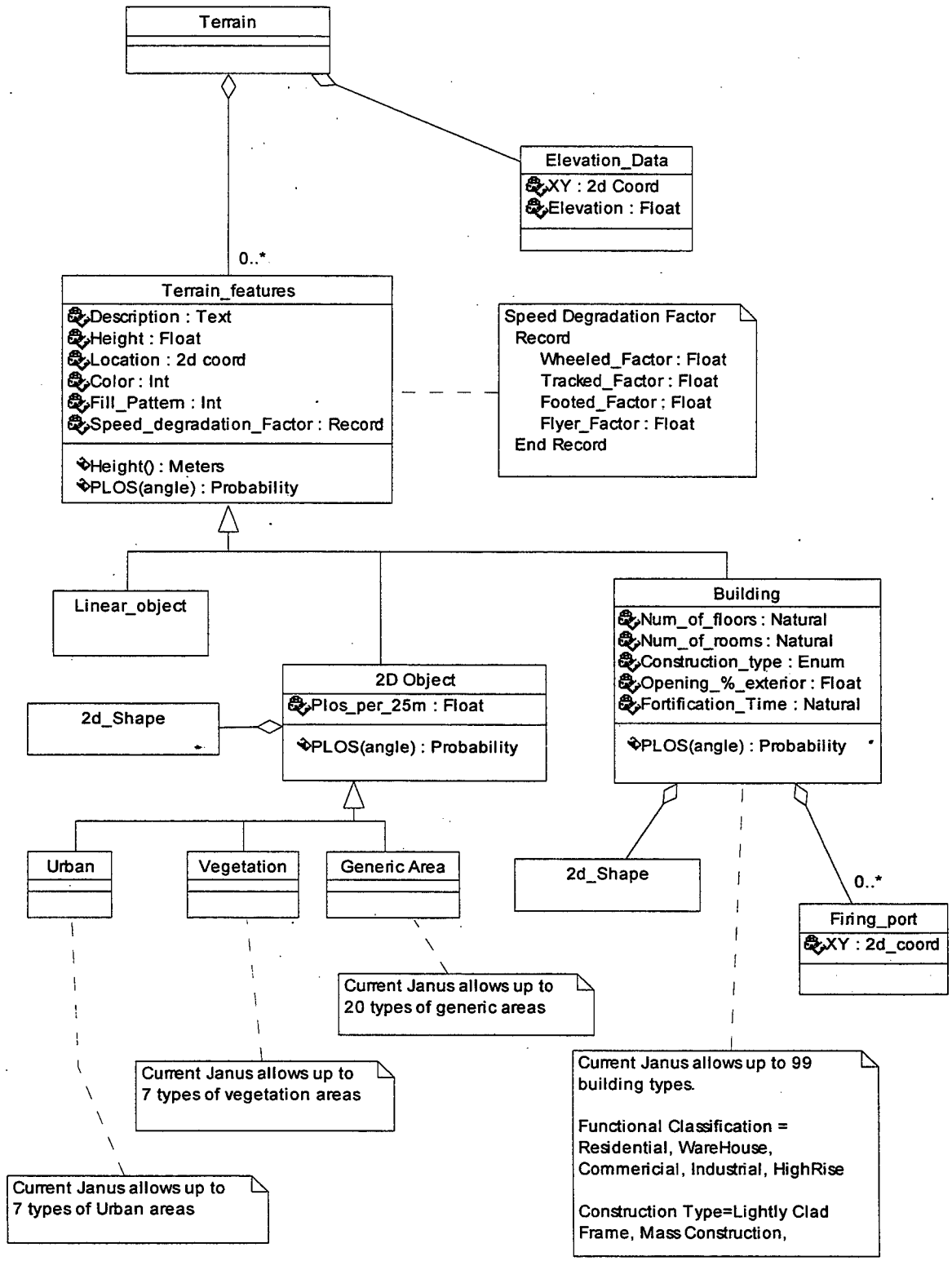
1. The overall structure of the JANUS core elements object model.



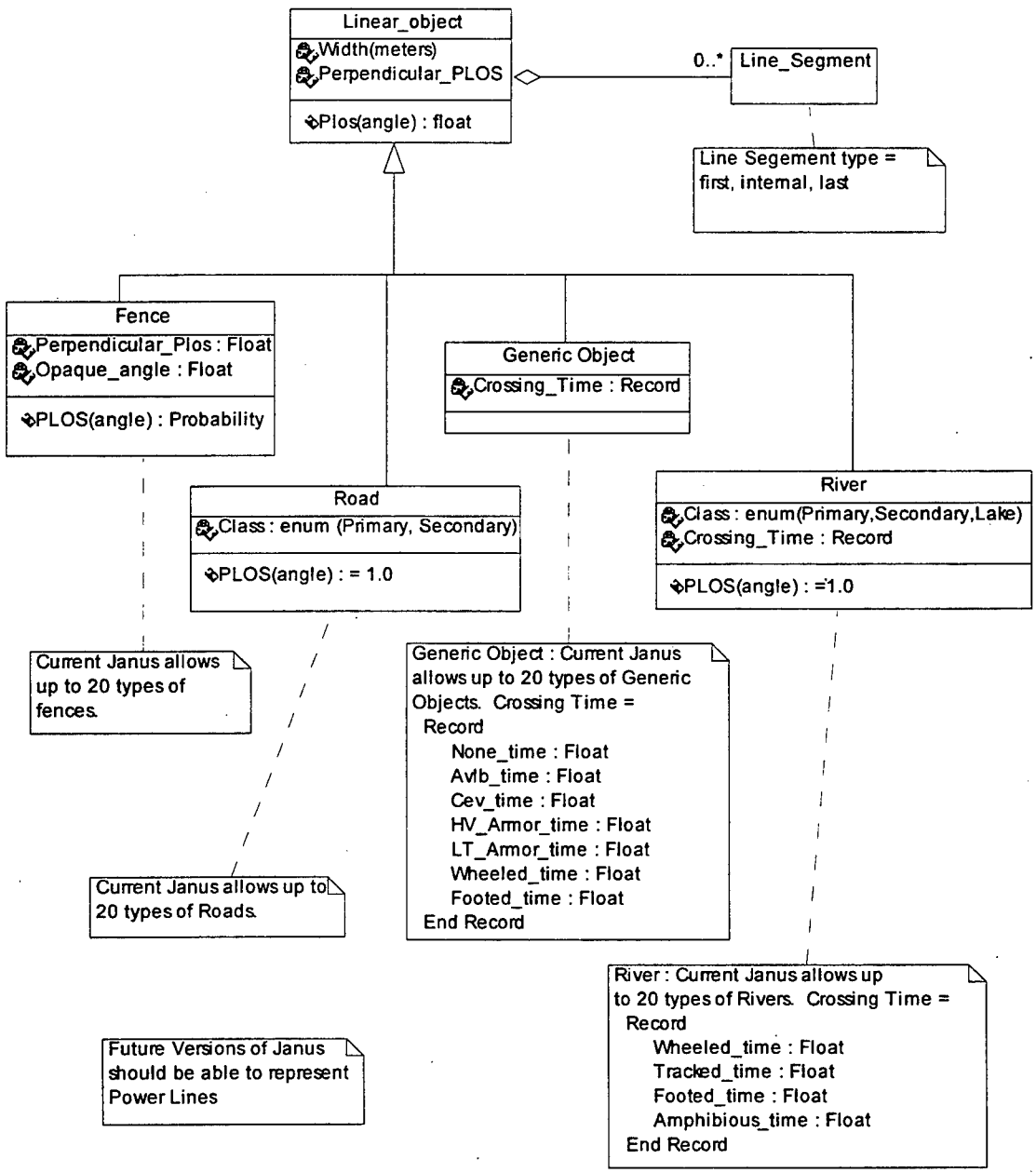
2. The structure of the Environment class and the Weather_Data class.



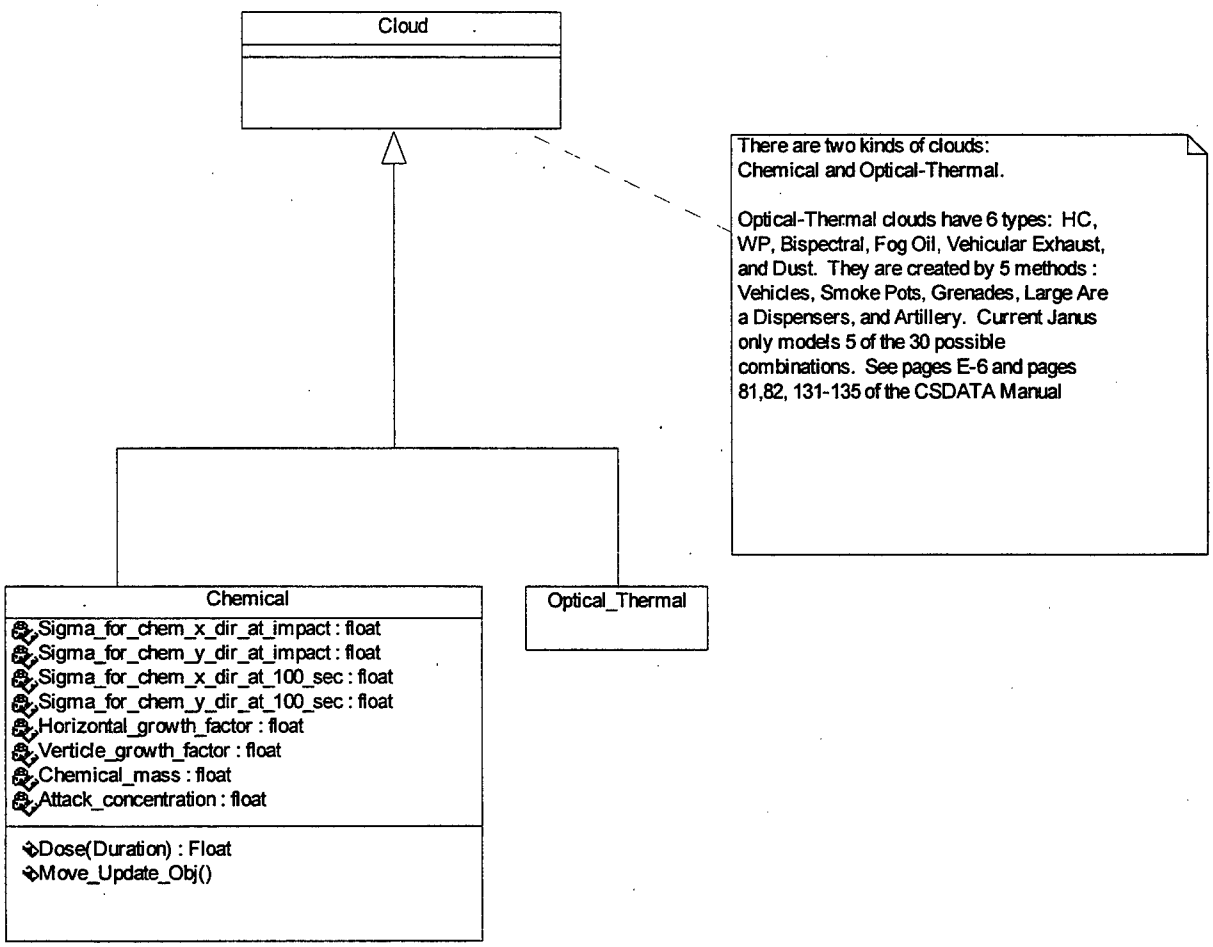
3. The structure of the Terrain class, an aggregate of the Environment class.



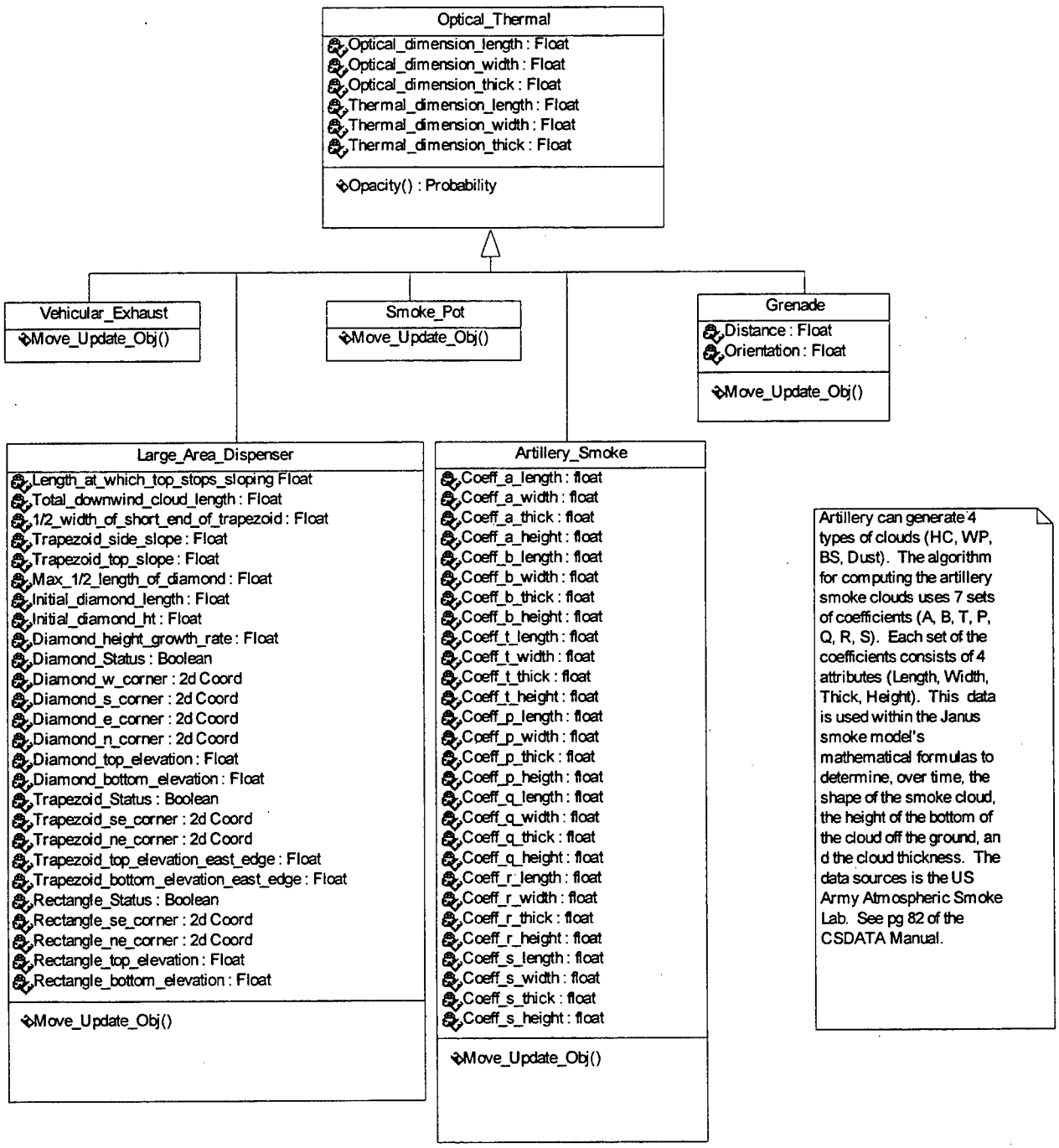
4. The structure of the Linear_Object class, a subclass of the Terrain class.



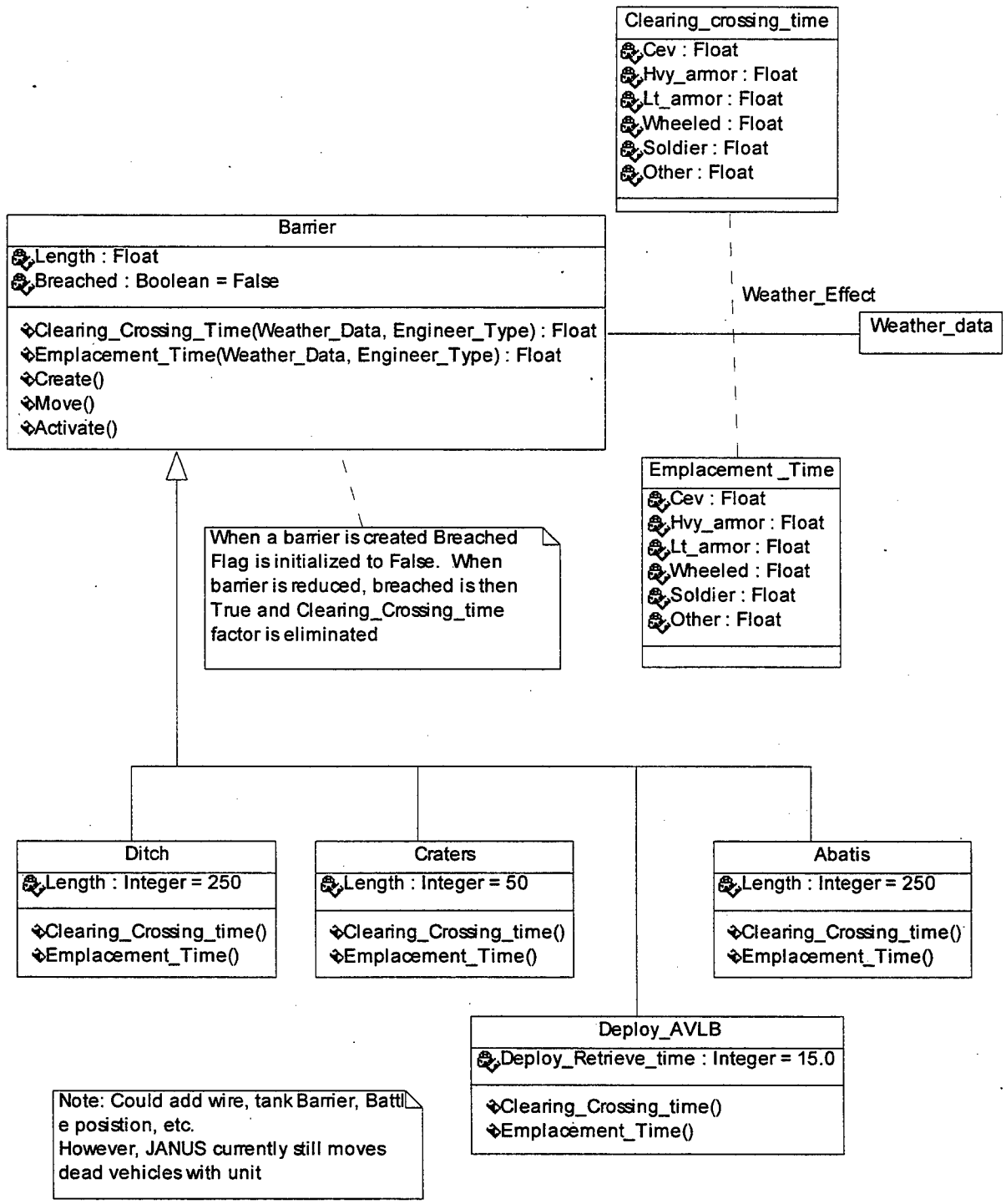
5. The structure of the Cloud class, a subclass of the Combat_Element class.



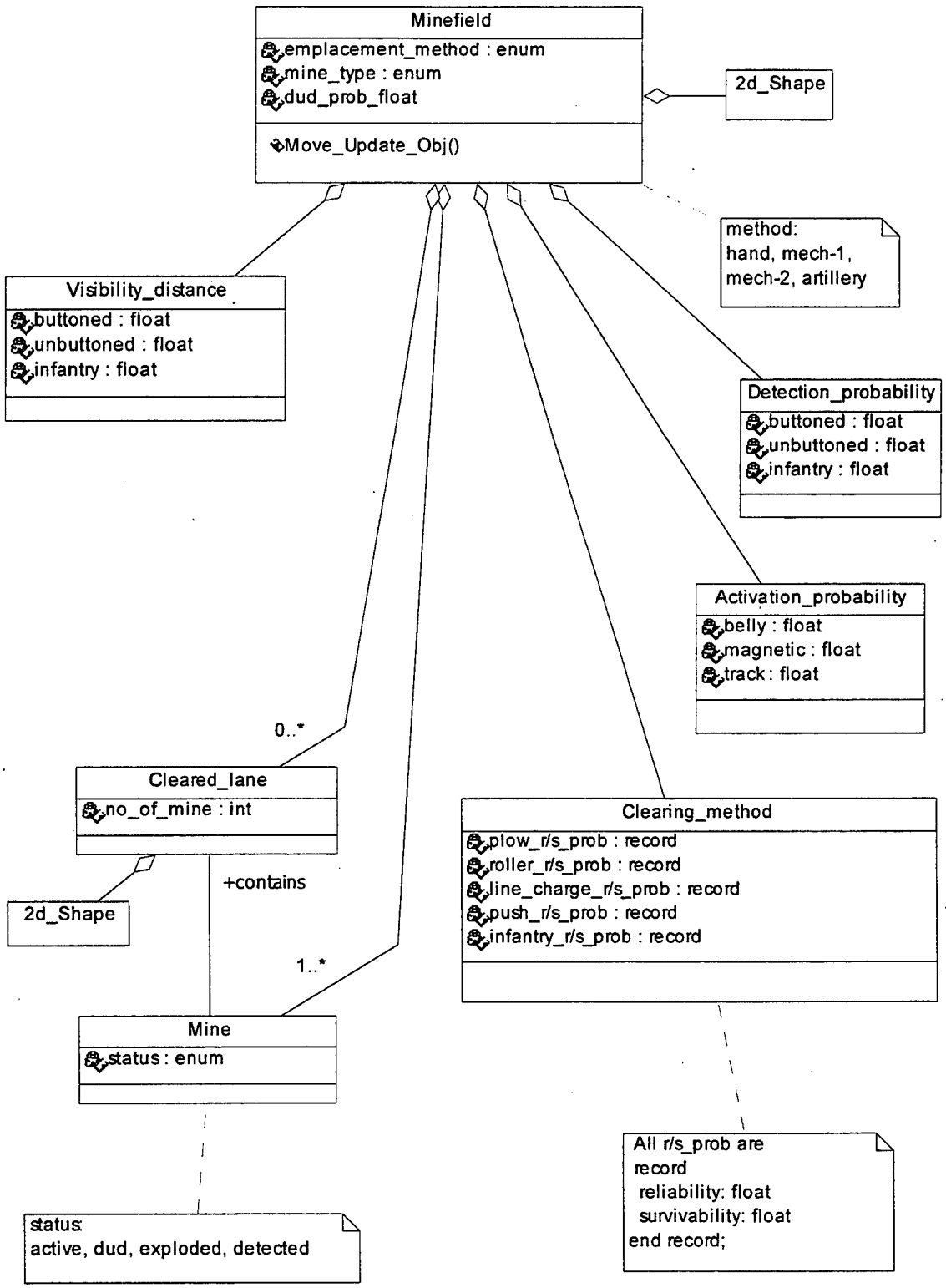
6. The structure of the Optical_Thermal class, a subclass of the Cloud class.



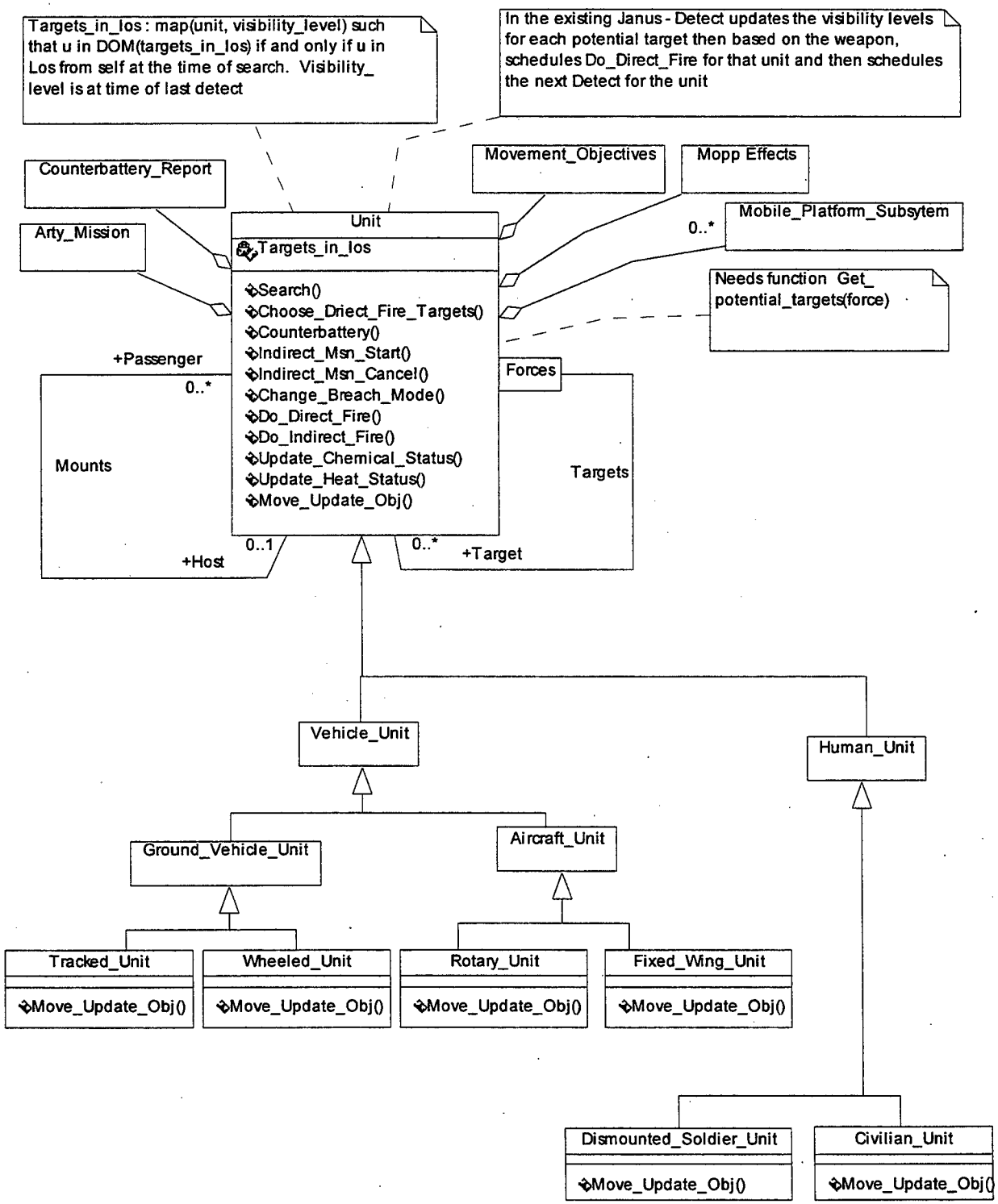
7. The structure of the Barrier class, a subclass of the Combat_Element class.



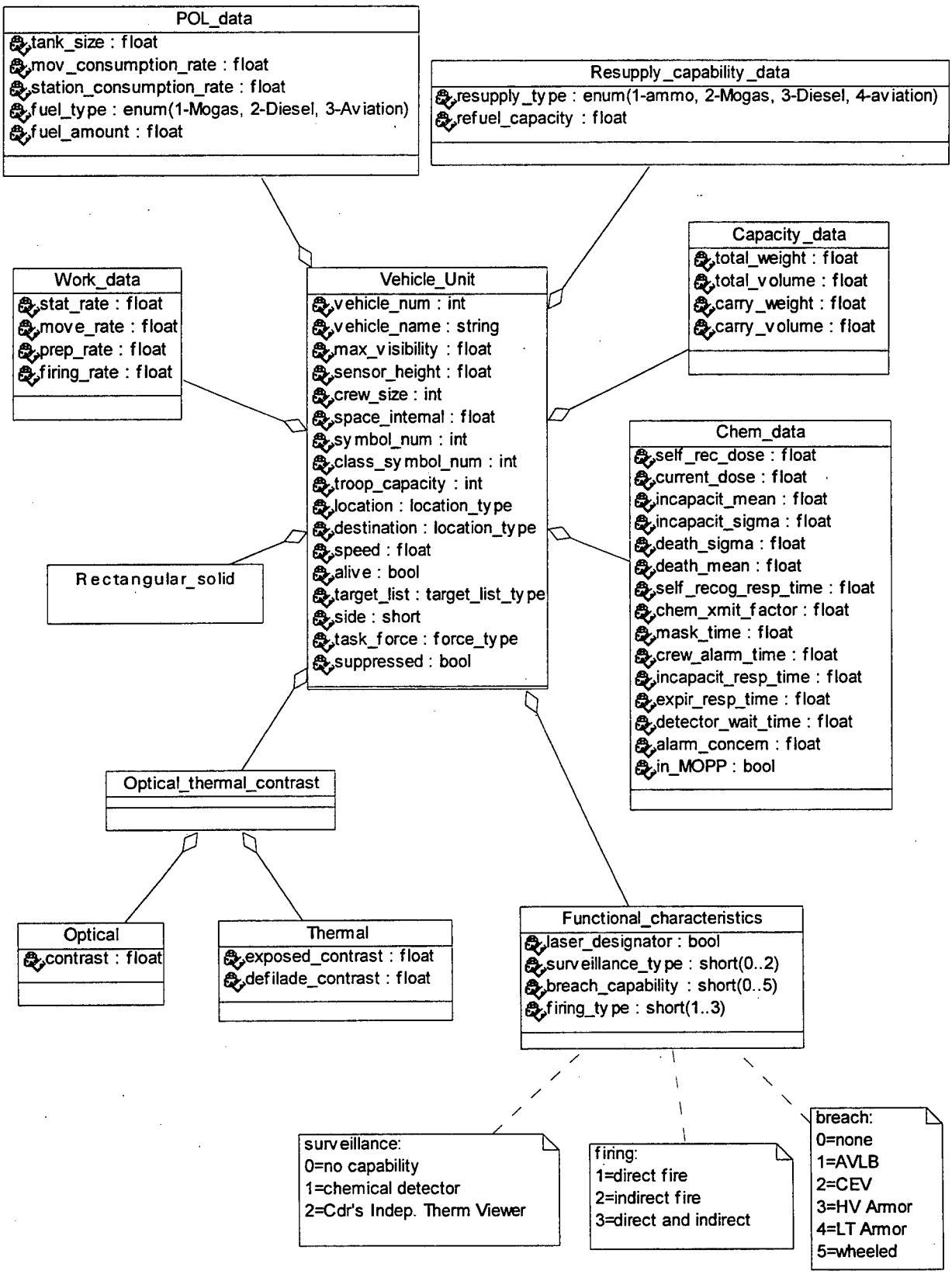
8. The structure of the Minefield class, a subclass of the Combat_Element class.



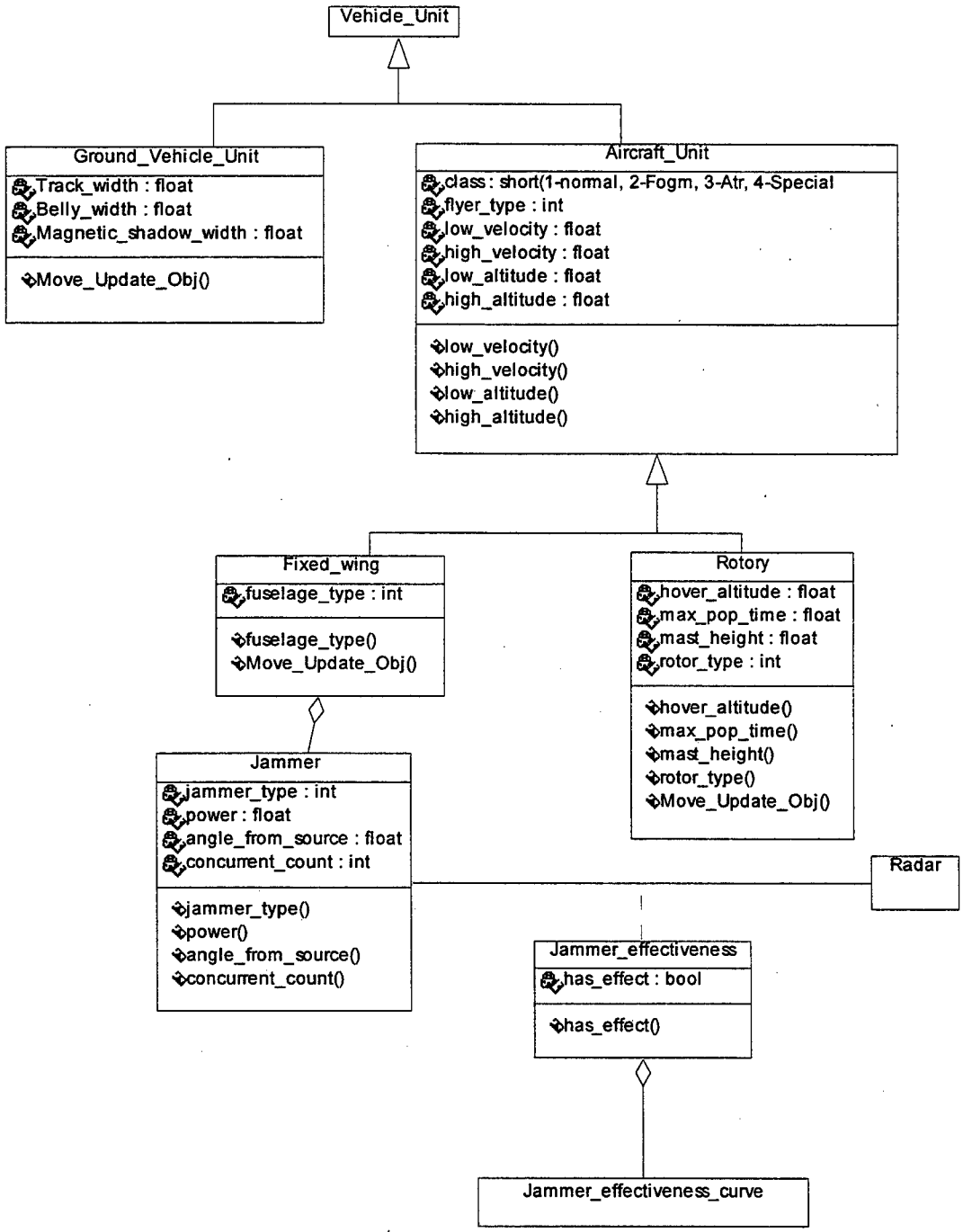
9. The structure of the Unit class, a subclass of the Combat_Element class.



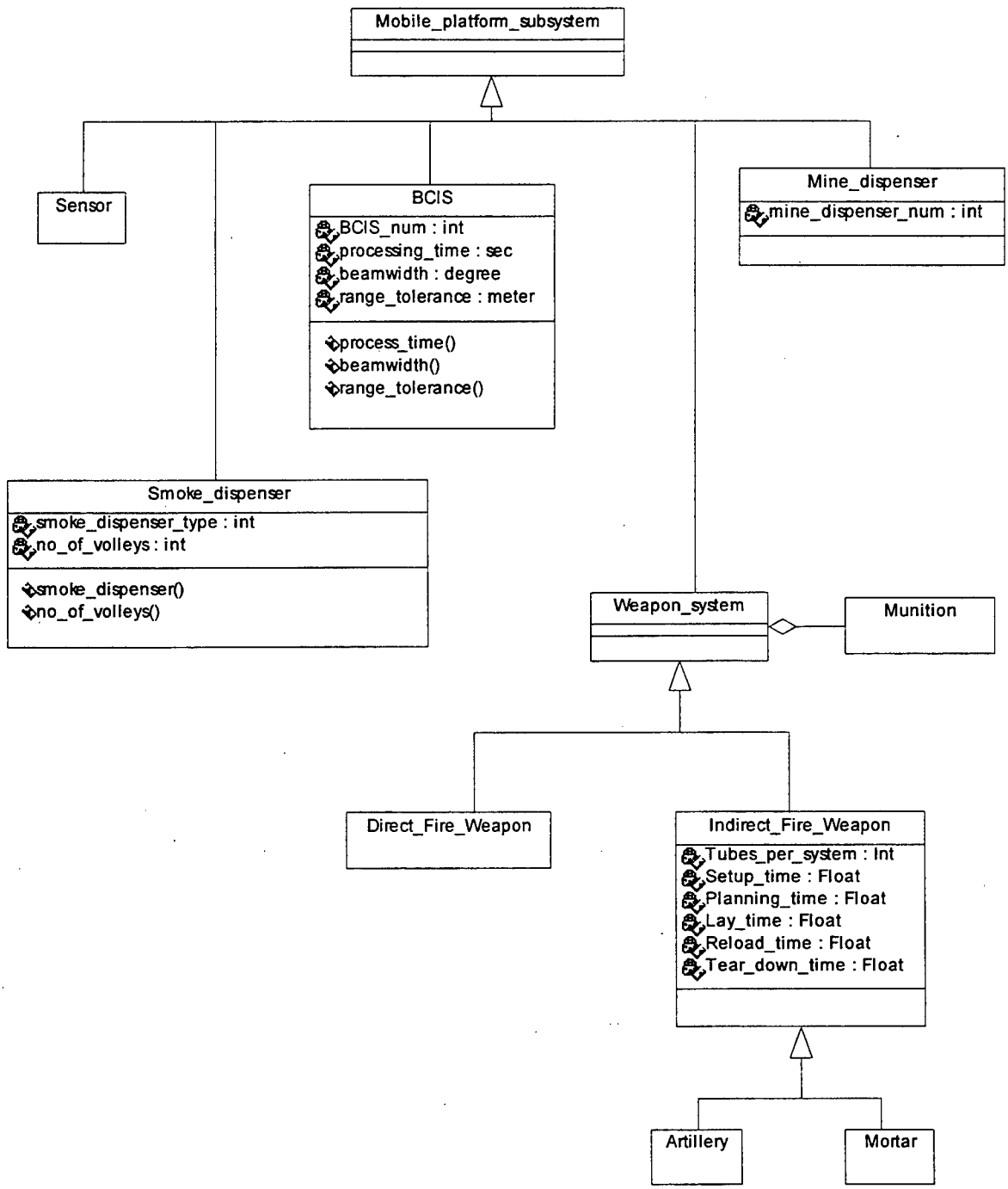
10. The structure of the Vehicle_Unit class, a subclass of the Unit class.



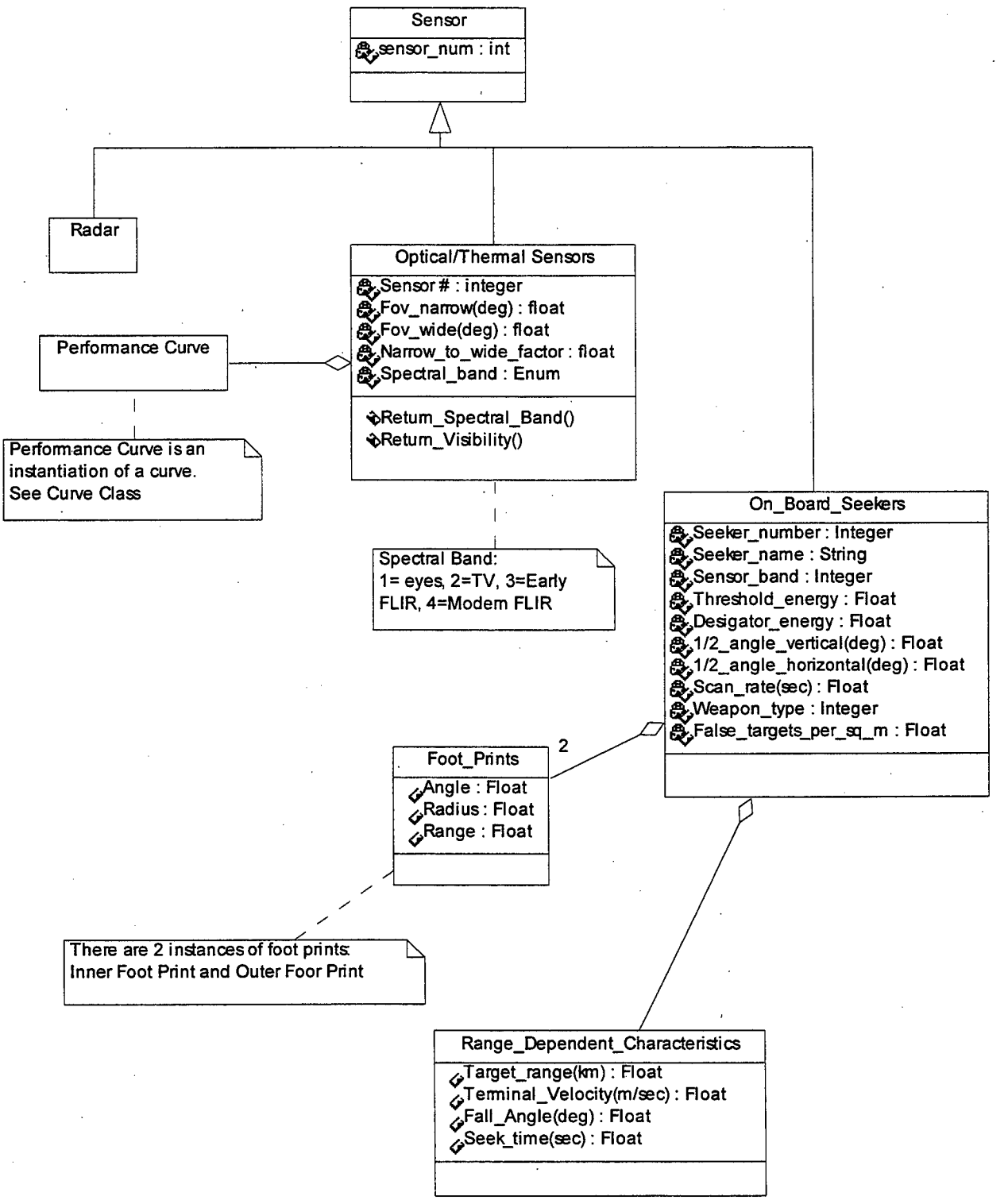
11. The structure of the Aircraft_Unit class and the Ground_Vehicle_Unit, subclasses of the Unit class.



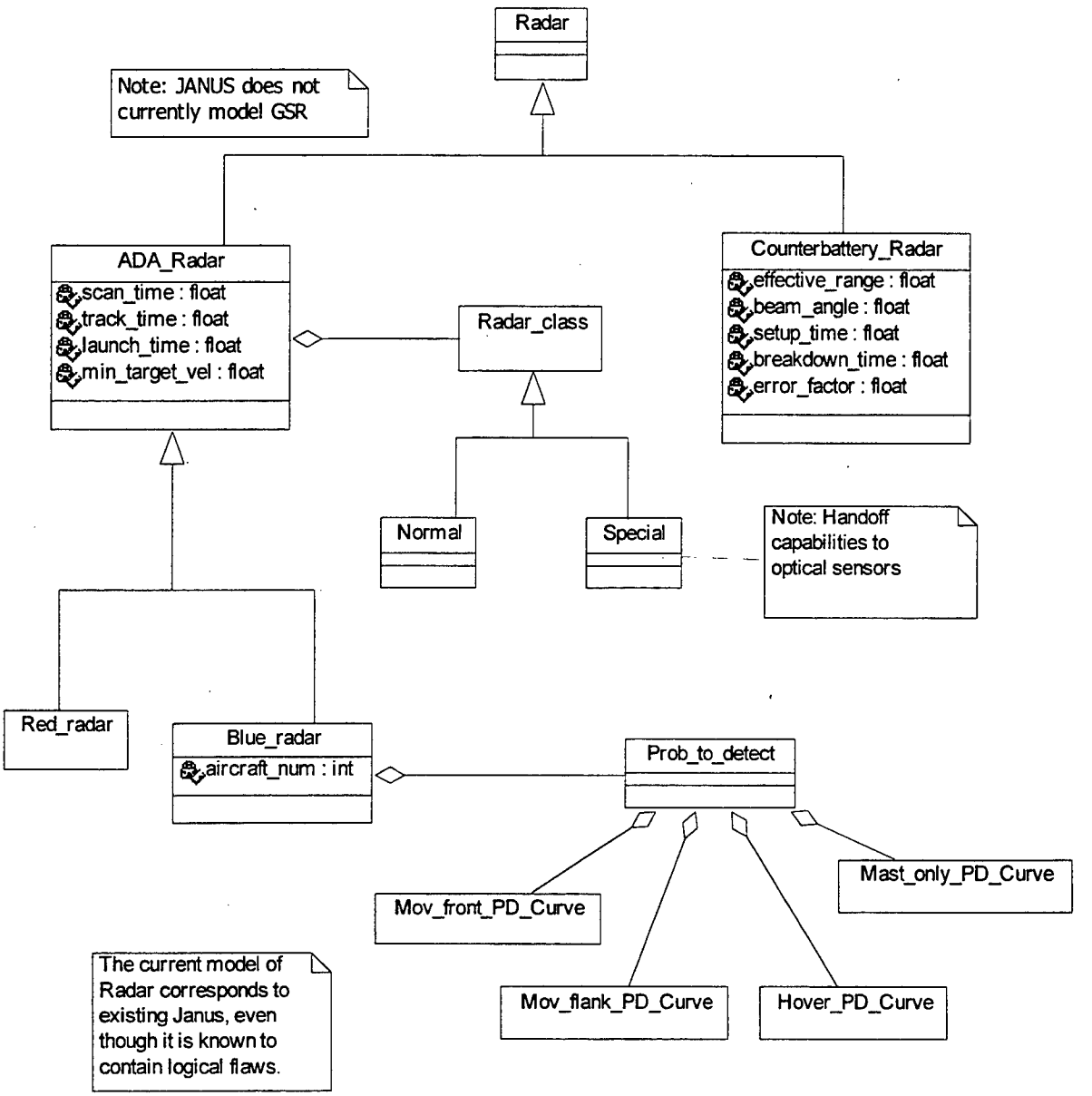
12. The structure of the Mobile_Platform_Subsystem class, an aggregate of the Unit class.



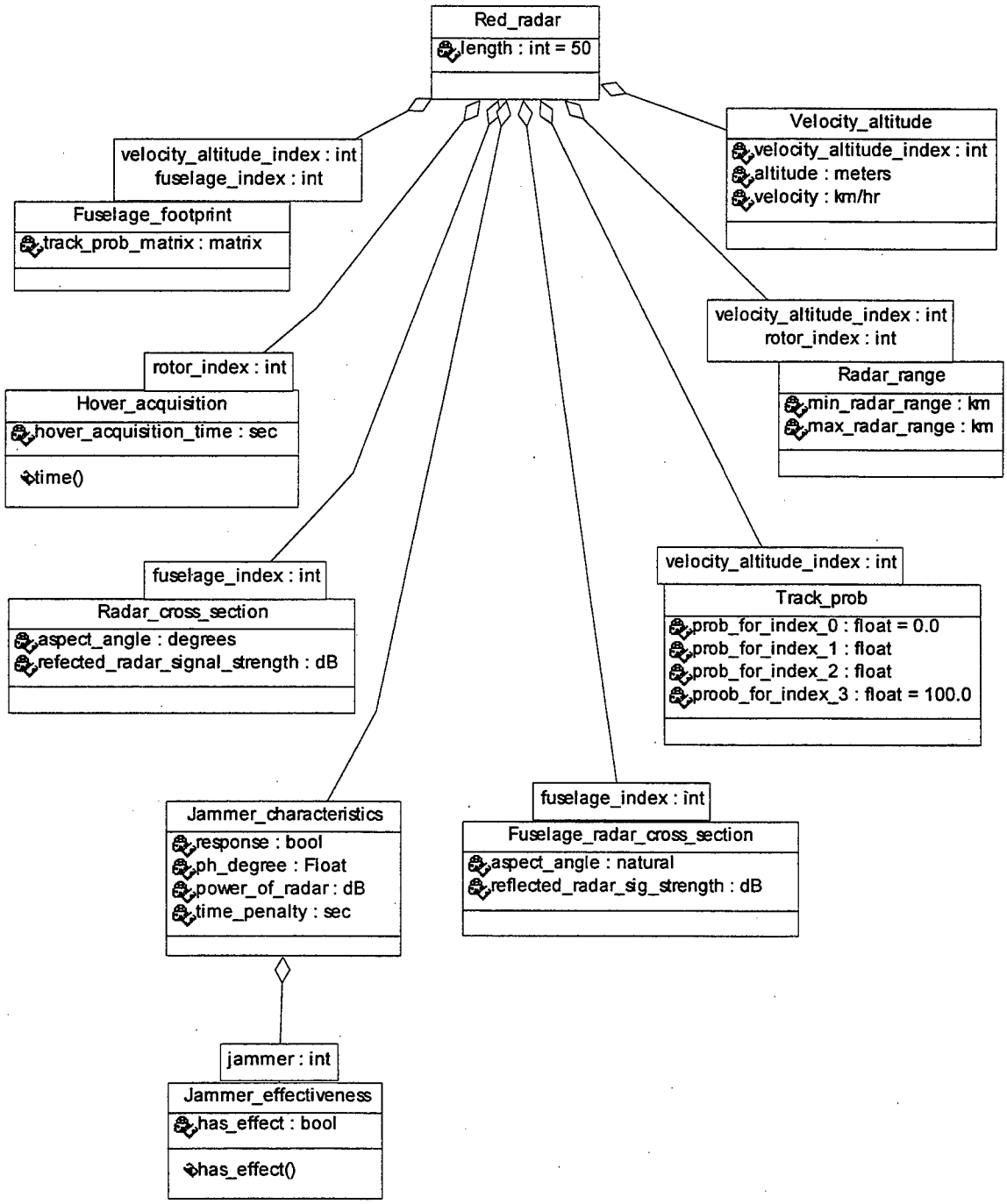
13. The structure of the Sensor class, a subclass of the Mobile_Platform_Subsystem class.



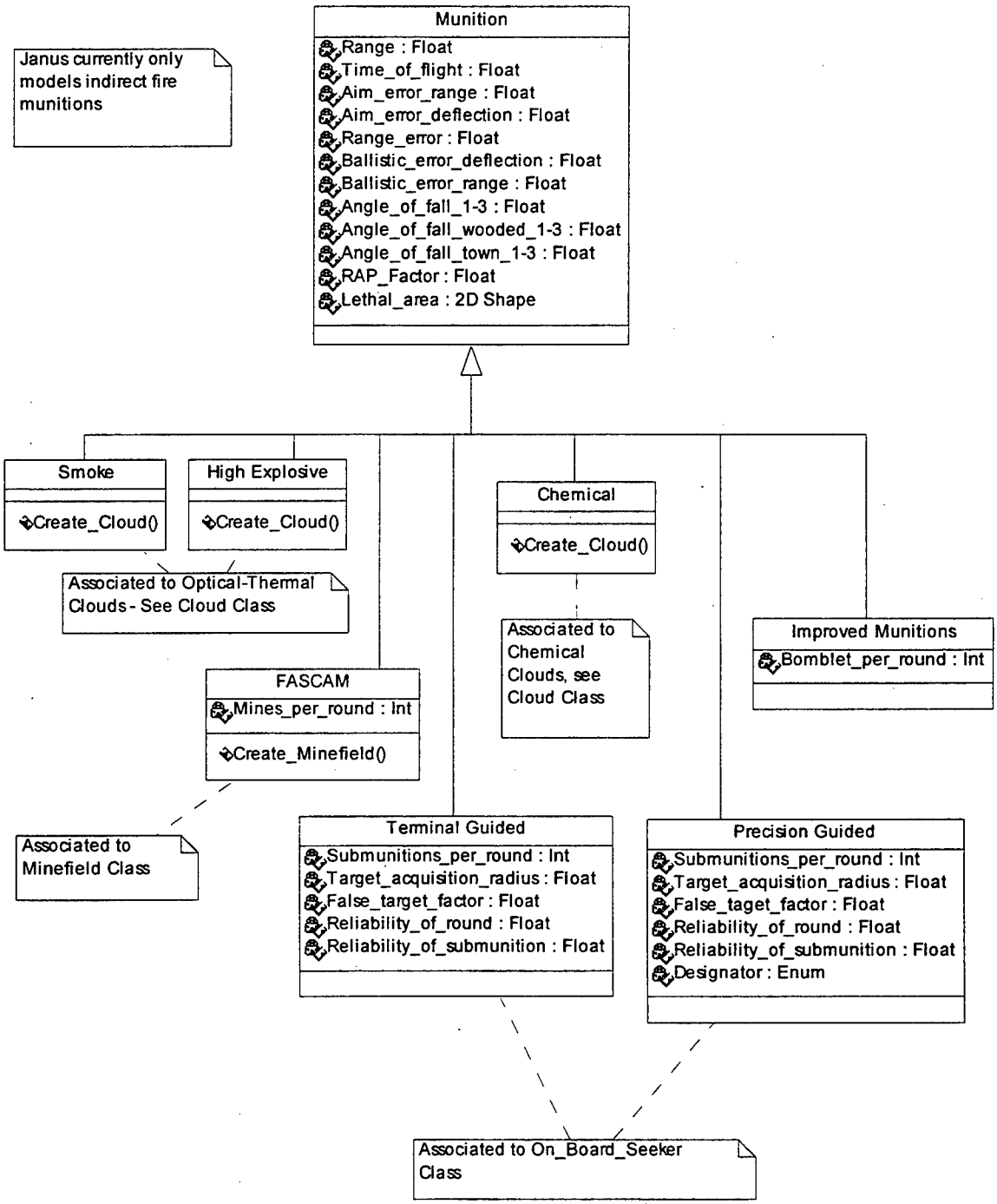
14. The structure of the Radar class, a subclass of the Sensor class.



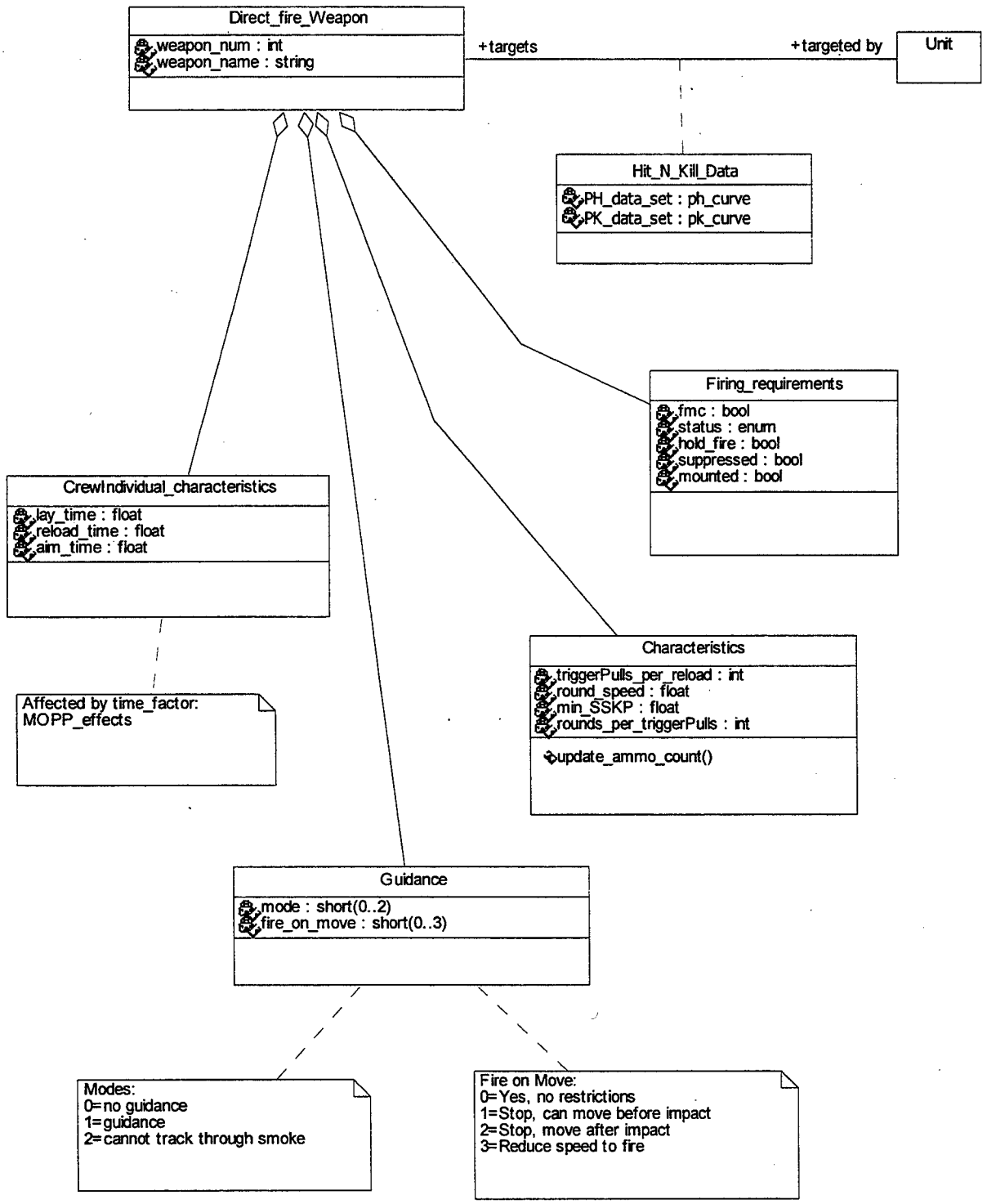
15. The structure of the Red_Radar class, a subclass of the ADA_Radar class.



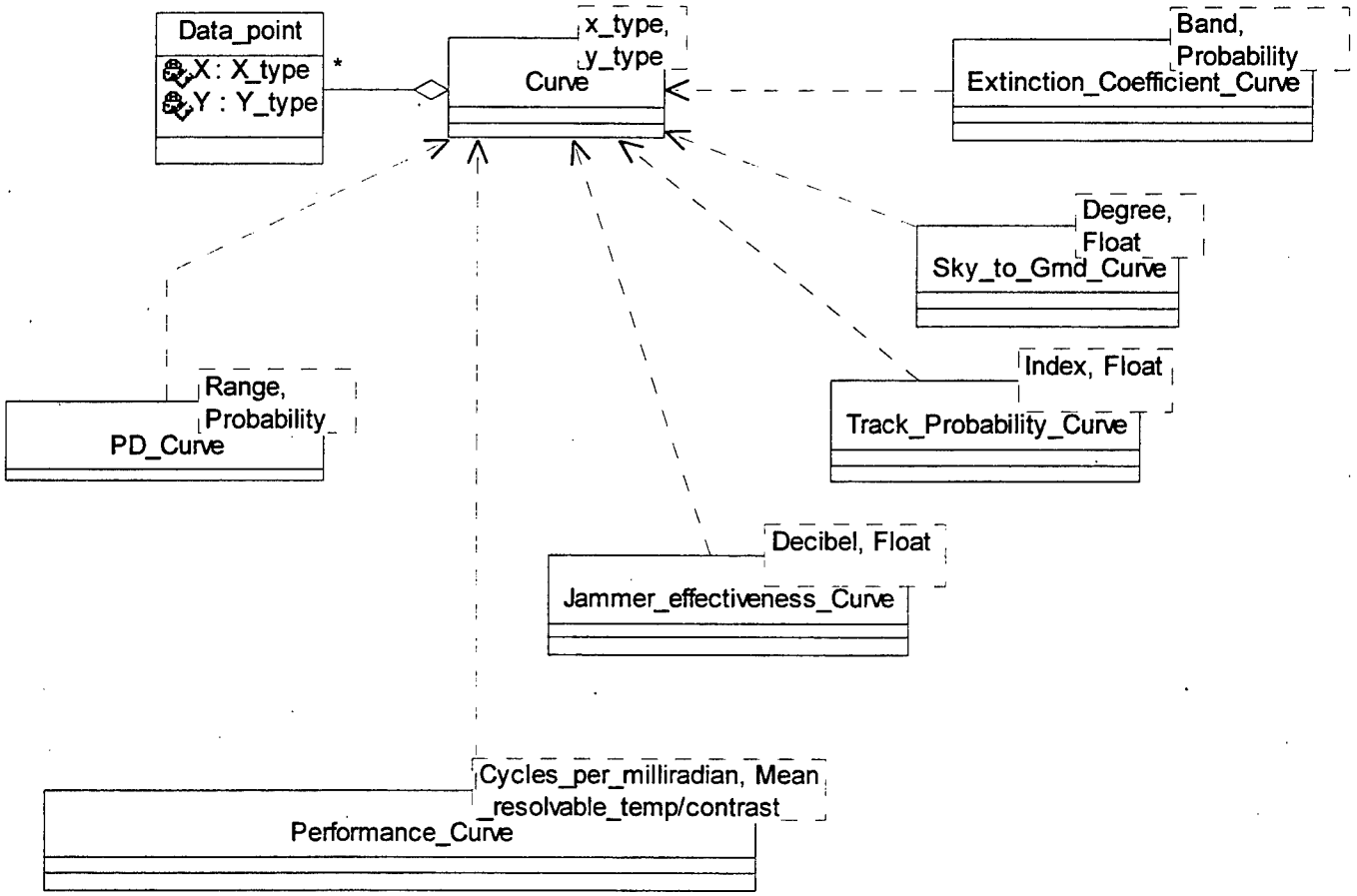
16. The structure of the Munition_Type class, an aggregate of the Unit class.



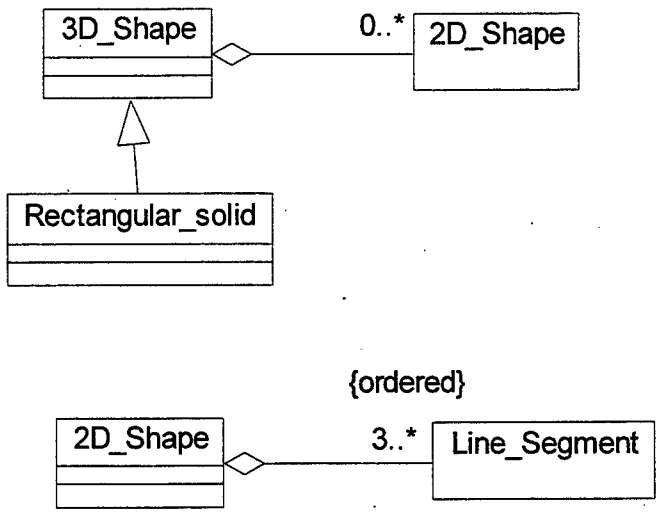
17. The structure of the Direct_Fire_Weapon class, a subclass of the Weapon_System class.



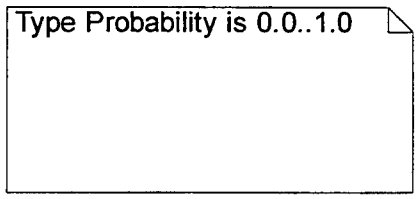
18. The structure of the Curve class, a generic data structure.

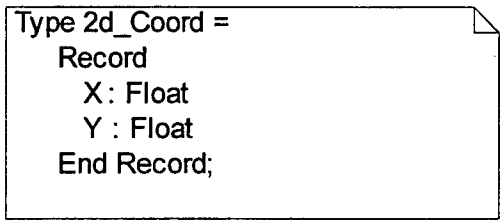


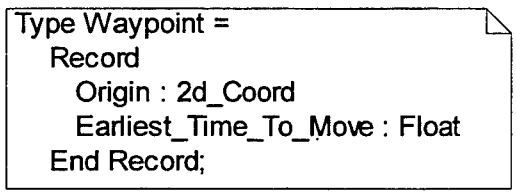
19. The structure of the 3D_Shape, 2D_Shape and Line_Segment classes.



20. The definition of the Probability, 2D_Coordinate, and Waypoint data types.







APPENDIX F. The PSDL specification for the executable prototype.

```
TYPE event_type
SPECIFICATION
END

IMPLEMENTATION ada event_type
END

TYPE event_queue_type
SPECIFICATION

  OPERATOR empty_queue
  SPECIFICATION
    OUTPUT q: event_queue_type
  END

END

IMPLEMENTATION ada event_queue_type
END

TYPE statistics_type
SPECIFICATION
END

IMPLEMENTATION ada statistics_type
END

TYPE scenario_type
SPECIFICATION

  OPERATOR empty_scenario
  SPECIFICATION
    OUTPUT s: scenario_type
  END

END

IMPLEMENTATION ada scenario_type
END

TYPE statistics_request_type
SPECIFICATION
END

IMPLEMENTATION ada statistics_request_type
END

TYPE replay_request_type
SPECIFICATION
END

IMPLEMENTATION ada replay_request_type
END

TYPE user_interaction_type
SPECIFICATION

  OPERATOR stop_simulation
  SPECIFICATION
    OUTPUT x: user_interaction_type
  END
```

```
END

IMPLEMENTATION ada user_interaction_type
END

TYPE location_type
SPECIFICATION
END

IMPLEMENTATION ada location_type
END

TYPE game_time_type
SPECIFICATION

  OPERATOR zero
  SPECIFICATION
    OUTPUT z: game_time_type
  END

END

IMPLEMENTATION ada game_time_type
END

OPERATOR gui_3
  SPECIFICATION
    INPUT statistics: statistics_type
    INPUT replay: location_type
    OUTPUT scenario: scenario_type
    OUTPUT user_interaction: user_interaction_type
    OUTPUT replay_request: replay_request_type
    OUTPUT statistics_request: statistics_request_type
    STATES scenario: scenario_type INITIALLY.scenario_type.empty_scenario
    STATES new_y: float INITIALLY 0.0
    STATES new_x: float INITIALLY 0.0
    STATES first_time: boolean INITIALLY TRUE
  END

  IMPLEMENTATION
    GRAPH
      VERTEX enter_new_plan_75_74
      VERTEX get_y_68_67
      VERTEX get_x_65_64
      VERTEX get_re_30_29
      VERTEX get_st_27_26
      VERTEX edit_plan_24_23
      VERTEX get_user_in_21_20
      VERTEX gui_event_monitor_18_17: 50 MS
      VERTEX display_st_31_30
      VERTEX display_re_37_36
      VERTEX initial_scenario_40_39
      EDGE new_plan_entered enter_new_plan_75_74 -> edit_plan_24_23
      EDGE new_y get_y_68_67 -> edit_plan_24_23
      EDGE new_x get_x_65_64 -> edit_plan_24_23
      EDGE scenario edit_plan_24_23 -> edit_plan_24_23
      EDGE scenario edit_plan_24_23 -> EXTERNAL
      EDGE statistics_request get_st_27_26 -> EXTERNAL
      EDGE replay_request get_re_30_29 -> EXTERNAL
      EDGE user_interaction get_user_in_21_20 -> EXTERNAL
      EDGE statistics EXTERNAL -> display_st_31_30
      EDGE scenario initial_scenario_40_39 -> EXTERNAL
```

```
      EDGE replay EXTERNAL -> display_re_37_36
      EDGE first_time initial_scenario_40_39 -> initial_scenario_40_39
    DATA STREAM
      new_plan_entered: boolean
    CONTROL CONSTRAINTS
      OPERATOR enter_new_plan_75_74
      OPERATOR get_y_68_67
      OPERATOR get_x_65_64
      OPERATOR get_re_30_29
      OPERATOR get_st_27_26
      OPERATOR edit_plan_24_23
        TRIGGERED BY ALL new_plan_entered
      OPERATOR get_user_in_21_20
      OPERATOR gui_event_monitor_18_17
        PERIOD 300 MS
        FINISH WITHIN 300 MS
      OPERATOR display_st_31_30
      OPERATOR display_re_37_36
      OPERATOR initial_scenario_40_39
        TRIGGERED IF (first_time = TRUE)
  END

OPERATOR warrior_1
  SPECIFICATION
    STATES replay_position: integer INITIALLY 1
    STATES replay_request: replay_request_type INITIALLY
replay_request_type.off
  END

  IMPLEMENTATION
    GRAPH
      VERTEX gui_3_2
      VERTEX post_processor_6_5
      VERTEX janus_9_8
      VERTEX jaaws_12_11
      EDGE replay_position jaaws_12_11 -> jaaws_12_11
      EDGE replay_request jaaws_12_11 -> jaaws_12_11
      EDGE scenario gui_3_2 -> janus_9_8
      EDGE user_interaction gui_3_2 -> janus_9_8
      EDGE replay_request gui_3_2 -> jaaws_12_11
      EDGE statistics_request gui_3_2 -> post_processor_6_5
      EDGE statistics post_processor_6_5 -> gui_3_2
      EDGE replay jaaws_12_11 -> gui_3_2
      EDGE simulation_history janus_9_8 -> jaaws_12_11
      EDGE simulation_history janus_9_8 -> post_processor_6_5
    DATA STREAM
      scenario: scenario_type,
      user_interaction: user_interaction_type,
      statistics_request: statistics_request_type,
      statistics: statistics_type,
      replay: location_type,
      simulation_history: sequence[e: event_type]
    CONTROL CONSTRAINTS
      OPERATOR gui_3_2
      OPERATOR post_processor_6_5
        TRIGGERED BY ALL statistics_request
      OPERATOR janus_9_8
      OPERATOR jaaws_12_11
        TRIGGERED IF (sequence.length(simulation_history) > 0)
  END

OPERATOR enter_new_plan_75
  SPECIFICATION
```

```
    OUTPUT new_plan_entered: boolean
  END

  IMPLEMENTATION tae enter_new_plan_75
  END

OPERATOR get_y_68
  SPECIFICATION
    OUTPUT new_y: float
  END

  IMPLEMENTATION tae get_y_68
  END

OPERATOR get_x_65
  SPECIFICATION
    OUTPUT new_x: float
  END

  IMPLEMENTATION tae get_x_65
  END

OPERATOR get_re_30
  SPECIFICATION
    OUTPUT replay_request: replay_request_type
  END

  IMPLEMENTATION tae get_re_30
  END

OPERATOR get_st_27
  SPECIFICATION
    OUTPUT statistics_request: statistics_request_type
  END

  IMPLEMENTATION tae get_st_27
  END

OPERATOR edit_plan_24
  SPECIFICATION
    INPUT new_plan_entered: boolean
    INPUT new_y: float
    INPUT new_x: float
    INPUT scenario: scenario_type
    OUTPUT scenario: scenario_type
  END

  IMPLEMENTATION ada edit_plan_24
  END

OPERATOR get_user_in_21
  SPECIFICATION
    OUTPUT user_interaction: user_interaction_type
  END

  IMPLEMENTATION tae get_user_in_21
  END

OPERATOR gui_event_monitor_18
  SPECIFICATION
    MAXIMUM EXECUTION TIME 50 MS
  END
```

```
  IMPLEMENTATION ada gui_event_monitor_18
  END

OPERATOR display_st_31
  SPECIFICATION
    INPUT statistics: statistics_type
  END

  IMPLEMENTATION tae display_st_31
  END

OPERATOR display_re_37
  SPECIFICATION
    INPUT replay: location_type
  END

  IMPLEMENTATION tae display_re_37
  END

OPERATOR initial_scenario_40
  SPECIFICATION
    INPUT first_time: boolean
    OUTPUT scenario: scenario_type
    OUTPUT first_time: boolean
  END

  IMPLEMENTATION ada initial_scenario_40
  END

OPERATOR post_processor_6
  SPECIFICATION
    INPUT statistics_request: statistics_request_type
    INPUT simulation_history: sequence[e: event_type]
    OUTPUT statistics: statistics_type
  END

  IMPLEMENTATION ada post_processor_6
  END

OPERATOR janus_9
  SPECIFICATION
    INPUT scenario: scenario_type
    INPUT user_interaction: user_interaction_type
    OUTPUT simulation_history: sequence[e: event_type]
    STATES game_time: game_time_type INITIALLY game_time_type.zero
    STATES event_q: event_queue_type INITIALLY event_queue_type.empty
  END

  IMPLEMENTATION
    GRAPH
      VERTEX create_new_events_114_113
      VERTEX do_event_66_65: 100 MS
      VERTEX create_user_event_69_68
      EDGE game_time do_event_66_65 -> create_user_event_69_68
      EDGE game_time do_event_66_65 -> do_event_66_65
      EDGE event_q do_event_66_65 -> do_event_66_65
      EDGE simulation_history do_event_66_65 -> do_event_66_65
      EDGE event_q create_new_events_114_113 -> do_event_66_65
      EDGE game_time do_event_66_65 -> create_new_events_114_113
      EDGE event_q do_event_66_65 -> create_new_events_114_113
      EDGE event_q create_user_event_69_68 -> do_event_66_65
      EDGE event_q do_event_66_65 -> create_user_event_69_68
      EDGE scenario EXTERNAL -> create_new_events_114_113
```

```
      EDGE simulation_history do_event_66_65 -> EXTERNAL
      EDGE user_interaction EXTERNAL -> create_user_event_69_68
    CONTROL CONSTRAINTS
      OPERATOR create_new_events_114_113
        TRIGGERED IF not(scenario_type.is_empty(scenario))
      OPERATOR do_event_66_65
        TRIGGERED IF not(event_queue_type.is_empty(event_q))
        PERIOD 1000 MS
      OPERATOR create_user_event_69_68
        TRIGGERED IF (user_interaction = stop_simulation)
  END

OPERATOR create_new_events_114
  SPECIFICATION
    INPUT game_time: game_time_type
    INPUT event_q: event_queue_type
    INPUT scenario: scenario_type
    OUTPUT event_q: event_queue_type
  END

  IMPLEMENTATION ada create_new_events_114
  END

OPERATOR do_event_66
  SPECIFICATION
    INPUT game_time: game_time_type
    INPUT simulation_history: sequence[e: event_type]
    INPUT event_q: event_queue_type
    OUTPUT game_time: game_time_type
    OUTPUT event_q: event_queue_type
    OUTPUT simulation_history: sequence[e: event_type]
    MAXIMUM EXECUTION TIME 100 MS
  END

  IMPLEMENTATION ada do_event_66
  END

OPERATOR create_user_event_69
  SPECIFICATION
    INPUT game_time: game_time_type
    INPUT event_q: event_queue_type
    INPUT user_interaction: user_interaction_type
    OUTPUT event_q: event_queue_type
  END

  IMPLEMENTATION ada create_user_event_69
  END

OPERATOR jaaws_12
  SPECIFICATION
    INPUT replay_position: integer
    INPUT replay_request: replay_request_type
    INPUT simulation_history: sequence[e: event_type]
    OUTPUT replay_position: integer
    OUTPUT replay_request: replay_request_type
    OUTPUT replay: location_type
  END

  IMPLEMENTATION ada jaaws_12
  END
```

APPENDIX G. The Ada/C Source Code of the prototype.

1. warrior_1.adb

```
  with WARRIOR_1_STATIC_SCHEDULERS; use WARRIOR_1_STATIC_SCHEDULERS;
  with WARRIOR_1_DYNAMIC_SCHEDULERS; use WARRIOR_1_DYNAMIC_SCHEDULERS;
  with CAPS_HARDWARE_MODEL; use CAPS_HARDWARE_MODEL;
procedure WARRIOR_1 is
begin
  init_hardware_model;
  start_static_schedule;
  start_dynamic_schedule;
end WARRIOR_1;
```

2. warrior_1_drivers.ads

```
package WARRIOR_1_DRIVERS is
  procedure POST_PROCESSOR_6_5_DRIVER;
  procedure JAAWS_12_11_DRIVER;
  procedure ENTER_NEW_PLAN_75_74_DRIVER;
  procedure GET_Y_68_67_DRIVER;
  procedure GET_X_65_64_DRIVER;
  procedure GET_RE_30_29_DRIVER;
  procedure GET_ST_27_26_DRIVER;
  procedure EDIT_PLAN_24_23_DRIVER;
  procedure GET_USER_IN_21_20_DRIVER;
  procedure GUI_EVENT_MONITOR_18_17_DRIVER;
  procedure DISPLAY_ST_31_30_DRIVER;
  procedure DISPLAY_RE_37_36_DRIVER;
  procedure INITIAL_SCENARIO_40_39_DRIVER;
  procedure CREATE_NEW_EVENTS_114_113_DRIVER;
  procedure DO_EVENT_66_65_DRIVER;
  procedure CREATE_USER_EVENT_69_68_DRIVER;
end WARRIOR_1_DRIVERS;
```

3. warrior_1_drivers.adb

```
  -- with/use clauses for atomic components.
  with EVENT_TYPE_PKG; use EVENT_TYPE_PKG;
  with EVENT_QUEUE_TYPE_PKG; use EVENT_QUEUE_TYPE_PKG;
  with STATISTICS_TYPE_PKG; use STATISTICS_TYPE_PKG;
  with SCENARIO_TYPE_PKG; use SCENARIO_TYPE_PKG;
  with STATISTICS_REQUEST_TYPE_PKG; use STATISTICS_REQUEST_TYPE_PKG;
  with REPLAY_REQUEST_TYPE_PKG; use REPLAY_REQUEST_TYPE_PKG;
  with USER_INTERACTION_TYPE_PKG; use USER_INTERACTION_TYPE_PKG;
  with LOCATION_TYPE_PKG; use LOCATION_TYPE_PKG;
  with GAME_TIME_TYPE_PKG; use GAME_TIME_TYPE_PKG;
  with ENTER_NEW_PLAN_75_PKG; use ENTER_NEW_PLAN_75_PKG;
  with GET_Y_68_PKG; use GET_Y_68_PKG;
  with GET_X_65_PKG; use GET_X_65_PKG;
  with GET_RE_30_PKG; use GET_RE_30_PKG;
  with GET_ST_27_PKG; use GET_ST_27_PKG;
  with EDIT_PLAN_24_PKG; use EDIT_PLAN_24_PKG;
  with GET_USER_IN_21_PKG; use GET_USER_IN_21_PKG;
  with GUI_EVENT_MONITOR_18_PKG; use GUI_EVENT_MONITOR_18_PKG;
  with DISPLAY_ST_31_PKG; use DISPLAY_ST_31_PKG;
  with DISPLAY_RE_37_PKG; use DISPLAY_RE_37_PKG;
  with INITIAL_SCENARIO_40_PKG; use INITIAL_SCENARIO_40_PKG;
  with POST_PROCESSOR_6_PKG; use POST_PROCESSOR_6_PKG;
  with CREATE_NEW_EVENTS_114_PKG; use CREATE_NEW_EVENTS_114_PKG;
```

```
  with DO_EVENT_66_PKG; use DO_EVENT_66_PKG;
  with CREATE_USER_EVENT_69_PKG; use CREATE_USER_EVENT_69_PKG;
  with JAAWS_12_PKG; use JAAWS_12_PKG;
  -- with/use clauses for generated packages.
  with WARRIOR_1_EXCEPTIONS; use WARRIOR_1_EXCEPTIONS;
  with WARRIOR_1_STREAMS; use WARRIOR_1_STREAMS;
  with WARRIOR_1_TIMERS; use WARRIOR_1_TIMERS;
  with WARRIOR_1_INSTANTIATIONS; use WARRIOR_1_INSTANTIATIONS;
  -- with/use clauses for CAPS library packages.
  with DS_DEBUG_PKG; use DS_DEBUG_PKG;
  with PSDL_STREAMS; use PSDL_STREAMS;
  with PSDL_STRING_PKG; use PSDL_STRING_PKG;
  with PSDL_TIMERS;
package body WARRIOR_1_DRIVERS is

  procedure POST_PROCESSOR_6_5_DRIVER is
    LV_STATISTICS_REQUEST :
            STATISTICS_REQUEST_TYPE_PKG.STATISTICS_REQUEST_TYPE;
    LV_SIMULATION_HISTORY : EVENT_TYPE_SEQUENCE;
    LV_STATISTICS : STATISTICS_TYPE_PKG.STATISTICS_TYPE;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
     -- Data trigger checks.
    if not (DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5.NEW_DATA) then
      return;
    end if;

     -- Data stream reads.
    begin
      DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5.BUFFER.READ(
                                              LV_STATISTICS_REQUEST);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("STATISTICS_REQUEST_POST_PROCESSOR_6_5",
                                                  "POST_PROCESSOR_6_5");
    end;
    begin
      DS_SIMULATION_HISTORY_POST_PROCESSOR_6_5.BUFFER.READ(
                                              LV_SIMULATION_HISTORY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("SIMULATION_HISTORY_POST_PROCESSOR_6_5",
                                                  "POST_PROCESSOR_6_5");
    end;

     -- Execution trigger condition check.
    if True then
      begin
      POST_PROCESSOR_6(
        STATISTICS_REQUEST => LV_STATISTICS_REQUEST,
        SIMULATION_HISTORY => LV_SIMULATION_HISTORY,
        STATISTICS => LV_STATISTICS);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("POST_PROCESSOR_6_5");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_STATISTICS_DISPLAY_ST_31_30.BUFFER.WRITE(LV_STATISTICS);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("STATISTICS_DISPLAY_ST_31_30",
                                                    "POST_PROCESSOR_6_5");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "POST_PROCESSOR_6_5",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end POST_PROCESSOR_6_5_DRIVER;


procedure JAAWS_12_11_DRIVER is
  LV_SIMULATION_HISTORY : EVENT_TYPE_SEQUENCE;
  LV_REPLAY_POSITION : INTEGER;
  LV_REPLAY_REQUEST : REPLAY_REQUEST_TYPE_PKG.REPLAY_REQUEST_TYPE;
  LV_REPLAY : LOCATION_TYPE_PKG.LOCATION_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_REPLAY_POSITION_JAAWS_12_11.BUFFER.READ(LV_REPLAY_POSITION);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("REPLAY_POSITION_JAAWS_12_11",
                                                    "JAAWS_12_11");
  end;
  begin
    DS_REPLAY_REQUEST_JAAWS_12_11.BUFFER.READ(LV_REPLAY_REQUEST);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("REPLAY_REQUEST_JAAWS_12_11",
                                                    "JAAWS_12_11");
  end;
  begin
    DS_SIMULATION_HISTORY_JAAWS_12_11.BUFFER.READ(LV_SIMULATION_HISTORY);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SIMULATION_HISTORY_JAAWS_12_11",
                                                    "JAAWS_12_11");
  end;

-- Execution trigger condition check.
  if  (LENGTH(LV_SIMULATION_HISTORY) > 0)  then
    begin
    JAAWS_12(
      SIMULATION_HISTORY => LV_SIMULATION_HISTORY,
```

```
      REPLAY_POSITION => LV_REPLAY_POSITION,
      REPLAY_REQUEST => LV_REPLAY_REQUEST,
      REPLAY => LV_REPLAY);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("JAAWS_12_11");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_POSITION_JAAWS_12_11.BUFFER.WRITE(LV_REPLAY_POSITION);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_POSITION_JAAWS_12_11",
                                                    "JAAWS_12_11");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_REQUEST_JAAWS_12_11.BUFFER.WRITE(LV_REPLAY_REQUEST);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_REQUEST_JAAWS_12_11",
                                                    "JAAWS_12_11");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_DISPLAY_RE_37_36.BUFFER.WRITE(LV_REPLAY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_DISPLAY_RE_37_36", "JAAWS_12_11");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "JAAWS_12_11",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end JAAWS_12_11_DRIVER;


procedure ENTER_NEW_PLAN_75_74_DRIVER is
  LV_NEW_PLAN_ENTERED : BOOLEAN;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
```

```
-- Execution trigger condition check.
  if ENTER_NEW_PLAN_75_PKG.has_new_input then
    begin
    ENTER_NEW_PLAN_75(
      NEW_PLAN_ENTERED => LV_NEW_PLAN_ENTERED);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("ENTER_NEW_PLAN_75_74");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23.BUFFER.WRITE(
                                                  LV_NEW_PLAN_ENTERED);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("NEW_PLAN_ENTERED_EDIT_PLAN_24_23",
                                                  "ENTER_NEW_PLAN_75_74");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "ENTER_NEW_PLAN_75_74",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end ENTER_NEW_PLAN_75_74_DRIVER;


procedure GET_Y_68_67_DRIVER is
  LV_NEW_Y : FLOAT;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
   -- Data trigger checks.

   -- Data stream reads.

   -- Execution trigger condition check.
  if GET_Y_68_PKG.has_new_input then
    begin
    GET_Y_68(
      NEW_Y => LV_NEW_Y);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_Y_68_67");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_NEW_Y_EDIT_PLAN_24_23.BUFFER.WRITE(LV_NEW_Y);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("NEW_Y_EDIT_PLAN_24_23", "GET_Y_68_67");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_Y_68_67",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_Y_68_67_DRIVER;


procedure GET_X_65_64_DRIVER is
  LV_NEW_X : FLOAT;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if GET_X_65_PKG.has_new_input then
    begin
    GET_X_65(
      NEW_X => LV_NEW_X);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_X_65_64");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_NEW_X_EDIT_PLAN_24_23.BUFFER.WRITE(LV_NEW_X);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("NEW_X_EDIT_PLAN_24_23", "GET_X_65_64");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
```

```
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_X_65_64",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_X_65_64_DRIVER;


procedure GET_RE_30_29_DRIVER is
  LV_REPLAY_REQUEST : REPLAY_REQUEST_TYPE_PKG.REPLAY_REQUEST_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if GET_RE_30_PKG.has_new_input then
    begin
    GET_RE_30(
      REPLAY_REQUEST => LV_REPLAY_REQUEST);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_RE_30_29");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_REQUEST_JAAWS_12_11.BUFFER.WRITE(LV_REPLAY_REQUEST);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_REQUEST_JAAWS_12_11",
                                                    "GET_RE_30_29");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_RE_30_29",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_RE_30_29_DRIVER;


procedure GET_ST_27_26_DRIVER is
  LV_STATISTICS_REQUEST :
                        STATISTICS_REQUEST_TYPE_PKG.STATISTICS_REQUEST_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.
```

```
-- Data stream reads.

-- Execution trigger condition check.
   if GET_ST_27_PKG.has_new_input then
     begin
     GET_ST_27(
       STATISTICS_REQUEST => LV_STATISTICS_REQUEST);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("GET_ST_27_26");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
     begin
       DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5.BUFFER.WRITE(
                                                  LV_STATISTICS_REQUEST);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("STATISTICS_REQUEST_POST_PROCESSOR_6_5",
                                                       "GET_ST_27_26");
     end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "GET_ST_27_26",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
end GET_ST_27_26_DRIVER;


procedure EDIT_PLAN_24_23_DRIVER is
   LV_NEW_PLAN_ENTERED : BOOLEAN;
   LV_NEW_Y : FLOAT;
   LV_NEW_X : FLOAT;
   LV_SCENARIO : SCENARIO_TYPE_PKG.SCENARIO_TYPE;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.
   if not (DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23.NEW_DATA) then
     return;
   end if;

-- Data stream reads.
   begin
     DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23.BUFFER.READ(LV_NEW_PLAN_ENTERED);
   exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("NEW_PLAN_ENTERED_EDIT_PLAN_24_23",
```

```
                                                        "EDIT_PLAN_24_23");
  end;
  begin
    DS_NEW_Y_EDIT_PLAN_24_23.BUFFER.READ(LV_NEW_Y);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("NEW_Y_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
  end;
  begin
    DS_NEW_X_EDIT_PLAN_24_23.BUFFER.READ(LV_NEW_X);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("NEW_X_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
  end;
  begin
    DS_SCENARIO_EDIT_PLAN_24_23.BUFFER.READ(LV_SCENARIO);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SCENARIO_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
  end;

-- Execution trigger condition check.
  if True then
    begin
    EDIT_PLAN_24(
      NEW_PLAN_ENTERED => LV_NEW_PLAN_ENTERED,
      NEW_Y => LV_NEW_Y,
      NEW_X => LV_NEW_X,
      SCENARIO => LV_SCENARIO);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("EDIT_PLAN_24_23");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_SCENARIO_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_CREATE_NEW_EVENTS_114_113",

                                                              "EDIT_PLAN_24_23");
    end;
    begin
      DS_SCENARIO_EDIT_PLAN_24_23.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
    end;
```

```
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "EDIT_PLAN_24_23",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end EDIT_PLAN_24_23_DRIVER;


procedure GET_USER_IN_21_20_DRIVER is
  LV_USER_INTERACTION : USER_INTERACTION_TYPE_PKG.USER_INTERACTION_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if GET_USER_IN_21_PKG.has_new_input then
    begin
    GET_USER_IN_21(
      USER_INTERACTION => LV_USER_INTERACTION);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_USER_IN_21_20");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_USER_INTERACTION_CREATE_USER_EVENT_69_68.BUFFER.WRITE(
                                                LV_USER_INTERACTION);
    exception
        when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW(
          "USER_INTERACTION_CREATE_USER_EVENT_69_68", "GET_USER_IN_21_20");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_USER_IN_21_20",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_USER_IN_21_20_DRIVER;


procedure GUI_EVENT_MONITOR_18_17_DRIVER is

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
```

```
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if True then
    begin
    GUI_EVENT_MONITOR_18;
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GUI_EVENT_MONITOR_18_17");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GUI_EVENT_MONITOR_18_17",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GUI_EVENT_MONITOR_18_17_DRIVER;


procedure DISPLAY_ST_31_30_DRIVER is
  LV_STATISTICS : STATISTICS_TYPE_PKG.STATISTICS_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_STATISTICS_DISPLAY_ST_31_30.BUFFER.READ(LV_STATISTICS);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("STATISTICS_DISPLAY_ST_31_30",
                                                    "DISPLAY_ST_31_30");
  end;

-- Execution trigger condition check.
  if True then
    begin
    DISPLAY_ST_31(
      STATISTICS => LV_STATISTICS);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("DISPLAY_ST_31_30");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
```

```
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DISPLAY_ST_31_30",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end DISPLAY_ST_31_30_DRIVER;


procedure DISPLAY_RE_37_36_DRIVER is
  LV_REPLAY : LOCATION_TYPE_PKG.LOCATION_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_REPLAY_DISPLAY_RE_37_36.BUFFER.READ(LV_REPLAY);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("REPLAY_DISPLAY_RE_37_36",
                                                    "DISPLAY_RE_37_36");
  end;

-- Execution trigger condition check.
  if True then
    begin
    DISPLAY_RE_37(
      REPLAY => LV_REPLAY);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("DISPLAY_RE_37_36");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DISPLAY_RE_37_36",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end DISPLAY_RE_37_36_DRIVER;
```

```
procedure INITIAL_SCENARIO_40_39_DRIVER is
  LV_SCENARIO : SCENARIO_TYPE_PKG.SCENARIO_TYPE;
  LV_FIRST_TIME : BOOLEAN;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_FIRST_TIME_INITIAL_SCENARIO_40_39.BUFFER.READ(LV_FIRST_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("FIRST_TIME_INITIAL_SCENARIO_40_39",
                                                    "INITIAL_SCENARIO_40_39");
  end;

-- Execution trigger condition check.
  if  (LV_FIRST_TIME = true)  then
    begin
    INITIAL_SCENARIO_40(
      SCENARIO => LV_SCENARIO,
      FIRST_TIME => LV_FIRST_TIME);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("INITIAL_SCENARIO_40_39");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 - Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_SCENARIO_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_CREATE_NEW_EVENTS_114_113",
                                                    "INITIAL_SCENARIO_40_39");
    end;
    begin
      DS_SCENARIO_EDIT_PLAN_24_23.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_EDIT_PLAN_24_23",
                                                    "INITIAL_SCENARIO_40_39");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_FIRST_TIME_INITIAL_SCENARIO_40_39.BUFFER.WRITE(LV_FIRST_TIME);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("FIRST_TIME_INITIAL_SCENARIO_40_39",
                                                    "INITIAL_SCENARIO_40_39");
    end;
  end if;
```

```
-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "INITIAL_SCENARIO_40_39",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end INITIAL_SCENARIO_40_39_DRIVER;


procedure CREATE_NEW_EVENTS_114_113_DRIVER is
  LV_GAME_TIME : GAME_TIME_TYPE_PKG.GAME_TIME_TYPE;
  LV_SCENARIO : SCENARIO_TYPE_PKG.SCENARIO_TYPE;
  LV_EVENT_Q : EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_GAME_TIME_CREATE_NEW_EVENTS_114_113.BUFFER.READ(LV_GAME_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("GAME_TIME_CREATE_NEW_EVENTS_114_113",
                                "CREATE_NEW_EVENTS_114_113");
  end;
  begin
    DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.READ(LV_EVENT_Q);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                "CREATE_NEW_EVENTS_114_113");
  end;
  begin
    DS_SCENARIO_CREATE_NEW_EVENTS_114_113.BUFFER.READ(LV_SCENARIO);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SCENARIO_CREATE_NEW_EVENTS_114_113",
                                "CREATE_NEW_EVENTS_114_113");
  end;

-- Execution trigger condition check.
  if not (SCENARIO_TYPE_PKG.IS_EMPTY(LV_SCENARIO))  then
    begin
    CREATE_NEW_EVENTS_114(
      GAME_TIME => LV_GAME_TIME,
      SCENARIO => LV_SCENARIO,
      EVENT_Q => LV_EVENT_Q);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("CREATE_NEW_EVENTS_114_113");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.
```

```
-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                                 "CREATE_NEW_EVENTS_114_113");
    end;
    begin
      DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                                 "CREATE_NEW_EVENTS_114_113");
    end;
    begin
      DS_EVENT_Q_DO_EVENT_66_65.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_DO_EVENT_66_65",
                                                 "CREATE_NEW_EVENTS_114_113");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "CREATE_NEW_EVENTS_114_113",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end CREATE_NEW_EVENTS_114_113_DRIVER;


procedure DO_EVENT_66_65_DRIVER is
  LV_GAME_TIME : GAME_TIME_TYPE_PKG.GAME_TIME_TYPE;
  LV_EVENT_Q : EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE;
  LV_SIMULATION_HISTORY : EVENT_TYPE_SEQUENCE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_GAME_TIME_DO_EVENT_66_65.BUFFER.READ(LV_GAME_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("GAME_TIME_DO_EVENT_66_65",
                                                 "DO_EVENT_66_65");
  end;
  begin
    DS_SIMULATION_HISTORY_DO_EVENT_66_65.BUFFER.READ(
                                                 LV_SIMULATION_HISTORY);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SIMULATION_HISTORY_DO_EVENT_66_65",
                                                 "DO_EVENT_66_65");
  end;
  begin
    DS_EVENT_Q_DO_EVENT_66_65.BUFFER.READ(LV_EVENT_Q);
  exception
```

```
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EVENT_Q_DO_EVENT_66_65",
                                                    "DO_EVENT_66_65");
    end;

  -- Execution trigger condition check.
    if not (EVENT_QUEUE_TYPE_PKG.IS_EMPTY(LV_EVENT_Q))  then
      begin
      DO_EVENT_66(
        GAME_TIME => LV_GAME_TIME,
        EVENT_Q => LV_EVENT_Q,
        SIMULATION_HISTORY => LV_SIMULATION_HISTORY);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DO_EVENT_66_65");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

  -- Exception Constraint translations.

  -- Other constraint option translations.

  -- Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_GAME_TIME_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_GAME_TIME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GAME_TIME_CREATE_NEW_EVENTS_114_113",
                                                    "DO_EVENT_66_65");
      end;
      begin
        DS_GAME_TIME_DO_EVENT_66_65.BUFFER.WRITE(LV_GAME_TIME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GAME_TIME_DO_EVENT_66_65",
                                                    "DO_EVENT_66_65");
      end;
      begin
        DS_GAME_TIME_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_GAME_TIME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GAME_TIME_CREATE_USER_EVENT_69_68",
                                                    "DO_EVENT_66_65");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                                    "DO_EVENT_66_65");
      end;
      begin
        DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                                    "DO_EVENT_66_65");
```

```
    end;
    begin
      DS_EVENT_Q_DO_EVENT_66_65.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_DO_EVENT_66_65",
                                                    "DO_EVENT_66_65");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_SIMULATION_HISTORY_POST_PROCESSOR_6_5.BUFFER.WRITE(
                                                LV_SIMULATION_HISTORY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SIMULATION_HISTORY_POST_PROCESSOR_6_5",
                                                    "DO_EVENT_66_65");
    end;
    begin
      DS_SIMULATION_HISTORY_JAAWS_12_11.BUFFER.WRITE(
                                                LV_SIMULATION_HISTORY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SIMULATION_HISTORY_JAAWS_12_11",
                                                    "DO_EVENT_66_65");
    end;
    begin
      DS_SIMULATION_HISTORY_DO_EVENT_66_65.BUFFER.WRITE(
                                                LV_SIMULATION_HISTORY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SIMULATION_HISTORY_DO_EVENT_66_65",
                                                    "DO_EVENT_66_65");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DO_EVENT_66_65",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end DO_EVENT_66_65_DRIVER;


procedure CREATE_USER_EVENT_69_68_DRIVER is
  LV_GAME_TIME : GAME_TIME_TYPE_PKG.GAME_TIME_TYPE;
  LV_USER_INTERACTION : USER_INTERACTION_TYPE_PKG.USER_INTERACTION_TYPE;
  LV_EVENT_Q : EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_GAME_TIME_CREATE_USER_EVENT_69_68.BUFFER.READ(LV_GAME_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("GAME_TIME_CREATE_USER_EVENT_69_68",
                                          "CREATE_USER_EVENT_69_68");
  end;
```

```
  begin
    DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.READ(LV_EVENT_Q);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                "CREATE_USER_EVENT_69_68");
  end;
  begin
    DS_USER_INTERACTION_CREATE_USER_EVENT_69_68.BUFFER.READ(
                                LV_USER_INTERACTION);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("USER_INTERACTION_CREATE_USER_EVENT_69_68",
                                "CREATE_USER_EVENT_69_68");
  end;

-- Execution trigger condition check.
  if  (LV_USER_INTERACTION = STOP_SIMULATION)  then
    begin
    CREATE_USER_EVENT_69(
      GAME_TIME => LV_GAME_TIME,
      USER_INTERACTION => LV_USER_INTERACTION,
      EVENT_Q => LV_EVENT_Q);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("CREATE_USER_EVENT_69_68");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 -- Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
     begin
       DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_EVENT_Q);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                  "CREATE_USER_EVENT_69_68");
     end;
     begin
       DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_EVENT_Q);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                  "CREATE_USER_EVENT_69_68");
     end;
     begin
       DS_EVENT_Q_DO_EVENT_66_65.BUFFER.WRITE(LV_EVENT_Q);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_DO_EVENT_66_65",
                                  "CREATE_USER_EVENT_69_68");
     end;
   end if;

 -- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
```

```
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "CREATE_USER_EVENT_69_68",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end CREATE_USER_EVENT_69_68_DRIVER;
end WARRIOR_1_DRIVERS;
```

4. warrior_1_exceptions.ads

```
package WARRIOR_1_EXCEPTIONS is
  -- PSDL exception type declaration
  type PSDL_EXCEPTION is (UNDECLARED_ADA_EXCEPTION);
end WARRIOR_1_EXCEPTIONS;
```

5. warrior_1_instantiations.ads

```
  with EVENT_TYPE_PKG; use EVENT_TYPE_PKG;
  -- Generic type packages
  with
    SEQUENCE_PKG;
package WARRIOR_1_INSTANTIATIONS is
  -- Ada Generic package instantiations

  package EVENT_TYPE_SEQUENCE_PKG is new
    SEQUENCE_PKG(EVENT_TYPE_PTR);

  type EVENT_TYPE_SEQUENCE is new
    EVENT_TYPE_SEQUENCE_PKG.SEQUENCE;

end WARRIOR_1_INSTANTIATIONS;
```

6. warrior_1_streams.ads

```
  -- with/use clauses for atomic type packages
  with EVENT_TYPE_PKG; use EVENT_TYPE_PKG;
  with EVENT_QUEUE_TYPE_PKG; use EVENT_QUEUE_TYPE_PKG;
  with STATISTICS_TYPE_PKG; use STATISTICS_TYPE_PKG;
  with SCENARIO_TYPE_PKG; use SCENARIO_TYPE_PKG;
  with STATISTICS_REQUEST_TYPE_PKG; use STATISTICS_REQUEST_TYPE_PKG;
  with REPLAY_REQUEST_TYPE_PKG; use REPLAY_REQUEST_TYPE_PKG;
  with USER_INTERACTION_TYPE_PKG; use USER_INTERACTION_TYPE_PKG;
  with LOCATION_TYPE_PKG; use LOCATION_TYPE_PKG;
  with GAME_TIME_TYPE_PKG; use GAME_TIME_TYPE_PKG;
  -- with/use clauses for generated packages.
  with WARRIOR_1_EXCEPTIONS; use WARRIOR_1_EXCEPTIONS;
  with WARRIOR_1_INSTANTIATIONS; use WARRIOR_1_INSTANTIATIONS;
  -- with/use clauses for CAPS library packages.
  with PSDL_STREAMS; use PSDL_STREAMS;
  with PSDL_STRING_PKG; use PSDL_STRING_PKG;
package WARRIOR_1_STREAMS is
  -- Local stream instantiations

  package DS_USER_INTERACTION_CREATE_USER_EVENT_69_68 is new
    PSDL_STREAMS.SAMPLED_BUFFER(USER_INTERACTION_TYPE);

  package DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5 is new
    PSDL_STREAMS.FIFO_BUFFER(STATISTICS_REQUEST_TYPE);
```

```
package DS_STATISTICS_DISPLAY_ST_31_30 is new
  PSDL_STREAMS.SAMPLED_BUFFER(STATISTICS_TYPE);

package DS_REPLAY_DISPLAY_RE_37_36 is new
  PSDL_STREAMS.SAMPLED_BUFFER(LOCATION_TYPE);

package DS_SIMULATION_HISTORY_POST_PROCESSOR_6_5 is new
  PSDL_STREAMS.SAMPLED_BUFFER(EVENT_TYPE_SEQUENCE);

package DS_SIMULATION_HISTORY_JAAWS_12_11 is new
  PSDL_STREAMS.SAMPLED_BUFFER(EVENT_TYPE_SEQUENCE);

package DS_SIMULATION_HISTORY_DO_EVENT_66_65 is new
  PSDL_STREAMS.SAMPLED_BUFFER(EVENT_TYPE_SEQUENCE);

package DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23 is new
  PSDL_STREAMS.FIFO_BUFFER(BOOLEAN);

-- State stream instantiations

package DS_REPLAY_POSITION_JAAWS_12_11 is new
        PSDL_STREAMS.STATE_VARIABLE(INTEGER, 1);

package DS_REPLAY_REQUEST_JAAWS_12_11 is new
        PSDL_STREAMS.STATE_VARIABLE(
              REPLAY_REQUEST_TYPE_PKG.REPLAY_REQUEST_TYPE,
                                    REPLAY_REQUEST_TYPE_PKG.OFF);

package DS_SCENARIO_CREATE_NEW_EVENTS_114_113 is new
        PSDL_STREAMS.STATE_VARIABLE(
              SCENARIO_TYPE_PKG.SCENARIO_TYPE,
                                 SCENARIO_TYPE_PKG.EMPTY_SCENARIO);

package DS_SCENARIO_EDIT_PLAN_24_23 is new
        PSDL_STREAMS.STATE_VARIABLE(
              SCENARIO_TYPE_PKG.SCENARIO_TYPE,
                              SCENARIO_TYPE_PKG.EMPTY_SCENARIO);

package DS_NEW_Y_EDIT_PLAN_24_23 is new
        PSDL_STREAMS.STATE_VARIABLE(FLOAT, 0.0);

package DS_NEW_X_EDIT_PLAN_24_23 is new
        PSDL_STREAMS.STATE_VARIABLE(FLOAT, 0.0);

package DS_FIRST_TIME_INITIAL_SCENARIO_40_39 is new
        PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, true);

package DS_GAME_TIME_CREATE_NEW_EVENTS_114_113 is new
        PSDL_STREAMS.STATE_VARIABLE(
                    GAME_TIME_TYPE_PKG.GAME_TIME_TYPE,
                                       GAME_TIME_TYPE_PKG.ZERO);

package DS_GAME_TIME_DO_EVENT_66_65 is new
        PSDL_STREAMS.STATE_VARIABLE(
                    GAME_TIME_TYPE_PKG.GAME_TIME_TYPE,
                                       GAME_TIME_TYPE_PKG.ZERO);
```

```
    package DS_GAME_TIME_CREATE_USER_EVENT_69_68 is new
            PSDL_STREAMS.STATE_VARIABLE(
                        GAME_TIME_TYPE_PKG.GAME_TIME_TYPE,
                                        GAME_TIME_TYPE_PKG.ZERO);

    package DS_EVENT_Q_CREATE_USER_EVENT_69_68 is new
            PSDL_STREAMS.STATE_VARIABLE(
                        EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE,
                                        EVENT_QUEUE_TYPE_PKG.EMPTY);

    package DS_EVENT_Q_CREATE_NEW_EVENTS_114_113 is new
            PSDL_STREAMS.STATE_VARIABLE(
                        EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE,
                                        EVENT_QUEUE_TYPE_PKG.EMPTY);

    package DS_EVENT_Q_DO_EVENT_66_65 is new
            PSDL_STREAMS.STATE_VARIABLE(
                        EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE,
                                        EVENT_QUEUE_TYPE_PKG.EMPTY);

  end WARRIOR_1_STREAMS;
```

7. warrior_1_timers.ads

```
  with PSDL_TIMERS;
package WARRIOR_1_TIMERS is
  -- Timer instantiations
end WARRIOR_1_TIMERS;
```

8. warrior_1_dynamic_schedulers.ads

```
package warrior_1_DYNAMIC_SCHEDULERS is
  procedure START_DYNAMIC_SCHEDULE;
  procedure STOP_DYNAMIC_SCHEDULE;
end warrior_1_DYNAMIC_SCHEDULERS;
```

9. warrior_1_dynamic_schedulers.adb

```
with warrior_1_DRIVERS;  use warrior_1_DRIVERS;
with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
package body warrior_1_DYNAMIC_SCHEDULERS is

  task type DYNAMIC_SCHEDULE_TYPE is
    pragma priority (DYNAMIC_SCHEDULE_PRIORITY);
    entry START;
  end DYNAMIC_SCHEDULE_TYPE;
  for DYNAMIC_SCHEDULE_TYPE'STORAGE_SIZE use 100_000;
  DYNAMIC_SCHEDULE : DYNAMIC_SCHEDULE_TYPE;

  done : boolean := false;
  procedure STOP_DYNAMIC_SCHEDULE is
  begin
    done := true;
  end STOP_DYNAMIC_SCHEDULE;

  task body DYNAMIC_SCHEDULE_TYPE is
  begin
```

```
    accept START;
    loop
      enter_new_plan_75_74_DRIVER;
      exit when done;

      get_y_68_67_DRIVER;
      exit when done;

      get_x_65_64_DRIVER;
      exit when done;

      get_re_30_29_DRIVER;
      exit when done;

      get_st_27_26_DRIVER;
      exit when done;

      get_user_in_21_20_DRIVER;
      exit when done;

      initial_scenario_40_39_DRIVER;
      exit when done;

      create_new_events_114_113_DRIVER;
      exit when done;

      edit_plan_24_23_DRIVER;
      exit when done;

      create_user_event_69_68_DRIVER;
      exit when done;

      jaaws_12_11_DRIVER;
      exit when done;

      post_processor_6_5_DRIVER;
      exit when done;

      display_re_37_36_DRIVER;
      exit when done;

      display_st_31_30_DRIVER;
      exit when done;

    end loop;
  end DYNAMIC_SCHEDULE_TYPE;

  procedure START_DYNAMIC_SCHEDULE is
  begin
    DYNAMIC_SCHEDULE.START;
  end START_DYNAMIC_SCHEDULE;

end warrior_1_DYNAMIC_SCHEDULERS;
```

10. warrior_1_static_schedulers.ads

```
package warrior_1_STATIC_SCHEDULERS is
  procedure START_STATIC_SCHEDULE;
  procedure STOP_STATIC_SCHEDULE;
end warrior_1_STATIC_SCHEDULERS;
```

11. warrior_1_static_schedulers.adb

```
with warrior_1_DRIVERS;  use warrior_1_DRIVERS;
with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
with PSDL_TIMERS;  use PSDL_TIMERS;
with TEXT_IO;  use TEXT_IO;
package body warrior_1_STATIC_SCHEDULERS is

  task type STATIC_SCHEDULE_TYPE is
    pragma priority (STATIC_SCHEDULE_PRIORITY);
    entry START;
  end STATIC_SCHEDULE_TYPE;
  for STATIC_SCHEDULE_TYPE'STORAGE_SIZE use 200_000;
  STATIC_SCHEDULE : STATIC_SCHEDULE_TYPE;

  done : boolean := false;
  procedure STOP_STATIC_SCHEDULE is
  begin
    done := true;
  end STOP_STATIC_SCHEDULE;

  task body STATIC_SCHEDULE_TYPE is
    PERIOD : duration;
    gui_event_monitor_18_17_START_TIME1 : duration;
    gui_event_monitor_18_17_STOP_TIME1 : duration;
    do_event_66_65_START_TIME2 : duration;
    do_event_66_65_STOP_TIME2 : duration;
    gui_event_monitor_18_17_START_TIME3 : duration;
    gui_event_monitor_18_17_STOP_TIME3 : duration;
    gui_event_monitor_18_17_START_TIME4 : duration;
    gui_event_monitor_18_17_STOP_TIME4 : duration;
    gui_event_monitor_18_17_START_TIME5 : duration;
    gui_event_monitor_18_17_STOP_TIME5 : duration;
    do_event_66_65_START_TIME6 : duration;
    do_event_66_65_STOP_TIME6 : duration;
    gui_event_monitor_18_17_START_TIME7 : duration;
    gui_event_monitor_18_17_STOP_TIME7 : duration;
    gui_event_monitor_18_17_START_TIME8 : duration;
    gui_event_monitor_18_17_STOP_TIME8 : duration;
    gui_event_monitor_18_17_START_TIME9 : duration;
    gui_event_monitor_18_17_STOP_TIME9 : duration;
    do_event_66_65_START_TIME10 : duration;
    do_event_66_65_STOP_TIME10 : duration;
    gui_event_monitor_18_17_START_TIME11 : duration;
    gui_event_monitor_18_17_STOP_TIME11 : duration;
    gui_event_monitor_18_17_START_TIME12 : duration;
    gui_event_monitor_18_17_STOP_TIME12 : duration;
    gui_event_monitor_18_17_START_TIME13 : duration;
    gui_event_monitor_18_17_STOP_TIME13 : duration;
    schedule_timer : TIMER := NEW_TIMER;
  begin
    accept START;
    PERIOD := TARGET_TO_HOST(duration( 3.00000E+00));
    gui_event_monitor_18_17_START_TIME1 := TARGET_TO_HOST(
                                           duration( 0.00000E+00));
    gui_event_monitor_18_17_STOP_TIME1 := TARGET_TO_HOST(
                                           duration( 5.00000E-02));
    do_event_66_65_START_TIME2 := TARGET_TO_HOST(duration( 5.00000E-02));
    do_event_66_65_STOP_TIME2 := TARGET_TO_HOST(duration( 1.50000E-01));
    gui_event_monitor_18_17_START_TIME3 := TARGET_TO_HOST(
                                           duration( 3.00000E-01));
```

```
gui_event_monitor_18_17_STOP_TIME3 := TARGET_TO_HOST(
                                        duration( 3.50000E-01));
gui_event_monitor_18_17_START_TIME4 := TARGET_TO_HOST(
                                        duration( 6.00000E-01));
gui_event_monitor_18_17_STOP_TIME4 := TARGET_TO_HOST(
                                        duration( 6.50000E-01));
gui_event_monitor_18_17_START_TIME5 := TARGET_TO_HOST(
                                        duration( 9.00000E-01));
gui_event_monitor_18_17_STOP_TIME5 := TARGET_TO_HOST(
                                        duration( 9.50000E-01));
do_event_66_65_START_TIME6 := TARGET_TO_HOST(duration( 1.05000E+00));
do_event_66_65_STOP_TIME6 := TARGET_TO_HOST(duration( 1.15000E+00));
gui_event_monitor_18_17_START_TIME7 := TARGET_TO_HOST(
                                        duration( 1.20000E+00));
gui_event_monitor_18_17_STOP_TIME7 := TARGET_TO_HOST(
                                        duration( 1.25000E+00));
gui_event_monitor_18_17_START_TIME8 := TARGET_TO_HOST(
                                        duration( 1.50000E+00));
gui_event_monitor_18_17_STOP_TIME8 := TARGET_TO_HOST(
                                        duration( 1.55000E+00));
gui_event_monitor_18_17_START_TIME9 := TARGET_TO_HOST(
                                        duration( 1.80000E+00));
gui_event_monitor_18_17_STOP_TIME9 := TARGET_TO_HOST(
                                        duration( 1.85000E+00));
do_event_66_65_START_TIME10 := TARGET_TO_HOST(duration( 2.05000E+00));
do_event_66_65_STOP_TIME10 := TARGET_TO_HOST(duration( 2.15000E+00));
gui_event_monitor_18_17_START_TIME11 := TARGET_TO_HOST(
                                        duration( 2.15000E+00));
gui_event_monitor_18_17_STOP_TIME11 := TARGET_TO_HOST(
                                        duration( 2.20000E+00));
gui_event_monitor_18_17_START_TIME12 := TARGET_TO_HOST(
                                        duration( 2.40000E+00));
gui_event_monitor_18_17_STOP_TIME12 := TARGET_TO_HOST(
                                        duration( 2.45000E+00));
gui_event_monitor_18_17_START_TIME13 := TARGET_TO_HOST(
                                        duration( 2.70000E+00));
gui_event_monitor_18_17_STOP_TIME13 := TARGET_TO_HOST(
                                        duration( 2.75000E+00));

START(schedule_timer);
loop
  delay(gui_event_monitor_18_17_START_TIME1 -
                              HOST_DURATION(schedule_timer));
  gui_event_monitor_18_17_DRIVER;
  if HOST_DURATION(schedule_timer) >
                             gui_event_monitor_18_17_STOP_TIME1 then
    PUT_LINE("timing error from operator gui_event_monitor_18_17");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                  gui_event_monitor_18_17_STOP_TIME1);
  end if;
  exit when done;

  delay(do_event_66_65_START_TIME2 - HOST_DURATION(schedule_timer));
  do_event_66_65_DRIVER;
  if HOST_DURATION(schedule_timer) > do_event_66_65_STOP_TIME2 then
    PUT_LINE("timing error from operator do_event_66_65");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                        do_event_66_65_STOP_TIME2);
  end if;
  exit when done;

  delay(gui_event_monitor_18_17_START_TIME3 -
                                  HOST_DURATION(schedule_timer));
  gui_event_monitor_18_17_DRIVER;
```

```
if HOST_DURATION(schedule_timer) >
                                    gui_event_monitor_18_17_STOP_TIME3 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME3);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME4 -
                                          HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                                    gui_event_monitor_18_17_STOP_TIME4 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME4);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME5 -
                                    HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                                    gui_event_monitor_18_17_STOP_TIME5 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME5);
end if;
exit when done;

delay(do_event_66_65_START_TIME6 - HOST_DURATION(schedule_timer));
do_event_66_65_DRIVER;
if HOST_DURATION(schedule_timer) > do_event_66_65_STOP_TIME6 then
  PUT_LINE("timing error from operator do_event_66_65");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                          do_event_66_65_STOP_TIME6);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME7 -
                                          HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                                    gui_event_monitor_18_17_STOP_TIME7 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME7);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME8 -
                                          HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                                    gui_event_monitor_18_17_STOP_TIME8 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME8);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME9 -
                                          HOST_DURATION(schedule_timer));
```

```
      gui_event_monitor_18_17_DRIVER;
      if HOST_DURATION(schedule_timer) >
                                    gui_event_monitor_18_17_STOP_TIME9 then
        PUT_LINE("timing error from operator gui_event_monitor_18_17");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME9);
      end if;
      exit when done;

      delay(do_event_66_65_START_TIME10 - HOST_DURATION(schedule_timer));
      do_event_66_65_DRIVER;
      if HOST_DURATION(schedule_timer) > do_event_66_65_STOP_TIME10 then
        PUT_LINE("timing error from operator do_event_66_65");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                           do_event_66_65_STOP_TIME10);
      end if;
      exit when done;

      delay(gui_event_monitor_18_17_START_TIME11 -
                                           HOST_DURATION(schedule_timer));
      gui_event_monitor_18_17_DRIVER;
      if HOST_DURATION(schedule_timer) >
                                gui_event_monitor_18_17_STOP_TIME11 then
        PUT_LINE("timing error from operator gui_event_monitor_18_17");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME11);
      end if;
      exit when done;

      delay(gui_event_monitor_18_17_START_TIME12 -
                                           HOST_DURATION(schedule_timer));
      gui_event_monitor_18_17_DRIVER;
      if HOST_DURATION(schedule_timer) >
                                gui_event_monitor_18_17_STOP_TIME12 then
        PUT_LINE("timing error from operator gui_event_monitor_18_17");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME12);
      end if;
      exit when done;

      delay(gui_event_monitor_18_17_START_TIME13 -
                                           HOST_DURATION(schedule_timer));
      gui_event_monitor_18_17_DRIVER;
      if HOST_DURATION(schedule_timer) >
                                gui_event_monitor_18_17_STOP_TIME13 then
        PUT_LINE("timing error from operator gui_event_monitor_18_17");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME13);
      end if;
      exit when done;

      delay(PERIOD - HOST_DURATION(schedule_timer));
      RESET(schedule_timer);
    end loop;
  end STATIC_SCHEDULE_TYPE;

  procedure START_STATIC_SCHEDULE is
  begin
    STATIC_SCHEDULE.START;
  end START_STATIC_SCHEDULE;

end warrior_1_STATIC_SCHEDULERS;
```

12. warrior_event_monitor_task_pkg.ads

```
  -- The wrapper task to provide mutual exclusion
  -- for calls from the prototype to TAE.

  with PRIORITY_DEFINITIONS; use PRIORITY_DEFINITIONS;
  with statistics_type_pkg; use statistics_type_pkg;
  with location_type_pkg; use location_type_pkg;
package warrior_event_monitor_task_pkg is
  task warrior_event_monitor_task is
    pragma priority (BUFFER_PRIORITY);
    entry event_monitor_entry;
    entry display_st_31_entry(statistics: statistics_type);
    entry display_re_37_entry(replay: location_type);
    entry end_task;
  end warrior_event_monitor_task;
end warrior_event_monitor_task_pkg;
```

13. warrior_event_monitor_task_pkg.adb

```
  -- The wrapper task to provide mutual exclusion
  -- for calls from the prototype to TAE.

  with generated_tae_event_monitor_pkg;
  with panel_gui_3;
with text_io;
package body warrior_event_monitor_task_pkg is
  task body warrior_event_monitor_task is
    done : boolean := false;
  begin
    panel_gui_3.initialize_gui;
    loop
      select
        accept event_monitor_entry do
         if not done then
            generated_tae_event_monitor_pkg.generated_tae_event_monitor;
         end if;
        end event_monitor_entry;
      or
        accept display_st_31_entry(statistics: statistics_type) do
         if not done then
            panel_gui_3.display_st_31(statistics);
         end if;
        end display_st_31_entry;
      or
        accept display_re_37_entry(replay: location_type) do
         if not done then
            panel_gui_3.display_re_37(replay);
         end if;
        end display_re_37_entry;
      or
        accept end_task do
         raise Program_Error;
        end end_task;
      end select;
    end loop;

  end warrior_event_monitor_task;
end warrior_event_monitor_task_pkg;
```

14. create_new_events_114_pkg.ads

```
  with game_time_type_pkg;    use game_time_type_pkg;
  with event_queue_type_pkg; use event_queue_type_pkg;
  with scenario_type_pkg;     use scenario_type_pkg;

package create_new_events_114_pkg is

  procedure create_new_events_114( game_time: in     game_time_type;
                                    event_q: in out event_queue_type;
                                   scenario: in     scenario_type );
end create_new_events_114_pkg;
```

15. create_new_events_114_pkg.adb

```
  with simulation_object_Pkg;       USE Simulation_Object_Pkg;
  with event_type_pkg;              USE event_type_pkg;
  with event_type_pkg.move_pkg;     use event_type_pkg.move_pkg;
  with text_io;

package body create_new_events_114_pkg is
  procedure create_new_events_114( game_time: in     game_time_type;
                                    event_q: in out event_queue_type;
                                   scenario: in     scenario_type ) is

      Event      : Event_Type_Ptr;
      Object_Ptr : Simulation_Object_Ptr;

  begin --
    Object_Ptr := Get_Unit ( Scenario ); -- Just one unit in this version
    if Can_move(Object_Ptr.all) and      -- Just one kind of initial event
       not Get_Is_Scheduled(Object_Ptr.all)
    then
       -- since this is currently the only type of event, move
       Event := event_type_pkg.move_pkg.Construct_Event (Object_Ptr,
                                                         Game_Time );
       Schedule_Event( Event, Event_Q );
       Set_Is_Scheduled(Object_Ptr.all, true);
    end if;
  end create_new_events_114;
end create_new_events_114_pkg;
```

16. create_user_event_69_pkg.ads

```
  with game_time_type_pkg;         use game_time_type_pkg;
  with event_queue_type_pkg;       use event_queue_type_pkg;
  with user_interaction_type_pkg; use user_interaction_type_pkg;

package create_user_event_69_pkg is

  procedure create_user_event_69( game_time: in     game_time_type;
                                    event_q: in out event_queue_type;
                            user_interaction: in     user_interaction_type );
end create_user_event_69_pkg;
```

17. create_user_event_69_pkg.adb

```
  WITH Simulation_Object_Pkg;  USE Simulation_Object_Pkg;
  WITH Event_Type_Pkg;         USE Event_Type_Pkg;
  with event_type_pkg.end_sim_pkg; use event_type_pkg.end_sim_pkg;

package body create_user_event_69_pkg is

  procedure create_user_event_69( game_time: in     game_time_type;
                                    event_q: in out event_queue_type;
                            user_interaction: in     user_interaction_type ) is

      Event      : Event_Type_Ptr;
      Object_Ptr : Simulation_Object_Ptr := NULL;

  begin --
    if User_Interaction = stop_simulation then
       -- Only one kind of user interaction in this version.
       Event := event_type_pkg.end_sim_pkg.Construct_Event( Object_Ptr,
                                                            Game_Time );
       Schedule_Event( Event, event_q );
    end if;

  end create_user_event_69;
end create_user_event_69_pkg;
```

18. delimiter_pkg.ads

```
package delimiter_pkg is
  type delimiter_array is array (character) of boolean;
  function initialize_delimiter_array return delimiter_array;
end delimiter_pkg;
```

19. delimiter_pkg.adb

```
package body delimiter_pkg is
  function initialize_delimiter_array return delimiter_array is
  begin
    return (' ' | ascii.ht | ascii.cr | ascii.lf => true, others => false);
  end initialize_delimiter_array;
end delimiter_pkg;
```

20. display_re_37_pkg.ads

```
with location_type_pkg; use location_type_pkg;
package display_re_37_pkg is
  procedure display_re_37(replay: location_type);
end display_re_37_pkg;
```

21. display_re_37_pkg.adb

```
  with warrior_event_monitor_task_pkg;
  use warrior_event_monitor_task_pkg;
package body display_re_37_pkg is
  procedure display_re_37(replay: location_type) is
  begin
    warrior_event_monitor_task.display_re_37_entry(replay);
  end display_re_37;
end display_re_37_pkg;
```

22. display_st_31_pkg.ads

```
with statistics_type_pkg; use statistics_type_pkg;
package display_st_31_pkg is
  procedure display_st_31(statistics: statistics_type);
end display_st_31_pkg;
```

23. display_st_31_pkg.adb

```
  with warrior_event_monitor_task_pkg;
  use warrior_event_monitor_task_pkg;
package body display_st_31_pkg is
  procedure display_st_31(statistics: statistics_type) is
  begin
    warrior_event_monitor_task.display_st_31_entry(statistics);
  end display_st_31;
end display_st_31_pkg;
```

24. do_event_66_pkg.ads

```
  with game_time_type_pkg;        use game_time_type_pkg;
  with event_queue_type_pkg;      use event_queue_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions;      use warrior_1_exceptions;

package do_event_66_pkg is

  procedure do_event_66( game_time: in out game_time_type;
                simulation_history: in out event_type_sequence;
                           event_q: in out event_queue_type );
end do_event_66_pkg;
```

25. do_event_66_pkg.adb

```
  with simulation_object_pkg;        use simulation_object_pkg;
  with event_type_pkg;               use event_type_pkg;
  with event_type_pkg.move_pkg;      use event_type_pkg.move_pkg;
  with event_type_pkg.end_sim_pkg;   use event_type_pkg.end_sim_pkg;


package body do_event_66_pkg is

  procedure do_event_66( game_time: in out game_time_type;
                         simulation_history: in out event_type_sequence;
                         event_q: in out event_queue_type ) is

      Next_Time    : Game_Time_Type;
      Event        : Event_Type_Ptr;
  begin --
    Get_Next_Event( Event, event_q );      -- get event from event queue
    Next_Time := Execute_Event( Event.ALL );  -- execute event and get next
                                              -- execution time
    Game_Time := Get_Event_Time( Event.ALL ); -- update game time to time of
                                              -- event
    Simulation_History := Add( Copy_Event( Event.ALL ), Simulation_History );
    if Next_Time /= NEVER then
       Set_Event_Time( Event.ALL, Next_Time );
       Schedule_Event( Event, event_q );
    end if;

  end do_event_66;
end do_event_66_pkg;
```

26. edit_plan_24_pkg.ads

```
with scenario_type_pkg;         use scenario_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions;     use warrior_1_exceptions;

package edit_plan_24_pkg is

  procedure edit_plan_24( new_plan_entered: in boolean;
                          new_y: in     float;
                          new_x: in     float;
                        scenario: in out scenario_type );
end edit_plan_24_pkg;
```

27. edit_plan_24_pkg.adb

```
with Location_Type_pkg;      use Location_Type_pkg;
  with Simulation_Object_Pkg; use Simulation_Object_Pkg;

package body edit_plan_24_pkg is

  procedure edit_plan_24( new_plan_entered: in boolean;
                          new_y: in     float;
                          new_x: in     float;
                        scenario: in out scenario_type ) is

    unit: Simulation_Object_Ptr;
    destination: Location_Type;
```

```
  begin --
    destination := To_Location(new_x, new_y);
    unit := get_unit(scenario);
    Set_Destination(unit.all, destination);

  end edit_plan_24;
end edit_plan_24_pkg;
```

28. enter_new_plan_75_pkg.ads

```
package enter_new_plan_75_pkg is
  procedure enter_new_plan_75(new_plan_entered : out boolean);
  procedure record_input(new_plan_entered : in boolean);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end enter_new_plan_75_pkg;
```

29. enter_new_plan_75_pkg.adb

```
with psdl_streams; use psdl_streams;
package body enter_new_plan_75_pkg is
  package new_plan_entered_buffer is new
    sampled_buffer(boolean);
  use new_plan_entered_buffer;

  procedure enter_new_plan_75(new_plan_entered : out boolean) is
  begin
    -- Get the value from new_plan_entered_buffer
    buffer.read(new_plan_entered);
  end enter_new_plan_75;

  procedure record_input(new_plan_entered : in boolean) is
  begin
    -- Save the value in new_plan_entered_buffer
    buffer.write(new_plan_entered);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of new_plan_entered_buffer
    return new_data;
  end has_new_input;

end enter_new_plan_75_pkg;
```

30. event_queue_type_pkg.ads

```
  WITH sorted_list_pkg;
  WITH Event_Type_Pkg; use Event_Type_Pkg;

package Event_Queue_Type_Pkg is

  type Event_Queue_Type is private;

  PROCEDURE Schedule_Event (Event   : IN     Event_Type_Ptr;
                            Event_Q : IN OUT Event_Queue_Type);

  PROCEDURE Get_Next_Event(Event   : out    Event_Type_Ptr;
                           Event_Q : IN OUT Event_Queue_Type);
```

```
  FUNCTION Is_Empty(Event_Q : IN Event_Queue_Type) RETURN BOOLEAN;

  FUNCTION Empty RETURN Event_Queue_Type;

private
  package e_q_pkg is new sorted_list_pkg(element_type => Event_Type_Ptr,
                                         "<" => "<");
  type Event_Queue_Type is new e_q_pkg.sorted_list;

end Event_Queue_Type_Pkg;
```

31. event_queue_type_pkg.adb

```
  with event_type_pkg.move_pkg;     use event_type_pkg.move_pkg;
  with event_type_pkg.end_sim_pkg;  use event_type_pkg.end_sim_pkg;
  with ada.text_io;

package body Event_Queue_Type_Pkg is

  PROCEDURE Schedule_Event
            (Event   : IN     Event_Type_Ptr;
             Event_Q : IN OUT Event_Queue_Type) is
  begin
    add(Event_Q, Event);
  end Schedule_Event;



  PROCEDURE Get_Next_Event
            (Event   : out    Event_Type_Ptr;
             Event_Q : IN OUT Event_Queue_Type) is
  begin
    get_smallest(Event_Q, Event);
  end Get_Next_Event;



  FUNCTION Is_Empty(Event_Q : IN Event_Queue_Type) RETURN BOOLEAN is
  begin
    return e_q_pkg.is_empty(e_q_pkg.sorted_list(Event_Q));
  end Is_Empty;



  FUNCTION Empty RETURN Event_Queue_Type is
  begin
    return Event_Queue_Type(e_q_pkg.empty);
  end Empty;
end Event_Queue_Type_Pkg;
```

32. event_type_pkg.ads

```
------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.ads
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation
--|              model.
```

```
-------------------------------------------------------------------------
WITH Simulation_Object_Pkg; USE Simulation_Object_Pkg;
WITH Game_Time_Type_Pkg;    USE Game_Time_Type_pkg;

PACKAGE Event_Type_Pkg IS

    TYPE Event_Action_Type IS ( MoveUpdateObj, EndSimulation );

    TYPE Event_Type IS ABSTRACT TAGGED PRIVATE;
    TYPE Event_Type_Ptr IS ACCESS ALL Event_Type'Class;

    -------------------------------------------------------------------
    --| FUNCTION Get_Event_Time
    --| Pre:     An unexecuted event exsist.
    --| Post:    Start time for the event is returned.
    -------------------------------------------------------------------
    FUNCTION Get_Event_Time (Event: IN Event_Type'Class)
                                              RETURN Game_Time_Type;

    -------------------------------------------------------------------
    --| PROCEDURE Set_Event_Time
    --| Pre:
    --| Post:
    -------------------------------------------------------------------
    PROCEDURE Set_Event_Time (Event: IN OUT Event_Type'Class;
                                        Time: IN Game_Time_Type);

    -------------------------------------------------------------------
    --| FUNCTION Get_Object
    --| Pre: An event exist.
    --| Post: The object designated within the event is returned.
    -------------------------------------------------------------------
    FUNCTION Get_Object (Event: IN Event_Type'Class)
                                       RETURN Simulation_Object_Ptr;

    -------------------------------------------------------------------
    --| FUNCTION Get_Action
    --| Pre: An event exist.
    --| Post: The action on the object in the event is returned.
    -------------------------------------------------------------------
    FUNCTION Get_Action (Event: IN Event_Type'Class)
                                       RETURN Event_Action_Type;

    -------------------------------------------------------------------
    --| FUNCTION "<"
    --| Pre: Two event types exist.
    --| Post: The least valued event is returned.
    -------------------------------------------------------------------
    FUNCTION "<" (Left, Right: IN Event_Type_Ptr) RETURN Boolean;


    -------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     A move event has been extracted from the event queue
    --|          and needs to be executed.
    --| Post:    Move event is executed and time executed is returned.
    -------------------------------------------------------------------
```

```
    FUNCTION Execute_Event (Event: IN Event_Type)
                                                    ETURN Game_Time_Type;

    ------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:      An move event exist.
    --| Post:     The move event is copied and a pointer to the copy is
    --|           returned.
    ------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Event_Type)
                                                    RETURN Event_Type_Ptr;

PRIVATE
    TYPE Event_Type IS ABSTRACT TAGGED
      RECORD
        Action      : Event_Action_Type;              -- desired action
                                                      -- to be performed
        Object_Ptr  : Simulation_Object_Ptr := NULL; -- pointer to
                                                      -- simulation
                                                      -- object
        Time        : Game_Time_Type;                 -- time to start
                                                      -- event action
    END RECORD;
END Event_Type_Pkg;
```

33. event_type_pkg.adb

```
------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.adb
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation
--|              model.
------------------------------------------------------------------------
with ada.text_io;
PACKAGE BODY Event_Type_Pkg IS

    ------------------------------------------------------------------------
    --| FUNCTION Get_Event_Time
    --| Pre:      An unexecuted event exsist.
    --| Post:     Start time for the event is returned.
    ------------------------------------------------------------------------
    FUNCTION Get_Event_Time (Event: IN Event_Type'Class)
                                              RETURN Game_Time_Type IS
    BEGIN -- Get_Event_Time
        RETURN Event.Time;
    END Get_Event_Time;


    ------------------------------------------------------------------------
    --| PROCEDURE Set_Event_Time
    --| Pre:
    --| Post:
```

```
-------------------------------------------------------------------
PROCEDURE Set_Event_Time (Event: IN OUT Event_Type'Class;
                                        Time: IN Game_Time_Type) IS
BEGIN -- Set_Event_Time
    Event.Time := Time;
END Set_Event_Time;


-------------------------------------------------------------------
--| FUNCTION Get_Object
--| Pre: An event exist.
--| Post: The object designated within the event is returned.
-------------------------------------------------------------------
FUNCTION Get_Object (Event: IN Event_Type'Class)
                                  RETURN Simulation_Object_Ptr IS
BEGIN -- Get_Object
  RETURN Event.Object_Ptr;
END Get_Object;


-------------------------------------------------------------------
--| FUNCTION Get_Action
--| Pre: An event exist.
--| Post: The action on the object in the event is returned.
-------------------------------------------------------------------
FUNCTION Get_Action (Event: IN Event_Type'Class)
                                    RETURN Event_Action_Type IS
BEGIN -- Get_Action
  RETURN Event.Action;
END Get_Action;


-------------------------------------------------------------------
--| FUNCTION "<"
--| Pre: Two event types exist.
--| Post: The least valued event is returned.
-------------------------------------------------------------------
FUNCTION "<" (Left, Right: IN Event_Type_Ptr) RETURN Boolean IS
  Reply : Boolean;
BEGIN -- "<"
  IF Left.ALL.Time < Right.ALL.Time THEN
     Reply := True;
  ELSIF  Left.ALL.Time > Right.ALL.Time THEN
     Reply := False;
  ELSE
     Reply := ( Left.ALL.Action < Right.ALL.Action );
  END IF;
RETURN Reply;
END "<";

-------------------------------------------------------------------
--| FUNCTION Execute_Event
--| Pre:     A move event has been extracted from the event queue
--|          and needs to be executed.
--| Post:    Move event is executed and time executed is returned.
-------------------------------------------------------------------
FUNCTION Execute_Event (Event: IN Event_Type)
```

```
                                                RETURN Game_Time_Type IS
    begin --
      ada.text_io.put_line("In the base execute event routine.");
      return 100;
    end execute_event;

    ------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:      An move event exist.
    --| Post:     The move event is copied and a pointer to the copy is
    --|           returned.
    ------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Event_Type) RETURN Event_Type_Ptr IS
    begin --
      ada.text_io.put_line("In the base copy routine");
      return null;
    end copy_event;

END Event_Type_Pkg;
```

34. event_type_pkg-end_sim_pkg.ads

```
------------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.End_Sim_Pkg.ads
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation model.
------------------------------------------------------------------------------
PACKAGE Event_Type_Pkg.End_Sim_Pkg IS

    TYPE End_Sim_Event_Type IS NEW Event_Type WITH PRIVATE;

    ------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:      An end simulation event has been extracted from the event
    --|           queue and needs to be executed.
    --| Post:     End Simulation is executed and time executed is returned.
    ------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN End_Sim_Event_Type)
                                                    RETURN Game_Time_Type;

    ------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:      No event exist.
    --| Post:     A move event is constructed and the event is returned.
    ------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                              Time: IN Game_Time_Type)
                              RETURN Event_Type_Ptr;

    ------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:      An event exist.
    --| Post:     The event is copied and the copy is returned.
    ------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN End_Sim_Event_Type) RETURN Event_Type_Ptr;
```

```
PRIVATE
    TYPE End_Sim_Event_Type IS NEW Event_Type WITH NULL RECORD;
END Event_Type_Pkg.End_Sim_Pkg;
```

35. event_type_pkg-end_sim_pkg.adb

```
-------------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.End_Sim_Pkg.adb
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
involving
--|              event types in the Warrior Combat Simulation model.
-------------------------------------------------------------------------------
WITH Warrior_1_Static_Schedulers;  USE Warrior_1_Static_Schedulers;
WITH Warrior_1_Dynamic_Schedulers; USE Warrior_1_Dynamic_Schedulers;
WITH Panel_Gui_3;
WITH warrior_event_monitor_task_pkg;

PACKAGE BODY Event_Type_Pkg.End_Sim_Pkg IS

    -------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     An end simulation event has been extracted from the event
    --|          queue and needs to be executed.
    --| Post:    End simulation is executed and time executed is returned.
    -------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN End_Sim_Event_Type)
                                                    RETURN Game_Time_Type IS
        Time : Game_Time_Type := Event.Time;
    BEGIN -- Execute_Event
      Stop_Static_Schedule;
      Stop_Dynamic_Schedule;
      Panel_Gui_3.End_Simulation;
      warrior_event_monitor_task_pkg.warrior_event_monitor_task.end_task;
      RETURN Time;
    END Execute_Event;

    -------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:     No event exist.
    --| Post:    A move event is constructed and the event is returned.
    -------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                              Time: IN Game_Time_Type)
                              RETURN Event_Type_Ptr IS
      Event: Event_Type_Ptr;

    BEGIN --
      Event := NEW End_Sim_Event_Type'(Action => EndSimulation,
                                       Object_Ptr => Object_Ptr,
                                            Time => Time);

      RETURN Event;

    END Construct_Event;

    -------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:     An event exist.
```

```
    --| Post:    The event is copied and the copy is returned.
    ------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN End_Sim_Event_Type) RETURN Event_Type_Ptr IS
        Copy: Event_Type_Ptr;
    BEGIN -- Copy_Event
        Copy := Construct_Event( Get_Object( Event ), Get_Event_Time( Event ));
      RETURN Copy;
    END Copy_Event;

END Event_Type_Pkg.End_Sim_Pkg;
```

36. event_type_pkg-move_pkg.ads

```
------------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.Move_Pkg.ads
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation model.
------------------------------------------------------------------------------
PACKAGE Event_Type_Pkg.Move_Pkg IS

    TYPE Move_Event_Type IS NEW Event_Type WITH PRIVATE;

    ------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     A move event has been extracted from the event queue and needs
    --|          to be executed.
    --| Post:    Move event is executed and time executed is returned.
    ------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN Move_Event_Type) RETURN Game_Time_Type;

    ------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:     No event exist.
    --| Post:    A move event is constructed and the event is returned.
    ------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                              Time: IN Game_Time_Type)
                              RETURN Event_Type_Ptr;
    ------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:     An move event exist.
    --| Post:    The move event is copied and a pointer to the copy is
    --|          returned.
    ------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Move_Event_Type) RETURN Event_Type_Ptr;


PRIVATE

    TYPE Move_Event_Type IS NEW Event_Type WITH NULL RECORD;

END Event_Type_Pkg.Move_Pkg;
```

37. event_type_pkg-move_pkg.adb

```
-------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.Move_Pkg.adb
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation
--|              model.
-------------------------------------------------------------------------
PACKAGE BODY Event_Type_Pkg.Move_Pkg IS

    ---------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     An move event has been extracted from the event queue
    --|          and needs to be executed.
    --| Post:    Move event is executed and time executed is returned.
    ---------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN Move_Event_Type)
                                          RETURN Game_Time_Type IS
        Time: Game_Time_Type;
    BEGIN -- Execute_Event
        Time := Get_Event_Time(Event);
        Move_Update_Obj( Get_Object(Event).ALL, Time );
        RETURN Time;
    END Execute_Event;

    ---------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:     No event exist.
    --| Post:    A move event is constructed and the event is returned.
    ---------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                                  Time: IN Game_Time_Type)
                                  RETURN Event_Type_Ptr IS
      Event: Event_Type_Ptr;

    BEGIN --
      Event := NEW Move_Event_Type'(Action => MoveUpdateObj,
                                Object_Ptr => Object_Ptr,
                                     Time => Time);

      RETURN Event;
    END Construct_Event;

    ---------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:     An event exist.
    --| Post:    The event is copied and the copy is returned.
    ---------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Move_Event_Type)
                                          RETURN Event_Type_Ptr IS
        Copy: Event_Type_Ptr;
```

```
    BEGIN -- Copy_Event
      IF Get_Object( Event ) /= NULL THEN
        Copy := Construct_Event( Copy_Obj( Get_Object(Event).ALL ),
                                              Get_Event_Time( Event ) );
      ELSE
        Copy := Construct_Event( NULL, Get_Event_Time(Event) );
      END IF;
      RETURN Copy;
    END Copy_Event;

END Event_Type_Pkg.Move_Pkg;
```

38. game_time_type_pkg.ads

```
package game_time_type_pkg is
  subtype game_time_type is integer range -1 .. integer'last;

  never: constant game_time_type := -1;

  function zero return game_time_type;

end game_time_type_pkg;
```

39. game_time_type_pkg.adb

```
package body game_time_type_pkg is

  function zero return game_time_type is
  begin
    return game_time_type(0);
  end zero;
end game_time_type_pkg;
```

40. generated_tae_event_monitor_pkg.ads

```
with Interfaces.C;
use Interfaces.C;

with linker_options_pragma_pkg;

package generated_tae_event_monitor_pkg is
       procedure generated_tae_event_monitor;
       pragma Import(C, generated_tae_event_monitor,
                                  "generated_tae_event_monitor");

end generated_tae_event_monitor_pkg;
```

41. get_re_30_pkg.ads

```
with replay_request_type_pkg; use replay_request_type_pkg;
package get_re_30_pkg is
  procedure get_re_30(replay_request : out replay_request_type);
  procedure record_input(replay_request : in replay_request_type);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_re_30_pkg;
```

42. get_re_30_pkg.adb

```
  with psdl_streams; use psdl_streams;
package body get_re_30_pkg is
  package replay_request_buffer is new
    sampled_buffer(replay_request_type);
  use replay_request_buffer;

  procedure get_re_30(replay_request : out replay_request_type) is
  begin
    -- Get the value from replay_request_buffer
    buffer.read(replay_request);
  end get_re_30;

  procedure record_input(replay_request : in replay_request_type) is
  begin
    -- Save the value in replay_request_buffer
    buffer.write(replay_request);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of replay_request_buffer
    return new_data;
  end has_new_input;
end get_re_30_pkg;
```

43. get_st_27_pkg.ads

```
with statistics_request_type_pkg; use statistics_request_type_pkg;
package get_st_27_pkg is
  procedure get_st_27(statistics_request : out statistics_request_type);
  procedure record_input(statistics_request : in statistics_request_type);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_st_27_pkg;
```

44. get_st_27_pkg.adb

```
  with psdl_streams; use psdl_streams;
package body get_st_27_pkg is
  package statistics_request_buffer is new
    sampled_buffer(statistics_request_type);
  use statistics_request_buffer;

  procedure get_st_27(statistics_request : out statistics_request_type) is
  begin
    -- Get the value from statistics_request_buffer
    buffer.read(statistics_request);
  end get_st_27;

  procedure record_input(statistics_request : in statistics_request_type) is
  begin
    -- Save the value in statistics_request_buffer
    buffer.write(statistics_request);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of statistics_request_buffer
```

```
    return new_data;
  end has_new_input;
end get_st_27_pkg;
```

45. get_user_in_21_pkg.ads

```
with user_interaction_type_pkg; use user_interaction_type_pkg;
package get_user_in_21_pkg is
  procedure get_user_in_21(user_interaction : out user_interaction_type);
  procedure record_input(user_interaction : in user_interaction_type);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_user_in_21_pkg;
```

46. get_user_in_21_pkg.adb

```
  with psdl_streams; use psdl_streams;
package body get_user_in_21_pkg is
  package user_interaction_buffer is new
    sampled_buffer(user_interaction_type);
  use user_interaction_buffer;

  procedure get_user_in_21(user_interaction : out user_interaction_type) is
  begin
    -- Get the value from user_interaction_buffer
    buffer.read(user_interaction);
  end get_user_in_21;

  procedure record_input(user_interaction : in user_interaction_type) is
  begin
    -- Save the value in user_interaction_buffer
    buffer.write(user_interaction);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of user_interaction_buffer
    return new_data;
  end has_new_input;
end get_user_in_21_pkg;
```

47. get_x_65_pkg.ads

```
package get_x_65_pkg is
  procedure get_x_65(new_x : out float);
  procedure record_input(new_x : in float);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_x_65_pkg;
```

48. get_x_65_pkg.adb

```
  with psdl_streams; use psdl_streams;
package body get_x_65_pkg is
  package new_x_buffer is new
    sampled_buffer(float);
  use new_x_buffer;
```

```
  procedure get_x_65(new_x : out float) is
  begin
    -- Get the value from new_x_buffer
    buffer.read(new_x);
  end get_x_65;

  procedure record_input(new_x : in float) is
  begin
    -- Save the value in new_x_buffer
    buffer.write(new_x);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of new_x_buffer
    return new_data;
  end has_new_input;
end get_x_65_pkg;
```

49. get_y_68_pkg.ads

```
package get_y_68_pkg is
  procedure get_y_68(new_y : out float);
  procedure record_input(new_y : in float);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_y_68_pkg;
```

50. get_y_68_pkg.adb

```
  with psdl_streams; use psdl_streams;
package body get_y_68_pkg is
  package new_y_buffer is new
    sampled_buffer(float);
  use new_y_buffer;

  procedure get_y_68(new_y : out float) is
  begin
    -- Get the value from new_y_buffer
    buffer.read(new_y);
  end get_y_68;

  procedure record_input(new_y : in float) is
  begin
    -- Save the value in new_y_buffer
    buffer.write(new_y);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of new_y_buffer
    return new_data;
  end has_new_input;
end get_y_68_pkg;
```

51. gui_event_monitor_18_pkg.ads

```
package gui_event_monitor_18_pkg is
  procedure gui_event_monitor_18;
end gui_event_monitor_18_pkg;
```

52. gui_event_monitor_18_pkg.adb

```
  with warrior_event_monitor_task_pkg;
  use warrior_event_monitor_task_pkg;
package body gui_event_monitor_18_pkg is
  procedure gui_event_monitor_18 is
  begin
    warrior_event_monitor_task.event_monitor_entry;
  end gui_event_monitor_18;
end gui_event_monitor_18_pkg;
```

53. initial_scenario_40_pkg.ads

```
with scenario_type_pkg; use scenario_type_pkg;
package initial_scenario_40_pkg is
  procedure initial_scenario_40(scenario : out scenario_type;
                                first_time : in out boolean);
end initial_scenario_40_pkg;
```

54. initial_scenario_40_pkg.adb

```
package body initial_scenario_40_pkg is
  procedure initial_scenario_40(scenario : out scenario_type;
                                first_time : in out boolean) is
  begin
    initialize_scenario(scenario);
    first_time := false;
  end initial_scenario_40;
end initial_scenario_40_pkg;
```

55. jaaws_12_pkg.ads

```
  with replay_request_type_pkg; use replay_request_type_pkg;
  with location_type_pkg; use location_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions; use warrior_1_exceptions;

package jaaws_12_pkg is

  procedure jaaws_12(
    simulation_history: in event_type_sequence;
    replay_request: in out replay_request_type;
    replay_position: in out integer;
    replay: out location_type );
end jaaws_12_pkg;
```

56. jaaws_12_pkg.adb

```
  with simulation_object_pkg; use simulation_object_pkg;
  with event_type_pkg; use event_type_pkg;

package body jaaws_12_pkg is

  procedure jaaws_12(
    simulation_history: in event_type_sequence;
    replay_request: in out replay_request_type;
    replay_position: in out integer;
    replay: out location_type ) is
      -- Precondition: not is_empty(simulation_history)
      -- Precondition: 1 <= replay_position <= length(simulation_history)
```

```
      i: integer;
      e: event_type_ptr;
      o: simulation_object_ptr;
  begin
    -- replay_position = previous snapshot location or 1
    -- Set replay to the previous snapshot.
    e := fetch(simulation_history, replay_position);
    if get_action(e.all) = MoveUpdateObj then
       o := get_object(e.all);
       replay := get_location(o.all);
    else -- the previous position is not at a move event
         replay := origin;
    end if;

    -- Set i to the tentative new replay position
    if replay_request = on then -- reset to the beginning
       replay_request := off;
       i := 1;
       -- e := fetch(simulation_history, i);
       -- o := get_object(e.all);
       -- replay := get_location(o.all);
       -- replay_position := i;
    elsif replay_position < length(simulation_history) then
          i := replay_position + 1;
    else i := replay_position; -- Already at the end, stay there.
    end if;

    -- Advance i to the location of the next move event if there is one.
    -- Invariant: 1 <= i <= length(simulation_history)

    e := fetch(simulation_history, i);
    while get_action(e.all) /= MoveUpdateObj loop
       if i < length(simulation_history) then
          i := i + 1;
          e := fetch(simulation_history, i);
       else -- There is no next move event, stay at the previous position.
            -- i is at the last simulation history event and it is not
            -- a MoveUpdateObj event, so do nothing
            -- replay_position maintains the old value
            -- replay maintains either old value or origin
            return;
       end if;
    end loop;
    -- i is at a new MoveUpdateObj event position
    o := get_object(e.all);
    replay := get_location(o.all);
    replay_position := i;
  end jaaws_12;
end jaaws_12_pkg;
```

57. linker_options_pragma_pkg.ads

```
package linker_options_pragma_pkg is
   pragma Linker_Options("warrior_tae.c");
   pragma Linker_Options("warrior_pan_gui_3.c");
   pragma Linker_Options("warrior_creat_init.c");
   pragma Linker_Options("warrior_init_pan.c");
   pragma Linker_Options("-I/local/tae/include");
   pragma Linker_Options("/local/tae/lib/sun4/libwpt.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libXtae.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libddo.a");
   pragma Linker_Options("/local/tae/lib/sun4/libwmw.a");
```

```
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libIV.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libxterm.a");
   pragma Linker_Options("/usr/lib/libXm.a");
   pragma Linker_Options("/usr/lib/libXt.a");
   pragma Linker_Options("/usr/lib/libXmu.a");
   pragma Linker_Options("/usr/lib/libXext.a");
   pragma Linker_Options("/usr/lib/libX11.a");
   pragma Linker_Options("/local/tae/lib/sun4/libtaec.a");
   pragma Linker_Options("/local/tae/lib/sun4/libtae.a");
   pragma Linker_Options("/usr/lib/libtermlib.a");
   pragma Linker_Options("/usr/lib/libm.a");
   pragma Linker_Options("/usr/local/lib/libcxx.a");
end linker_options_pragma_pkg;
```

58. location_type_pkg.ads

```
package location_type_pkg is

   type Location_Type is record
         X : Float := 0.0;
         Y : Float := 0.0;
         Z : Float := 0.0;
      end record;

   FUNCTION "+" (L1,L2 : Location_Type) RETURN Location_Type;

   FUNCTION "-" (L1,L2 : Location_Type) RETURN Location_Type;

   FUNCTION "*" (C : Float; L : Location_Type) RETURN Location_Type;

   FUNCTION Length (L : Location_Type) RETURN Float;

   FUNCTION "=" (L1,L2 : Location_Type) RETURN Boolean;

   FUNCTION Get_X (L : Location_Type) RETURN Float;

   FUNCTION Get_Y (L : Location_Type) RETURN Float;

   FUNCTION Origin RETURN Location_Type;

   FUNCTION To_Location(X, Y: Float) RETURN Location_Type;
end location_type_pkg;
```

59. location_type_pkg.adb

```
--  ------------------------------------------------------------------------
--  File Name:    Location_Type_Pkg.Adb
--  ------------------------------------------------------------------------
WITH Ada.Numerics.Elementary_Functions;  --Used for Square Root
USE  Ada.Numerics.Elementary_Functions;

PACKAGE BODY Location_Type_Pkg IS

   FUNCTION "+" (L1,L2 : Location_Type) RETURN Location_Type IS
      BEGIN
         RETURN (X=> L1.X + L2.X, Y=> L1.Y + L2.Y, Z=> L1.Z + L2.Z);
      END;

   FUNCTION "-" (L1,L2 : Location_Type) RETURN Location_Type IS
      BEGIN
         RETURN (X=> L1.X - L2.X, Y=> L1.Y - L2.Y, Z=> L1.Z - L2.Z);
      END;
```

```
   FUNCTION "*" (C : Float; L : Location_Type) RETURN Location_Type IS
      BEGIN
         RETURN (X=> C * L.X, Y=> C * L.Y, Z=> C * L.Z);
      END;

   FUNCTION Length (L : Location_Type) RETURN Float IS
      BEGIN
         RETURN Sqrt((L.X * L.X) + (L.Y * L.Y) + (L.Z * L.Z));
      END;

   FUNCTION "=" (L1,L2 : Location_Type) RETURN Boolean IS
      BEGIN
         RETURN (L1.X=L2.X AND L1.Y=L2.Y AND L1.Z=L2.Z);
      END;

   FUNCTION Get_X (L : Location_Type) RETURN Float IS
      BEGIN
         RETURN L.X;
      END;

   FUNCTION Get_Y (L : Location_Type) RETURN Float IS
      BEGIN
         RETURN L.Y;
      END;

   FUNCTION Origin RETURN Location_Type IS
         L : Location_Type:=(X=>0.0, Y=>0.0, Z=>0.0);
      BEGIN
         RETURN L;
      END;

   FUNCTION To_Location(X, Y: Float) RETURN Location_Type IS
         L : Location_Type:=(X=>X, Y=>Y, Z=>0.0);
      BEGIN
         RETURN L;
      END;

END Location_Type_Pkg;
```

60. lookahead_stream_pkg.ads

```
with io_exceptions;
  with delimiter_pkg; use delimiter_pkg;
package lookahead_stream_pkg is
  function token return character;
    -- Returns the next non-blank character without removing it.
    -- Raises constraint_error if no more tokens in the buffer.

  procedure skip_char; -- removes the current character.

  end_error: exception renames io_exceptions.end_error;
    -- Attempt to read past end of file.
end lookahead_stream_pkg;
```

61. lookahead_stream_pkg.adb

```
  with text_io; use text_io;
package body lookahead_stream_pkg is
  blank: constant delimiter_array := initialize_delimiter_array;
  buffer: character;
  empty: boolean := true;
```

```
  -- (~empty => buffer is the next character in the stream).

  function peek return character is
  begin
    if empty then get(buffer); empty := false; end if;
    return buffer;
  end peek;

  function token return character is
   -- Blank is a constant array, see top of package body.
  begin
     -- Advance the lookahead stream to a non-blank character.
   while blank(peek) loop skip_char; end loop;
     -- Return the character without removing it from the stream.
   return peek;
  end token;

  procedure skip_char is
  begin
    if empty then get(buffer);  -- Read and discard next character.
    else empty := true; end if;  -- Discard character in the buffer.
  end skip_char;
end lookahead_stream_pkg;
```

62. natural_set_io_pkg.ads

```
with natural_set_pkg;
with text_io;
with integer_io;

package natural_set_io_pkg is

  procedure put(ns: in natural_set_pkg.set);

end natural_set_io_pkg;
```

63. natural_set_io_pkg.adb

```
package body natural_set_io_pkg is

  package natural_io is new text_io.integer_io(NATURAL);

  procedure put_n(i: in natural) is
  begin
    natural_io.put(i);
  end put_n;

  procedure mput is new natural_set_pkg.generic_put(put_n);

  procedure put(ns: in natural_set_pkg.set) is
  begin
    mput(ns);
  end put;

end natural_set_io_pkg;
```

64. natural_set_pkg.ads

```
with set_pkg;

package natural_set_pkg is NEW set_pkg(NATURAL,"=");
```

65. panel_gui_3.ads

```
        with Interfaces.C;
        use Interfaces.C;

        with statistics_type_pkg;
        use statistics_type_pkg;
        with location_type_pkg;
        use location_type_pkg;
package panel_gui_3 is

        -- procedures for calling c routines to display info to GUI

        procedure display_st_31(statistics: in statistics_type);

        procedure display_re_37(replay: in location_type);


        -- procedures to be called by the c routines to handle push button events

        procedure set_user_interaction;
        pragma Export(C, set_user_interaction, "set_user_interaction");

        procedure set_statistics_request;
        pragma Export(C, set_statistics_request, "set_statistics_request");

        procedure set_replay_request;
        pragma Export(C, set_replay_request, "set_replay_request");

        procedure set_new_plan;
        pragma Export(C, set_new_plan, "set_new_plan");

        procedure end_simulation;
        pragma Import(C, end_simulation, "end_simulation");

        procedure set_x(x : in double);
        pragma Export(C, set_x, "set_x");

        procedure set_y(y : in double);
        pragma Export(C, set_y, "set_y");

        procedure initialize_gui;
        pragma Import(C, initialize_gui, "initialize_gui");

end panel_gui_3;
```

66. panel_gui_3.adb

```
        with Interfaces.C;
        use Interfaces.C;

        with statistics_type_pkg;
        use statistics_type_pkg;
        with location_type_pkg;
        use location_type_pkg;
        with replay_request_type_pkg;
        use replay_request_type_pkg;
        with statistics_request_type_pkg;
        use statistics_request_type_pkg;
        with user_interaction_type_pkg;
        use user_interaction_type_pkg;
```

```
        with get_user_in_21_pkg;
        with get_st_27_pkg;
        with get_re_30_pkg;
        with get_x_65_pkg;
        with get_y_68_pkg;
        with enter_new_plan_75_pkg;

        with text_io;
        with ada.float_text_io;
        use ada.float_text_io;

package body panel_gui_3 is


        procedure display_fuel_consumption(c: in double);
        pragma Import(C, display_fuel_consumption, "display_fuel_consumption");

        procedure display_xloc(x: in double);
        pragma Import(C, display_xloc, "display_xloc");

        procedure display_yloc(y: in double);
        pragma Import(C, display_yloc, "display_yloc");

        procedure display_mover(x, y: in double);
        pragma Import(C, display_mover, "display_mover");

        procedure display_st_31(statistics: statistics_type) is
              d : double := double(statistics_type_pkg.convert(statistics));
        begin

              display_fuel_consumption(d);
        end display_st_31;

        procedure display_re_37(replay: location_type) is
              x, y : double;
        begin
              -- need code to extract x, y from location type;
              -- set x, y to dummy value 5.0, -5.0 for the time being
              x := double(location_type_pkg.get_x(replay));
              y := double(location_type_pkg.get_y(replay));
              display_xloc(x);
              display_yloc(y);
              display_mover(x, y);
        end display_re_37;

        procedure set_user_interaction is
              v : user_interaction_type
                        := user_interaction_type_pkg.stop_simulation;
        begin
              get_user_in_21_pkg.record_input(v);
        end set_user_interaction ;

        procedure set_statistics_request is
              v : statistics_request_type := statistics_request_type_pkg.on;
        begin
              get_st_27_pkg.record_input(v);
        end set_statistics_request ;

        procedure set_replay_request is
              v : replay_request_type := replay_request_type_pkg.on;
        begin
              get_re_30_pkg.record_input(v);
        end set_replay_request;
```

```
        procedure set_new_plan is
        begin
              enter_new_plan_75_pkg.record_input(true);
        end set_new_plan;

         procedure set_x(x : in double) is
         begin
              get_x_65_pkg.record_input(float(x));
         end set_x;

         procedure set_y(y : in double) is
         begin
              get_y_68_pkg.record_input(float(y));
         end set_y;

end panel_gui_3;
```

67. post_processor_6_pkg.ads

```
with statistics_request_type_pkg; use statistics_request_type_pkg;
  with statistics_type_pkg; use statistics_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions; use warrior_1_exceptions;

package post_processor_6_pkg is

  procedure post_processor_6(
    statistics_request: in statistics_request_type;
    simulation_history: in event_type_sequence;
    statistics: out statistics_type );
end post_processor_6_pkg;
```

68. post_processor_6_pkg.adb

```
  with simulation_object_pkg; use simulation_object_pkg;
  with event_type_pkg; use event_type_pkg;
package body post_processor_6_pkg is

  procedure post_processor_6(
    statistics_request: in statistics_request_type;
    simulation_history: in event_type_sequence;
    statistics: out statistics_type ) is
      o: simulation_object_ptr;
      e: event_type_ptr;
      fuel_used: float := 0.0;
  begin
    -- This version assumes a vehicle never refuels
    for i IN 1 .. length(simulation_history) loop
        e := fetch(simulation_history, i);
        if get_action(e.all) = MoveUpdateObj then
           o := get_object(e.all);
           fuel_used := get_fuel_used(o.all);
        end if;
    end loop;
    statistics := convert(fuel_used);
  end post_processor_6;
end post_processor_6_pkg;
```

69. replay_request_type_pkg.ads

```
package replay_request_type_pkg is
  type replay_request_type is private;

  function on return replay_request_type;

  function off return replay_request_type;

private
  type replay_request_type is new boolean;
end replay_request_type_pkg;
```

70. replay_request_type_pkg.adb

```
package body replay_request_type_pkg is

  function on return replay_request_type is
  begin
    return true;
  end on;

  function off return replay_request_type is
  begin
    return false;
  end off;
end replay_request_type_pkg;
```

71. scenario_type_pkg.ads

```
with Simulation_Object_Pkg; use Simulation_Object_Pkg;
package scenario_type_pkg is
  type scenario_type is private;

  PROCEDURE Initialize_Scenario(Sc1 : OUT Scenario_Type);

  function empty_scenario return scenario_type;

  function is_empty(SC1 : Scenario_Type) return boolean;

  function get_unit(SC1 : Scenario_Type) RETURN Simulation_Object_Ptr;

private
   type scenario_type is record
         Scenario_Name : String(1..20) := "empty scenario      ";
         Unit          : Simulation_Object_Ptr := NULL;  --Now only 1 obj,
could be List
         --Terrain     : Terrain_Type;
         --Weather     : Weather_Type;
      end record;

end scenario_type_pkg;
```

72. scenario_type_pkg.adb

```
WITH Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;
  USE  Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;

PACKAGE BODY Scenario_Type_Pkg IS
```

```
  function get_unit(SC1 : Scenario_Type) RETURN Simulation_Object_Ptr IS
     BEGIN
        RETURN SC1.Unit;
     END;

  function empty_scenario return scenario_type is
       dummy : scenario_type;
  begin
    return dummy;
  end empty_scenario;

  function is_empty(SC1 : Scenario_Type) return boolean is
  begin
    return SC1.Unit = null;
  end is_empty;

   PROCEDURE Initialize_Scenario(Sc1 : OUT Scenario_Type) IS

      BEGIN
         Sc1.Scenario_Name:="Scenario One        ";
         Sc1.Unit := Construct_Obj(Scheduled => False,
                                   Name      => "M1A1                ",
                                   Symbol    => 1,
                                   Force     => 1,
                                   Move_Period => 10,
                                   Active    => True,
                                   Location_x => -100.0,
                                   Location_y => -100.0,
                                   Destination_x => 3000.0,
                                   Destination_y => 3000.0,
                                   Speed =>  10.0,
                                   Max_Speed =>  25.0,
                                   Fuel => 500.0,
                                   Consumption =>   0.36);
      END;

END Scenario_Type_Pkg;
```

73. sequence_pkg.ads

```
with natural_set_pkg;
with text_io;
use  text_io;

generic
  type t is private;

package sequence_pkg is
  type    sequence    is private;
  subtype natural_set is natural_set_pkg.set;
  function empty return sequence;
  procedure empty(ss : out sequence);
  function add(x : t; s : sequence) return sequence;
  procedure add(x : in t; s : in sequence; ss : out sequence);
  generic
    with function equal(x, y : t) return boolean is <>;
  function remove(x : t; s : sequence) return sequence;
  function append(s1, s2 : sequence) return sequence;
  procedure append(s1, s2 : in sequence; ss : out sequence);
  function fetch(s : sequence; n : natural) return t;
  procedure fetch(s : in sequence; n : in natural; tt : out t);
  function fetch(s1 : sequence; low, high : natural) return sequence;
```

```
procedure fetch(s1 : in sequence; low, high : in natural; ss : out sequence);
function length(s : sequence) return natural;
procedure length(s : in sequence; nn : out natural);
function domain(s : sequence) return natural_set;
procedure domain(s : in sequence; ns : out natural_set);
generic
  with function equal(x, y : t) return boolean is <>;
function is_in(x : t; s : sequence) return boolean;
generic
  with function equal(x, y : t) return boolean is <>;
function part_of(s1, s2 : sequence) return boolean;
generic
  with function equal(x, y : t) return boolean is <>;
function generic_equal(s1, s2 : sequence) return boolean;
generic
  with function "<" (x, y : t) return boolean is <>;
function less_than(s1, s2 : sequence) return boolean;
generic
  with function "<" (x, y : t) return boolean is <>;
  with function equal(x, y : t) return boolean is <>;
function less_than_or_equal(s1, s2 : sequence) return boolean;
generic
  with function "<" (x, y : t) return boolean is <>;
function greater_than(s1, s2 : sequence) return boolean;
generic
  with function "<" (x, y : t) return boolean is <>;
  with function equal(x, y : t) return boolean is <>;
function greater_or_equal(s1, s2 : sequence) return boolean;
generic
  with function equal(x, y : t) return boolean is <>;
function subsequence(s1, s2 : sequence) return boolean;
generic
  with function "<" (x, y : t) return boolean is <>;
  with function successor(x : t) return t;
function interval(x1, x2 : t) return sequence;
generic
  type et is private;
  type st is private;
  with function f(x : et) return t;
  with function length(s : st) return natural is <>;
  with function fetch(s : st; n : natural) return et is <>;
function apply(s1 : st) return sequence;
generic
  with function f(x, y : t) return t;
  identity : t;
function reduce(s : sequence) return t;
generic
  with function f(x, y : t) return t;
function reduce1(s : sequence) return t;
generic
  with procedure generate(x1 : in t);
procedure scan(s : sequence);
generic
  with function input return t is <>;
function generic_input return sequence;
generic
  with function input return t is <>;
function generic_file_input(file : file_type) return sequence;
generic
  with procedure put(item : in t) is <>;
procedure generic_put(item : in sequence);
generic
  with procedure put(item : in t) is <>;
```

```
  procedure generic_file_put(file : in file_type; item : in sequence);
  bounds_error              : exception;
  empty_reduction_undefined : exception;
private
  type sequence_record;
  type sequence_ptr   is access sequence_record;
  type sequence is record
    p : sequence_ptr := null;
  end record;
end sequence_pkg;
```

74. sequence_pkg.adb

```
with lookahead_stream_pkg;

use  lookahead_stream_pkg;


package body sequence_pkg is
  use  natural_set_pkg;

  type sequence_record is record
    value : t;
    rest  : sequence;
  end record;

  function empty return sequence is
    s : sequence;

  begin
    return s;
  end empty;

  procedure empty(ss : out sequence) is
  begin
    ss := empty;
  end empty;

  function add(x : t; s : sequence) return sequence is
    s1 : sequence;

  begin
    if s = empty then
      s1.p := new sequence_record'(value => x, rest => s);

    else
      s1.p := new sequence_record'(value => s.p.value,
                                   rest => add(x, s.p.rest));
    end if;
    return s1;
  end add;

  procedure add(x : in t; s : in sequence; ss : out sequence) is
  begin
    ss := add(x, s);
  end add;
```

```
function remove(x : t; s : sequence) return sequence is
  ss      : sequence;
  local_x : t := x;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

    if not equal(local_x, y) then
      ss := add(y, ss);
    end if;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;          -- of generator loop
  return ss;
end remove;

function append(s1, s2 : sequence) return sequence is
  ss : sequence;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(x : t) is
    begin
    ss := add(x, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
    null;
  end;                                -- of generator loop
  declare                             -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(x : t) is
    begin
    ss := add(x, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new
                                    scan(generator_loop_body);
  begin
    execute_generator_loop(s2);
  exception
    when exit_from_generator_loop =>
    null;
  end;    -- of generator loop
```

```
  return ss;
end append;

procedure append(s1, s2 : in sequence; ss : out sequence) is
begin
  ss := append(s1, s2);
end append;

function fetch(s : sequence; n : natural) return t is
  index : natural := 1;

begin    -- begin generator loop
  declare
    generator_loop_return_value : t;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(y : t) is
    begin

    if index = n then
      generator_loop_return_value := y;
      raise return_from_generator_loop;
    end if;
    index := index + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;                              -- of generator loop
  raise bounds_error;
end fetch;

procedure fetch(s : in sequence; n : in natural; tt : out t) is
begin
  tt := fetch(s, n);
end fetch;

function fetch(s1 : sequence; low, high : natural) return sequence is
  ss : sequence;

begin
  for i in low .. high loop
    ss := add(fetch(s1, i), ss);
  end loop;
  return ss;
end fetch;

procedure fetch(s1 : in sequence; low, high : in natural;
                                                ss : out sequence) is
begin
  ss := fetch(s1, low, high);
```

```
end fetch;

function length(s : sequence) return natural is
  index : natural := 0;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
    index := index + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;                                -- of generator loop
  return index;
end length;

procedure length(s : in sequence; nn : out natural) is
begin
  nn := length(s);
end length;

function domain(s : sequence) return natural_set is
  ns : natural_set := empty;

begin
  for i in 1 .. length(s) loop
    ns := add(i, ns);
  end loop;
  return ns;
end domain;

procedure domain(s : in sequence; ns : out natural_set) is
begin
  ns := domain(s);
end domain;

function is_in(x : t; s : sequence) return boolean is
  local_x : t := x;

begin    -- begin generator loop
  declare
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(y : t) is
    begin

    if equal(local_x, y) then
      generator_loop_return_value := true;
      raise return_from_generator_loop;
```

```
    end if;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;                                -- of generator loop
  return false;
end is_in;

function part_of(s1, s2 : sequence) return boolean is
  n : natural := 0;

  function matches_at(s1, s2 : sequence; n : natural)
                                                  return boolean is
    i : natural := 0;

  begin
    while i < length(s1) loop
    if equal(fetch(s1, i + 1), fetch(s2, n + i)) then
      i := i + 1;

    else
      return false;
    end if;
    end loop;
    return true;
  end matches_at;

begin
  while n + length(s1) <= length(s2) loop
    if matches_at(s1, s2, n + 1) then
    return true;

    else
    n := n + 1;
    end if;
  end loop;
  return false;
end part_of;

function generic_equal(s1, s2 : sequence) return boolean is
  i        : natural  := 1;
  local_s2 : sequence := s2;

begin
  if length(s1) /= length(s2) then
    return false;
  end if;
  declare    -- begin generator loop
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
```

```
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(x : t) is
    begin

    if not equal(x, fetch(local_s2, i)) then
      generator_loop_return_value := false;
      raise return_from_generator_loop;
    end if;
    i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;    -- of generator loop
  return true;
end generic_equal;

function less_than(s1, s2 : sequence) return boolean is
  i        : natural  := 1;
  y        : t;
  local_s2 : sequence := s2;

begin    -- begin generator loop
  declare
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(x : t) is
    begin
    y := fetch(local_s2, i);

    if x < y then
      generator_loop_return_value := true;
      raise return_from_generator_loop;

    elsif y < x then
      generator_loop_return_value := false;
      raise return_from_generator_loop;
    end if;
    i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;                                    -- of generator loop
```

```
  return (length(s1) < length(s2));
end less_than;

function less_than_or_equal(s1, s2 : sequence) return boolean is
  function lt is new less_than;
  function seq_equal is new generic_equal(equal);

begin
  return lt(s1, s2) or else seq_equal(s1, s2);
end less_than_or_equal;

function greater_than(s1, s2 : sequence) return boolean is
  function lt is new less_than;

begin
  return lt(s2, s1);
end greater_than;
function greater_or_equal(s1, s2 : sequence) return boolean is
  function lt is new less_than;
  function seq_equal is new generic_equal(equal);

begin
  return lt(s2, s1) or else seq_equal(s1, s2);
end greater_or_equal;

function subsequence(s1, s2 : sequence) return boolean is
  i        : natural  := 0;
  local_s1 : sequence := s1;

begin
  if s1 = empty then
    return (true);
  end if;
  declare    -- begin generator loop
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(x : t) is
    begin

    if equal(x, fetch(local_s1, i + 1)) then
      i := i + 1;

      if i = length(local_s1) then
        generator_loop_return_value := (true);
        raise return_from_generator_loop;
      end if;
    end if;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s2);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
```

```
    return generator_loop_return_value;
  end;    -- of generator loop
  return false;
end subsequence;

function interval(x1, x2 : t) return sequence is
  ss : sequence;
  y  : t := x1;

begin
  while (y < x2) loop
    ss := add(y, ss);
    y := successor(y);
  end loop;

  if y = x2 then
    ss := add(y, ss);
  end if;
  return ss;
end interval;

function apply(s1 : st) return sequence is
  ss : sequence;

begin
  for i in 1 .. length(s1) loop
    ss := add(f(fetch(s1, i)), ss);
  end loop;
  return ss;
end apply;

function reduce(s : sequence) return t is
  x : t := identity;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
    x := f(y, x);
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;                                -- of generator loop
  return x;
end reduce;

function reduce1(s : sequence) return t is
  x : t;
  i : natural := 1;

begin
```

```
  if s = empty then
    raise empty_reduction_undefined;
  end if;
  x := fetch(s, 1);
  declare    -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

    if i > 1 then
      x := f(y, x);
    end if;
    i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                              scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;    -- of generator loop
  return x;
end reduce1;

procedure scan(s : sequence) is
  ss : sequence := s;

begin
  while ss.p /= null loop
    generate(ss.p.value);
    ss := ss.p.rest;
  end loop;
end scan;

function generic_input return sequence is
  x  : t;
  ss : sequence;

begin
  if token /= ascii.l_bracket then
    raise data_error;
  end if;
  skip_char;
  while token /= ascii.r_bracket loop
    x := input;
    ss := add(x, ss);

    if token = ',' then
    skip_char;

    elsif token /= ascii.r_bracket then
    raise data_error;
    end if;
  end loop;
  skip_char;
  return ss;
```

```
  exception
    when others =>
      raise data_error;
  end generic_input;

  function generic_file_input(file : file_type) return sequence is
    function get_sequence is new generic_input;
    s : sequence;

  begin
    set_input(file);
    s := get_sequence;
    set_input(standard_input);
    return s;
  end generic_file_input;

  procedure generic_put(item : in sequence) is
  begin
    put(ascii.l_bracket);

    if length(item) >= 1 then
      put(fetch(item, 1));
    end if;
    for i in 2 .. length(item) loop
      put(", ");
      put(fetch(item, i));
    end loop;
    put(ascii.r_bracket);
  end generic_put;

  procedure generic_file_put(file : in file_type;
                                                item : in sequence) is
    procedure put_sequence is new generic_put;

  begin
    set_output(file);
    put_sequence(item);
    set_output(standard_output);
  end generic_file_put;
end sequence_pkg;
```

75. set_pkg.ads

```
with text_io;
use  text_io;

generic
  type t is private;
  with function t_equal(x, y : t) return boolean is "=";

package set_pkg is
  type set is private;
  function empty return set;
  procedure empty(ss : out set);
  function add(x : t; s : set) return set;
  procedure add(x : in t; s : in set; ss : out set);
  function remove(x : t; s : set) return set;
  procedure remove(x : in t; s : in set; ss : out set);
```

```
  function is_in(x : t; s : set) return boolean;
  procedure is_in(x : in t; s : in set; bb : out boolean);
  function union(s1, s2 : set) return set;
  procedure union(s1, s2 : in set; ss : out set);
  function difference(s1, s2 : set) return set;
  procedure difference(s1, s2 : in set; ss : out set);
  function intersection(s1, s2 : set) return set;
  procedure intersection(s1, s2 : in set; ss : out set);
  function choose(s : set) return t;
  procedure choose(s : in set; tt : out t);
  function size(s : set) return natural;
  procedure size(s : in set; nn : out natural);
  function equal(s1, s2 : set) return boolean;
  procedure equal(s1, s2 : in set; bb : out boolean);
  function subset(s1, s2 : set) return boolean;
  procedure subset(s1, s2 : in set; bb : out boolean);
  generic
    with function "<" (x, y : t) return boolean is <>;
    with function successor(x : t) return t;
  function interval(x1, x2 : in t) return set;
  generic
    type et is private;
    type st is private;
    with function f(x : t) return et is <>;
    with function empty return st is <>;
    with function add(x : et; s : st) return st is <>;
  function apply(s : set) return st;
  generic
    with function f(x, y : t) return t;
    identity : t;
  function reduce(s : set) return t;
  generic
    with function f(x, y : t) return t;
  function reduce1(s : set) return t;
  generic
    with procedure generate(x1 : in t);
  procedure scan(s : set);
  empty_set                  : exception;
  empty_reduction_undefined : exception;
  generic
    with function input return t is <>;
  function generic_input return set;
  generic
    with function input return t is <>;
  function generic_file_input(file : in file_type) return set;
  generic
    with procedure put(item : in t) is <>;
  procedure generic_put(item : in set);
  generic
    with procedure put(item : in t) is <>;
  procedure generic_file_put(file : in file_type; item : in set);
private
  type set_record;
  type set_ptr   is access set_record;
  type set is record
    p : set_ptr := null;
  end record;
end set_pkg;
```

76. set_pkg.adb

```
with lookahead_stream_pkg;

use  lookahead_stream_pkg;


package body set_pkg is
  type set_record is record
    value : t;
    rest  : set;
  end record;

  function empty return set is
    s : set;

  begin
    return s;
  end empty;

  procedure empty(ss : out set) is
  begin
    ss := empty;
  end empty;

  function add(x : t; s : set) return set is
    ss : set;

  begin
    if is_in(x, s) then
      return s;

    else
      ss.p := new set_record'(value => x, rest => s);
      return ss;
    end if;
  end add;

  procedure add(x : in t; s : in set; ss : out set) is
  begin
    ss := add(x, s);
  end add;

  function remove(x : t; s : set) return set is
    ss : set := empty;

  begin    -- begin generator loop
    declare
      exit_from_generator_loop : exception;
      procedure generator_loop_body(y : t) is
      begin

       if not (t_equal(x, y)) then
         ss := add(y, ss);
       end if;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(s);
    exception
      when exit_from_generator_loop =>
```

```
    null;
  end;                                -- of generator loop
  return ss;
end remove;

procedure remove(x : in t; s : in set; ss : out set) is
begin
  ss := remove(x, s);
end remove;

function is_in(x : t; s : set) return boolean is
begin    -- begin generator loop
  declare
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(y : t) is
    begin

     if t_equal(x, y) then
       generator_loop_return_value := true;
       raise return_from_generator_loop;
     end if;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
    when return_from_generator_loop =>
     return generator_loop_return_value;
  end;     -- of generator loop
  return false;
end is_in;

procedure is_in(x : in t; s : in set; bb : out boolean) is
begin
  bb := is_in(x, s);
end is_in;

function union(s1, s2 : set) return set is
  ss : set := empty;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     ss := add(y, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
     null;
  end;                                -- of generator loop
  declare                             -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     ss := add(y, ss);
```

```
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s2);
  exception
    when exit_from_generator_loop =>
     null;
  end;    -- of generator loop
  return ss;
end union;

procedure union(s1, s2 : in set; ss : out set) is
begin
  ss := union(s1, s2);
end union;

function difference(s1, s2 : set) return set is
  ss : set := empty;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

     if not is_in(y, s2) then
       ss := add(y, ss);
     end if;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
     null;
  end;                                -- of generator loop
  return ss;
end difference;

procedure difference(s1, s2 : in set; ss : out set) is
begin
  ss := difference(s1, s2);
end difference;

function intersection(s1, s2 : set) return set is
  ss : set := empty;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

     if is_in(y, s2) then
       ss := add(y, ss);
     end if;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
     null;
```

```
  end;                                  -- of generator loop
  return ss;
end intersection;

procedure intersection(s1, s2 : in set; ss : out set) is
begin
  ss := intersection(s1, s2);
end intersection;

function choose(s : set) return t is
begin
  if size(s) > 0 then
    return s.p.value;

  else
    raise empty_set;
  end if;
end choose;

procedure choose(s : in set; tt : out t) is
begin
  tt := choose(s);
end choose;

function size(s : set) return natural is
  k : natural := 0;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     k := k + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
  end;                                  -- of generator loop
  return k;
end size;

procedure size(s : in set; nn : out natural) is
begin
  nn := size(s);
end size;

function equal(s1, s2 : set) return boolean is

begin
  return subset(s1, s2) and then subset(s2, s1);
end equal;

procedure equal(s1, s2 : in set; bb : out boolean) is
begin
  bb := equal(s1, s2);
end equal;
```

```
function subset(s1, s2 : set) return boolean is
  begin    -- begin generator loop
    declare
      generator_loop_return_value : boolean;
      return_from_generator_loop  : exception;
      exit_from_generator_loop    : exception;
      procedure generator_loop_body(y : t) is
      begin

       if not (is_in(y, s2)) then
         generator_loop_return_value := false;
         raise return_from_generator_loop;
       end if;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(s1);
    exception
      when exit_from_generator_loop =>
       null;
      when return_from_generator_loop =>
       return generator_loop_return_value;
    end;     -- of generator loop
    return true;
  end subset;

  procedure subset(s1, s2 : in set; bb : out boolean) is
  begin
    bb := subset(s1, s2);
  end subset;

  function interval(x1, x2 : in t) return set is
    ss : set := empty;
    y  : t   := x1;

  begin
    while not (x2 < y) loop
      ss := add(y, ss);
      y := successor(y);
    end loop;
    return ss;
  end interval;

  function apply(s : set) return st is
    ss : st := empty;

  begin    -- begin generator loop
    declare
      exit_from_generator_loop : exception;
      procedure generator_loop_body(y : t) is
      begin
       ss := add(f(y), ss);
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(s);
    exception
      when exit_from_generator_loop =>
       null;
    end;                              -- of generator loop
    return ss;
  end apply;
```

```
function reduce(s : set) return t is
  x : t := identity;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     x := f(y, x);
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
  end;                              -- of generator loop
  return x;
end reduce;

function reduce1(s : set) return t is
  x : t;
  i : natural := 1;

begin
  if size(s) = 0 then
    raise empty_reduction_undefined;
  end if;
  declare    -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

     if i = 1 then
       x := y;

     else
       x := f(y, x);
     end if;
     i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
  end;    -- of generator loop
  return x;
end reduce1;

procedure scan(s : set) is
  ss : set := s;

begin
  while ss.p /= null loop
    generate(ss.p.value);
    ss := ss.p.rest;
  end loop;
end scan;
```

```
function generic_input return set is
    x  : t;
    ss : set := empty;

  begin
    if token /= '{' then
      raise data_error;
    end if;
    skip_char;
    while token /= '}' loop
      x := input;
      ss := add(x, ss);

      if token = ',' then
       skip_char;

      elsif token /= '}' then
       raise data_error;
      end if;
    end loop;
    skip_char;
    return ss;
  exception
    when others =>
      raise data_error;
  end generic_input;

  function generic_file_input(file : in file_type) return set is
    function get_set is new generic_input;
    s : set;

  begin
    set_input(file);
    s := get_set;
    set_input(standard_input);
    return s;
  end generic_file_input;

  procedure generic_put(item : in set) is
    i : natural := 1;

  begin
    put(ascii.l_brace);
    declare    -- begin generator loop
      return_from_generator_loop : exception;
      exit_from_generator_loop   : exception;
      procedure generator_loop_body(y : t) is
      begin

       if i > 1 then
         put(", ");
       end if;
       put(y);
       i := i + 1;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(item);
    exception
      when exit_from_generator_loop =>
       null;
    end;                                    -- of generator loop
    put(ascii.r_brace);
```

```
  end generic_put;

  procedure generic_file_put(file : in file_type; item : in set) is
    procedure put_set is new generic_put;

  begin
    set_output(file);
    put_set(item);
    set_output(standard_output);
  end generic_file_put;
end set_pkg;
```

77. simulation_object_pkg.ads

```
--  ------------------------------------------------------------------------
--  File Name:    Simulation_Object_Pkg.Ads
--  Discription:  This package describes the basis for the Simulation Hierarchy
--  ------------------------------------------------------------------------
WITH game_time_type_pkg;  USE game_time_type_pkg;
WITH Location_Type_Pkg;   USE Location_Type_Pkg;

PACKAGE Simulation_Object_Pkg IS

   TYPE Simulation_Object IS ABSTRACT TAGGED PRIVATE;-- Basis of Simulation
Hierarchy
   TYPE Simulation_Object_Ptr IS ACCESS ALL Simulation_Object'Class;


   --  ------------------------------------------------------------------------
   --  PROCEDURE:    Move_Update_Obj
   --  PRE:          Obj is of type Simulation_Object and exists
   --                Time contains data (value is not never)
   --  POST:         Updates Object's location.  Time represents when to
   --                reschedule.
   --  ------------------------------------------------------------------------
   PROCEDURE Move_Update_Obj(Obj  : IN OUT Simulation_Object;
                             Time : IN OUT game_time_type);


   --  ------------------------------------------------------------------------
   --  FUNCTION:     Can_move
   --  ------------------------------------------------------------------------
   FUNCTION Can_move(Obj : Simulation_Object) RETURN boolean;

   --  ------------------------------------------------------------------------
   --  FUNCTION:     Copy_Obj
   --  PRE:          Obj is of type Simulation_Object and exists
   --  Return:       Makes a copy of the obj and returns a pointer to the new
   --                obj
   --  ------------------------------------------------------------------------
   FUNCTION Copy_Obj(Obj : Simulation_Object) RETURN Simulation_Object_Ptr;


   --  ------------------------------------------------------------------------
   --  FUNCTION:     Get_Is_Scheduled
   --  PRE:          Obj is of type Simulation_Object and exists
   --  Return:       Returns the value of Is_Scheduled which is a boolean type
   --  ------------------------------------------------------------------------
   FUNCTION Get_Is_Scheduled(Obj : Simulation_Object'Class) RETURN Boolean;
```

```
   --  ------------------------------------------------------------------------
   --  PROCEDURE:     Set_Is_Scheduled
   --  PRE:           Obj is of type Simulation_Object and exists
   --  POST:          Assigns Value to Is_Scheduled
   --  ------------------------------------------------------------------------
   PROCEDURE Set_Is_Scheduled(Obj    : IN OUT Simulation_Object'Class;
                              Value :        Boolean);


   --  ------------------------------------------------------------------------
   --  FUNCTION:      Get_Destination
   --  PRE:           Obj is of type Simulation_Object and exists
   --  Return:        Returns the destination
   --  ------------------------------------------------------------------------
   FUNCTION Get_Destination(Obj : Simulation_Object'Class)
                                                    RETURN Location_Type;

   --  ------------------------------------------------------------------------
   --  PROCEDURE:  Set_Destination
   --  PRE:           Obj is of type Simulation_Object and exists
   --  POST:          Assigns Value to the Destination
   --  ------------------------------------------------------------------------
   PROCEDURE Set_Destination(Obj : in out Simulation_Object'Class;
                                                     Value: in Location_Type);


   --  ------------------------------------------------------------------------
   --  FUNCTION:      Get_Location
   --  PRE:           Obj is of type Simulation_Object and exists
   --  Return:        Returns the location
   --  ------------------------------------------------------------------------
   FUNCTION Get_Location(Obj : Simulation_Object'Class) RETURN Location_Type;


   --  ------------------------------------------------------------------------
   --  FUNCTION:      Get_Fuel_Used
   --  PRE:           Obj is of type Simulation_Object and exists
   --  Return:        Returns the float
   --  ------------------------------------------------------------------------
   FUNCTION Get_Fuel_Used(Obj : Simulation_Object) RETURN Float;


PRIVATE

   TYPE Simulation_Object IS TAGGED RECORD
      Is_Scheduled      : Boolean:=False;
      Name              : String(1..20);
      Graphic_Symbol    : Natural;
      Force             : Natural;  --IE 1..6
      Move_Period       : Integer;
      Active            : Boolean;  --True = active part of sim, ie alive
      Location          : Location_Type;
      Destination       : Location_Type; --Could be a sequence of
                                         --Location_Types
      Speed             : Float; --In M/sec
      Max_Speed         : Float; --In M/sec
   END RECORD;

END Simulation_Object_Pkg;
```

78. simulation_object_pkg.adb

```
--  ------------------------------------------------------------------------
--  File Name:    Simulation_Object_Pkg.Adb
--  ------------------------------------------------------------------------

PACKAGE BODY Simulation_Object_Pkg IS

   --  ------------------------------------------------------------------------
   --  PROCEDURE:    Move_Update_Obj
   --  ------------------------------------------------------------------------
   PROCEDURE Move_Update_Obj(Obj  : IN OUT Simulation_Object;
                             Time : IN OUT game_time_type) IS
         Time_Elapsed : Float;  -- In seconds
         Distance     : Float;    -- In meters
         Displacement : Location_Type;
         Velocity     : Location_Type;
      BEGIN
         -- Stop motion if the object cannot move.
         IF not Can_move(Simulation_Object'Class(Obj)) THEN
            Obj.Speed := 0.0;
            Obj.Is_Scheduled := false;
            Time := never; -- Do not reschedule a move event for this object
            return;
         END IF;

         -- How far are we
         Time_Elapsed := Float(Obj.Move_Period);
         Displacement := Obj.Destination - Obj.Location;
         Distance := Length(Displacement);

         -- Set the speed
         -- Future versions will take terrain and weather into account here.
         IF Distance > Obj.Max_Speed * Time_Elapsed
         THEN Obj.Speed := Obj.Max_Speed;
         ELSE Obj.Speed := Distance/Time_Elapsed;
         END IF;

         -- Move and schedule the next move.
         Velocity := (Obj.Speed/Distance) * Displacement;
         Obj.Location := Obj.Location + (Time_Elapsed * Velocity);
         Time := Time + Obj.Move_Period; --Schedules next event in
                                         --Move_Period seconds

         -- Note: the above code works without modification
         -- for both two and three dimensions.
      END Move_Update_Obj;

   --  ------------------------------------------------------------------------
   --  FUNCTION:     Can_move
   --  ------------------------------------------------------------------------
   FUNCTION Can_move(Obj : Simulation_Object) RETURN boolean IS
        Min_Distance : Constant Float := 10.0;
        Distance : Float;
      BEGIN
        Distance := length(Obj.Destination - Obj.Location);
        RETURN Obj.Active -- must be alive to move
               and then Distance > Min_Distance;
                        -- must not already be at the planned destination
      END;
```

```
--  ------------------------------------------------------------------------
--  FUNCTION: Copy_Obj
--  ------------------------------------------------------------------------
FUNCTION Copy_Obj(Obj : Simulation_Object) RETURN Simulation_Object_Ptr IS
   BEGIN
      RETURN NULL;  --All are dispatched to leaves of hierarchy
   END Copy_Obj;


--  ------------------------------------------------------------------------
--  FUNCTION: Get_Is_Scheduled
--  ------------------------------------------------------------------------
FUNCTION Get_Is_Scheduled(Obj : Simulation_Object'Class) RETURN Boolean IS
   BEGIN
      RETURN Obj.Is_Scheduled;
   END Get_Is_Scheduled;


--  ------------------------------------------------------------------------
--  PROCEDURE:  Set_Is_Scheduled
--  ------------------------------------------------------------------------
PROCEDURE Set_Is_Scheduled(Obj   : IN OUT Simulation_Object'Class;
                           Value :        Boolean) IS
   BEGIN
      Obj.Is_Scheduled := Value;
   END Set_Is_Scheduled;


--  ------------------------------------------------------------------------
--  FUNCTION:  Get_Destination
--  ------------------------------------------------------------------------
FUNCTION Get_Destination(Obj : Simulation_Object'Class)
                                          RETURN Location_Type IS
   BEGIN
      RETURN Obj.Destination;
   END Get_Destination;

--  ------------------------------------------------------------------------
--  PROCEDURE:  Set_Destination
--  ------------------------------------------------------------------------
PROCEDURE Set_Destination(Obj : in out Simulation_Object'Class;
                                          Value: in Location_Type) IS
   BEGIN
      Obj.Destination := Value;
   END Set_Destination;


--  ------------------------------------------------------------------------
--  FUNCTION:  Get_Location
--  ------------------------------------------------------------------------
FUNCTION Get_Location(Obj : Simulation_Object'Class) RETURN Location_Type IS
   BEGIN
      RETURN Obj.Location;
   END Get_Location;


--  ------------------------------------------------------------------------
--  FUNCTION:     Get_Fuel_Used
--  ------------------------------------------------------------------------
FUNCTION Get_Fuel_Used(Obj : Simulation_Object) RETURN Float IS
   BEGIN
      RETURN 0.0;
   END Get_Fuel_Used;
```

```
END Simulation_Object_Pkg;
```

79. simulation_object_pkg-ground_object_pkg.ads

```
--  ------------------------------------------------------------------------
--  File Name:    Simulation_Object_Pkg.Ground_Object_Pkg.Ads
--  ------------------------------------------------------------------------

PACKAGE Simulation_Object_Pkg.Ground_Object_Pkg IS

   TYPE Ground_Object IS ABSTRACT NEW Simulation_Object WITH PRIVATE;

PRIVATE
   TYPE Ground_Object IS ABSTRACT NEW Simulation_Object WITH NULL RECORD;


END Simulation_Object_Pkg.Ground_Object_Pkg;
```

80. simulation_object_pkg-ground_object_pkg-tank_pkg.ads

```
--  ------------------------------------------------------------------------
--  File Name: Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg.Ads
--  ------------------------------------------------------------------------

PACKAGE Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg IS

   TYPE Tank_Type IS NEW Ground_Object WITH PRIVATE;


   --  ------------------------------------------------------------------------
   --  PROCEDURE:    Move_Update_Obj
   --  Description:  Overloaded Simulaiton_Object's Method to work on Tank
   --                Objects
   --  ------------------------------------------------------------------------
   PROCEDURE Move_Update_Obj(Obj  : IN OUT Tank_Type;
                             Time : IN OUT game_time_type);

   --  ------------------------------------------------------------------------
   --  FUNCTION:     Can_move
   --  ------------------------------------------------------------------------
   FUNCTION Can_move(Obj : Tank_Type) RETURN boolean;


   --  ------------------------------------------------------------------------
   --  FUNCTION:     Get_Fuel_Used
   --  Description:  Overloads the SImulation_Object's Method
   --  ------------------------------------------------------------------------
   FUNCTION Get_Fuel_Used(Obj : Tank_Type) RETURN Float;

   --  ------------------------------------------------------------------------
   --  FUNCTION:     Copy_Obj
   --  Description:  Overloads the SImulation_Object's Method
   --  ------------------------------------------------------------------------
   FUNCTION Copy_Obj(Obj : Tank_Type) RETURN Simulation_Object_Ptr;

   --  ------------------------------------------------------------------------
   --  FUNCTION:     Construct_Obj
   --  Description:  Constructs a simulation obj
   --  ------------------------------------------------------------------------
   FUNCTION Construct_Obj(Scheduled     : Boolean;
                          Name          : String;
```

```
                              Symbol         : Natural;
                              Force          : Natural;
                              Move_Period    : Integer;
                              Active         : Boolean;
                              Location_x     : Float;
                              Location_y     : Float;
                              Destination_x  : Float;
                              Destination_y  : Float;
                              Speed          : Float;
                              Max_Speed      : Float;
                              Fuel           : Float;
                              Consumption    : Float) RETURN Simulation_Object_Ptr;

PRIVATE

   TYPE Tank_Type IS NEW Ground_Object WITH RECORD
      Fuel              : Float; --In Gallons
      Fuel_Consumption  : Float; --Gallons/Second
      Fuel_Used         : Float:= 0.0;
   END RECORD;


END Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;
```

81. simulation_object_pkg-ground_object_pkg-tank_pkg.adb

```
--  ------------------------------------------------------------------------
--  File Name: Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg.Adb
--  ------------------------------------------------------------------------

PACKAGE BODY Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg IS

   --  ------------------------------------------------------------------------
   --  PROCEDURE:    Move_Update_Obj
   --  ------------------------------------------------------------------------
   PROCEDURE Move_Update_Obj(Obj  : IN OUT Tank_Type;
                             Time : IN OUT game_time_type) IS
      Time_Elapsed : Float;  --In Seconds
   BEGIN
      -- Stop motion if the object cannot move.
      IF not Can_move(Obj) THEN
         Obj.Speed := 0.0;
         Obj.Is_Scheduled := false;
         Time := never; -- Do not reschedule a move event for this object
         return;
      END IF;

      -- Move the tank using the general purpose move method
      -- from the most general superclass.
      Move_Update_Obj(Simulation_Object(Obj), Time);

      -- Now do the fuel consumption bookkeeping.
      Time_Elapsed := Float(Obj.Move_Period);
      Obj.Fuel := Obj.Fuel - (Obj.Fuel_Consumption * Time_Elapsed);
      Obj.Fuel_Used := Obj.Fuel_Used + (Obj.Fuel_Consumption
                                                  * Time_Elapsed);
   END;
```

```
--  ------------------------------------------------------------------------
--  FUNCTION:     Can_move
--  ------------------------------------------------------------------------
FUNCTION Can_move(Obj : Tank_Type) RETURN boolean IS
   BEGIN
     RETURN Can_move(Simulation_Object(Obj)) -- must satisfy inherited
                                             -- general constraints
            and then Obj.Fuel > 0.0;         -- must also have fuel to move
   END;


--  ------------------------------------------------------------------------
--  FUNCTION:     Get_Fuel_Used
--  Description:  Overloads the SImulation_Object's Method
--  ------------------------------------------------------------------------
FUNCTION Get_Fuel_Used(Obj : Tank_Type) RETURN Float IS
   BEGIN
      RETURN Obj.Fuel_Used;
   END;


--  ------------------------------------------------------------------------
--  FUNCTION:     Copy_Obj
--  ------------------------------------------------------------------------
FUNCTION Copy_Obj(Obj : Tank_Type) RETURN Simulation_Object_Ptr IS
      Obj_Ptr  : Simulation_Object_Ptr;
   BEGIN
      Obj_Ptr:= NEW Tank_Type'(Is_Scheduled      => Obj.Is_scheduled,
                               Name              => Obj.Name,
                               Graphic_Symbol    => Obj.Graphic_Symbol,
                               Force             => Obj.Force,
                               Move_Period       => Obj.Move_Period,
                               Active            => Obj.Active,
                               Location          => Obj.Location,
                               Destination       => Obj.Destination,
                               Speed             => Obj.Speed,
                               Max_Speed         => Obj.Max_Speed,
                               Fuel              => Obj.Fuel,
                               Fuel_Consumption  => Obj.Fuel_Consumption,
                               Fuel_Used         => Obj.Fuel_Used);
      RETURN Obj_Ptr;
   END;

--  ------------------------------------------------------------------------
--  FUNCTION:     Construct_Obj
--  Description:  Constructs a simulation obj
--  ------------------------------------------------------------------------
FUNCTION Construct_Obj(Scheduled     : Boolean;
                       Name          : String;
                       Symbol        : Natural;
                       Force         : Natural;
                       Move_Period   : Integer;
                       Active        : Boolean;
                       Location_X    : Float;
                       Location_Y    : Float;
                       Destination_X : Float;
                       Destination_Y : Float;
                       Speed         : Float;
                       Max_Speed     : Float;
                       Fuel          : Float;
                       Consumption   : Float)
                                     RETURN Simulation_Object_Ptr IS
      Obj_Ptr     : Simulation_Object_Ptr;
```

```
      Location    : Location_Type;
      Destination : Location_Type;
   BEGIN
      Location.X    := Location_X;
      Location.Y    := Location_Y;
      Destination.X := Destination_X;
      Destination.Y := Destination_Y;
      Obj_Ptr:= NEW Tank_Type'(Is_Scheduled      => Scheduled,
                               Name              => Name,
                               Graphic_Symbol    => Symbol,
                               Force             => Force,
                               Move_Period       => Move_Period,
                               Active            => Active,
                               Location          => Location,
                               Destination       => Destination,
                               Speed             => Speed,
                               Max_Speed         => Max_Speed,
                               Fuel              => Fuel,
                               Fuel_Consumption  => Consumption,
                               Fuel_Used         => 0.0);
      RETURN Obj_Ptr;
   END;

END Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;
```

82. sorted_list_pkg.ads

```
generic
  type element_type is private;
  with function "<"(x, y: element_type) return boolean;
package sorted_list_pkg is
  type sorted_list is private;

  function empty return sorted_list;
    -- Returns an empty sorted list.

  function is_empty(s: sorted_list) return boolean;
    -- True if and only if s has no elements.

  procedure add(s: in out sorted_list; x: in element_type);
    -- s := s U {x}.

  procedure get_smallest(s: in out sorted_list; x: out element_type);
    -- sets x to the smallest element of s and removes x from s.
    -- raises no_elements if s is empty.

  no_elements: exception;
private
  type sorted_list_record is
    record
      data: element_type;
      next: sorted_list;
    end record;
    -- The list is kept sorted in increasing order wrt "<".
  type sorted_list is access sorted_list_record;
end sorted_list_pkg;
```

83. sorted_list_pkg.adb

```
-- generic
--    type element_type is private;
--    with function "<"(x, y: element_type) return boolean;
package body sorted_list_pkg is
   free_list: sorted_list := null;

   procedure free(node: sorted_list) is
   begin
     node.next := free_list;
     free_list := node;
   end free;

   function new_node(x: element_type; s: sorted_list)
          return sorted_list is
     node: sorted_list;
   begin
     if free_list = null then
        return new sorted_list_record'(data => x, next => s);
     else node := free_list;
         free_list := free_list.next;
         node.data := x;
         node.next := s;
          return node;
     end if;
   end new_node;

   function empty return sorted_list is
   begin
     return null;
   end empty;

   function is_empty(s: sorted_list) return boolean is
   begin
     return (s = null);
   end is_empty;

   procedure add(s: in out sorted_list; x: in element_type) is
   begin
     if is_empty(s) then s := new_node(x, s);
     elsif x < s.data then s := new_node(x, s);
     else add(s.next, x);
     end if;
   end add;

   procedure get_smallest(s: in out sorted_list; x: out element_type) is
     head: sorted_list := s;
   begin
     if is_empty(s) then raise no_elements;
     else x := s.data;
         s := s.next;
         free(head);
     end if;
   end get_smallest;

end sorted_list_pkg;
```

84. statistics_request_type_pkg.ads

```
package statistics_request_type_pkg is
  type statistics_request_type is private;

  function on return statistics_request_type;

  function off return statistics_request_type;

private
  type statistics_request_type is new boolean;
end statistics_request_type_pkg;
```

85. statistics_request_type_pkg.adb

```
package body statistics_request_type_pkg is

  function on return statistics_request_type is
  begin
    return true;
  end on;

  function off return statistics_request_type is
  begin
    return false;
  end off;
end statistics_request_type_pkg;
```

86. statistics_type_pkg.ads

```
package statistics_type_pkg is
  type statistics_type is private;

  function convert(x: statistics_type) return float;

  function convert(x: float) return statistics_type;

private
  type statistics_type is new float;
end statistics_type_pkg;
```

87. statistics_type_pkg.adb

```
package body statistics_type_pkg is
  function convert(x: statistics_type) return float is
  begin
    return float(x);
  end convert;

  function convert(x: float) return statistics_type is
  begin
    return statistics_type(x);
  end convert;
end statistics_type_pkg;
```

88. user_interaction_type_pkg.ads

```
package user_interaction_type_pkg is
  type user_interaction_type is private;

  function stop_simulation return user_interaction_type;

private
  type user_interaction_type is new boolean;
end user_interaction_type_pkg;
```

89. user_interaction_type_pkg.adb

```
package body user_interaction_type_pkg is

  function stop_simulation return user_interaction_type is
  begin
    return true;
  end stop_simulation;
end user_interaction_type_pkg;
```

90. warrior_global.h

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        global.h *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * This global header file is automatically "#include"d in each panel
 * file.  You can insert references to global variables here.
 *
 * REGENERATED:
 * This file is generated only once.  Therefore, you may modify it without
 * impacting automatic code merge.
 *
 * CHANGE LOG:
 * 15-Oct-98    Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_GLOBAL                              /* prevent double include */
#define I_GLOBAL        0

/*      macros for access to parameter values
 *
 * These macros obtain parameter values given the name of
 * a Vm object and the name string of the parameter.
 * The Vm objects are created by the Initialize_All_Panels
 * function for a resource file.
 *
 * Reference scalar parameters as follows:
 *
 *      StringParm(myPanelTarget, "s")     -- string pointer
 *      IntParm(myPanelTarget, "i")        -- integer value
 *      RealParm(myPanelTarget, "r")       -- real value
 *
 *  For vector parameters, do the following:
 *
 *      TAEINT  *ival;
 *      ival = &IntParm(myPanelTarget, "i");
 *      printf ("%d %d %d", ival[0], ival[1], ival[2]);
 *
 */


#define StringParm(vmId, name)     (SVAL(*Vm_Find(vmId, name),0))
#define IntParm(vmId, name)        (IVAL(*Vm_Find(vmId, name), 0))
#define RealParm(vmId, name)       (RVAL(*Vm_Find(vmId, name), 0))


/*      Dispatch Table typedef          */

typedef VOID (*FUNCTION_PTR) ();
typedef struct DISPATCH
    {
    TEXT                *parmName;
    FUNCTION_PTR        eventFunction;
    } Dispatch;

#define EVENT_HANDLER static VOID           /* a flag for documentation */
```

```
/*      Display Id for use by direct Xlib calls: */

extern Display   *Default_Display;


/*      Externally define wptEvent so event handlers have access to it */

extern WptEvent  wptEvent;  /* event structure returned by Wpt_NextEvent */


#define SET_APPLICATION_DONE \
    { \
    extern BOOL Application_Done; \
    Application_Done = TRUE; \
    }

#endif

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                              *** */
/* *** mode:                          c              *** */
/* *** c-indent-level:                0              *** */
/* *** c-continued-statement-offset:  4              *** */
/* *** c-brace-offset:                4              *** */
/* *** c-brace-imaginary-offset:      4              *** */
/* *** c-argdecl-indent:              4              *** */
/* *** c-label-offset:                -4             *** */
/* *** c-continued-brace-offset:      -4             *** */
/* *** comment-column:                45             *** */
/* *** comment-multi-line:            nil            *** */
/* *** End:                                          *** */
```

91. warrior_pan_gui_3.h

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        pan_gui_3.h *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  gui_3
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     The panel's name is changed (not title)
 * For panel:  gui_3
 *
 * CHANGE LOG:
 * 15-Oct-98    Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_gui_3                          /* prevent double include */
#define I_PAN_gui_3      0

/* Vm objects and panel Id.    */
extern Id gui_3Target, gui_3View, gui_3Id;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH gui_3Dispatch[];

/* Initialize gui_3Target and gui_3View */
```

```
extern VOID gui_3_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID gui_3_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID gui_3_Destroy_Panel ();

/* Connect to this panel.  Create it or change its state */
extern VOID gui_3_Connect_Panel ();

/*
    extern VOID warrior_Initialize_All_Panels ();
    extern VOID warrior_Create_Initial_Panels ();
*/

/*# MTS 10-15-98
    added the following procedure declarations
#*/

extern VOID set_user_interaction();
extern VOID set_statistics_request();
extern VOID set_replay_request();
extern VOID set_new_plan();

/*# MTS 10-23-98
    added the following function declarations
#*/

extern VOID set_x();
extern VOID set_y();

FUNCTION VOID display_fuel_consumption();
FUNCTION VOID display_xloc();
FUNCTION VOID display_yloc();
FUNCTION VOID display_mover();
FUNCTION VOID end_simulation();


#endif

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                          *** */
/* *** mode:                          c          *** */
/* *** c-indent-level:                0          *** */
/* *** c-continued-statement-offset:  4          *** */
/* *** c-brace-offset:                4          *** */
/* *** c-brace-imaginary-offset:      4          *** */
/* *** c-argdecl-indent:              4          *** */
/* *** c-label-offset:                -4         *** */
/* *** c-continued-brace-offset:      -4         *** */
/* *** comment-column:                45         *** */
/* *** comment-multi-line:            nil        *** */
/* *** End:                                      *** */
```

92. warrior_creat_init.c

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        warrior_creat_init.c *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * Displays all panels in the initial panel set of this resource file
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     A panel is added to the initial panel set
 *     A panel is deleted from the initial panel set
 *     An initial panel's name is changed (not title)
 * For the set of initial panels:
 *   gui_3
 *
 * CHANGE LOG:
 * MERGE NOTE: Add Change Log entries BELOW this line.
 * 15-Oct-98    Initially generated...TAE
 * MERGE NOTE: Add Change Log entries ABOVE this line.
 * ************************************************************************
 */
#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "warrior_global.h"                  /* Application globals */

/* One include for each panel in initial panel set */
#include        "warrior_pan_gui_3.h"

/* MERGE NOTE: Add additional includes and functions BELOW this line. */
/* MERGE NOTE: Add additional includes and functions ABOVE this line. */


FUNCTION VOID warrior_Create_Initial_Panels ()
    {
    /* MERGE NOTE: Add additional variables and code BELOW this line. */
    /* MERGE NOTE: Add additional variables and code ABOVE this line. */

    /* Show panels */

    gui_3_Create_Panel (NULL, WPT_PREFERRED);

    /* MERGE NOTE: Add additional code BELOW this line. */
    /* MERGE NOTE: Add additional code ABOVE this line. */
    }

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                              *** */
/* *** mode:                          c              *** */
/* *** c-indent-level:                0              *** */
/* *** c-continued-statement-offset:  4              *** */
/* *** c-brace-offset:                4              *** */
/* *** c-brace-imaginary-offset:      4              *** */
/* *** c-argdecl-indent:              4              *** */
/* *** c-label-offset:                -4             *** */
/* *** c-continued-brace-offset:      -4             *** */
/* *** comment-column:                45             *** */
/* *** comment-multi-line:            nil            *** */
/* *** End:                                          *** */
```

93. warrior_init_pan.c

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        warrior_init_pan.c *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * Initialize all panels in the resource file.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     A panel is deleted
 *     A new panel is added
 *     A panel's name is changed (not title)
 * For the panels:
 *   gui_3
 *
 * CHANGE LOG:
 * MERGE NOTE: Add Change Log entries BELOW this line.
 * 15-Oct-98    Initially generated...TAE.
 * MERGE NOTE: Add Change Log entries ABOVE this line.
 * ************************************************************************
 */

#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "symtab.inc"
#include        "warrior_global.h"                  /* Application globals */

/* One "include" for each panel in resource file */
#include        "warrior_pan_gui_3.h"

/* MERGE NOTE: Add additional includes and functions BELOW this line. */
/* MERGE NOTE: Add additional includes and functions ABOVE this line. */


FUNCTION VOID warrior_Initialize_All_Panels (resfileSpec)
    TEXT        *resfileSpec;
    {
    Id vmCollection;

    /* MERGE NOTE: Add additional variables and code BELOW this line. */
    /* MERGE NOTE: Add additional variables and code ABOVE this line. */

    /* read resource file */
    vmCollection = Co_New (P_ABORT);
    Co_ReadFile (vmCollection, resfileSpec, P_ABORT);

    /* initialize view and target Vm objects for each panel */
    gui_3_Initialize_Panel (vmCollection);

    /* MERGE NOTE: Add additional code BELOW this line. */
    /* MERGE NOTE: Add additional code ABOVE this line. */
    }

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                          *** */
/* *** mode:                          c          *** */
/* *** c-indent-level:                0          *** */
/* *** c-continued-statement-offset:  4          *** */
/* *** c-brace-offset:                4          *** */
/* *** c-brace-imaginary-offset:      4          *** */
```

```
/* *** c-argdecl-indent:                4       *** */
/* *** c-label-offset:                  -4      *** */
/* *** c-continued-brace-offset:        -4      *** */
/* *** comment-column:                  45      *** */
/* *** comment-multi-line:              nil     *** */
/* *** End:                                     *** */
```

94. warrior_pan_gui_3.c

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        pan_gui_3.c *** */
/* *** Generated:   Nov  2 14:32:41 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  gui_3
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_gui_3.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * NOTES:
 * For each parameter that you have defined to be "event-generating" in
 * this panel, there is an event handler procedure below.  Each handler
 * has a name that is a concatenation of the parameter name and "_Event".
 * Add application-dependent logic to each event handler.  (As generated
 * by the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * For best automatic code merging results, you should put as many
 * modifications as possible between the lines of the MERGE NOTE comments.
 * Modifications outside the MERGE NOTEs will often merge correctly, but
 * must sometimes be merged by hand.  If the modifications cannot be
 * automatically merged, a reject file (*.rej) will be generated which
 * will contain your modifications.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     The panel's name is changed (not title)
 * For panel:  gui_3
 *
 * The following WorkBench operations will also cause regeneration:
 *     An item is deleted
 *     A new item is added to this panel
 *     An item's name is changed (not title)
 *     An item's data type is changed
 *     An item's generates events flag is changed
 *     An item's valids changed (if item is type string and connected)
 *     An item's connection information is changed
 * For the panel items:
 *   enter_new_plan,  get_re_30,       get_st_27,       get_user_in_21,
 *   get_x,           get_y
 *
 * CHANGE LOG:
 * MERGE NOTE: Add Change Log entries BELOW this line.
 * 2-Nov-98     Initially generated...TAE
 * MERGE NOTE: Add Change Log entries ABOVE this line.
 * ************************************************************************
 */

#include        "taeconf.inp"
```

```
#include        "wptinc.inp"
#include        "warrior_global.h"                  /* Application globals */
#include        "warrior_pan_gui_3.h"

/* One "include" for each connected panel */

/* MERGE NOTE: Add includes, vars, and functions BELOW this line. */
/* MERGE NOTE: Add includes, vars, and functions ABOVE this line. */

Id gui_3Target, gui_3View, gui_3Id;
/* gui_3Dispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID gui_3_Initialize_Panel (vmCollection)
    Id vmCollection;
    {
    gui_3View = Co_Find (vmCollection, "gui_3_v");
    gui_3Target = Co_Find (vmCollection, "gui_3_t");
    }


/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID gui_3_Create_Panel (relativeWindow, flags)
    Window      relativeWindow;
    COUNT       flags;
    {
    /* MERGE NOTE: Add code BELOW this line for gui_3_Create_Panel. */
    /* MERGE NOTE: Add code ABOVE this line for gui_3_Create_Panel. */

    if (gui_3Id)
        printf ("Panel (gui_3) is already displayed.\n");
    else
        gui_3Id = Wpt_NewPanel (Default_Display, gui_3Target, gui_3View,
                                relativeWindow, gui_3Dispatch, flags);
    }


/* ************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID gui_3_Destroy_Panel ()
    {
    /* MERGE NOTE: Add code BELOW this line for gui_3_Destroy_Panel. */
    /* MERGE NOTE: Add code ABOVE this line for gui_3_Destroy_Panel. */

    Wpt_PanelErase(gui_3Id);
    gui_3Id=0;
    }


/* ************************************************************************
 * Connect to this panel.  Create it or change its state.
 */
FUNCTION VOID gui_3_Connect_Panel (relativeWindow, flags)
    Window      relativeWindow;
    COUNT       flags;
    {
    /* MERGE NOTE: Add code BELOW this line for gui_3_Connect_Panel. */
```

```
    /* MERGE NOTE: Add code ABOVE this line for gui_3_Connect_Panel. */

    if (gui_3Id)
        Wpt_SetPanelState (gui_3Id, flags);
    else
        gui_3_Create_Panel (relativeWindow, flags);
    }


/* ************************************************************************
 * Handle event from parameter:  enter_new_plan
 */

EVENT_HANDLER enter_new_plan_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: enter_new_plan */
    /* MERGE NOTE: Add code BELOW this line for parm: enter_new_plan. */
    /* MERGE NOTE: Add code ABOVE this line for parm: enter_new_plan. */


    set_new_plan();

/*#
    printf ("Panel gui_3, parm enter_new_plan: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: enter_new_plan */


/* ************************************************************************
 * Handle event from parameter:  get_re_30
 */
EVENT_HANDLER get_re_30_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_re_30 */
    /* MERGE NOTE: Add code BELOW this line for parm: get_re_30. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_re_30. */

    set_replay_request();


/*#
    printf ("Panel gui_3, parm get_re_30: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: get_re_30 */


/* ************************************************************************
 * Handle event from parameter:  get_st_27
 */
EVENT_HANDLER get_st_27_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_st_27 */
    /* MERGE NOTE: Add code BELOW this line for parm: get_st_27. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_st_27. */

    set_statistics_request();

/*#
```

```
    printf ("Panel gui_3, parm get_st_27: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: get_st_27 */


/* ************************************************************************
 * Handle event from parameter:  get_user_in_21
 */
EVENT_HANDLER get_user_in_21_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_user_in_21 */
    /* MERGE NOTE: Add code BELOW this line for parm: get_user_in_21. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_user_in_21. */

    set_user_interaction();

/*#
    printf ("Panel gui_3, parm get_user_in_21: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: get_user_in_21 */


/* ************************************************************************
 * Handle event from parameter:  get_x
 */
EVENT_HANDLER get_x_Event (value, count)
    TAEFLOAT    value[];                    /* real vector     */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_x */
    /* MERGE NOTE: Add code BELOW this line for parm: get_x. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_x. */

    set_x((double)value[0]);

/*#
    printf ("Panel gui_3, parm get_x: value = %f\n",
            count > 0 ? value[0] : 0);
#*/
    }                                       /* parm: get_x */


/* ************************************************************************
 * Handle event from parameter:  get_y
 */
EVENT_HANDLER get_y_Event (value, count)
    TAEFLOAT    value[];                    /* real vector     */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_y */
    /* MERGE NOTE: Add code BELOW this line for parm: get_y. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_y. */

    set_y((double)value[0]);

/*#
    printf ("Panel gui_3, parm get_y: value = %f\n",
            count > 0 ? value[0] : 0);
#*/
    }                                       /* parm: get_y */
```

```
struct DISPATCH gui_3Dispatch[] = {
    {"enter_new_plan", enter_new_plan_Event},
    {"get_re_30", get_re_30_Event},
    {"get_st_27", get_st_27_Event},
    {"get_user_in_21", get_user_in_21_Event},
    {"get_x", get_x_Event},
    {"get_y", get_y_Event},
    {NULL, NULL}                              /* terminator entry */
};


/* MERGE NOTE: Add additional functions BELOW this line. */
/*# MTS 10-15-98
    added the following routines to display info to gui
#*/
FUNCTION VOID display_fuel_consumption(c)
double c;
{
        Wpt_SetReal(gui_3Id, "display_st_31", (TAEFLOAT)c);

}

FUNCTION VOID display_xloc(x)
double x;
{
        Wpt_SetReal(gui_3Id, "xloc", (TAEFLOAT)x);

}

FUNCTION VOID display_yloc(y)
double y;
{
        Wpt_SetReal(gui_3Id, "yloc", (TAEFLOAT)y);

}

FUNCTION VOID display_mover(x, y)
double x, y;
{
        TAEFLOAT value[2];
        value[0] = (TAEFLOAT)x;
        value[1] = (TAEFLOAT)y;
        Vm_SetReal (gui_3Target, "display_re_37", 2, value, P_UPDATE);
        Wpt_ParmUpdate (gui_3Id, "display_re_37");

}

FUNCTION VOID end_simulation()
{
       Wpt_PanelErase(gui_3Id);
       Wpt_Finish();
        SET_APPLICATION_DONE;
}



/* MERGE NOTE: Add additional functions ABOVE this line. */

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                          *** */
/* *** mode:                          c          *** */
/* *** c-indent-level:                0          *** */
/* *** c-continued-statement-offset:  4          *** */
```

```
/* *** c-brace-offset:                 4       *** */
/* *** c-brace-imaginary-offset:       4       *** */
/* *** c-argdecl-indent:               4       *** */
/* *** c-label-offset:                 -4      *** */
/* *** c-continued-brace-offset:       -4      *** */
/* *** comment-column:                 45      *** */
/* *** comment-multi-line:             nil     *** */
/* *** End:                                    *** */
```

95. warrior_tae.c

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        warrior.c *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * This the main program of an application generated by the TAE Plus Code
 * Generator.
 *
 * REGENERATED:
 * This file is generated only once.  Therefore, you may modify it without
 * impacting automatic code merge.
 *
 * NOTES:
 * To turn this into a real application, do the following:
 *
 * 1.  Each panel that has event generating parameters is encapsulated by
 * a separate file, named by concatenating the string "pan_" with the
 * panel name (followed by a ".c").  Each parameter that you have defined
 * to be "event-generating", has an event handler procedure in the
 * appropriate panel file.  Each handler has a name that is a
 * concatenation of the parameter name and the string "_Event".  Add
 * application-dependent logic to each event handler.  (As generated by
 * the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * 2.  To build the program, type "make".  If the symbols TAEINC, ...,
 * are not defined, the TAE shell (source) scripts $TAE/bin/csh/taesetup
 * will define them.
 *
 * ADDITIONAL NOTES:
 * 1.  Each event handler has two arguments: (a) the value vector
 * associated with the parameter and (b) the number of components.  Note
 * that for scalar values, we pass the value as if it were a vector with
 * count 1.
 *
 * Though it's unlikely that you are interested in the value of a button
 * event parameter, the values are always passed to the event handler for
 * consistency.
 *
 * 2.  You gain access to non-event parameters by calling the Vm package
 * using the targetId Vm objects that are created in
 * Initialize_All_Panels.  There are macros defined in global.h to assist
 * in accessing values in Vm objects.
 *
 * To access panel Id, target, and view, of other panels, add an
 * "#include" statement for each appropriate panel header file.
 *
 * CHANGE LOG:
```

```
 * 15-Oct-98    Initially generated...TAE
 * ************************************************************************
 */

#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "symtab.inc"
#include        "warrior_global.h"                  /* Application globals */
#include        "warrior_pan_gui_3.h"                  /* Application globals
*/


/*      Globally defined variables */

Display *Default_Display;
WptEvent wptEvent;  /* event structure returned by Wpt_NextEvent */
BOOL    Application_Done;

/*# MTS 10-15-98
    replace main routine by initialize_gui and generated_tae_event_monitor

main (argc, argv)

    FUNINT      argc;
    TEXT        *argv[];


    {

#*/
    CODE        eventType;

    COUNT       termLines, termCols;
    CODE        termType;


    /* PROGRAMMER NOTE:
     * add similar extern's for each resource file in this application
     */

    extern VOID warrior_Initialize_All_Panels ();
    extern VOID warrior_Create_Initial_Panels ();

    struct DISPATCH             *dp;        /* working dispatch pointer */
    struct VARIABLE             *parmv;     /* pointer to event VARIABLE */


/*# MTS 10-15-98
    add the statement void initialize_gui()
#*/

void initialize_gui()
{
    /* initialize terminal without clearing screen */
    t_pinit (&termLines, &termCols, &termType);

    /* permit upper/lowercase file names */
    f_force_lower (FALSE);

    Default_Display = Wpt_Init (NULL);

    /* PROGRAMMER NOTE:
     * To enable scripting, uncomment the following line.  See the
```

```
     * taerecord man page.
     */
    /* Wpt_ScriptInit ("warrior"); */

    /* initialize resource file */
    /* PROGRAMMER NOTE:
     * For each resource file in this application, calls to the appropriate
     * Initialize_All_Panels and Create_Initial_Panels must be added.
     */
    warrior_Initialize_All_Panels ("warrior.res");
    warrior_Create_Initial_Panels ();

/*# MTS 10-15-98
    add the following initialization here
#*/
    Application_Done = FALSE;

}


/*# MTS 10-15-98
    commented out the loop and
    add the statment generated_tae_event_monitor()


    /#* main event loop *#/
    /#* PROGRAMMER NOTE:
     #* use SET_APPLICATION_DONE in "quit" event handler to exit loop.
     #* (SET_APPLICATION_DONE is defined in global.h)
     *#/
    while (!Application_Done)

#*/
void generated_tae_event_monitor()
   {
   if (Wpt_Pending())
        {
        eventType = Wpt_NextEvent (&wptEvent); /* get next WPT event */

        switch (eventType)
            {
        case WPT_PARM_EVENT:

            /* Event has occurred from a Panel Parm.  Lookup the event
             * in the dispatch table and call the associated event
             * handler function.
             */

            dp = (struct DISPATCH *) wptEvent.p_userContext;
            for (;  (*dp).parmName != NULL;   dp++)
                if (s_equal ((*dp).parmName, wptEvent.parmName))
                    {
                    parmv = Vm_Find (wptEvent.p_dataVm, wptEvent.parmName);
                    (*(*dp).eventFunction)
                        ((*parmv).v_cvp, (*parmv).v_count);
                    break;
                    }
            break;

        case WPT_FILE_EVENT:

            /* PROGRAMMER NOTE:
             * Add code here to handle file events.
```

```
             * Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
             * event sources.
             */
            printf ("No EVENT_HANDLER for event from external source.\n");
            break;

        case WPT_WINDOW_EVENT:

            /* PROGRAMMER NOTE:
             * Add code here to handle window events.
             * WPT_WINDOW_EVENT can be caused by windows which you directly
             * create with X (not TAE panels), or by user acknowledgement
             * of a Wpt_PanelMessage (therefore no default print statement
             * is generated here).
             */
            break;

        case WPT_TIMEOUT_EVENT:

            /* PROGRAMMER NOTE:
             * Add code here to handle timeout events.
             * Timeout events occur when an application has not received any
             * user input within the interval specified by Wpt_SetTimeOut.
             */
            printf ("No EVENT_HANDLER for timeout event.\n");
            break;

        case WPT_TIMER_EVENT:

            /* PROGRAMMER NOTE:
             * Add code here to handle timer events.
             * Timer events occur on (or after) the interval specified when the
             * event is registered using Wpt_AddTimer.  Use Wpt_RemoveTimer to
             * remove timers.
             */
            printf ("No EVENT_HANDLER for event from timer source.\n");
            break;

        default:

            printf("Unknown WPT Event\n");
            break;
            }
        }
    else if (Application_Done)
        {


        Wpt_Finish();  /* close down all display connections */

        }

    }                                       /* end main */

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                          *** */
/* *** mode:                          c          *** */
/* *** c-indent-level:                0          *** */
/* *** c-continued-statement-offset:  4          *** */
/* *** c-brace-offset:                4          *** */
/* *** c-brace-imaginary-offset:      4          *** */
/* *** c-argdecl-indent:              4          *** */
/* *** c-label-offset:                -4         *** */
```

```
/* *** c-continued-brace-offset:        -4      *** */
/* *** comment-column:                  45      *** */
/* *** comment-multi-line:              nil     *** */
/* *** End:                                     *** */
```

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5100	2
3.	Research Office, Code 09 Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Dr. David Hislop U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	1
5.	LTC Michael McGinnes TRAC-Monterey PO Box 8692 Monterey, CA 93943	1
6.	Dr. Man-Tak Shing, CS/Sh Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
7.	Dr. Valdis Berzins, CS/Be Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
8.	Dr. Luqi, CS/Lq Computer Science Department Naval Postgraduate School Monterey, CA 93943	6