
ASSET Documentation Set

Ian Mayo, PlanetMayo Ltd. <info@planetmayo.co.uk>

Abstract

The ASSET Documentation Set provides a full introduction to the ASSET Framework, useful to users, developers and maintainers. The individual books within the set are self-contained, but online versions do contain occasional inter-book links.

ASSET Overview

ASSET Overview

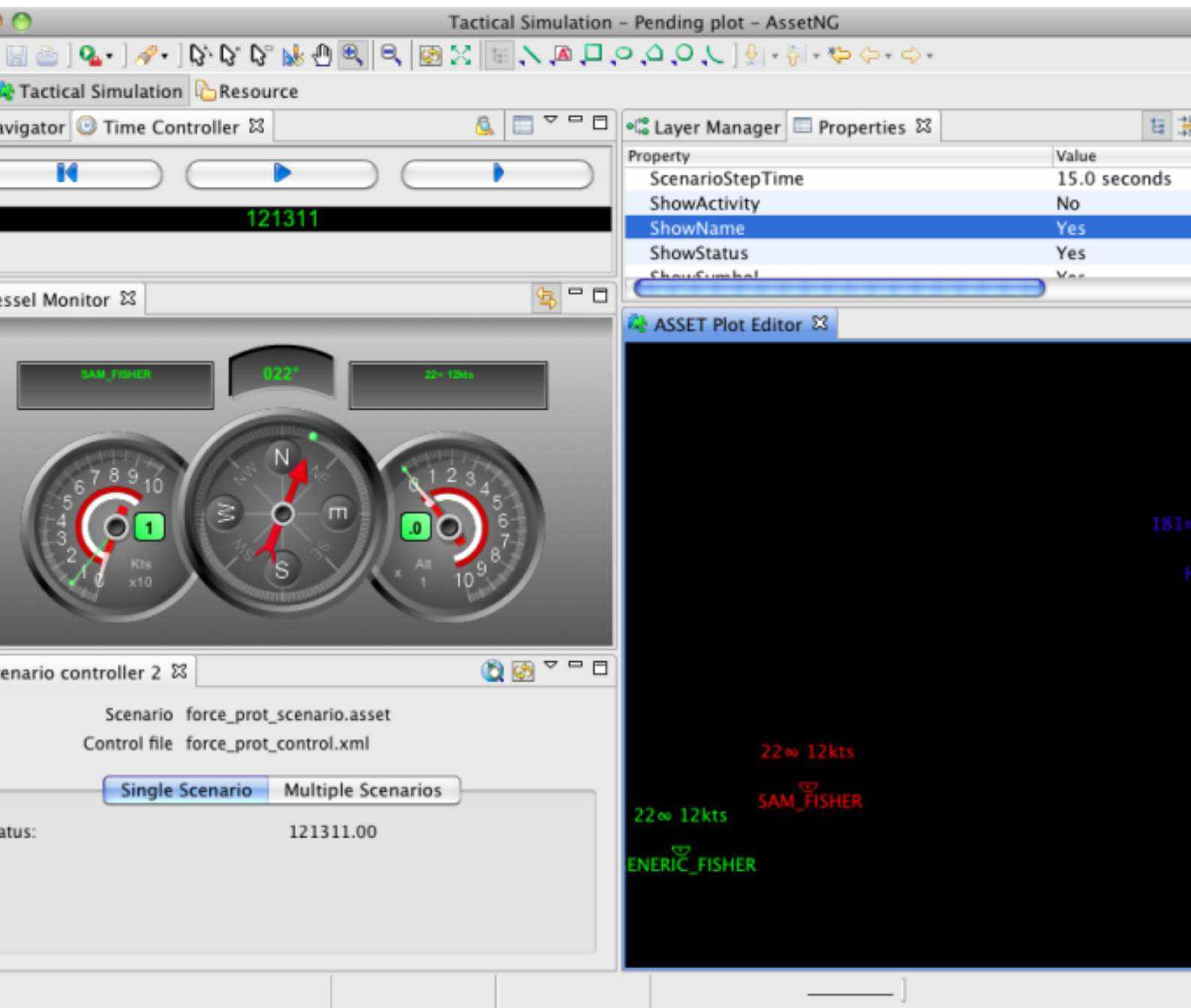
Table of Contents

An introduction to ASSET	1
1. Breadth of modelling	2
2. Integrated Analysis	6
3. Modular construction	8
4. Modern, open, standards	10

An introduction to ASSET

This document provides an overview to the ASSET maritime tactical modelling environment.

Figure 1. Screenshot of ASSET Workbench front-end



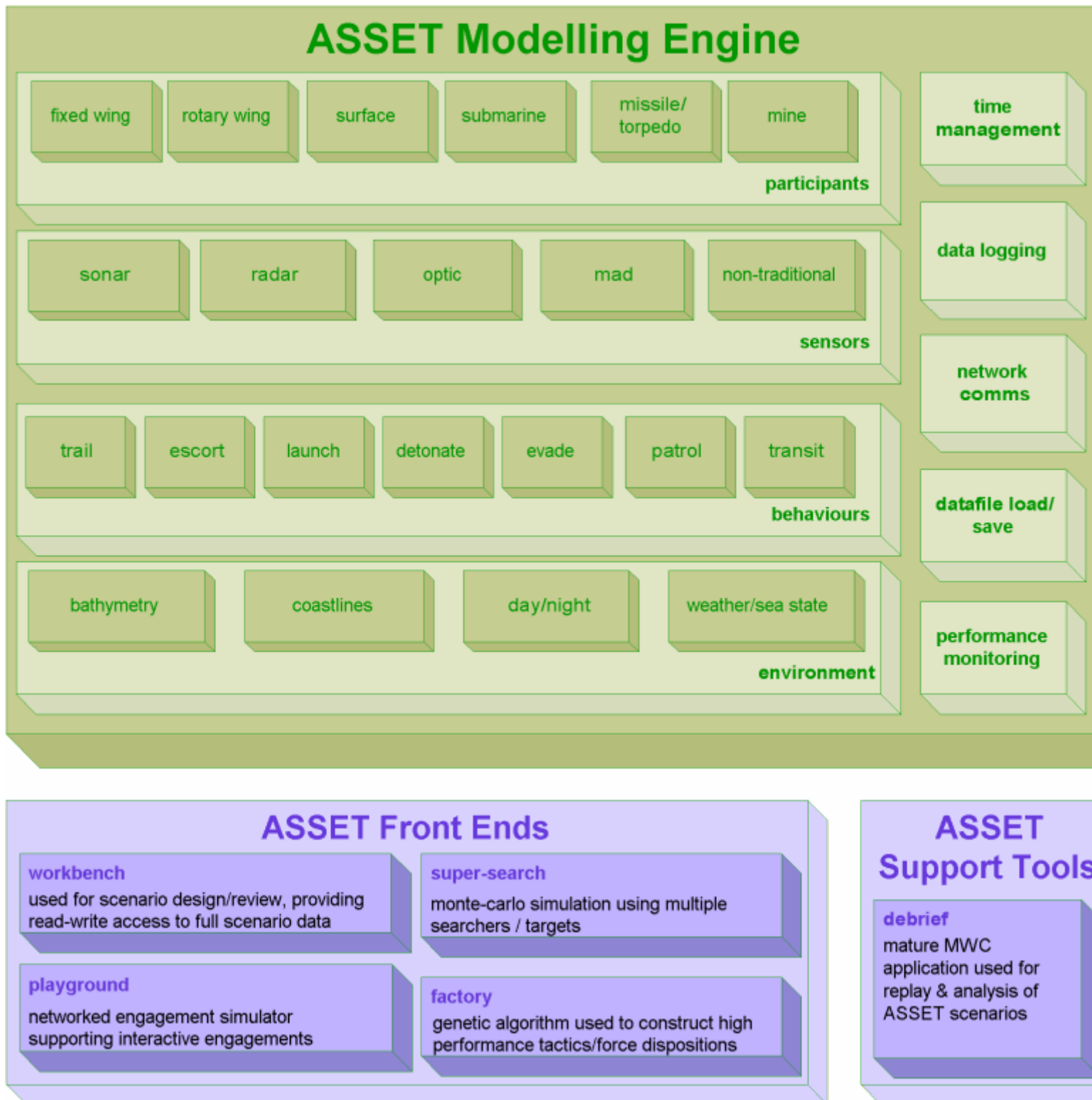
In typical use ASSET presents the user with a 2-d view of a tactical engagement and provides drill-down into current participant status, the behaviours of participants and views of sensor detections. The user is able to enable logging of data, copy screenshots to disk for insertion into reports, or even open a 3-d view of events as they unfold.

Through drag and drop the user is able to load new scenarios, scenario participants, sensors and behaviours.

1. Breadth of modelling

ASSET comprises a highly optimised modular modelling engine with a number of graphical front-ends tailored to specific approaches in the solution of tactical problems. The front-ends used by ASSET allow computer-controlled, human-in-the-loop and composite engagements to be investigated by scientific and non-scientific users, providing outputs for subsequent After Action Review.

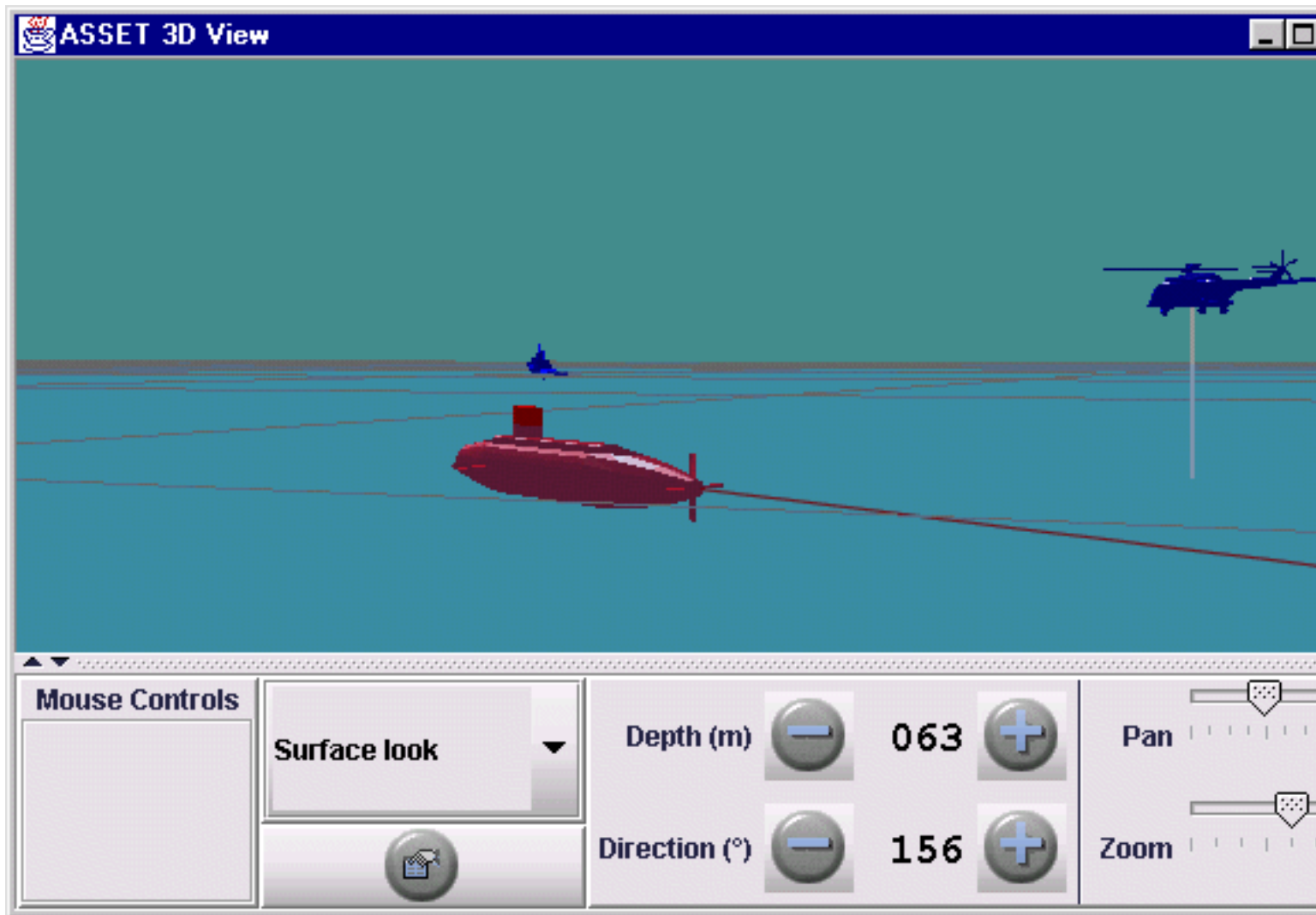
Figure 2. Overview of ASSET capabilities



Not only does ASSET have strong modelling capabilities, it also has a strong visualisation capability. The earlier screenshot shows a typical ASSET 2 dimensional view of a scenario unfolding (a helo investigating two contacts of interest) using recognised NATO symbology. Where the 2 dimensional view is insufficient then a 3 dimensional dynamic view can be opened. The following screenshot shows a friendly helo about to prosecute a hostile submarine, itself attempting to prosecute a friendly warship. The 3 dimensional view has a range of options to improve legibility.

These include magnifying the size of the participants themselves (as in the screenshot), configurable grid-lines, ocean shading, and snail trails.

Figure 3. Demonstration of ASSET 3d visualisation



ASSET has been built from the ground up to provide tactical modelling facilities to support the needs of maritime tactical development. Fundamental to this strategy has been the identification of an acceptable fidelity in modelling, where ASSET can be used in initial investigations into new tactical problems - acknowledging that it may be necessary to employ a higher fidelity model in the later stages of analysis, and that frequently the only true test for a tactic is real-world employment. Nevertheless, ASSET can be employed to great benefit in the following methods of analysis:

- *Quantitative analysis.* The direct comparison of different tactics allows for sensitivity analysis of different characteristics such as initial manoeuvres, evasion headings/speeds and protective screen dispositions. Different engagements can be compared through analysis of time-to-detection, closest point of approach, number of kills or a wide variety of other user-defined performance measures.
- *Collaborative brainstorming.* Through ASSET users are able to visualise ideas for new tactics, recording the engagement for replay by peers or interactively modifying a particular tactic to make rapid progress whilst brainstorming.
- *Mission rehearsal.* It is no longer necessary to wait for a sea trial before progressing an area of tactical development. Trial tactics can be synthetically modelled in ASSET, discarding

poorly performing tactics to concentrate sparse sea-time on optimally performing solutions. Additionally, an exercise-order can be rehearsed in-house prior to being sent to sea to test for completeness and highlight areas where greater detail is needed or remove exercise serials which will clearly not progress the problem under investigation.

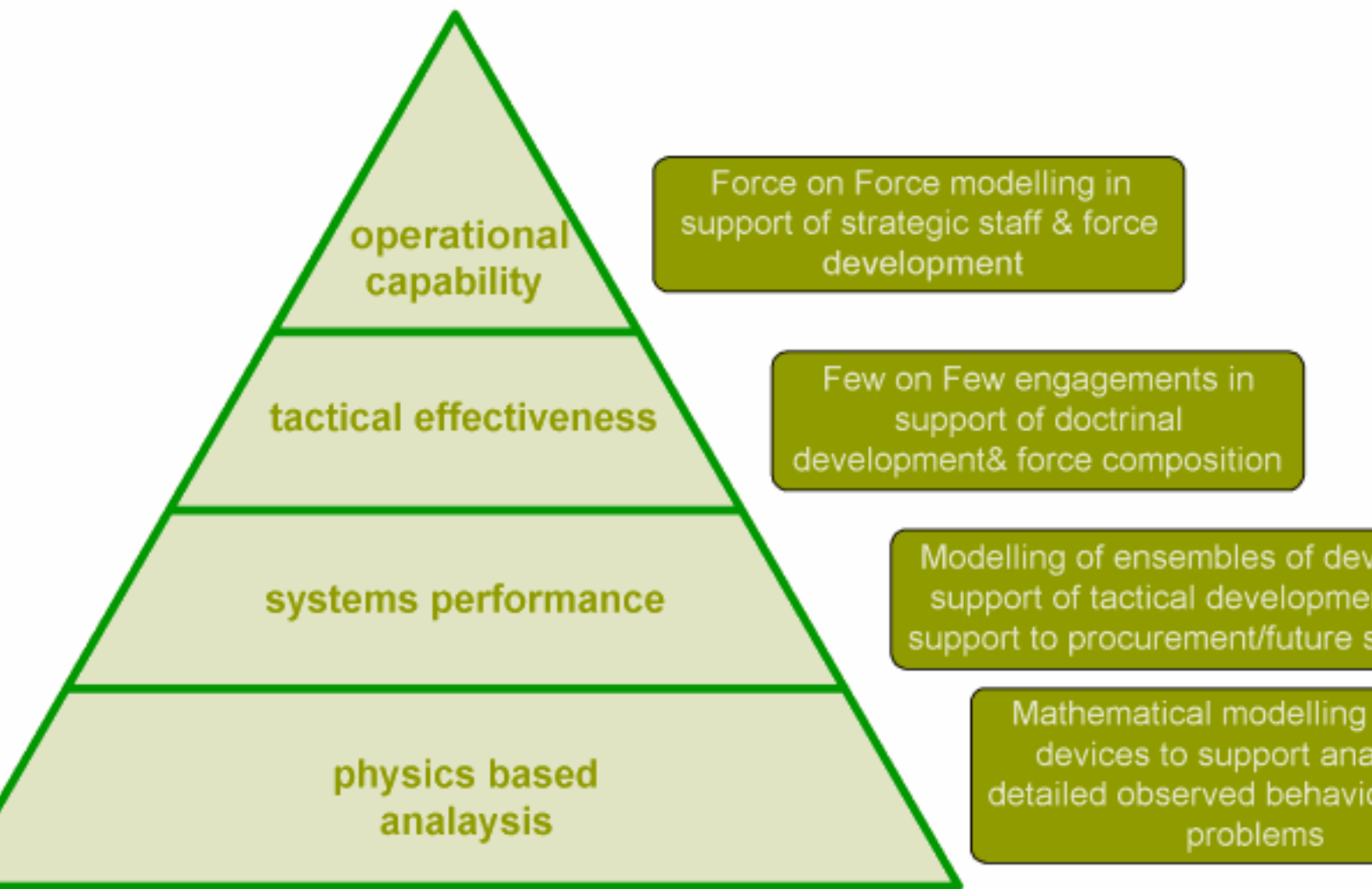
- *After action review.* In post-exercise review ASSET can be used to recreate tactical engagements with different environmental conditions, rules of engagement, or sensor performance, leading to a greater understanding of exercise events and improving the quality of analysis.

The adoption of ASSET can lead to advances in the following:

- *Operational capability.* ASSET provides a quick turnaround, low cost means of trialling new ways of conducting joint warfare. Expert users can experiment collaboratively to trial new force capabilities, quickly determining strong and weak points in command and control structures or platform capabilities. Lessons learnt from ASSET can leverage effort invested in sea-trials to quickly produce new or improved operational capabilities at short notice.
- *Operational performance.* The operational performance of existing systems can be improved through the development of more efficient tactics and procedures, or through the use of ASSET components in tactical decision aids.
- *Exercise preparation.* ASSET can be used in exercise preparation to validate speed-time transit calculations, verify that manoeuvres are achievable, and highlight to participants areas which need particularly detailed record-keeping.
- *Mission and threat analysis.* The repeated simulation of particular scenarios with slight variations allows for sensitivity analysis to be performed, leading to a greater understanding of the role of particular participants and rules of engagement. This leads to a greater understanding of the mission/threat itself and can be used to leverage the existing analyst's knowledge towards more thorough tactical development.
- *Performance prediction.* Once the agreed characteristics of a future sensor/weapon/platform have been defined, ASSET can model these equipments and provide performance predictions to allow for early recognition of potential advantages or shortcomings possibly years ahead of their use at sea. When the equipments are first available for use effort can be efficiently concentrated on specific areas highlighted by ASSET.

Additionally, ASSET is capable of modelling maritime engagements at a number of levels from a Force on Force engagement (such as an RN Task Force breaking through a blockade to rescue commonwealth dependants), down through one-on-one engagements (such as optimal firing position of helo on FAC) to single system analysis (to determine individual sensor performance).

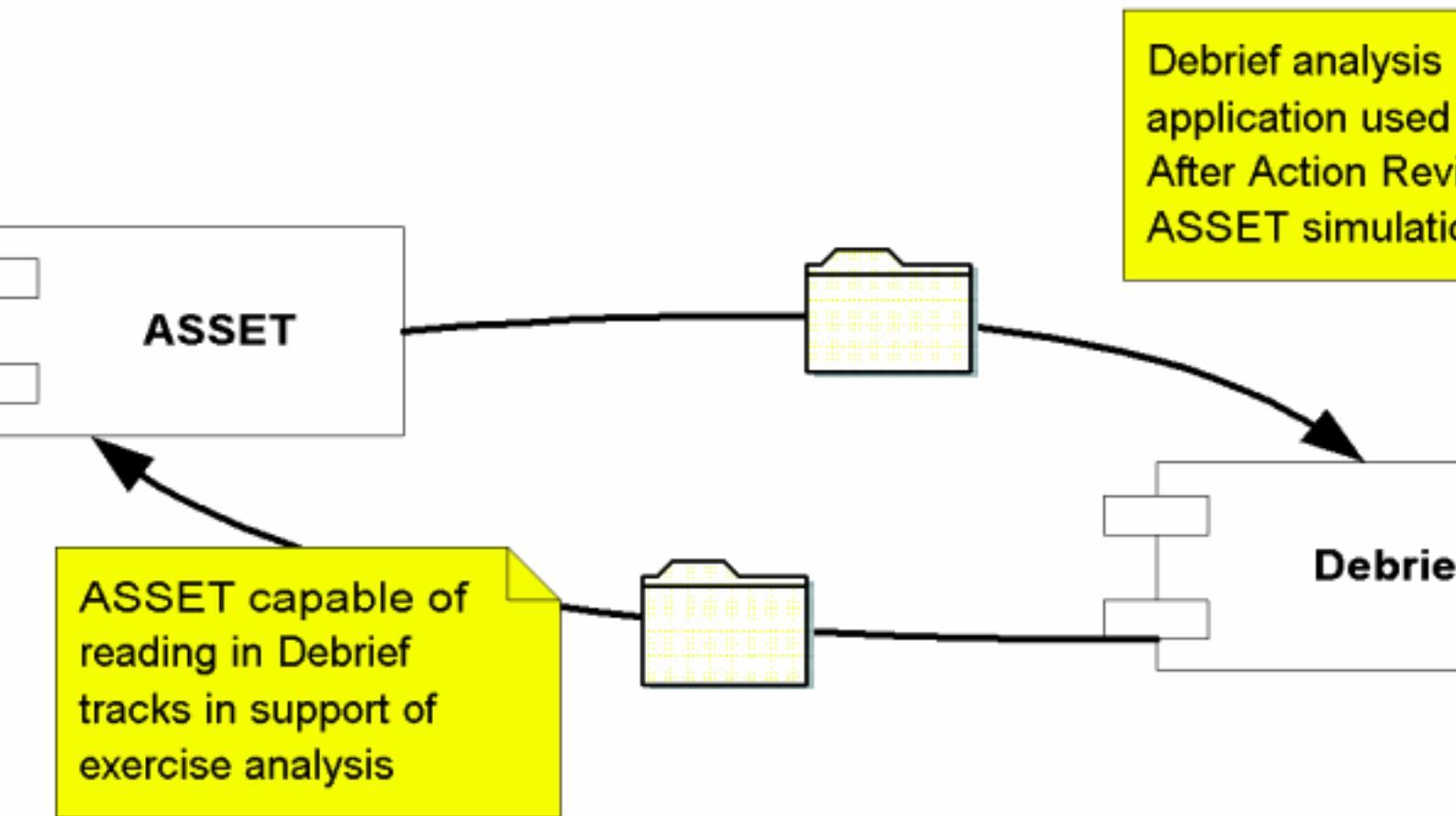
Figure 4. Range of modelling supported by ASSET



2. Integrated Analysis

Obviously, the complete analysis of a problem requires the analyst to be able to replay ASSET engagements. To this end, the ASSET integrated analysis suite includes the Debrief analysis tool, already in use at the Maritime Warfare Centre, COMSUBDEVRON 12, VX-1, Canadian Forces MWC, etc.

Figure 5. Integrated tactical development using ASSET and Debrief

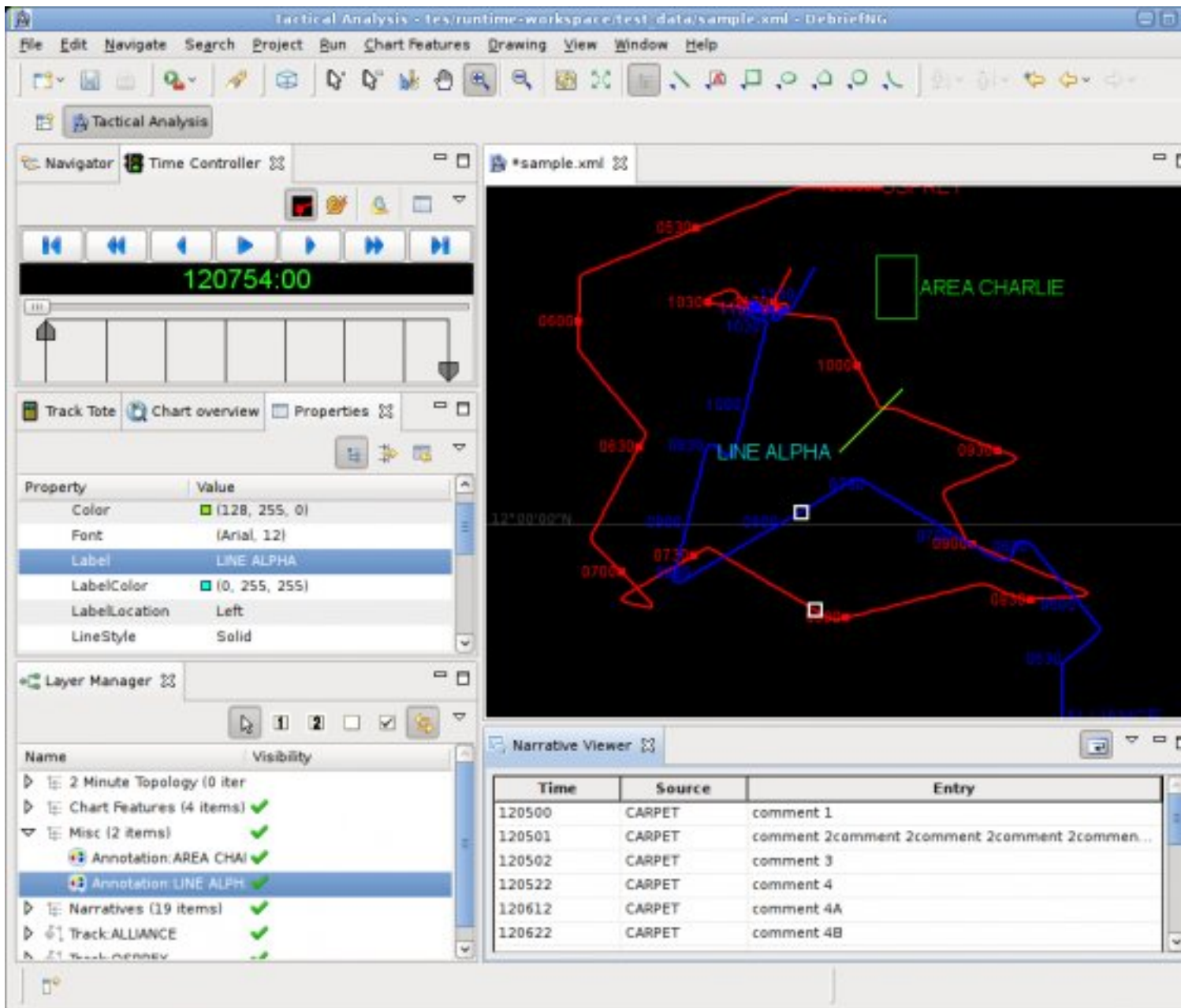


This integration is achieved through the use of agreed file formats:

- ASSET is capable of exporting data in Debrief's .REP file format, so allowing in-depth analysis of an engagement to be conducted with a tool which is both already familiar to many MWC personnel, and one of the best tools in its field
- Additionally, ASSET is capable of reading in the Debrief REP format, allowing an analyst to load a "real-world" vessel track into a scenario and determine for example, its search performance against a much larger body of targets.

The Debrief analysis tool is available online [www.debrief.info], and provides extensive functionality for the analysis of maritime engagements, supported by the following features:

- 3-d view of engagement
- Graphing support of time-variables, with export features to copy data to MS Excel for more versatile plotting.
- Global coastline and bathymetric support
- Mature plot export support, for subsequent insertion into reports/presentations.

Figure 6. The Debrief analysis tool

3. Modular construction

3.1. Modular modelling

The modular modelling mechanisms contained within ASSET allow the engine to conduct modelling at a range of resolutions - all configured dynamically at run-time, not build-time. For example, the modular modelling allows the user to elect to model an acoustic sensor using cookie-cutter detections or full sonar equations by specifying his requirement in the ASSET scenario file. Additionally, varying levels of complexity can be combined to suit the analysis problem at hand - the designer getting support from informed guidance in the ASSET modelling handbook.

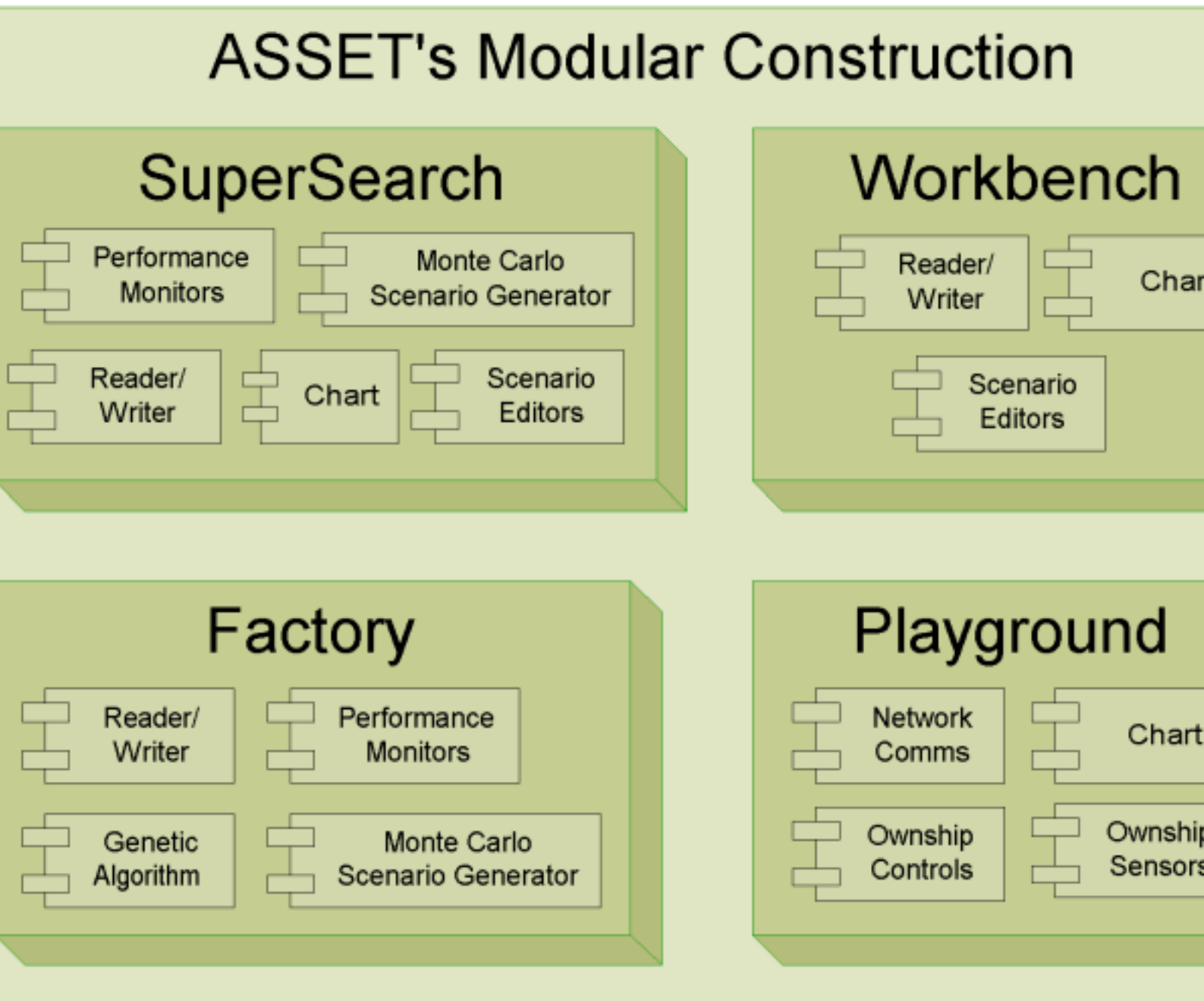
Despite the wide range of modelling modules in ASSET, they remain "thin" software components - that is, they rely to a great extent on the parameters passed to them from datafiles. This inherent reliance on datafiles provides a great degree of flexibility to the software - and as before, this is determined at run-time by the user not at build-time by the developer.

3.2. Modular applications

ASSET is build from a variety of high and low level components. Fundamentally it uses COTS graphical user interface (GUI) components taken from the Maritime Warfare Centre's Debrief application. Added to these components are further modules to handle file read/write, time management, and networking. Assembling different combinations of these modules provides a number of applications supporting various analysis/tactical development requirements, from non-graphic plain text applications (used for monitoring number crunching of large numbers of scenarios), to detailed graphic environments aimed to replicate onboard displays (so adding to the realism of the simulation whilst reducing the learning curve for understanding the results displayed).

During the normal course of events new analysis requirements will normally be met by producing new ASSET scenarios and for the required ASSET front-end. Exceptionally however, a new application may be required. The ASSET modules can quickly be re-combined using the ASSET skeleton provided, thus adding a new application to the ASSET toolset.

Figure 7. The modular construction of ASSET



4. Modern, open, standards

Under the covers, the ASSET modelling environment is an example of a software development taking full advantage of modern development standards and working practices. The adoption of modern practices has facilitated the rapid development maturing of the ASSET software, and reduced documentation and maintenance costs.

4.1. ASSET File Format

XML (eXtensible Modelling Language) is a text-file format which can be used to represent a wide variety of types of data in support of a wide variety of applications. The XML file format was first trialled by the Maritime Warfare Centre in support of its Debrief application, since then adopted across the world by a number of analysis agencies/corporations. The XML format allows ASSET

to store data in a tightly defined structure only normally available in binary storage, where it would lose its human legibility.

Figure 8. Sample of ASSET file format

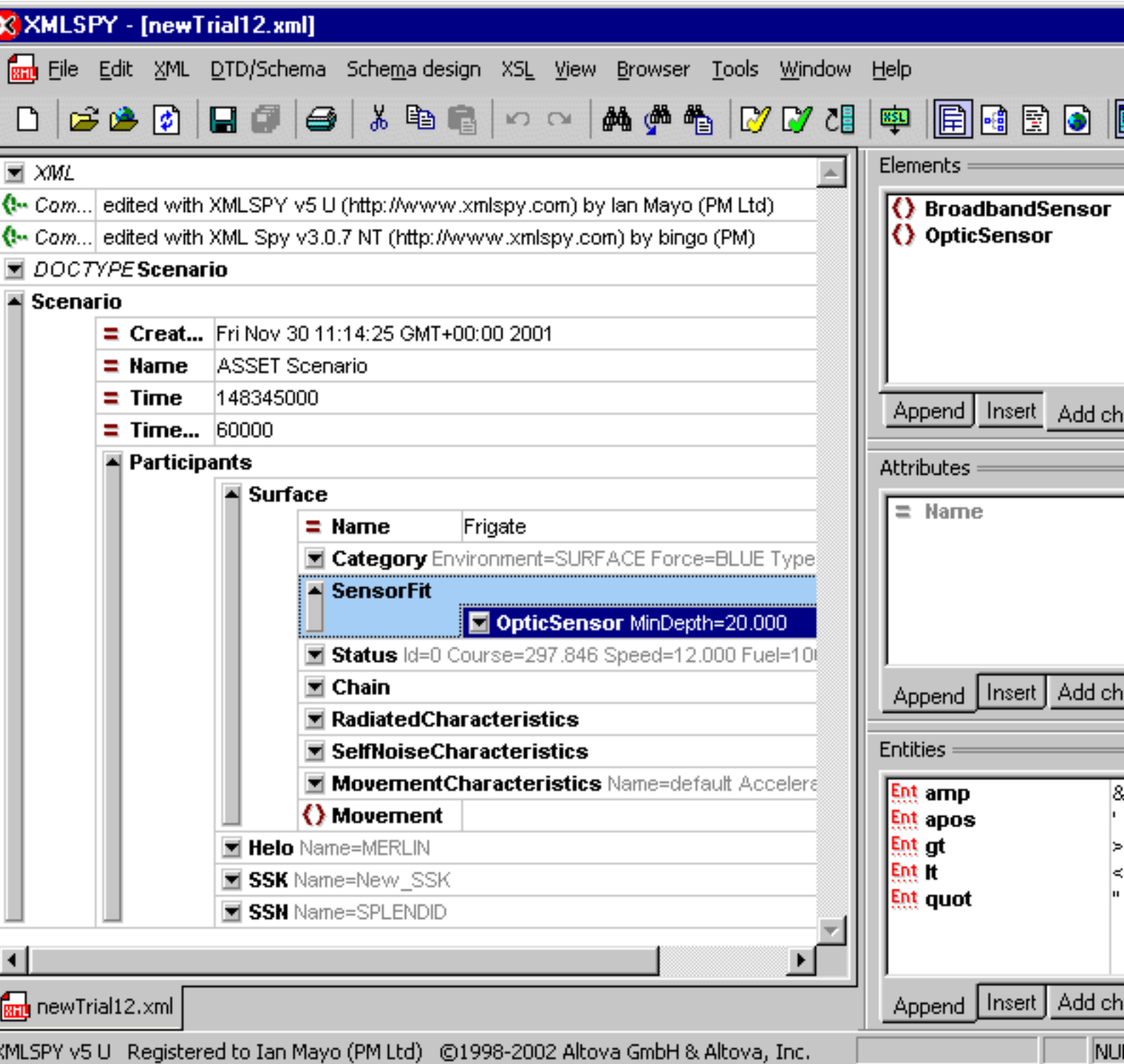
```
<Scenario Created="Thu May 17 15:07:28" Name="ASSET Scenario" Time="148345000"
  <Participants>
    <SSK Name="New_SSK">
      <Category Environment="SUBSURFACE" Force="RED" Type="SUBMARINE" />
      <SensorFit>
        <Broadband Aperture="120.000" />
        <Optic MinDepth="20.000" />
      </SensorFit>
      <Status Course="243.105" Speed="12.000" Fuel="92.800" Time="700102 17122"
        <Location>
          <shortLocation Lat="26.8605883" Long="53.3621488" Depth="40.000" />
        </Location>
      </Status>
      <DemandedStatus Course="3.105" Speed="12.000" Depth="40.000" />
      <Chain>
        <SSKRecharge Name="EmergencySnort" MinLevel="5.000" SafeLevel="20.000"
          SnortSpeed="4.000">
          <TargetType />
        </SSKRecharge>
        <Trail Name="Trail" TrailRange="2000.000" AllowableError="600.000">
          <TargetType>
            <Type Name="BLUE" />
            <Type Name="SURFACE" />
          </TargetType>
        </Trail>
      </Chain>
    </SSK>
  </Participants>
</Scenario>
```

The extensive adoption of XML in ASSET has provided the following opportunities:

- *Re-use of data.* The versatile and robust XML file format allows elements of ASSET data-files to be cut and pasted to form new scenarios, or placed under configuration management to form libraries of agreed performance/behaviour for friendly and threat platforms/sensors. This significantly reduces the traditionally expensive task of scenario generation/preparation, additionally allowing information to be passed to collaborating agencies for joint tactical development

- *Low cost development.* Many off-the-shelf libraries exist which support the read/write of XML data, ensuring that minimum time is invested in developing this necessary functionality. Successive software developments which need to read/write ASSET data-files are also able to exploit these free libraries.
- *Tool Support.* The world-wide adoption of XML as a standard file-format has resulted in a number of software tool suppliers providing dedicated XML editing applications, these applications have provided low-cost, high quality view and edit facilities for ASSET datafiles. The screenshot (below) shows an intelligent editor application inviting the user to add either a broadband or optic sensor (top-right) to the sensor fit of the frigate currently being edited (centre).

Figure 9. Sample of dedicated XML editor



4.2. Java programming language

The last three decades have seen an explosion in the variety and number of programming languages and environments available for software development. Sun Microsystems took the lessons learnt from previous languages such as Pascal, Fortran and c++ and produced the Java programming language. Very quickly after its release Java experienced wide industry adoption, and use for development of heavyweight applications. The adoption of Java within ASSET provides the following benefits:

- *Platform Independence* Through the use of Java, ASSET is able to run on a wide variety of operating systems, from standard desktop PC's to high performance Linux and Solaris workstations capable of number-crunching vast numbers tactical scenarios.
- *Object Orientation*. Tactical modelling inherently lends itself to object orientation; the representation of real-world objects as functional software units. For example, the object orientated support provided to ASSET by Java allows a software model of an SSK to quickly be produced taking the existing model of an SSN and adding battery usage, and snorting noise models. This significantly reduces both development and maintenance costs.
- *Rich Standard Library*. Java contains an extensive set of standard libraries, which have been exploited in ASSET to provide user interface, network communications, XML file storage, and 3 dimensional viewing. Use of standard libraries in addition of these areas of functionality to ASSET has reduced development costs, maintenance costs, and through-life costs (since they will never incur license charges).
- *Wide developer support*. The adoption of Java by industry, academic and research institutions world-wide significantly eases the problem of finding qualified development support - all recent computer science graduates will have a grounding in Java, and most will be fully competent in using it in developments such as ASSET. Thus an organisation making use of ASSET can be increasingly confident of finding suitably trained developmental effort.

ASSET Modelling Guide

Ian Mayo

ASSET Modelling Guide

Ian Mayo

Copyright © 2001,2002

The ASSET Modelling Guide

The ASSET Modelling guide gives details of the elements, characteristics and behaviours used inside the ASSET Engine. The modelling guide does not give any detail, nor does it refer to any front-ends through which the engine may be accessed. This information is contained in the ASSET User Guide

Table of Contents

1. Introduction	1
1. ASSET modelling philosophy	1
2. ASSET modelling sources	1
3. Overview	1
4. Model definition/approval process	1
5. Guidance to Authors	2
6. Modelling Template	4
7. Agreed Units	5
2. Scenario	6
1. Introduction	6
2. Characteristics	6
3. Structure	7
3. Environment	8
1. Introduction	8
2. General Characteristics	8
3. Oceanography/Meteorology	8
4. Mediums	8
4. Participants	12
1. Time cycle	12
2. Characteristics	15
3. Platform Types	18
4. Vessel Movement	20
5. Sensors	30
1. Lookup Sensor Model	30
6. Interaction	35
1. Detection	35
2. Communications	36
7. Behaviours	37
1. Introduction	37
2. State behaviours	37
3. Structural Behaviours	38
4. Transiting Behaviours	42
5. Composite Conditions	50
6. Composite Responses	52
7. Tactical Behaviours	52
8. Search Behaviours	59
9. High level tactical behaviours	64
10. Other behaviours	64
8. Scenario Control	66
1. Observers	66
2. Scenario Generation	70
9. Bibliography	75
1. ASSET Glossary	76
1. Glossary	76

Chapter 1. Introduction

1. ASSET modelling philosophy

ASSET has adopted a specific modelling philosophy which directly affects the resolution of modelling performed. The role of ASSET is seen as an advanced substitute for graph paper - used to look at vessel geometries and detections quickly, reliably and robustly.

ASSET does not aim to approach *real-world* levels of modelling, and there is no expectation of producing results which can replace tactical development undertaken at sea. ASSET does, however, aim to allow a user to run through a simplified version of the real world to gain a better understanding of a particular platform, sensor or tactic. Through the use of ASSET it is hoped that the Tactical Development process can be accelerated through the discarding of the low-value tactics allowing at-sea exercise time to focus on high-value tactics.

Of course the results produced by ASSET have to be relatively realistic, so vessels manoeuvre using specific turning circles and acceleration rates, sensors have specific recognition differentials, and energy travelling through the environment experiences propagation loss.



Important

The ASSET user should have a full understanding of the modelling abstraction used in a specific scenario, ASSET is not attempting to reproduce a "real-world" equivalent, and the user must decide what assumptions can be made of the results. A thorough understanding of the ASSET Modelling Guide is required before any analysis of results is conducted.

2. ASSET modelling sources

Large areas of the modelling used within ASSET directly follow that recommended in the two volumes of the *Sonar Modelling Handbook*, September 1998, DERA/S&P/UWS/CR980073 produced by DERA, *Principles of underwater sound* written by Robert J Urlick, 1983, and the Meteorologist's Glossary, [REF??@@]

3. Overview

Information in this modelling guide is structured according to the entity being modelled, scenario, environment, vessels, sensors and behaviours. This results in some related information being spread out through the guide; in particular the components of the sonar equations are spread out through several section. The following list should help:

Radiated Noise	Radiated Noise is covered in the Participants characteristics section, at: Section 2.4, "Radiated Noise" [16]
Self Noise	Self Noise is also covered in the Participants characteristics section, at: Section 2.5, "Self Noise" [18]
Propagation Loss	Propagation Loss is covered in the Environment section, at: Section 4, "Mediums" [8]
Detection Threshold	Detection Threshold is covered in the Sensors section, at: Chapter 5, <i>Sensors</i> [30]

4. Model definition/approval process

The following process will be used for the creation/incorporation of models into ASSET

1. Model requirement established either from ASSET analysis document or by domain expert
2. Algorithm researched by domain expert
3. Algorithm documented in Microsoft Word according to structure defined in Modelling Template
4. Author submits document to ASSET maintainer, providing amplification where necessary
5. Domain expert agrees applicable level of review with ASSET maintainer according to choices in note



Varying levels of algorithm review

- a. Review by ASSET maintainer
 - b. Review by other MWC resource as recommended by author
 - c. Review by ad-hoc group formed within MWC
 - d. Review by other MoD agency (such as DSTL, ADAC)
 - e. It is hoped that no ASSET algorithms will require external industry review. This requirement will addressed if it arises.
6. Review conducted
 7. ASSET maintainer incorporates algorithm in modelling guide
 8. Schema amended to support new model, reviewed by domain expert
 9. Software algorithms implemented
 10. Software test cases proven
 11. Software verification recorded (if necessary)

5. Guidance to Authors

This section provides guidance to authors of entity algorithm authors. The authoring of algorithms is not part of a specific phase of the ASSET development (as recorded in the ASSET Roadmap [<http://intranet2/coag/asset/docs/ASSETRoadmap.pdf>]), but is a continuing process with algorithms implemented as they are defined. In this document, information structures that form part of the ASSET modelling engine are provided with a hyperlink to the relevant point in the ASSET system documentation [<http://intranet2/coag/asset/api/index.html>].

The following types of vessel-related entity are modelled within ASSET:

- Vessels (including movement, fuel usage and radiated noise)
- Sensors
- Behaviours

The algorithm skeletons for these types of algorithm are detailed on the following pages. There may be requirements for new inputs and outputs in support of particular algorithms, which will be dealt with on a case by case basis, hopefully with decreasing frequency.

Information is passed between the vessel-related algorithms as described in the Section 1, “Time cycle” [12] - essential reading for a model author.

5.1. Vessel movement

The movement algorithm receives the following inputs:

Current vessel status [<http://intranet2/coag/asset/api/ASSET/Participants/Status.html>]

- Location : lat (degs), long (degs), depth (m)
- Course - degs
- Speed - m/sec
- Fuel level - % of capacity

Vessel movement characteristics [<http://intranet2/coag/asset/api/ASSET/Models/Movement/MovementCharacteristics.html>]

- Acceleration rate (m/sec)
- Dive rate (m/s)
- Climb rate (m/s)
- Fuel usage rate (units/m/sec)
- Maximum depth (m)
- Minimum depth (m)
- Maximum speed (m/sec)
- Minimum speed (m/sec)
- Turning circle (m), turn rate (deg/sec), or G-pulled

Using this information the movement algorithm produces an updated Status [<http://intranet2/coag/asset/api/ASSET/Participants/Status.html>] object. Vessel behaviour

5.2. Decision behaviour

The decision behaviour receives the following inputs:

Current vessel Status [<http://intranet2/coag/asset/api/ASSET/Participants/Status.html>]

- As described above in Section 5.1, “Vessel movement” [2]

Together with a detection list [<http://intranet2/coag/asset/api/ASSET/Models/Detection/DetectionList.html>]. This is a list of Detections [<http://intranet2/coag/asset/api/ASSET/Models/Detection/DetectionEvent.html>], each comprising a number of parameters, one or more of which may be represented by a 'Not-available' value:

Detection

- Bearing to target (degs)
- Course of target (degs)
- Range to target (m)
- Relative bearing to target (degs)
- Sensor location (lat/long degs)
- Speed of target (m/sec)
- Strength of detection (agreed units)

- Target type (Category [<http://intranet2/coag/asset/api/ASSET/Participants/Category.html>] of target - in as much detail as is available from this sensor)
- Target id (when available)

The Category [<http://intranet2/coag/asset/api/ASSET/Participants/Category.html>] of any ASSET participant is made up from 3 components: Environment, Force, and Type. The Force is one of Red, Blue or Green. The Environment is one of Subsurface, Surface, Airborne or Cross. Finally the Type value is used to list the actual type of platform, including submarine, torpedo, sonar buoy, carrier, frigate, oiler, helicopter, MPA, missile.

The decision behaviour returns a Demanded Status [<http://intranet2/coag/asset/api/ASSET/Participants/DemandedStatus.html>] object comprising:

- Course (degs)
- Speed (m/sec)
- Depth (m)
- Optionally, a change to a sensor line-up

6. Modelling Template

The following template is to be used as a basis for definition of model algorithms - but other sections may be added where applicable.

Table 1.1. Template for model algorithm authors

Section name	Description	Example
Title	Brief title for this algorithm	Merlin Mk x Fuel Usage
Author	Domain expert with main responsibility for algorithm	Ian Mayo
Description	What are we trying to represent?	Merlin is ASW helo responsible for
Inputs	Information required to configure behaviour (other than the standard behaviour inputs). Whilst internally the information will be stored in SI units, the input parameters should indicate any units used by convention	Target Location, Trail Speed (kts)
Algorithm	Series of steps representing algorithm. May be expressed in pseudo-code or pictorially via flowchart	fuel usage as in following table...
Source	Origin of algorithm, or <i>original invention</i>	Merlin simulator user guide, dated 12th Jan 2002
Test case	Simple reproducible scenario providing expected model outputs	With payload of xxx, and fuel load of yyy, climb to yyyy ft travel at zzz kts for 30 mins, resulting load of www
Modelling guidance	Guidance in the acceptable limits for use of the algorithm (bounds), sample input data.	This fuel usage algorithm is suitable for large area ASW search problems
Incorporation	The date at which the algorithm was incorporated into ASSET	Included in ASSET build 12th July 2003

The modelling templates included in this modelling guide may be used as examples for length, tone, and level of detail required.

7. Agreed Units

The following units will be used in ASSET data-files, largely taken from the SI ¹ standard. Internally however, the ASSET software may use different sets of units with conversion factors applied as necessary to decouple this dependency.

7.1. Spatial

X-Y axis	Latitude, Longitude in degrees
Z axis	metres, expressed as height
Distance/Range	Metres

7.2. Other

Noise levels	Decibels
--------------	----------

7.3. Dynamics

Course	Degrees (0..360)
Speed	m/sec
Acceleration	m/sec/sec

¹Systeme International; the universal, unified, self-consistent system of measurement units based on the mks (metre-kilogram-second) system.

Chapter 2. Scenario

1. Introduction

Each *scenario* is an *instance* of the real world, containing an environment, a set of entities to be modelled, and details of how ASSET should move through the model in time. A scenario is the largest unit which ASSET loads from disk (Section 2.5, “Permanent Storage (XML)” [3]), though a single workstation may contain an ASSET Server simultaneously running a number of scenarios.

Whilst a scenario may contain sufficient detail to fully investigate a tactical problem, it may usefully only contain a partial dataset. It may be used to define a single reference environment, or it may contain the combination of environment and red force for which a number of different blue forces are defined.

2. Characteristics

2.1. General Characteristics

2.1.1. Name

Each scenario takes a name, typically a short phrase of 20 letters or less. A name given to a scenario containing purely an environment may be *Gulf of Oman, Spring*, or one containing a detailed tactical problem may be *209 Barrier Penetration*.

2.1.2. Description

In addition to it's name, a scenario may store a description. The description can describe the environment, the participants, or the tactical problem under consideration. The number of characters contained in a description is not restricted.

2.2. Time Characteristics

2.2.1. Application Time Step

This is the *real-world* time interval between each model step, thus an application time step of 200 milliseconds will result in the *virtual-world* moving forward 5 times per second, provided processing power permits.



Note

The application time step is only used as a guideline. The processing power available may result in the model moving forward more slowly than the application time step. In such instances, the model will step forward immediately the last step is complete.

2.2.2. Scenario Time Step

This is the time moved forward by the *virtual world* each time it steps. A typical time step for a traditional ASW engagement would be 5 seconds (the approximate update for a sonar set), whilst for an AAW engagement examining Close-in weapon support a sub-second time step may be used.



Note

Obviously larger time steps mean that the model will be able to move through an engagement more quickly, with results obtained earlier - at the cost of definition. It is

recommended that the largest time step suitable be used, with specific time periods of interest re-run with smaller time steps where necessary.



Note

Within the ASSET sensor models, sensors only produce at the most one contact per target in each time step - sensors with a long integration rate may only produce a contact every few steps. The scenario designer should choose a step time applicable to the sensors and vehicles being modelled.

In each model step, the model performs the following processing:

1. Move the current time forward by the scenario time step
2. Cycle through the participant, instructing each one to step to the new current time

2.2.3. Start Time

The time value used as the Current Time when the scenario is first loaded, or when the scenario is started.

2.2.4. Current Time

This is the time currently being reflected in the model

3. Structure

The scenario contains the following entities:

Scenario Characteristics	The way this specific scenario behaves - abstract terms which do not relate to entities modelled
Participants	<p>The platforms modelled by the scenario. Each model consists:</p> <ul style="list-style-type: none"> • Sensors • Behaviour • Current/Demanded Status • Radiated noise characteristics • Movement characteristics
Environment	The definition of the environment within which the participants are modelled. The environment contains such details as the coastline, bathymetry, and definitions of the mediums modelled (acoustic propagation, optical dissipation, etc.).

Chapter 3. Environment

1. Introduction

The Environment is fundamental component of ASSET

2. General Characteristics

2.1. Dawn/Dusk times

These are the times of sunset/sunrise. They are either hard-coded into the environment, or calculated from the current time and location.

2.2. Shipping level

3. Oceanography/Meteorology

3.1. Coastline

The coastline used within ASSET is obtained from the NIMA [<http://www.nima.mil>] VMap Level 0 dataset stored in the Vector Product Format (VPF). The VMap Level 0 database contains, amongst many other things, a global coastline at approximately 500m resolution.

3.2. Bathymetry

The Bathymetric database used within ASSET is obtained from the ETOPO dataset. The ETOPO database is a gridded database providing global coverage at 5 minute resolution.

3.3. Elevation

3.4. Sea State

3.5. Wind Speed

3.6. Rain Levels

4. Mediums

Mediums are used within ASSET to represent energy forms which may be transmitted or detected by *participants*. ASSET participants contain a series of radiated noise characteristics, representing the one or more mediums in which they transmit energy. The meaning of radiated noise is intuitive for acoustic radiated noise, but more complex for others, read the relevant sections for more explanation.



Note

Within many mediums energy is transmitted across a broad frequency spectrum. ASSET uses a *centre-frequency* approach to this problem, within each medium a

single mid-range frequency is used. See each medium for more details of the centre-frequency.

Each medium is capable of calculating the following characteristics

Energy loss	The amount of energy lost between two 3-dimensional locations
Background energy	The amount of background energy experienced at a specific 3-d location in a particular bearing

4.1. Acoustic

Acoustic propagation has for decades been the predominant underwater detection, whether it be Narrowband (@@ frequency range) or Broadband (@@frequency range). The radiated acoustic noise levels for participants varies with relative bearing and vessel speed, and is covered in Section 2.4.1, “Acoustic Radiation” [16]. Acoustic propagation is represented by modelling spherical spreading combined with absorption [Urick] [75]p110. The handy rule provided by Urick to model this propagation is

$$TL = 20 \text{ Log } r + ar * 10^{-3}$$

TL Transmission Loss

r Range (yds)

a Absorption (dB/kyd)



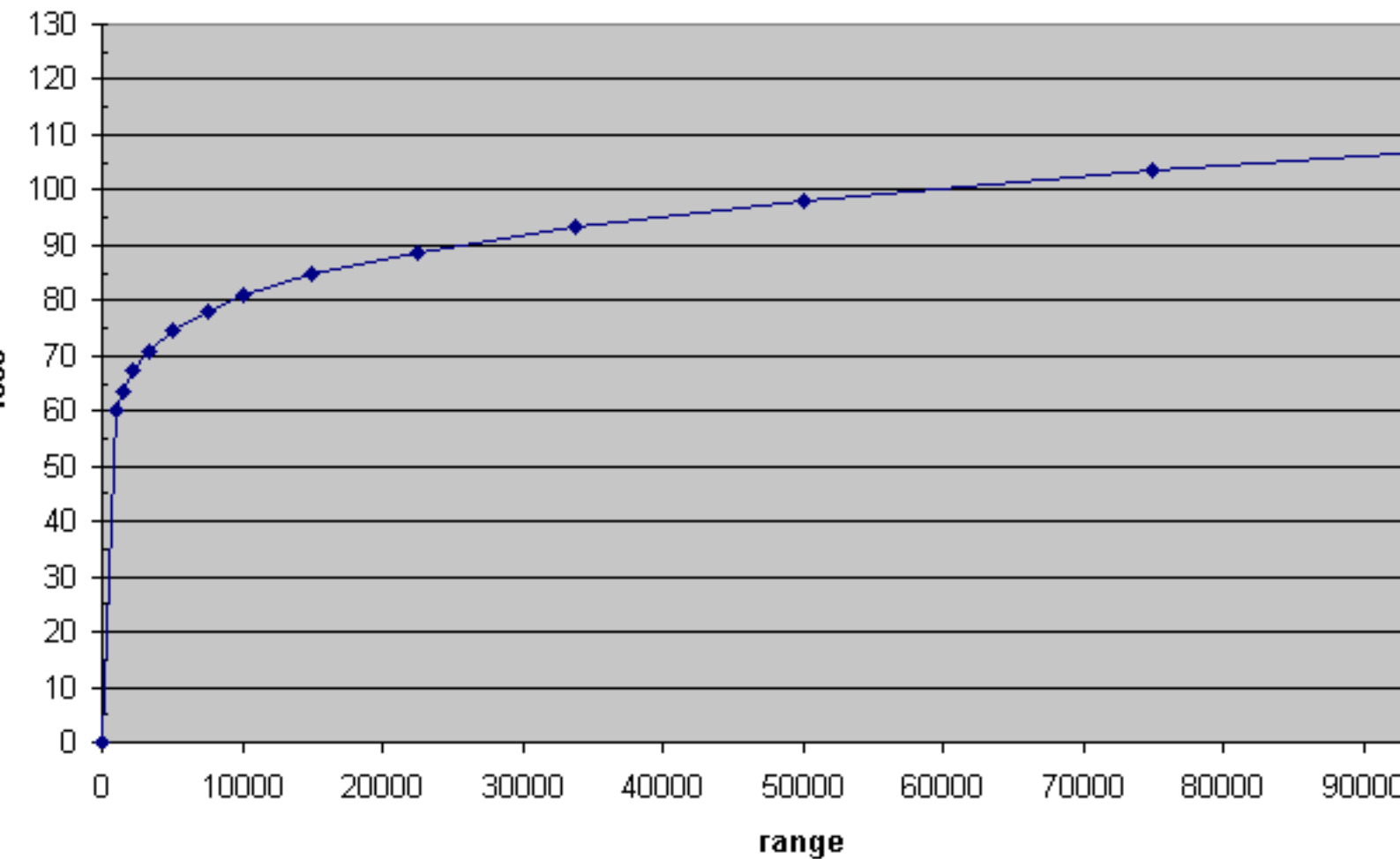
Note

Absorption coefficients are described on p109 of [Urick] [75]. Typical values of a are 0.08 at 1.2kHz in water at 20 deg C, 0.4 at 10kHz in water at 30 deg C.

4.1.1. Narrowband

4.1.2. Broadband

An absorption coefficient of 0.08 gives a proploss curve as shown below.

Figure 3.1. Example of broadband proploss curve.

Example of broadband proploss curve.

4.2. Non-acoustic

There are many forms of electromagnetic energy modelled by ASSET

4.2.1. Optic

The optic medium is used to represent visual detections. Visual sensors range from periscopes through to the classic Mk 1 Eyeball, and play a strong role in typical ASSET simulations - particularly since they provide instantaneous bearing, frequently with classification and range resolution soon after.¹

The radiated noise for the Optic medium represents the cross-sectional area of the participant on that particular bearing; a submarine at PD will show a very small optical radiated noise, on the surface when viewed head-on it will be larger, and even larger still when viewed on the beam.

The transmission loss of an optic sensor is a combination of two factors

- Curvature of the earth - depending on the sensor and target height and range between them, only a portion of the target may be visible. A curvature of the earth equation is used to reduce the "radiated-noise" of the target to account for seeing none or part of it.

¹This description of Optic detections is an initial, simplistic implementation intended only as a *placeholder* for a future validated version.

- Dissipation loss - this is purely a function of range and relates to the prevailing atmospheric conditions, obviously a foggy day will result in very short visual detection ranges even when sensor and target are relatively close.
- In ASSET the participant with the largest optical radiated noise is an aircraft carrier (having a cross-sectional area of 6000 square metres). Under the assumption that in clear weather an aircraft carrier will definitely be visible if in line of sight, ASSET dictates that at the maximum line-of-sight range, a loss of 5000 units is experienced. This loss is linearly reduced at smaller ranges - this at half of the maximum line of sight, a loss of 2500 units is experienced, and so on. Thus a sensor will have to be well within the maximum line-of-sight range of a submarine periscope to see it. An offset is applied to this linear relationship to reflect that very small targets (such as the submarine periscope) are in fact visible at several kilometers, not the hundreds of yards as would be calculated. The loss calculated is reduced by 700 to represent this "special case".

Equation for curvature of the earth

The formula used for calculation of the visual horizon (not including refraction) is:

$$d = \sqrt{2 * (h1 + h2) * a}$$

Where d represents the range(m), h1 represents the height of the sensor(m), h2 represents the height of the target(m), and a represents the radius of the earth, typically 6371229m. This formula is taken from the Meteorographer's Glossary [REF@@]

4.2.2. Communications Band

4.3. Non-traditional

4.3.1. Magnetic

4.3.2. Others

For the past few decades modern navies have investigated the use of "Footprint" type sensors which attempt to detect where a target has been, in contrast with traditional sensors which attempt to detect the current location of a target.

The existence, performance and detection medium of such sensors is a very sensitive matter and almost always classified. To overcome this sensitivity and allow this document to remain unclassified these sensors are treated in a generic, abstract manner, but one in which it should be possible to model their performance in a fidelity comparable to other sensors modelled within ASSET.

A Footprint medium is modelled by defining a spreading rate, decay rate, and limit of spread. The location of each participant which radiates the footprint medium is recorded at each model step as a footprint. Additionally at each step the previous "footprints" are expanded and decayed according to their characteristics. Sensors capable of detecting the footprint medium pass through this list of existing footprints, and produce a detection for each footprint they are able to detect. The strength of the detection is inversely proportional to the period of decay of the footprint.

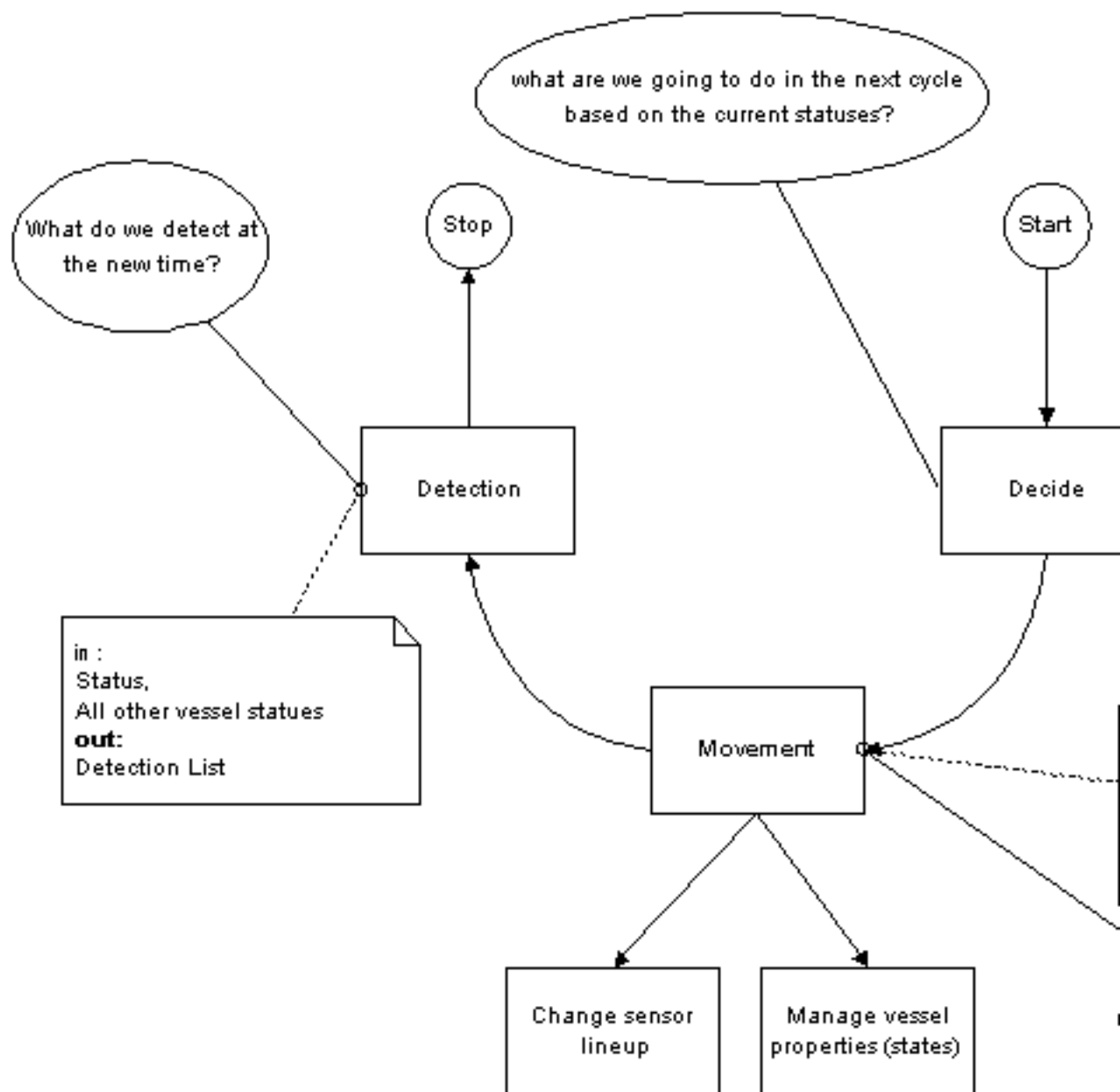
Chapter 4. Participants

Participants represent a fundamental unit within ASSET - a scenario means nothing until participants are added to it, they are the large unit modelled over which we would have control in the real world.

1. Time cycle

In each time step each ASSET participant cycles through detect, move, decide processes as illustrated in Figure 4.1, “ASSET time cycle” [12] .

Figure 4.1. ASSET time cycle



1.1. Movement

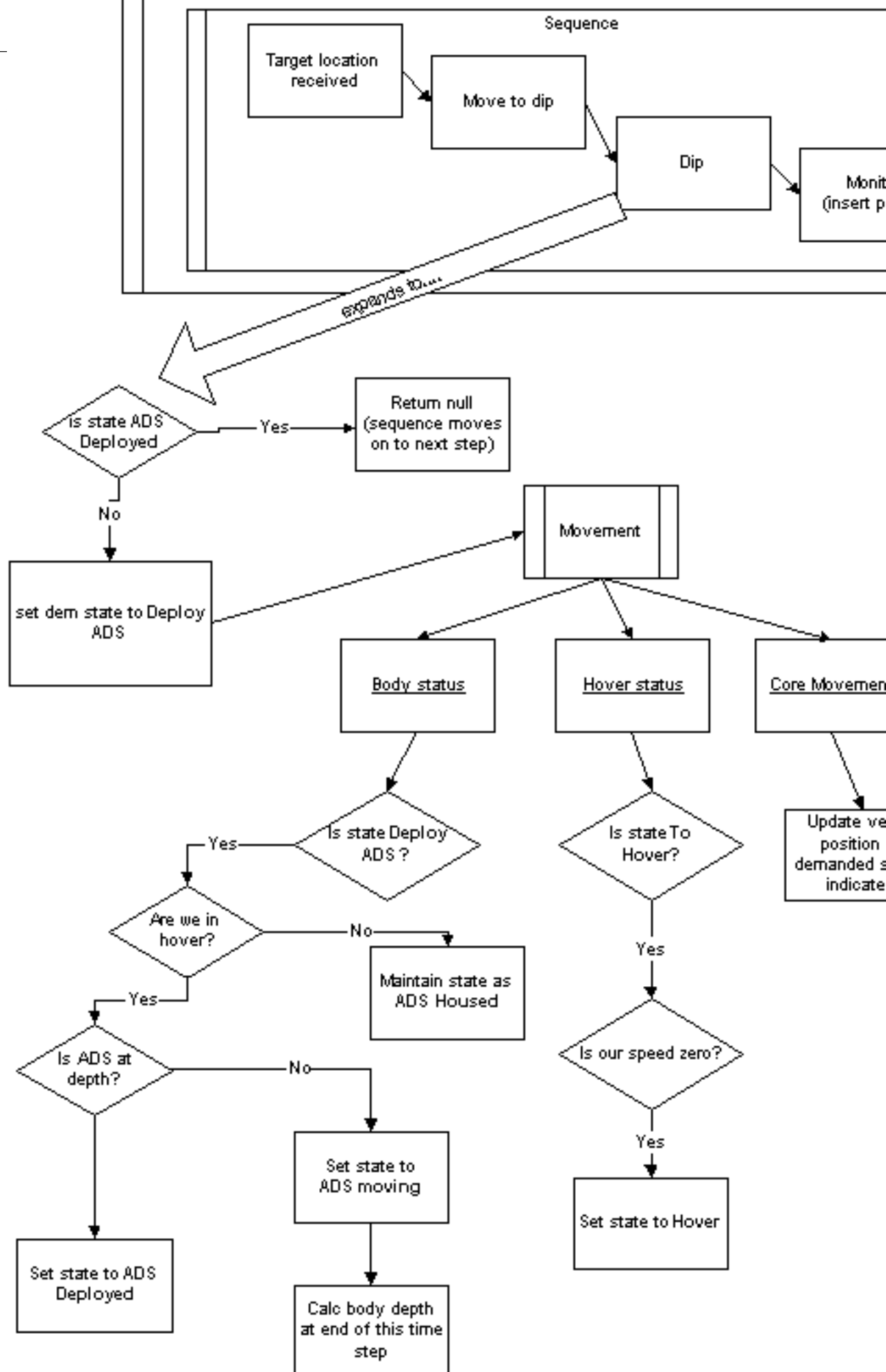
Within the movement step the participant first looks to see if there are any sensor line-up request changes (such as a request for a broadband sonar to go active). Then the series of property models

manage the transition between any sets of related properties (such as a helo moving into a hover). Lastly within this step the movement algorithm moves the vessel status towards its demanded state according to its manoeuvring characteristics.

1.1.1. State updates

The behaviour of some participants within ASSET is dependent upon their state at that time. The state may be dependent upon the current vessel status (SSK snorting at a particular speed/depth), some equipment lineup (helo in the dip, or SSN waiting for towed array steady), or some command and control state (surface ship at Action Stations).

Examples of varying vessel behaviour depending upon state are: a helo only travelling to a new dip when it isn't currently in the dip, a surface ship may not undergo replenishment at sea when at Action Stations, and an SSK may undergo a particular evasive manoeuvre when detecting an enemy vessel whilst snorting. Additionally, the fundamental characteristics of a vessel may change according to that vessel state: the recognition differential of a surface ship sensor may decrease whilst at Action Stations, the radiated noise levels of an SSK increase whilst snorting.



1.2. Detection

Within the detection step the model steps through each of the sensors on the current vessel. After checking whether the sensor is enabled, the model then examines each scenario participant to see if it is detectable on the medium detected by that sensor. If the vessel is detectable ASSET uses the sensor model to determine if a detection is produced. Detections obtained are added to a list, to be passed in turn to the decision step.

1.3. Decision

The decision step passes the current vessel status, demanded vessel status and the current list of detections to the high level decision model for the vessel. Typically this will be a Waterfall model, but any may be used. The Waterfall model then performs its processing, finally returning a demanded status - for use by the movement step in the next time cycle.

2. Characteristics

2.1. Category

Categorisation of participants allows a level of abstraction within ASSET - it is through categories that behaviours are able to refer to *Red Submarines* instead of having to refer to specific submarine instance. Participants define a single field within each of the types defined for categories, Force, Environment, and Type.

2.1.1. Force

Red	The opposing force used within ASSET simulations, as described in the Glossary
Blue	The friendly naval ASSETs, as described in the Glossary.
Green	The neutral ASSETs, as described in the Glossary.

2.1.2. Environment

Surface	Surface vessels; typically surface ships
Airborne	Airborne vessels; typically aircraft and missiles
Subsurface	Subsurface vessels; typically submarines and torpedoes

2.1.3. Type

SUBMARINE	A conventional or nuclear submarine
MINISUB	A minisubmarine, typically of less than 1000 tonnes
CARRIER	An aircraft carrier
FRIGATE	A frigate
DESTROYER	A destroyer
TROOP_CARRIER	A troop carrier
OILER	A replenishment oiler
HELICOPTER	A helicopter
MPA	A maritime patrol aircraft
AV_MISSILE	An airborne missile
TORPEDO	A torpedo, possibly having an airborne state (air-dropped), but predominantly underwater

2.2. Movement

All participants have movement characteristics such as minimum/maximum speed, acceleration rate and turning circle. Participants capable of movement in 3 dimensions also have rates of climb/descent.

Name	The name of this set of manoeuvring characteristics
AccelerationRate	The rate at which this vehicle accelerates (kt/sec)
DepthChangeRate	The rate at which this vehicle change depth (m/sec)
FuelUsageRate	The rate at which this vehicle consumes fuel (%/sec/kt) - modelled as a percentage of capacity
MaxDepth	The maximum depth of this vehicle (m)
MaxSpeed	The maximum speed of this vehicle (kt)
TurningCircle	The turning circle of this vehicle (m)



Note

Note that SSK's use a modified Movement model, which includes a recharge rate - this is the rate at which batteries are charged when on the surface.

ChargeRate	Rate at which batteries are charged (%/sec)
------------	---

2.3. Fuel usage

All participants within ASSET consume fuel. The fuel usage rate may be dominant in the behaviour/performance of an participant (such as helicopters & conventional submarines), and is normally a function of speed. Fuel is modelled as a %age of total capacity - this when a vehicle has full tanks its fuel level is at 100%, and the tanks are empty at zero %.

The fuel usage rate described in Section 2.2, "Movement" [16] is in %/sec/kt. These units represent the %age of fuel used per second for a given speed in knots.



Note

Fuel usage is not considered for Nuclear Submarines

2.4. Radiated Noise

Participants have a list of mediums under which they may be detected. Most radiated noise mediums contain a noise level, and some contain directional and/or speed related factors.

2.4.1. Acoustic Radiation

Acoustic radiated noise is the major detectable medium for underwater vehicles, and great efforts have been invested in both its reduction and detection. The dissipation of radiated noise through the water is covered in Section 4.1, "Acoustic" [9]

Typically vehicles have a base level of radiated noise at stationary which increases with speed. The radiated noise level may also be affected by machinery line-up or operating posture (such as an SSK snort). Radiated noise is the sum of *Machinery Noise*¹, *Propellor Noise*² and *Hydrodynamic Noise*³

¹That part of the total noise of the vessel caused by the ship's machinery

²a hybrid form of noise having features and an origin common to both machinery and hydrodynamic noise

Radiated acoustic noise is modelled in two ways, the first just takes a base-level and applies a speed-dependent offset to it. The second way takes a set of sample data values and interpolates the required data from it. Obviously the second method requires a greater investment in data entry but provides more accurate results

1. Base-level method. A single base noise level is provided for the indicated medium. This value is used as the *stationary* radiated noise.

Base Noise Level	The stationary radiated noise for this vehicle (dB)
------------------	---

The base noise is increased by a speed-dependent factor, initially following the following equation:

$$0.0000252 * \text{Speed}^5 - 0.001456 * \text{Speed}^4 + 0.02165 * \text{Speed}^3 + 0.04 * \text{Speed}^2 - 0.66 * \text{Speed}$$

This equation has been produced by taking the BB radiated noise of WWII submarines (recorded in p337 of [Urick] [75]) and stretching the curve to fill a 0-36 knot range.

2. Sample data method. A set of data samples are provided, each sample including radiated noise, speed, and bearing

Radiated noise	The radiated noise level at indicated speed/direction (dB).
----------------	---

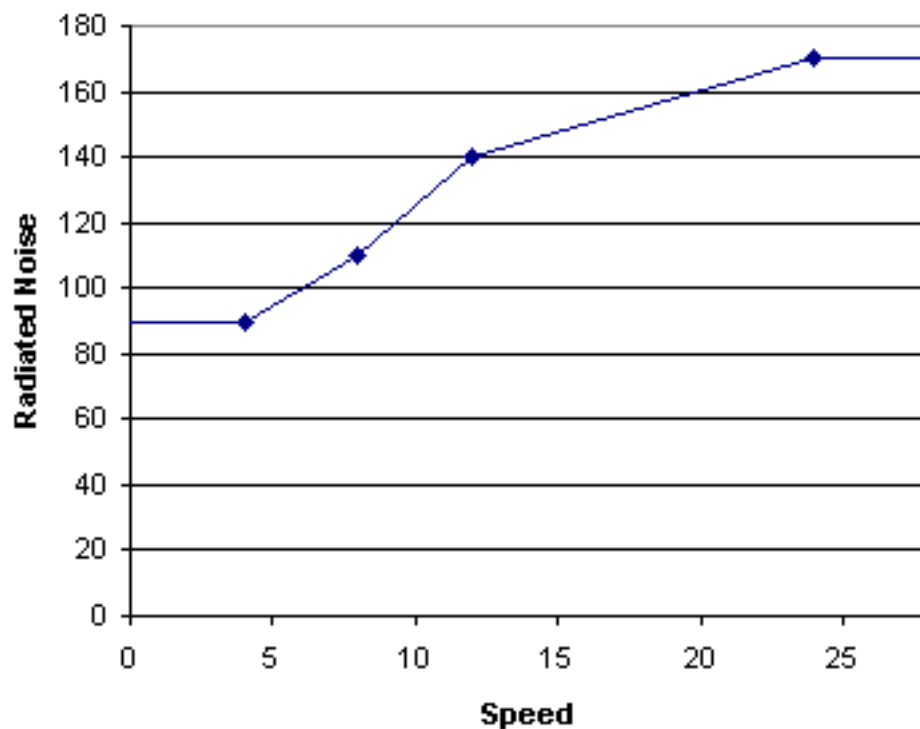
Speed	The vessel speed at which this noise level was recorded (kts)
-------	---

Direction	The direction which on which this noise level was recorded (degs +/-180)
-----------	--

A modified linear interpolation method is used. The user supplies a series of measured values of radiated noise versus speed, versus relative angle. The following sequence of steps is follows:

- a. First examine the relative angle for which the radiated noise is required. Take the set of radiated noise values for the nearest relative angle.
- b. At the requested speed, determine if this speed falls between the values provided (in which case conduct a linear interpolation to determine the radiated noise), else take the value of the lowest or highest radiated noise value (according to whether the requested speed falls below or above the set of values provided, resp).

³radiated noise originating in the irregular flow of water past the vessel moving through it and causing noise by a variety of hydrodynamic processes



Example of interpolation method used

The interpolation graph shows that if radiated noise was supplied at 4, 8, 12, and 24 knots, the radiated noise at a speed of 10 knots would be around 124dB, whilst the radiated noise at 2 knots would be 90dB and the noise at 27 knots would be 170dB.

The data supplied for the sample data method the following guidelines should be observed:

- For each relative bearing a full range of speed values should be provided, however sparse the samples are.
- [TBD]

2.5. Self Noise

Urick defines self-noise as: “Self noise differs from radiated noise in that the measurement hydrophone is located on board the noise-making vessel and travels with it, instead of being fixed in the sea at a location some distance away. Although the fundamental causes of noise are the same, the relative importance of the various noise sources is different. Moreover, in self-noise, the paths by which the noise reaches the hydrophone are many and varied and play a dominant role in affecting the magnitude and kind of noise received by the hydrophone on the moving vessel”.

The same modelling options are provided for self-noise as are used for Radiated Noise (Section 2.4, “Radiated Noise” [16]).

3. Platform Types

3.1. Airborne

Airborne participants include helos, aircraft and airborne missiles.



Note

Airborne ASSETs typically travel much more quickly than their surface or subsurface counterparts, and short scenario time steps should be used.

3.1.1. Aircraft

3.1.1.1. Vessel states

3.1.2. Helicopters

3.1.3. Airborne Missiles

3.1.4. Typical behaviours

An aircraft typically adopts one of two roles, it will either escort another participant (such as an surface HVU), or will conduct a search about a barrier or area. On detection of a potential target, the airborne ASSET will normally communicate this detection then close to prosecute it.

The Escort behaviour is used to instruct an airborne ASSET to escort another participant, with the aircraft loosely stationed at a relatively long range.

One of the Transit behaviours is used to put the aircraft in a search pattern, possibly involving a Transit to place it on station.

3.2. Submarine

Submarine platforms, by definition, travel in 3-dimensions, and typically favour use of their acoustic sensor.

3.2.1. Conventional Submarine

Within the Royal Naval use of ASSET the conventional submarine is, and probably will continue to be, the predominant *red force*.

3.2.2. Nuclear Submarine

3.2.3. Torpedo

Torpedoes modelled within ASSET may be air, surface or submarine launched. Sonar is their main sensor, and they are fuel-limited. The actual logic used by most submarines is beyond that which may be modelled - but within ASSET it typically follows the following pattern:

1. Follow a bearing
2. Search for a particular target type
3. Collide with the target (triggering the warhead)

3.2.4. Countermeasure

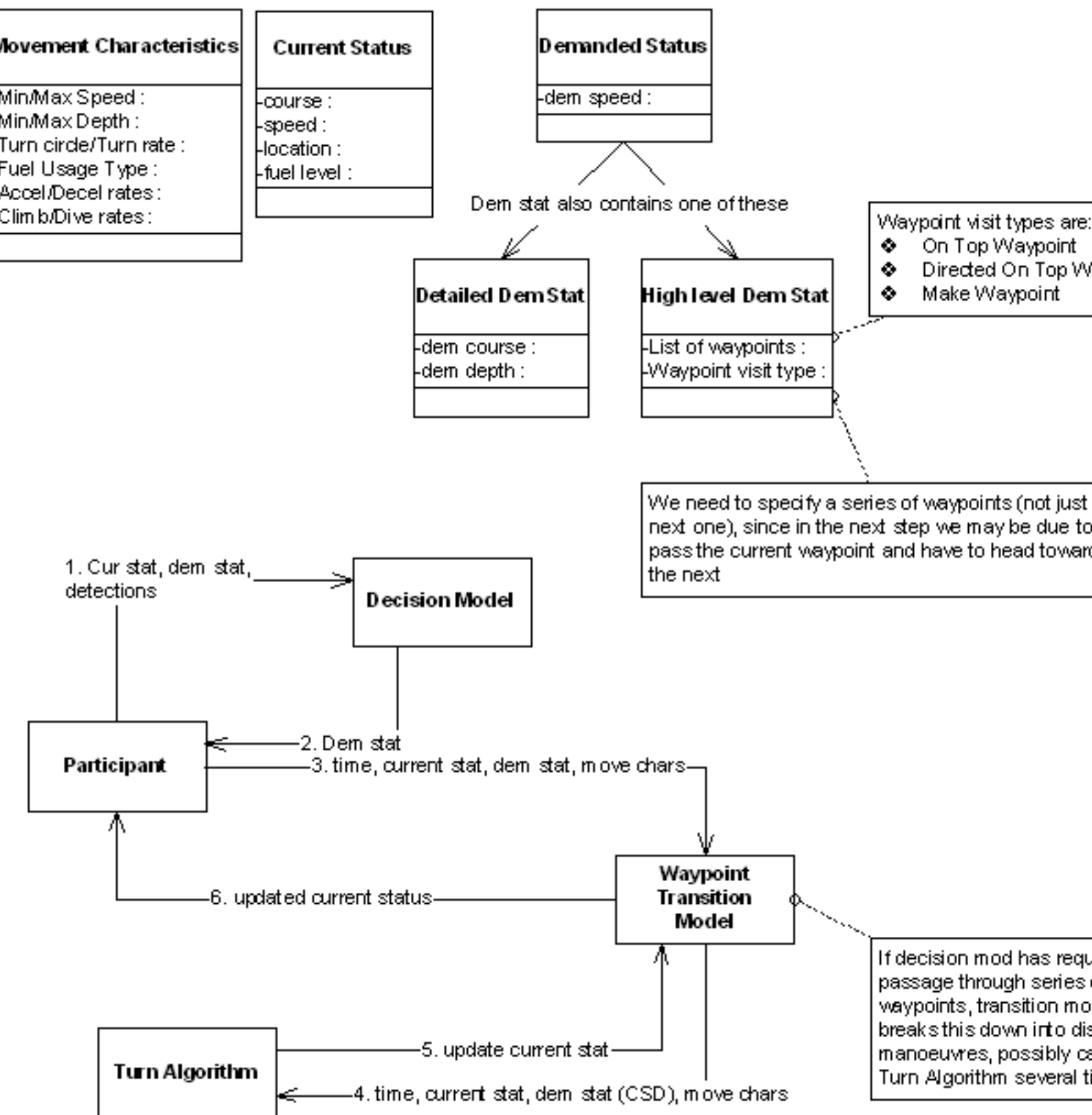
3.3. Surface

A surface platform is essentially a submarine platform which is constrained to travel at zero depth.

4. Vessel Movement

Vessel movement is managed within ASSET at two levels, firstly there is low level modelling where a participant makes a transition to a new course and speed using the correct acceleration/turn rates. Secondly a higher level modelling manages a participant's movement through a series of waypoints. Information is passed between the movement algorithms using the Current Status, the Demanded Status, and the set of vessel manoeuvring characteristics, as shown in the following diagram:

Figure 4.3. Overview of vessel manoeuvring



4.1. Turn Algorithm

4.1.1. Introduction

This low level algorithm was first created in the days of MWDC whilst investigating submarine TMA, and has been used many times since in the solution of ad hoc modelling problems.

4.1.2. Author

Ian Mayo coded the current implementation. If I remember correctly the original strategy for the algorithm was devised by Iain McKenna, c++ implementation by me, with review by Steve Little then John Golding back in around 1996 then 2000 resp.

4.1.3. Description

The Turn Algorithm is the low-level model which moves a vessel along a course and speed, and handles the transition from current course/speed/depth to demanded course/speed/depth within that platform's manoeuvring characteristics.

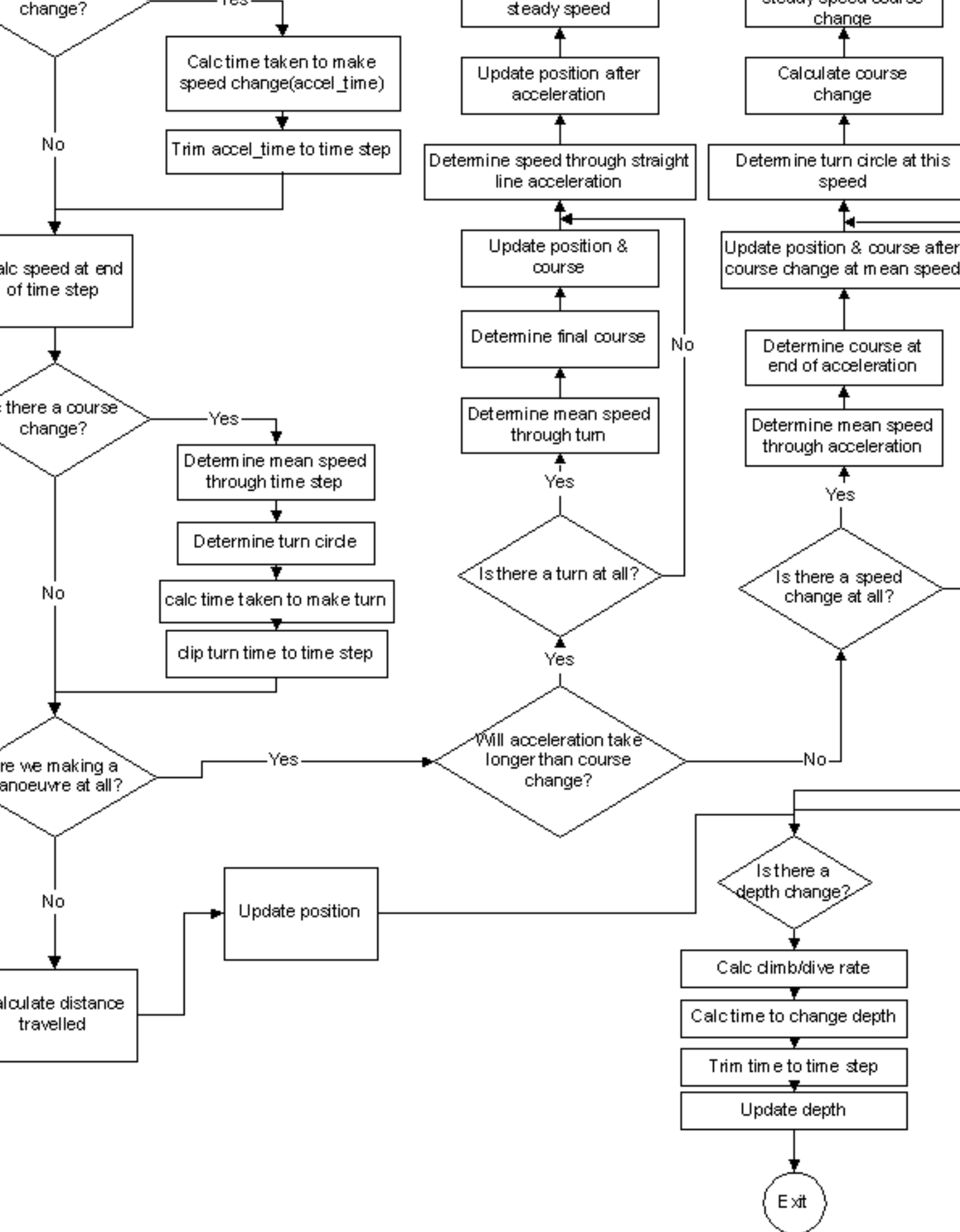
4.1.4. Inputs

Vessel manoeuvring characteristics (units described in source section)

4.1.5. Algorithm

The algorithm updates a vessel's course/speed/location according to the demanded course/speed/depth passed in and that vessel's set of manoeuvring characteristics. That is, the vessel will tend towards the demanded status within that platform's constraints.

Of significance within the algorithm is the decision regarding which of the course & speed change will take longer. Processing the first part of the manoeuvre (where both course and speed are changing), followed by the second part of the manoeuvre (where only one is changing) simplifies the calculation of mean speed (and this distance travelled) during each period. The algorithm is implemented as follows:



4.1.6. Source

Original invention by Submarine Division, MWDC circa 1995.

4.1.7. Test case

to be determined... Something like the following set:

1. Check vessel moves correct distance in correct direction on steady course/speed at different step sizes
2. Check vessel accelerates & decelerates at indicated rates, within indicated limits
3. Check vessel turns at set turning circle for different speeds, maintaining straight heading between
4. Check combination of course & speed changes, check that individual changes complete at correct point

4.1.8. Modelling guidance

ASSET users have no direct control over this algorithm - vessel movement is controlled via the vessel manoeuvring characteristics. The vehicle dynamics modelled within this algorithm are environment-independent, and may be seen to reflect the movement of submarines, aircraft or surface vessels. Higher level modelling algorithms (such as waypoint transition) are used to translate demanded location into demanded course and depth - including where platforms travel near to but do not actually visit a location.

The turn algorithm works at most sizes of time interval, though because of an smoothing function (when speed delta is less than 0.001 m/sec or course delta is less than 0.01 degrees just assume on speed/course) time intervals of the order of several milliseconds may not work.

4.1.9. Incorporation

July 2003

4.2. Waypoint transition

At a higher level, there is a requirement for modelling functionality to fill the void between the decision model requesting a particular action and the demanded course and speed required for the turn algorithm. An example of this is where a decision model requires that a participant pass through a particular waypoint and leave that waypoint already on a particular course (passing through a Directed On Top). The *old* way of modelling waypoints within ASSET was for the participant to just keep heading towards a waypoint until within a set minimum threshold of it, then assume the new course - which isn't sufficiently complex to handle waypoint transitions.

Where necessary the waypoint transition algorithm can break a single time step down into a number of discrete course/speed/depth changes, calling the Turn Algorithm repeatedly as required.

The waypoint transition algorithm provides support for a variety of styles of visiting waypoints:

On Top Waypoint	The participant must travel directly to the waypoint, passing over it
Directed On Top Waypoint	The participant must travel over the waypoint in such a fashion that it leaves the waypoint on a requested heading (or towards the next waypoint)
Make Waypoint	The participant must travel towards a waypoint but make a turn to head towards the next waypoint at the last possible

moment (the track of the aircraft will describe an arc which is the internal tangent to both the inbound and outbound tracks)

In general terms the algorithm first inspects the transition style, then performs processing dependent on the style requested.

4.2.1. On Top Waypoint

4.2.1.1. Author

Jon Walters, Merlin C OA. Flowchart drawn up by Ian Mayo

4.2.1.2. Description

An aircraft is required to reach a point in 3D space in the quickest way while maintaining a current speed.

4.2.1.3. Inputs

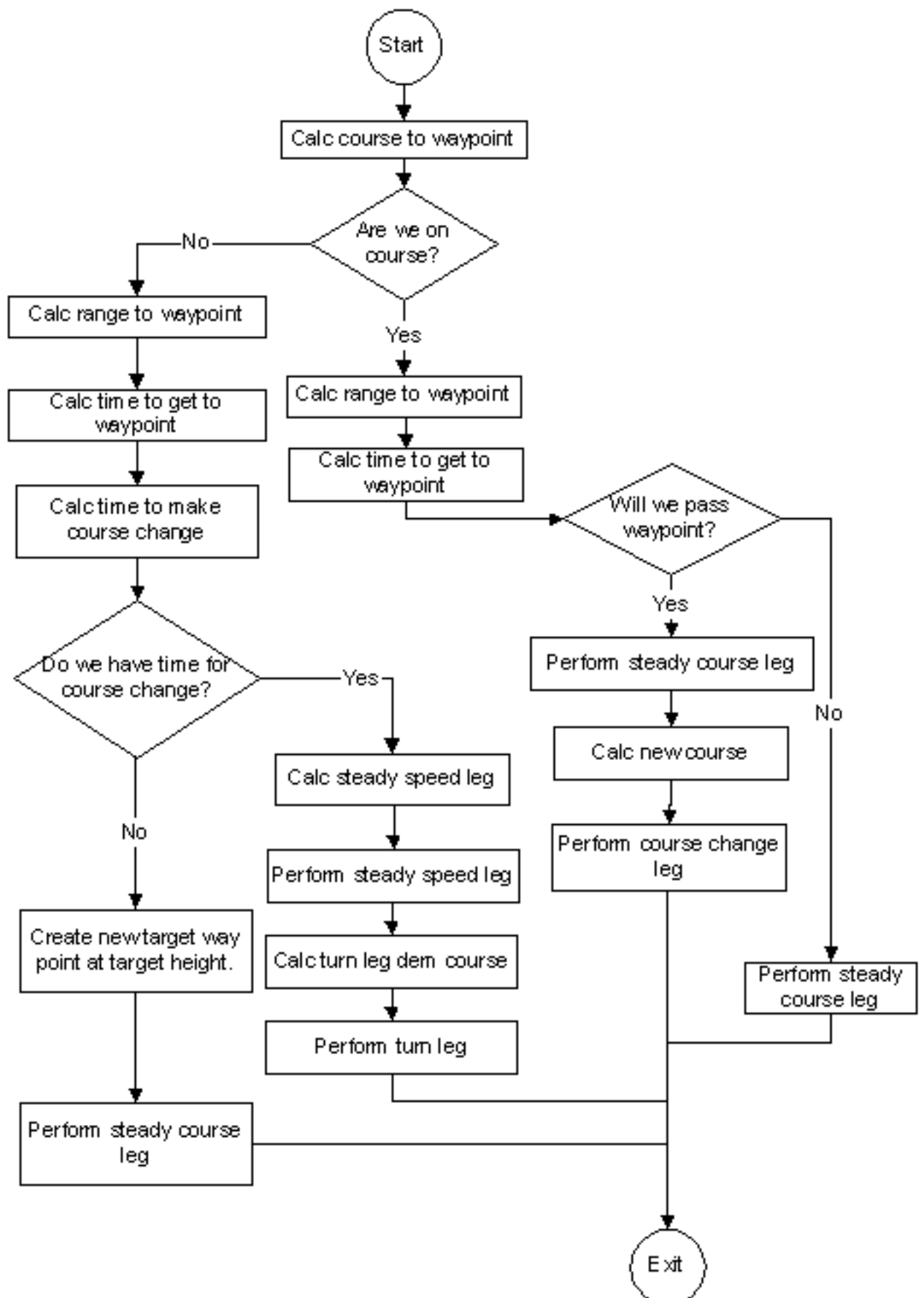
INPUT - waypoint x,y,z.

OUTPUT - state 'captured' or 'missed'

4.2.1.4. Algorithm

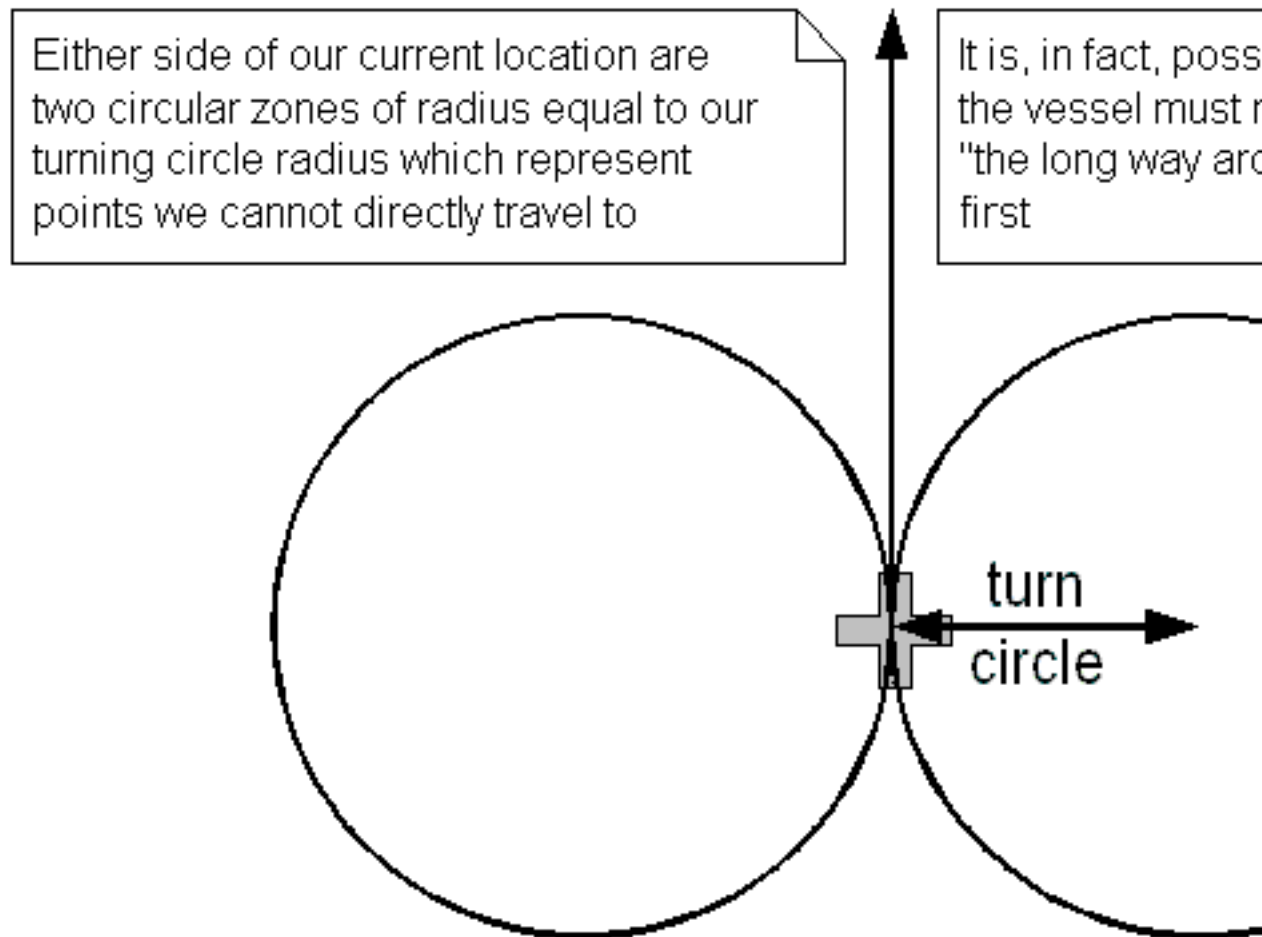
Calculate intercept course and height difference. Turn onto course and simultaneously Climb or Dive as necessary. Refine intercept course and Turn again if necessary. When within 50 m laterally and 100 ft vertically (capture criteria) of waypoint declare waypoint is 'captured' - continue on current course and set height to waypoint height. If aircraft never within 50 m or 100ft of waypoint then declare waypoint 'missed' and execute a Recapture Waypoint.

Figure 4.5. Flowchart showing On Top Waypoint algorithm



The following diagram illustrates the manner in which the algorithm decides whether a waypoint is attainable. If the waypoint is within either of the *unattainable* circles the participant cannot get to it by merely selecting the direct course as demanded course and sailing down it. If the waypoint is within one of the unattainable circles then the vessel must first turn away before heading towards the target location.

Figure 4.6. Unattainable waypoints



4.2.1.5. Source

Merlin Flying Guide

4.2.1.6. Test Case

Start at (0,0,200) heading 000 speed 60 m/s and make waypoint at (10000, 10000, 1000) in a time of xxxx (calculate). Return 'captured' as a result. Start at (0,0,1000) heading 090 speed 70 m/s and attempt to make waypoint (-500,0,100). Ensure 'missed' is returned at that the aircraft ends up at 1000 ft.

4.2.1.7. Modelling Guidance

Element of aircraft maneuver. The criteria of being within 50m and 100ft is called 'capturing' the waypoint. In practice there are just going to be some position, course and speeds (PCS) of the aircraft from where it just cannot capture a waypoint.

4.2.1.8. Incorporation

tbd

4.2.2. Directed On Top Waypoint

4.2.3. Make Waypoint

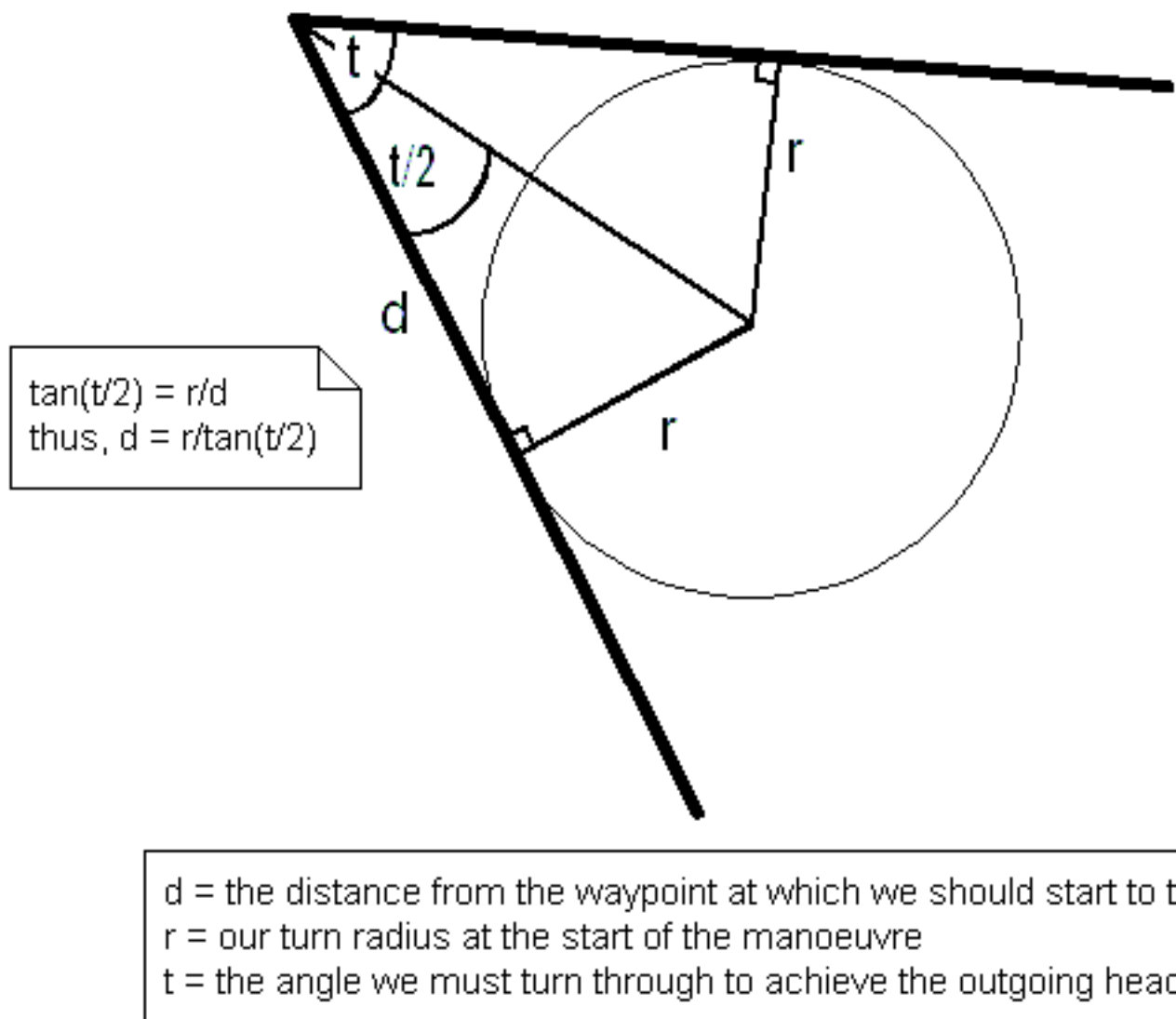
Author. Jon Walters

Description. An aircraft is required turn onto the next leg of a route without passing through the waypoint that marks the junction of its current leg with the next

Inputs. inbound leg track, out bound leg track, rate of turn

Algorithm. Assumption aircraft established on course of inbound leg. Commence a turn at the designated ROT and current speed that will make the aircraft describe an arc which is the internal tangent to both the inbound and outbound tracks. If this is geometrically impossible due to the length of the legs being shorter than radius of the turn, then carry out an On-Top WP behaviour instead.

Figure 4.7. Make Waypoint algorithm



Source. Merlin Flying Guide

Test case.

- Establish leg between start (0,0,1000) and end (20000,20000,1000) transit height 1000 ft. Place aircraft on leg with 5000 m to run. Outbound leg is from (20000,20000,1000) to (20000,0,1000). Check aircraft turn right through 45 degrees, starting with xxxx m to run to (20000, 20000, 1000) and joining the new track xxxx m down leg from (20000,20000,1000)
- Establish leg between start (0,0,1000) and end (1000,1000,1000) transit height 1000 ft. Place aircraft on leg with 800 m to run at speed 70 m/s, Rate of turn set to 1 deg/sec. Outbound leg is from (1000,1000,1000) to (1000,0,1000). Check aircraft reverts to On-Top WP behaviour and successfully arrives at the final waypoint (1000,0,1000)

Modelling guidance. Element of aircraft maneuver.

Incorporation. Sep 03

Chapter 5. Sensors

1. Lookup Sensor Model

In Autumn 2003 an improved above-water sensor model was defined within MWC.

This model covers all non-acoustic detection sensors used by aircraft. The following criteria were used in its formulation:

1. Must be simple, both in execution and in principle
2. Must be transparent, with all assumptions made obvious
3. Must be simple
4. Must be tweakable, with constants included that allowing tuning to the real life (RL@copy;) world expected performance
5. Must be simple
6. Must capture the vital parameters that most affect performance in RL@copy;
7. Must be simple

1.1. Basics of model

A modified "cookie-cutter" approach is proposed. Each entity will be in one of 4 states with respect to the sensor - UNDETECTED, DETECTED, CLASSIFIED and IDENTIFIED.

Each sensor will have a predicted range RP, which will be calculated for each combination of sensor and target and depend on (inter alia) target, target aspect, weather conditions. In addition each sensor will have a time between detection opportunities (TBDO in seconds) and variability in detection range - VDR (between 0 and 1.0). Every time an RP is generated a random element is applied to produce an instantaneous range (RI). For each completed TBDO there will be one possible detection opportunity, with the RI compared to the actual range (RA). If $RA \leq RI$ then the target will be in state DETECTED. Transition to CLASSIFIED will take place when range has reduced to a certain percentage of R *and* the contact has been DETECTED for a certain number of seconds - transition to IDENTIFIED will then take place in a similar manner. The parameters for these transitions will be specified for each sensor. If at any time the contact goes outside a maximum tracking range then it will revert to UNDETECTED status.

The sensor model will thus have the following parameters:

Predicted Range (RP)	Range at which there is a chance of making an initial detection - calculated for each sensor/target combination either by formula or through lookup table
Instantaneous detection range (RI)	The actual range determined for each combination of target and sensor obtained by adding a variable number to the RP
Variability in detection range (VDR)	A number by which RP is multiplied to obtain the standard deviation of the normal distribution of the instantaneous detection range
Time between detection opportunities (TBDO)	The time in seconds after which a detection chance occurs
Max Range Factor (MRF)	The number that RP must be multiplied by to obtain the maximum tracking range

Classification range factor (CRF)	The number the RP must be multiplied by to obtain the range at which CLASSIFICATION occurs
Classification time period (CTP)	The time after which a contact automatically goes from from DETECTED to CLASSIFIED
Identification range factor (IRF)	The number the RP must be multiplied by to obtain the range at which IDENTIFICATION occurs
Identification time period (ITP)	The time after which a contact automatically goes from from CLASSIFIED to IDENTIFIED

1.2. Definitions

Range	The instantaneous <i>slant</i> range between sensor and target
Undetected	completely unknown to the sensor
Detected	position known but no knowledge of type or identity
Classified	target known as a broad target type, ie ENEMY FRIGATE or FRIENDLY PERISCOPE
Identified	target known by name and class

1.3. Method of calculating RI

RI is obtained by multiplying RP by a random number generated on the normal distribution. Generate a random number (n) from the normal distribution with a mean of zero and a standard deviation equal to VDR. Obtain the RI from the following equation: $RI = RP * (1 + n)$

If RI is less than zero then set RI to zero.

This RI is generated once for every combination of target and sensor - that is every time an RP is generated the random element is applied to produce an RI.

1.4. Radar Model (for Merlin radar)

RP This is derived from the single transmission radar range equation with only target radar crosssectional area as a variable. RP is given by either $\sqrt{2 * R * \text{ownship_ht} / B} + \sqrt{2 * R * \text{tgt_ht} / B}$ [RADAR HORIZON range RH] or $K * (\text{sigma} * \text{seastate})^{(1/4)}$, whichever is less. R is the radius of the Earth (6370950m), and B is the fudge factor, 0.7366 for radar (back-calculated from the radar horizon formula in the Merlin Tacman. Sigma is the notional radar cross section in square metres, K is a constant dependent on the radar and seastate is factor dependent on sea state. Sigma is obtained from a lookup up table of target type and aspect of target - seastate from a lookup table on type of target and actual sea state.

VDR 0.02

TBDO 2

MRF 1.2 (if > RH then equal to RH)

CRF 0

CTP 0

IRF 0

ITP 0

Examples of lookup tables for radar sensor:

Table 5.1. Sigma

	Dead ahead	Bow	Beam	Quarter	Astern
Frigate	1000	3000	4000	3000	1000
CVS	2000	8000	10000	8000	2000
Periscope	0.5	0.5	0.5	0.5	0.5
Fishing boat	5	8	10	8	5

Table 5.2. Sea State

	0-1	2	3	4	5	6
Frigate	1	1	1	1	0.95	0.9
CVS	1	1	1	1	1	0.95
Periscope	1	0.8	0.75	0.7	0.5	0.3
Fishing boat	1	0.9	0.8	0.75	0.7	0.5

1.5. Eyesight

RP RP is derived from Beer's law, adjusted for sea state and with the contrast abstracted out to a target visibility factor.

$$RP = -1/Atten * \ln (V * seastate) * light\ or\ visual\ horizon$$

$$(VH) = \sqrt{2 * R * ownship_ht / B} + \sqrt{2 * R * tgt_ht / B}$$
 [RADAR HORIZON range RH], whichever is less

Where:

Atten is an atmospheric attenuation factor (mist, fog, clear day, haze, et al) obtained from a lookup table

V is a target visibility value based obtained from a lookup table

Sea State is a modifier for sea state found via a lookup table from target type and sea state.

Light is a factor allowing for darkness or the lack thereof

R Radius of the Earth (6370950m)

Factor 0.8279 (taken from Bowditch, the Practical Navigator, 1995, Table 12).

VDR 0.05

TBDO 10

MRF 1.05 (if > VH then equal to VH)

CRF 0.8

CTP 20

IRF 0.2

ITP 30

Examples of lookup tables for Eyesight sensor:

Table 5.3. Attenuation

	Very clear	Clear	Light Haze	Haze	Mist	Fog
Atten	8e-5	2e-4	5e-4	1e-3	2e-3	4e-3

Table 5.4. V

	CVS	Frigate	Periscope	Fishing Boat
V	0.2	0.2	0.12	0.16

Table 5.5. Sea State

	0-1	2	3	4	5	6
Frigate	1	1	1	1	0.95	0.9
CVS	1	1	1	1	1	0.95
Periscope	1	0.8	0.75	0.7	0.5	0.3
Fishing boat	1	0.9	0.8	0.75	0.7	0.5

Table 5.6. Light

	Daylight	Dusk	Moonlit Night	Dark Night
Light	1	0.4	0.3	0.05

1.6. MAD

RP RP is a simple lookup table based on target size (see below for unrealistic figures).

VDR 0.05

TBDO 1

MRF 1.0

CRF 0.6

CTP 1

IRF 0

ITP 0

Table 5.7. Predicted Range

	Range
Frigate	2000
CVS	3000
Submarine	1200
Fishing boat	400

1.7. ESM (Generic RWR)

This model will have to take into account whether the transmitter is on or off. At the moment of switching on the first detection possibility takes place, then additional detection opportunities spaced at TBDO intervals. We would simulate the single sweep of a submarine radar by switching off again within the TBDO of the radar.

RP is a simple, but large, lookup table based on target emitter. In principle this might be replaced by some form of equation, but that will require a lot of knowledge of receiver and transmitter properties.

VDR 0.1

TBDO Based on scan rate of emitter, obtained from lookup table

MRF 1.0

CRF 1.0

CTP 10

IRF 1.0

ITP 15

1.8. Electro-optics

Same as eyesight but with tweaked parameters depending on the type of sensor.

1.9. History

Version 1	Initial formulation of algorithms
Version 2	Incorporates modifications suggested by Iain Mckenna. These remove the probability of detection variable and replace it be a variable detection range at each scan, which is now renamed a detection opportunity. Also a small tweak to the ESM model requiring a variable TBDO found from a lookup table and dependent on the type of emitter.
Version 2.1	Modified the variable element so that it applies to each combination of target and sensor only once, and is not recalculated at each detection opportunity

Chapter 6. Interaction

1. Detection

The Passive and Active sonar equations from the Sonar Modelling Handbook are used to model detections. With a negative signal excess no detection will occur, with increasing positive signal excess a detection is more likely to occur.

This section records the modelling of sensors. Inevitably sensors are described according to the medium which they detect, but the following general equation is followed within all mediums:

$$SE = (TN - (ON - DI) - RD - BN) - PL$$

In the following sections, the mechanism for detections in particular mediums is described, including how the components of the detection diverge from the common meanings listed above.

1.1. Passive Acoustic Detections

As expected, in acoustic detections the passive and active sonar equations are used unchanged. The components of the passive equation (here) [../smh/volume1/1-3.pdf#page=4] are derived as follows:

SE The resultant energy arriving at the sensor, a positive signal excess producing a valid detection.

TN The radiated noise of the participant carrying the sensor, normally speed and aspect dependent

TN The radiated noise of the target participant, normally speed and aspect dependent

DI Directivity Index of the current sensor. For sensors which do not provide 360 degree coverage the DI is used to "blank-out" areas without coverage

RD The factor applied to components of the sonar equation to compensate for errors in observed and predicted sonar performance

PL The acoustic energy dissipated between target (source) and sensor, as illustrated in the earlier graph (Section 4.1.2, "Broadband" [9])

BN Background Noise in the direction of the source from the sensor

1.2. Active Acoustic Detections

Some of the components used in the Passive Sonar equation (here) [../smh/volume1/1-5.pdf#page=3] are modified for use in active detections:

SL The source level of the active sonar

PL Twice the acoustic energy dissipated between target (source) and sensor, as illustrated in the earlier graph (Section 4.1.2, "Broadband" [9])

1.3. Passive Narrowband Detections

The equation components used in Narrowband detection have the same meaning as those in the Broadband Passive detection shown above (though they will inevitably take different values).

1.3.1. Towed Arrays

The towed array is the traditional sensor used for passive narrowband detection. A significant characteristic of the towed array is the fact that it is only able to produce valid data when in a

roughly straight line. Whilst a number of algorithms exist to aid in the prediction of array steady time, ASSET uses a fixed steady time, configurable for each towed array sensor.

2. Communications

Communications within ASSET is implemented similar to detections. A transmitter is treated similarly to an active sensor, and a receiver is treated similarly to a passive receiver.

2.1. Message Type

The following tree of message types are supported:

1. Tactical
 - a. Detection
 - Target Type
 - Target Strength
 - Target course, speed, depth (where known)

In addition, all messages contain ownship position and the current DTG.

2.2. Lifetime

Chapter 7. Behaviours

1. Introduction

Many objects in ASSET possess their own behaviour, implemented through a form of artificial intelligence. The provision of behaviours prevents the need for a dedicated driver for each vessel, and allows agreed behaviours to be distributed throughout an organisation - so that all analysts are trialling tactics against the same target for example.

The behaviour model also allows a particular set of Rules of Engagement (ROE) to be defined for a vessel or group of vessels to give them a repeatable behaviour.

Each behaviour receives the current vessel *status* (course, speed, depth), the current list of *detections*, and a *scenario activity monitor*, returning a *demanded status* (course/speed). If the particular behaviour does currently apply, then a null *demanded status* is returned. When a participant receives a null *demanded status* it continues travelling on its current course and slows down to it's minimum speed.

Structural behaviours allow the combination of "real" behaviours - providing a complex behaviour pattern without actually making any tactical decision themselves. Structural behaviours control how their child behaviours are used/called. Some behaviours (such as Waterfall) allow the processing of one behaviour to be interrupted by a higher priority behaviour. When this happens the lower-level behaviour receives a message to inform it that it's processing was interrupted. Most behaviours will not make use of this message. Some transitting and tactical ones do though. The Ladder-Search, for example takes the participant back to it's last point on the search track prior to being interrupted.

All behaviours share some of the same attributes

Name	The name given to this instance of this behaviour. In one participant we may use the same type of behaviour (such as Move) a number of times, each should be given a unique name
IsAlive	Some behaviours may be defined in the scenario file, but to aid analysis you may choose to not make them active. They are still loaded into ASSET, but they are never used by their parent participant.

2. State behaviours

This set of behaviours manages the transition between vessel states - either a condition depending upon a state or the request that a vessel make the transition to a new state.

2.1. New State Request

Author. Ian Mayo

Description. Request that a participant make the transition to an new state - and keep on requesting until we reach that point,

Inputs. Status and demanded status - so we can check if we are already in the state, or already requesting the transition to the new state.

Algorithm. See if the status object already carries this state. If it does return null, if it doesn't then extend the current demanded status to incorporate the new state request and return that.

Source. Original invention

Test case. In test harness test that demanded status is produced when not already in that state, and not produced when that state achieved.

Modelling guidance. None so far.

Incorporation. August 03

3. Structural Behaviours

Typically the overall behaviour of a vessel will comprise a number of discrete behaviors. The overall behaviour is constructed using these structural behaviours. In them selves they do not cause a vessel to move or change state, but they do control which actual discrete behaviour is triggered

3.1. Sequence

3.1.1. Category

Behaviour/structural

3.1.2. Description

A sequence of behaviours, each of which is fired once

3.1.3. Behaviour parameters

- Series of behaviours
- Whether to restart on completion
- Current behaviour (default value of first)

3.1.4. Algorithm

1. Fire current behaviour
2. Is the next behaviour at the end of the list?
 - a. If no, fire the next behaviour, remember that one as the *current*
 - b. If yes, are we set to restart?
 - i. If yes, move the first behaviour
 - ii. If no, return null

3.1.5. Source

Original invention

3.1.6. Test Case

1. Create non-repeating behaviour using sequence of behaviours, check they are fired in correct order, returning null on completion.
2. Create repeating behaviour using sequence of behaviours, check they are fired in correct order, firing first again on completion.

3.1.7. Modelling Guidance

This is a way of producing a behaviour representing a complex set of steps, such as transiting to a specified point before conducting a specific search at that point

3.2. Waterfall

A series of behaviours in descending order, where if the highest priority behavior is not applicable control falls to the next highest priority, and so on. The normal implementation of a Waterfall

behaviour contains a lowest-priority behaviour which is always valid (wander, return home, etc). Where the lowest priority behaviour contains a test, if it fails no demanded state is produced and the vessel remains on steady course/speed.

3.2.1. Description

A prioritised list of behaviours

3.2.2. Behaviour parameters

Series of behaviours

Series of behaviours to be executed in descending order of priority

-
- Stay Alive

3.2.3. Algorithm

1. Fire behaviour at head of Waterfall
2. Did it return a demanded status?
 - a. If yes, return this demanded status
 - b. If no, descend through other behaviours until one returns a demanded status
3. If none return demanded status, return null

3.2.4. Source

Original invention

3.2.5. Test Case

Create behaviour for vessel where high level behaviours aren't always valid. Check that control falls to lower level behaviours when high level ones aren't valid, and that lower level ones are triggered when higher one(s) are valid.

3.2.6. Modelling Guidance

This is a fundamental behaviour component within ASSET, capable of producing a wide range of behaviours. Frequently the lowest-level behaviour will not have a condition - it will always return a result: such as wander - so that the participant is never 'uncontrolled'.

3.2.7. See Also

Also see the Wander behaviour

3.3. Switch

This structural behaviour allows different participant behaviours to be defined - only one of which is actually adopted within a scenario run. The multiple behaviour definitions allow a quick change in participant behaviour whilst in an interactive (Workbench) run, or for a range of target behaviours to be recorded when in a batch modelling (Monte Carlo) run.

3.3.1. Description

A range of behaviours, only one of which is active

3.3.2. Behaviour parameters

- Series of behaviours

- The index of the active behaviour

3.3.3. Algorithm

Actuate the indicated behaviour

3.3.4. Source

Original invention

3.3.5. Test Case

Define a series of behaviours, with an index indicating the active behaviour. Execute the Switch, ensuring that only the indicated behaviour is active.

3.3.6. Modelling Guidance

Frequently there are occasions when a participant could execute one of a number of behaviours. This could be represented within the scenario as definitions of multiple vessels - but since all vessel properties (other than the behaviour) are identical this incurs a maintenance/configuration cost. Use of the switch allows a single participant to be defined, with it's behaviour controlled at run-time. When performing Monte Carlo simulation use of the Switch allows the target population to randomly employ one of the predetermined behaviours (Aggressive vs Evasive for example).

3.3.7. See Also

In Monte Carlo generation the Choice scenario variance can be used to randomly vary the Switch index attribute - producing targets with random behaviour.

3.4. Composite

3.4.1. Description

A traditional condition/response structure allowing complex behaviors to be defined. Thus, if the condition is successful, the response is followed.

3.4.2. Behaviour parameters

- Condition behaviour
- Response behaviour

3.4.3. Algorithm

1. Perform the condition test.
2. Was a value returned?
 - a. If yes, pass value to the response behaviour, then return the demanded status
 - b. If no, return null

3.4.4. Source

Original invention

3.4.5. Test Case

Create behaviour for vessel where high level behaviours aren't always valid. Check that control falls to lower level behaviours when high level ones aren't valid, and that lower level ones are triggered when higher one(s) are valid.

3.4.6. Modelling Guidance

The versatility provided by this architectural behaviour lends it to being used in a wide variety of behaviours, though the results value passed between the condition and response needs to be compatible with both

3.4.7. See Also

See also the Condition and Response groups of behaviours

3.5. Conditions

3.5.1. Detection

This condition is triggered on detection of another participant, though a target-type structure allows for the categories (Section 2.1, “Category” [15]) of target detected to be filtered. For example, an ASW detection behaviour is only normally to be triggered on detection of a sub-surface target.

Additional parameters allow for the signal strength, maximum/minimum range, and relative bearing thresholds to be specified, applicable where the detecting sensor is capable of producing such data.

3.5.2. Location

This condition is triggered when the participant achieves a position relative to the location provided. The condition may be triggered if the participant either exceeds a set range from the location, or when the participant is less than a set range from the location, according to the setting of the SucceedIfCloser flag.

3.5.3. TimePoint

This condition is triggered once a specified time has been reached (so this condition will be successful at or after the indicated time)

3.6. Responses

3.6.1. Manoeuvre to course

This response directs the participant to steer a particular course, depth, and speed - with these values optionally being relative to the detected target (where course, speed, depth available - else current course, speed, depth retained).

3.6.2. Manoeuvre to Location

This response directs the participant to steer a course to a particular location, optionally including values for speed and depth. Where speed or depth are not provided the existing values are continued.

3.6.3. Launch Weapon

This response launches a weapon (read in from the indicated file). The response is normally triggered by a detection condition, from which it determines the bearing (and possibly range) to the target.

3.6.4. Denonate

This response is normally part of the behaviour of a weapon (torpedo or missile) and happens at a particular proximity to a specified type of target.

3.6.5. Transmit Signal

This response allows a participant to send a specific signal, typically reporting a detection. A signal reporting a detection may include absolute or relative directions to the target.

4. Transiting Behaviours

This section describes the set of behaviours which describe how a participant moves through the scenario space. The low-level movement of participants is not part of the behaviour-model, but is represented as the core maneuvering model. This allows the user-controlled movement behaviors to be implemented at a higher level.

4.1. Wander

Author. Ian Mayo

Description. Vessel travels within a region, turning back towards origin once range limit exceeded

Inputs.

- Origin point (lat/long)
- Maximum range to wander out to
- (Optional) speed to travel at
- (Optional) depth to travel at

Algorithm.

1. If speed and depth are provided, and the vessel isn't at that course and depth set demanded course and depth.
2. Measure range from origin.
 - a. If range greater than threshold, calculate course to origin. Add a random offset to this (0 to 120 degrees port or starboard). Set this demanded course

Source. Original invention

Test Case.

1. Origin, demanded course and demanded speed provided. Vessel is near origin within range of the threshold - but not at (achievable) demanded depth and speed. Vessel should adopt that course and speed - then when at range threshold check that new demanded course requested
2. Only origin provided. Vessel is near origin within range of the threshold. Vessel should continue at current course and speed - then when at range threshold check that new demanded course requested

Modelling Guidance.

- This behaviour may be used as a "fallback" behaviour used to keep a vessel in an operations area when no higher priority waterfall behaviours are operative.
- The vessel will not be expected to travel directly through the origin - even if a random offset wasn't applied. The vessel is given the demanded course when reaching the outer perimeter of the wander area, but following the course change it will no longer be heading directly for the origin (assuming it has a non-zero turning circle).

Incorporation. Jul 03

See Also. Online [http://intranet2/coag/asset/help/schemas/ASSET.xsd.html#type_WanderType] schema definition



Note

Note that the Rectangle Wander behaviour is a slightly modified wander behaviour where instead of travelling up to the indicated range from the origin the participant wanders around a rectangular area. Other aspects remain as the wander behaviour.

4.2. Transit

Author. Ian Mayo

Description. Vessel travels between two or more way-points in a straight line - possibly travelling back through the points in reverse or starting again from the start

Inputs.

- Set of way-points, each containing lat./long./depth.
- Transit speed (optional)
- Three-way flag indicating what to do on completion: quit, cycle again through way-points from start, or cycle back through way-points in reverse
- Threshold for how close vessel travels to way-point (large threshold needed for large time steps)

Algorithm.

1. Determine if we have passed next way-point (are we within threshold of way-point location)
 - a. If we have passed next way-point, determine if that was last way-point
 - i. If not, retrieve location of next way-point
 - ii. If we've passed last way-point determine if we are set to travel back in reverse or from start
 - A. If travelling back in reverse or from start, determine next way-point (point before last if backward, point after first if forward)
 - B. If not, return NULL
 - b. If not, retrieve location of next waypoint
2. Remember id of next way-point
3. Determine course to next way-point
4. Return demanded course and maintain speed to next way-point

Source. Original invention

Test Case.

1. Supply vessel at (0,0) with 3 way points (0,1), (1,1), (1,0). Request vessel transits through them at 12 kts, quitting on completion. Vessel should travel through way-points in ascending order, return null demanded status on completion.
2. Supply vessel at (1,1) with 3 way points (0,1), (0,0), (1,0). Request vessel transits through them at 12 kts, reversing back through on completion. Vessel should travel through way-points in ascending order, cycle back through to first, then travel forwards again. Test completes as vessel starts moving forwards through points second time.
3. Supply vessel at (1,1) with 3 way points (0,1), (0,0), (1,0). Request vessel transits through them at 12 kts, travelling through forwards again on completion. Vessel should travel through way-points in ascending order, travel directly back to first, then travel forwards again. Test completes as vessel starts moving forwards through points second time.

Modelling Guidance.

1. The range threshold for deciding if the vessel has passed the way-point is fundamental to this behavior - the range must be achievable within this time step for a vessel at this speed, otherwise vessel will 'step-over' way-point, turn back, head for same way-point again, and so on continuously. Using a particularly large threshold however may result in the vessel changing 'target' to the next way-point before actually visiting this one.
2. An alternative to the use of the Range-threshold would be to identify passing the way-point by it appearing in the stern-arcs: as soon as this occurs the way-point is met and we move on to the next one. *This choice needs further discussion.*
3. This behavior may be used for a traditional transit/long journey, but may also be used as a component in a composite behavior such as prosecuting target at datum: following the detection a single-point transit is created which enables the vessel to travel from where-ever it is to the target point, then drop-out for the next behavior to be used: such as dropping weapon.

Incorporation. Jul 03

4.3. Transit Waypoint

Author. Ian Mayo

Description. This extension of the Transit behaviour allows the specification of how the vessel should navigate the waypoints within the route- using one of waypoint transition modes.

Inputs.

- Route inputs as for Transit behaviour
- Waypoint transition mode (one of On Top Waypoint, Directed On Top Waypoint, Make Waypoint)

Algorithm. As for Transit behaviour, but instead of just determining course to next waypoint produce demanded course (possibly following a delay period) according to the waypoint transition mode.

Source. Transit behaviour, supplemented by Merlin flying guide

Test Case.

1. Supply vessel at (0,0) with 3 way points (0,1), (1,1), (1,0). Request vessel transits through them at 12 kts, quitting on completion. Vessel should travel through way-points in ascending order, return null demanded status on completion.
2. Supply vessel at (1,1) with 3 way points (0,1), (0,0), (1,0). Request vessel transits through them at 12 kts, reversing back through on completion. Vessel should travel through way-points in ascending order, cycle back through to first, then travel forwards again. Test completes as vessel starts moving forwards through points second time.
3. Supply vessel at (1,1) with 3 way points (0,1), (0,0), (1,0). Request vessel transits through them at 12 kts, travelling through forwards again on completion. Vessel should travel through way-points in ascending order, travel directly back to first, then travel forwards again. Test completes as vessel starts moving forwards through points second time.

Modelling Guidance.

1. The range threshold for deciding if the vessel has passed the way-point is fundamental to this behavior - the range must be achievable within this time step for a vessel at this speed, otherwise vessel will 'step-over' way-point, turn back, head for same way-point again, and so on continuously. Using a particularly large threshold however may result in the vessel changing 'target' to the next way-point before actually visiting this one.

2. An alternative to the use of the Range-threshold would be to identify passing the way-point by it appearing in the stern-arcs: as soon as this occurs the way-point is met and we move on to the next one. *This choice needs further discussion.*
3. This behavior may be used for a traditional transit/long journey, but may also be used as a component in a composite behavior such as prosecuting target at datum: following the detection a single-point transit is created which enables the vessel to travel from where-ever it is to the target point, then drop-out for the next behavior to be used: such as dropping weapon.

Incorporation. Jul 03

4.4. Working Transit

Author. Ian Mayo

Description. This extension of the Transit behaviour allows the specification of an activity to perform at regular intervals during the transit.

Inputs.

- Route inputs as for Transit behaviour
- The number of stops to make (at which points the activity will be conducted)
- The activity to perform at each stop

Algorithm. As for transit behaviour, with the exception that the activity is performed at regular intervals along the route. Note that the behaviour cuts off the corners of the transit route, to ensure that turns are only conducted at activity points.

Source. Original definition

Test Case. None yet.

Modelling Guidance. This behaviour has been defined to make complex transits easier to define, particularly when the transit involves frequent stops to perform some action such as Stern Arc Clearance, taking a submarine broadcast, etc.

Incorporation. Sep 04

4.5. Move

Whilst the other transiting behaviours represent absolute movements, the Move behaviour represents an instruction for a participant to perform a relocation relative to it's current location. The relocation is expressed in terms of speed, course, and the distance to travel.

Author. Ian Mayo

Description. Either relocate to another location, or switch to a demanded course/speed/depth

Inputs. INPUT - Either target location with speed

Algorithm. Check aircraft is not already on course. Convert the G pulled to the equivalent ROT. Choose direction of turn that minimises the angle turned through. Turn in that direction at current speed and at a rate of turn ROT until course is as demanded. Default ROT is 3 degrees per second, but in may vary between 1 and 10 degrees. If the angle of turn is exactly 180 degrees then turn right. This maneuver may be combined with the constant speed parts of either a Climb or Dive.

Source. Merlin flying guide

Test Case.

- On a course of 120 and demanded course of 210 ensure that the turn is to the right and takes 30 seconds.
- Repeat using equivalent G to a 3 degree per sec turn.

Modelling Guidance. Atomic element of aircraft maneuver. The limitation that speed changes cannot occur during a turn is chosen to simplify the modelling required.

Incorporation. Aug 03

4.6. Turn

Author. Jon Walters

Description. An aircraft is required to change its course from one value to another, turning by the shortest route

Inputs. INPUT - demanded course, desired rate of turn (ROT) or G pulled.

Algorithm. Check aircraft is not already on course. Convert the G pulled to the equivalent ROT. Choose direction of turn that minimises the angle turned through. Turn in that direction at current speed and at a rate of turn ROT until course is as demanded. Default ROT is 3 degrees per second, but in may vary between 1 and 10 degrees. If the angle of turn is exactly 180 degrees then turn right. This maneuver may be combined with the constant speed parts of either a Climb or Dive.

Source. Merlin flying guide

Test Case.

- On a course of 120 and demanded course of 210 ensure that the turn is to the right and takes 30 seconds.
- Repeat using equivalent G to a 3 degree per sec turn.

Modelling Guidance. Atomic element of aircraft maneuver. The limitation that speed changes cannot occur during a turn is chosen to simplify the modelling required.

Incorporation. Aug 03

4.7. Long Way Turn

Author. Jon Walters

Description. An aircraft is required to change its course from one value to another, turning by the longest route

Inputs. INPUT - demanded course, desired rate of turn (ROT) or G pulled

Algorithm. Check aircraft is not already on course. Convert the G pulled to the equivalent ROT. Choose direction of turn that maximises the angle turned through. Turn in that direction at current speed and at a rate of turn ROT until course is as demanded. Default ROT is 3 degrees per second, but in may vary between 1 and 10 degrees. If the angle of turn is exactly 180 degrees then turn right. This maneuver may be combined with constant speed parts of either a Climb or Dive.

Source. Merlin flying guide

Test Case.

- On a course of 240 and demanded course of 270 ensure that the turn is to the left and takes 110 seconds.
- Repeat using equivalent G to a 3 degree per sec turn.

Modelling Guidance. Atomic element of aircraft maneuver. The limitation that speed changes cannot occur during a turn is chosen to simplify the modelling required.

Incorporation.

4.8. Climb

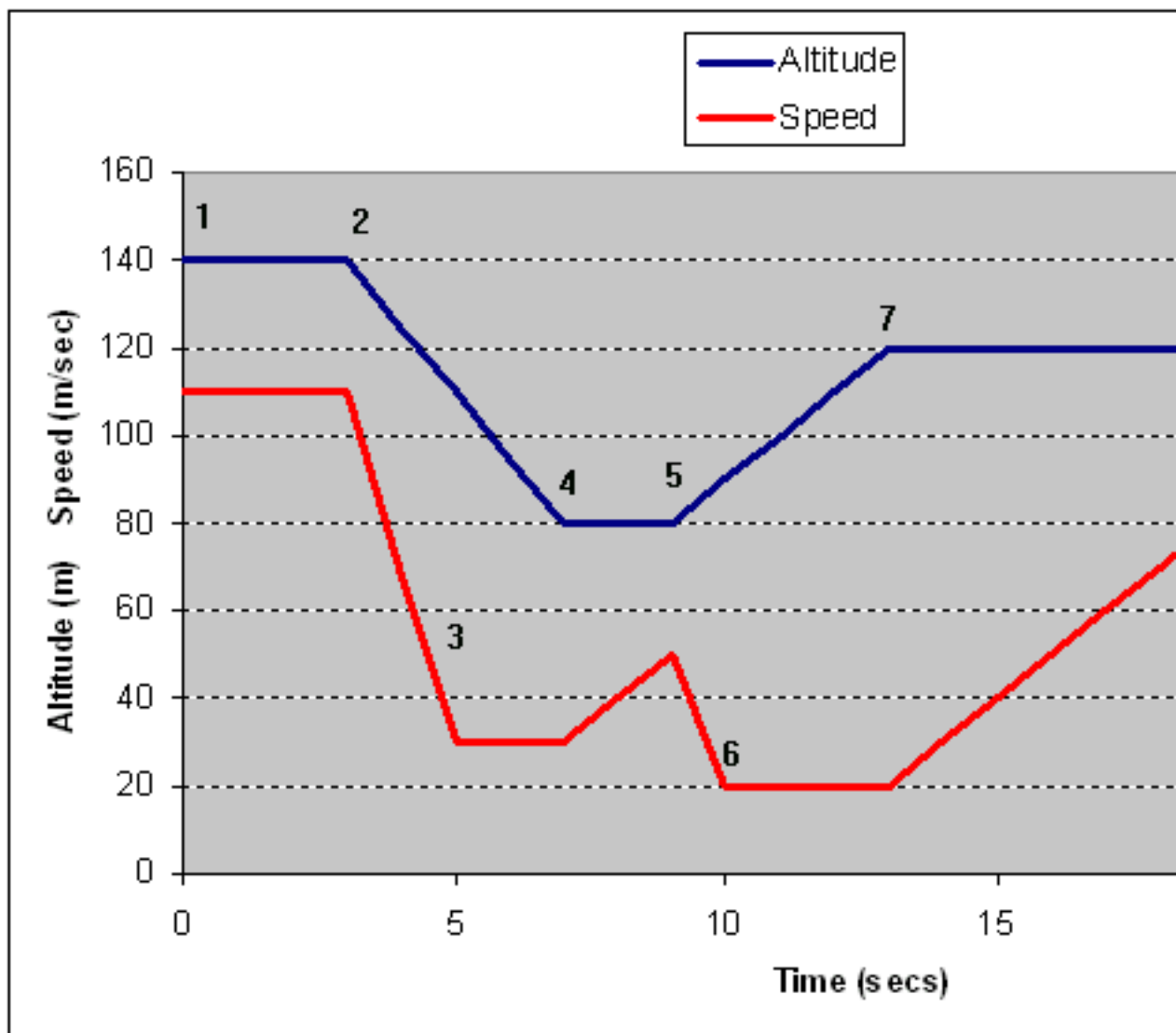
Author. Jon Walters

Description. Aircraft is to climb to a certain altitude

Inputs. INPUT - demanded altitude, rate-of-climb (ROC) and climb speed (CS)

Algorithm. Check that the aircraft is lower than the demanded altitude. If it is then change speed to the CS at the maximum AR or DR allowed then climb at a given ROC at that climb speed until demanded altitude is reached. Then change speed back to initial speed at maximum AR or DR. Default Merlin ROC is 15 feet/sec and climb speed is 35 m/sec but both these parameters may vary - other aircraft will have different values. The constant speed part of this maneuver may be combined with either a Turn or Long Way Turn. Figure 7.1, “Example of aircraft performing dive then climb at default speeds” [47] demonstrates the use of default climb and dive speeds in height alterations.

Figure 7.1. Example of aircraft performing dive then climb at default speeds



1. At point 1 the aircraft is travelling at a speed of 110m/sec at an altitude of 140m.
2. At point 2 the aircraft starts to dive to a demanded height of 80m. The aircraft has a default descend speed of 30m/sec so starts decelerating
3. At point 3 the aircraft has dropped to 30m/sec and deceleration stops
4. At point 4 the aircraft has reached the demanded altitude of 80m and starts to accelerate back to its original speed
5. At point 5 the aircraft starts to climb to a demanded height of 120m, so starts to decelerate to the default climb speed of 20m/sec
6. At point 6 the aircraft has dropped to 20m/sec and continues to climb at this speed
7. At point 7 the aircraft has finished the climb to 120m, and starts to accelerate back to its original speed
8. At point 8 the aircraft has resumed its original speed of 110m/sec

Source. Merlin flying guide

Test Case. From an altitude of 200 feet climb straight ahead for 1 minute - end height is 1100 feet and distance traveled is 2100 m.

Modelling Guidance. Atomic element of aircraft maneuver.

Incorporation.

4.9. Dive

Author. Jon Walters

Description. Aircraft is to descend to a certain altitude

Inputs. INPUT - demanded altitude, rate-of-descent (ROD) and dive speed (DS)

Algorithm. Check that the aircraft is higher than the demanded altitude. If it is then change speed to the DS at the maximum AR or DR allowed then dive at a given ROD at that dive speed until demanded altitude is reached. Then change speed back to initial speed at maximum AR or DR. Default Merlin ROD is -15 feet/sec and dive speed is 60 m/sec but both these parameters may vary - other aircraft will have different values. The constant speed part of this may be combined with either a Turn or a Long Way Turn.

Source. Merlin flying guide

Test Case. From an altitude of 3000 feet dive straight ahead for 2 minute - end height is 1800 feet and distance traveled is 7200 m.

Modelling Guidance. Atomic element of aircraft maneuver.

Incorporation.

4.10. Change Height

Author. Jon Walters

Description.

Inputs.

Algorithm.

Source.

Test Case.

Modelling Guidance.

Incorporation.

4.11. OnTop Waypoint

Author. Jon Walters

Description. An aircraft is required to reach a point in 3D space in the quickest way while maintaining a current speed.

Inputs. INPUT - waypoint x,y,z

Algorithm. Calculate an intercept course and either Turn or Long Way Turn onto it (the quickest way) - then fly a straight leg ending when the WP is reached - then continue on current course. On this leg execute Change Height to waypoint height. If there is no solution for an intercept course (WP is within radius of turn) then fly ahead until solution possible, then make the Turn or Long Way Turn. If there is insufficient time during the straight leg to make the altitude change required at the maximum ROC or ROD then delay the initial turn until there will be a long enough straight leg.

Source. Merlin Flying Guide

Test Case.

- Start at (0,0,200) heading 000 speed 60 m/s and make waypoint at (10000, 10000, 1000) in a time of xxxx (calculate).
- Start at (0,0,1000) heading 090 speed 70 m/s and attempt to make waypoint (-500,0,100). Ensure that a initial straight leg occurs before the first turn.

Modelling Guidance. Element of aircraft maneuver. Normally the user does not directly encounter this behaviour, it is invisibly used within higher level behaviours such as Fly Route, Fly Expanding Square Search

Incorporation. Sep 03

4.12. Fly Route

Author. Jon Walters

Description. An aircraft is required to fly an arbitrary route defined by a series of waypoints. The Fly Route behaviour is a wrapper around the Transit Waypoint behaviour ()

Inputs. INPUT - a series of legs as defined below.

Algorithm. A leg is defined by a start and end WP (x,y,z position) - it has a track (the bearing from start to finish) and a behaviour of the aircraft when it reaches the end on the leg, either fly directly on top the waypoint ('on-top') or turn prior to the waypoint to make good the next track ('make'). Carry out a Directed On-Top WP to the initial WP with an outbound heading of the initial leg track. Carry out either a Make WP or an On-Top WP (followed by a Regain Track to the track of the next leg) depending on the behaviour specified.

Source. Merlin Flying Guide

Test Case.

- Establish route (0,0,1000) - (10000,10000,1000) - (10000,0,1000) - (0,0,1000) and fly the route once with all behaviours set to 'on-top' and once with them set to 'make'. Check behaves correctly.
- Establish route (0,0,1000) - (1000,1000,1000) - (1000,0,1000) - (0,0,1000) and fly the route once with all behaviours set to 'make', a/c speed to 70 m/s and rate of turn to 1 deg/sec. Check aircraft uses On-Top WP and flies route correctly.

Modelling Guidance. Element of aircraft maneuver.

Incorporation.

5. Composite Conditions

5.1. Detection

5.1.1. Description

Condition triggered by a particular detection

5.1.2. Behaviour parameters

- Category of target to detect
- Strength threshold (optional)
- Min range to target (optional)
- Max range to target (optional)
- Relative bearing sector (min/max) to target (optional)

5.1.3. Algorithm

1. Run through current series of detections, check each against category of target. (Also check detection strength, range and relative bearing if applicable)
2. Was validation found?
 - a. If yes, return that detection as result
 - b. If no, return null

5.1.4. Source

Original invention

5.1.5. Test case

Create scenario where target entering then leaving detection range. Check null results, then valid results, then null results again

5.1.6. Modelling guidance

The behaviour of many vessels is depending upon the presence of others (both friendly & hostile). This condition is used to determine if ownship is in contact with others. As a condition, a detection is always combined with a response into a composite behaviour. Typical responses for this behaviour include transit to target location and conducting an evasive behaviour.

Use of the optional parameters allows us to produce a variety of behaviours depending upon whether the target is within or outside weapons range, or in/out of our weapon launch sector.

5.1.7. See also

The general Condition behaviour.

5.1.8. Incorporation

Not yet.

5.2. Location

5.2.1. Description

Condition triggered by proximity to particular location

5.2.2. Behaviour parameters

- Target location
- Minimum valid range (default is zero)
- Maximum valid range

5.2.3. Algorithm

1. Examine the current vessel status, measure range to target location
2. Are we within min & max thresholds of range
 - a. If yes, return range object
 - b. If no, return null

5.2.4. Source

Original invention

5.2.5. Test Case

1. Create scenario where target approaching then passing target location, with location condition with zero min range. Check null, valid, then null is returned
1. Create scenario where target approaching then passing target location, with location condition with positive min range. Check null, valid, null (for too close), then valid, then finally null is returned

5.2.6. Modelling guidance

This condition allows behaviour to be controlled dependent upon where the vessel is in relation to a particular point: allowing a response to be triggered both when we are very near or very far from a particular point.

Uses of the Location condition could include turning back to an origin after travelling a specific range from it, or always avoiding a particular location (wreck, Wolf Rock).

5.2.7. See also

The general Condition behaviour.

5.2.8. Incorporation

Not yet.

5.3. Timed

5.3.1. Description

Condition triggered by passing a specific time

5.3.2. Behaviour parameters

- Target time
- Whether to continue to fire (remain valid) after target time

5.3.3. Algorithm

1. Examine the current model time. Is it after the target time?
 - a. If yes, have we already fired?
 - i. If no, return the current model time
 - ii. If yes, Should we continue to fire?
 - A. If yes, return the current model time
 - B. If no, return null
 - a. If not after target time, return null

5.3.4. Source

Original invention

5.3.5. Test Case

1. Create simple scenario where vessel travels around. Timed (no continuing) condition should be set forward of scenario start time. Check timed condition fires once after target time.
2. Create simple scenario where vessel travels around. Timed (continuing) condition should be set forward of scenario start time. Check timed condition fires (& continues to fire) after target time.

5.3.6. Modelling Guidance

This condition allows vessels to conduct specific behaviours at set times: a timed rendezvous or signal transmission/receipt for example.

5.3.7. Incorporation

Not yet.

6. Composite Responses

7. Tactical Behaviours

7.1. Investigate

7.1.1. Description

Behaviour representing a participant performing investigations of a series of targets. The behaviour keeps track of targets whose investigation is complete, with completed investigations being defined as reaching a defined detection state.

7.1.2. Behaviour parameters

- Category of target to detect
- Level of detection required to complete investigation (one of the detection states)

7.1.3. Algorithm

1. If we have a current target then:
 - a. Run through current detections, see if we are still holding it
 - b. If we are holding it, see if we have reached the required detection level
 - c. If we have reached the required level, add this target to our "investigated list", and clear current target placeholder
2. If we do not have a current target (because we never held one, or because we've just identified one) then run through the current detections, find a valid target which has not already been investigated by us. If we find one, store it as the current target
3. If we now hold a current target, set course towards it

7.1.4. Source

Original invention

7.1.5. Test case

Create searcher in field of valid & invalid targets. Check that valid targets are investigated once only, & invalid targets are not investigated.

7.1.6. Modelling guidance

7.1.7. See also

7.1.8. Incorporation

April 04

7.2. Escort

7.2.1. Description

A vessel (escort) maintains a set range and bearing from another (target).

7.2.2. Behaviour parameters

- Target category (who to escort)
- Allowable error - the allowable error in the range to target. This acts as a buffer preventing continuous (potentially erratic) course and speed changes.
- Bearing - optional value representing bearing from target to escort station.
- Range - optional value representing the range at which the target should be escorted

7.2.3. Algorithm

1. Are we still in contact with the target?

- a. If not, see if we are in contact with another compatible target
 - i. If no compatible target return null
 - ii. If compatible target found, make that current
2. Are we set to station at a set range?
 - a. If so, check we have range to target, if not, drop out
 - b. If we have range to target, calc range to target and check if we are still within the allowable error for range
 - i. If we are out of range, change our speed by 5% up or down as necessary
3. Are we set to station at a set bearing? (as provided with the Bearing Trail behaviour)
 - a. If yes, calc bearing to desired location (relative to tgt) and set demanded bearing accordingly
 - b. If no, just set our course to that of target

7.2.4. Source

Original invention

7.2.5. Test Case

1. Supply vessel (HVU) at (0,0) with 3 way-points (0,1), (1,1), (1,0). Request vessel transits through them at 12 kts, quitting on completion. Vessel should travel through way-points in ascending order. Frigate assigned to escort Supply vessel. Frigate located ahead of HVU in contact range, instructed to escort at 1km on bearing of 145 degs. Frigate should reposition itself on station and adopt speed of HVU. After HVU performs turn, Frigate should reposition itself.
2. Supply vessel as before, submarine assigned to escort for'd of HVU, starting astern. Submarine starts in contact with HVU then loses contact whilst moving ahead - comes to halt.

7.2.6. Modelling Guidance

Being able to perform this behaviour relies on the escort being able to maintain contact on the target vessel with a sensor capable of producing range data.

7.3. Trail

The Trail behaviour is functionally identical to the Escort behaviour within ASSET.. When designing a target involving a Trail behaviour, an opposing vessel is specified using the TargetType instead of a friendly vessel.

7.4. Bearing Trail

The bearing trail behaviour is an extension of the Trail behaviour, utilising the demanded relative bearing attribute as described in the Escort behaviour algorithm.

7.5. Intercept

7.5.1. Description

The Intercept behaviour uses the known course and speed of the target to enable ownship to switch to an intercept course to the target.

7.5.2. Behaviour parameters

- Category of target to detect
- Whether ownship speed change is allowed

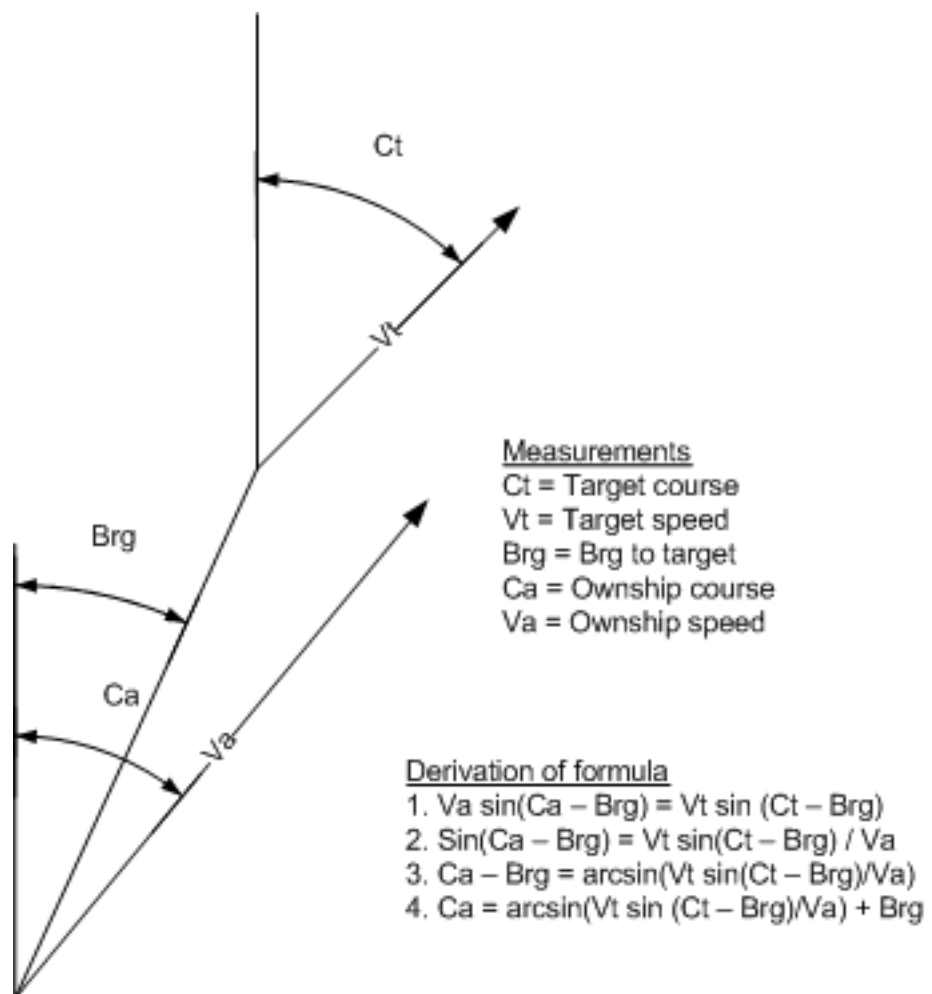
7.5.3. Algorithm

1. If we are in contact with existing target, or participant of target type then
 - a. If intercept is possible, set demanded course to intercept course
 - b. If intercept is not possible, re-perform calculation using maximum ownship speed
 - i. If intercept still not possible, return null demanded status
 - ii. If intercept is possible at max speed, set demanded course to intercept course, and demanded speed to maximum ownship speed according to ownship movement characteristics.



Note

Figure 7.2. Algorithm of intercept course calculation



7.5.4. Source

Merlin modelling guide, Spreadsheet titled "Missile_Approach_02.xls" from Tom Ferrier.

7.5.5. Test case

7.5.6. Modelling guidance

This behaviour was originally produced in support of the force protection scenario modelling, used for the Merlin to intercept unidentified fishing vessels.

7.5.7. See also

7.5.8. Incorporation

September 04

7.6. Prosecute

7.7. Launch Weapon

7.8. Lay Buoyfield

This behaviour is used to represent an aircraft (fixed or rotary) laying a buoy-pattern. The behaviour is configured according to the style of field being laid. For example, a barrier uses direction and separation properties.

ASSET provides functionality to model the laying of Sonar Buoyfields, that is a pattern of sonar buoys which are laid out in a particular pattern.

Although it may not be clearly visible in the following diagrams, all of the buoyfields are located with reference to a kingpin. This kingpin is located relative to a Jig Point (allowing for small, quick adjustments in the location of the field prior to dropping), as specified by a range and bearing.

ASSET supports the following patterns of buoyfield:

7.8.1. Examples of Buoyfields

7.8.1.1. Barrier

A Barrier is a straight line of buoys

Figure 7.3. Example of a barrier buoy-pattern

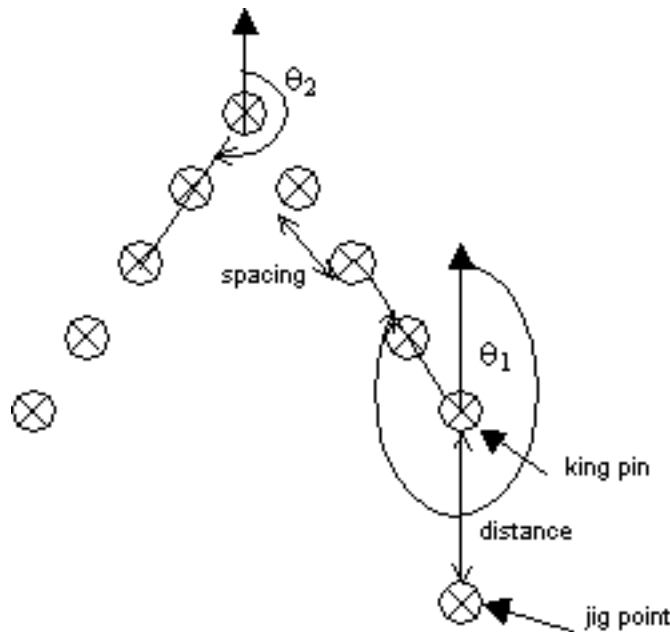


7.8.1.2. Wedge

A Wedge is a bent-line barrier of buoys, specified by buoy spacing and two true bearings. The first bearing is from the kingpin to the centre of the wedge and the second bearing is from the centre point to the last buoy. In ASSET, theta-one is termed the Right Hand Orientation (Orientation1) and

theta-two is termed the Left Hand Orientation (Orientation2). ASSET allows even or odd numbers of buoys, and will ensure that the sides of the wedge remain symmetric.

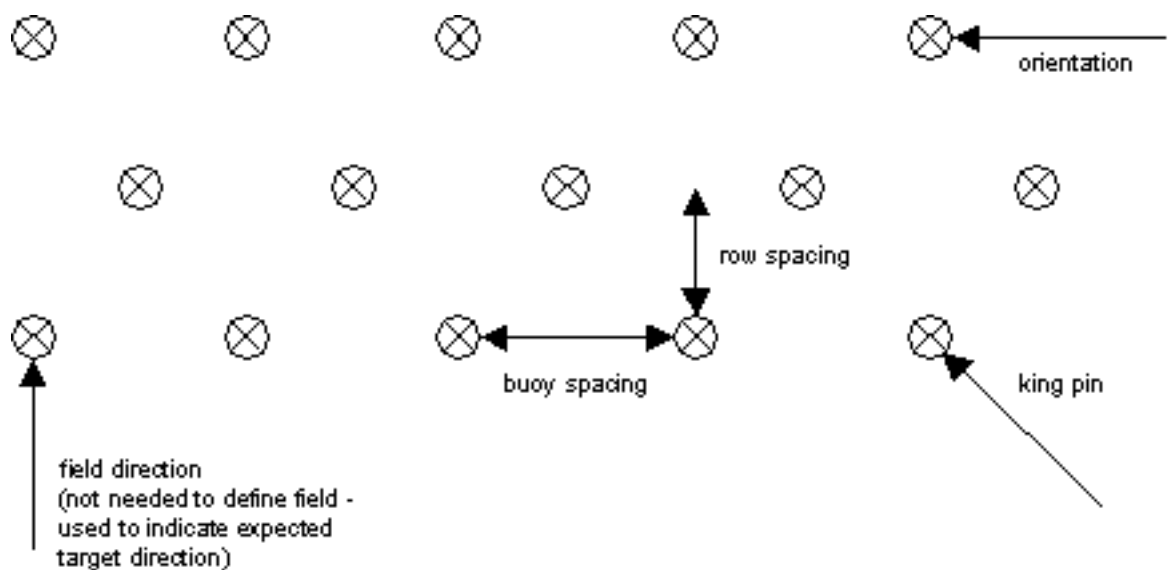
Figure 7.4. Example of a Wedge buoy-pattern



7.8.1.3. Field

A Field is a fixed pattern which contains more than two rows of buoys. The pattern is specified by buoy and row spacing, a true bearing, and an indication of whether the successive rows fall to the left, right, or centre of the kingpin. Note that whilst the diagram shows and the Naval specification order includes the Field Direction parameter, this is not required for buoyfield construction and is not taken from the user. Although the Field Direction is contained in the buoyfield creation order, it is included as an indication of expected target direction of travel, and has no graphic meaning.

Figure 7.5. Example of a Field buoy-pattern



The Left, Right, or Centre offset value is interpreted in the following way:

Left: is interpreted as the "alternate" rows starting offset a 1/2 buoy width further in the field orientation

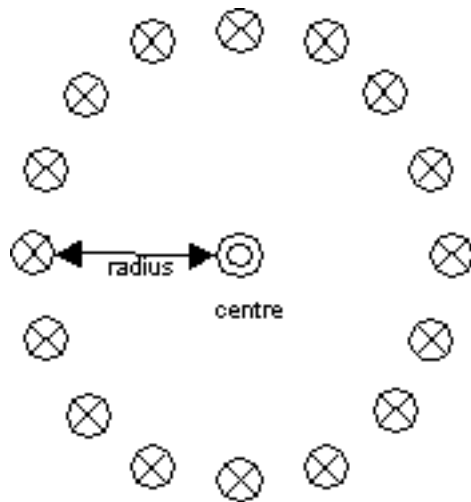
Right: is interpreted as the "alternate" rows being offset a 1/2 buoy width backwards along the field orientation

Centre: is interpreted as there being no offset in successive rows.

7.8.1.4. Circle

The Circle pattern describes a pattern of buoys arranged in a circle, specified by kingpin, radius, and number of buoys. An orientation parameter is also used, which indicates the bearing of the first buoy from the kingpin (centre), since it may not necessarily be to the North. A flag indicates if the circle is being laid clockwise or not.

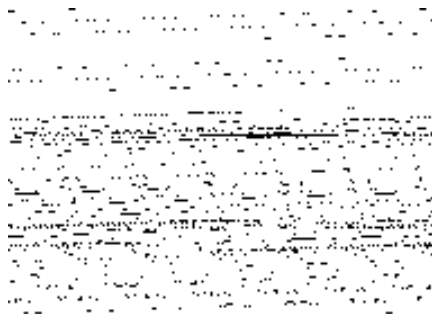
Figure 7.6. Example of a Circle buoy-pattern



7.8.1.5. Arc

The Arc pattern describes a pattern of buoys arranged in a segment of a circle. In addition to the kingpin and radius, an orientation parameter indicates the direction of the mid-point of the arc from the kingpin, and an arc value indicates the arc of coverage each side of the orientation. A flag indicates if the circle is being laid clockwise or not.

Figure 7.7. Example of an Arc buoy-pattern



8. Search Behaviours

Investigating search tactics is seen as a prime use of the ASSET environment. As such a number of high level search behaviours are included as standard.

8.1. Expanding Square Search

Author. Jon Walters

Description. An aircraft is required to carry out an expanding square search as defined in the Merlin Tacman

Inputs. INPUT - a datum point, initial track, track spacing, chirality of spiral (left or right) and an optional maximum number of legs (defaults to 1000)

Algorithm. Construct a route made of a series of waypoints defining a number of legs. WP1 is datum point, WP2 is distance track spacing in direction initial track, WP3 is distance track spacing in direction initial track + (if chirality = right) or - (if chirality = left). The next WP is calculated in the same fashion but is length 2 x track spacing. The leg lengths as multiples of the track spacing are (1,1,2,2,3,3,4,4,5,5 and so on). . Each time an even WP is generated increment a number of legs variable. If number of legs is equal to maximum number of legs stop generating route. Then Fly Route based on these WPs with all legs set to 'on-top'.



Note

Where a vehicle has a large turning circle, some of the early points within the search may be unattainable. Broadly speaking, it's not until the waypoints are approaching a turning-circle diameter apart that the vessel can reliably achieve them. There are a choice of ways in which to model this:

1. When the sequence of points is first created, only insert points which are attainable
2. When the vehicle is travelling through the search path determine if the current point is attainable - if not jump to the next one.

Although the second method is more accurate (since the first must make approximations regarding the actual speed/state of the vehicle as it reaches each waypoint), it is more computationally expensive and involves a more complex algorithm - so the model is implemented using the first method; if a successive point is more than $\frac{2}{3}$ of a turning circle diameter from the previous point (using the vehicle's speed when entering the search), then that point is not added to the search plan. The following diagrams illustrate this. Figure 7.9, "Expanding square search with all points attainable" [62] shows a search path where all search points are attainable at the initial speed, whilst Figure 7.10, "Expanding square search with inner points omitted (unattainable)" [63] shows a search plan where some of the inner points have been omitted since they are closer together than 60% of the vehicle's turning diameter when the vehicle starts the search.

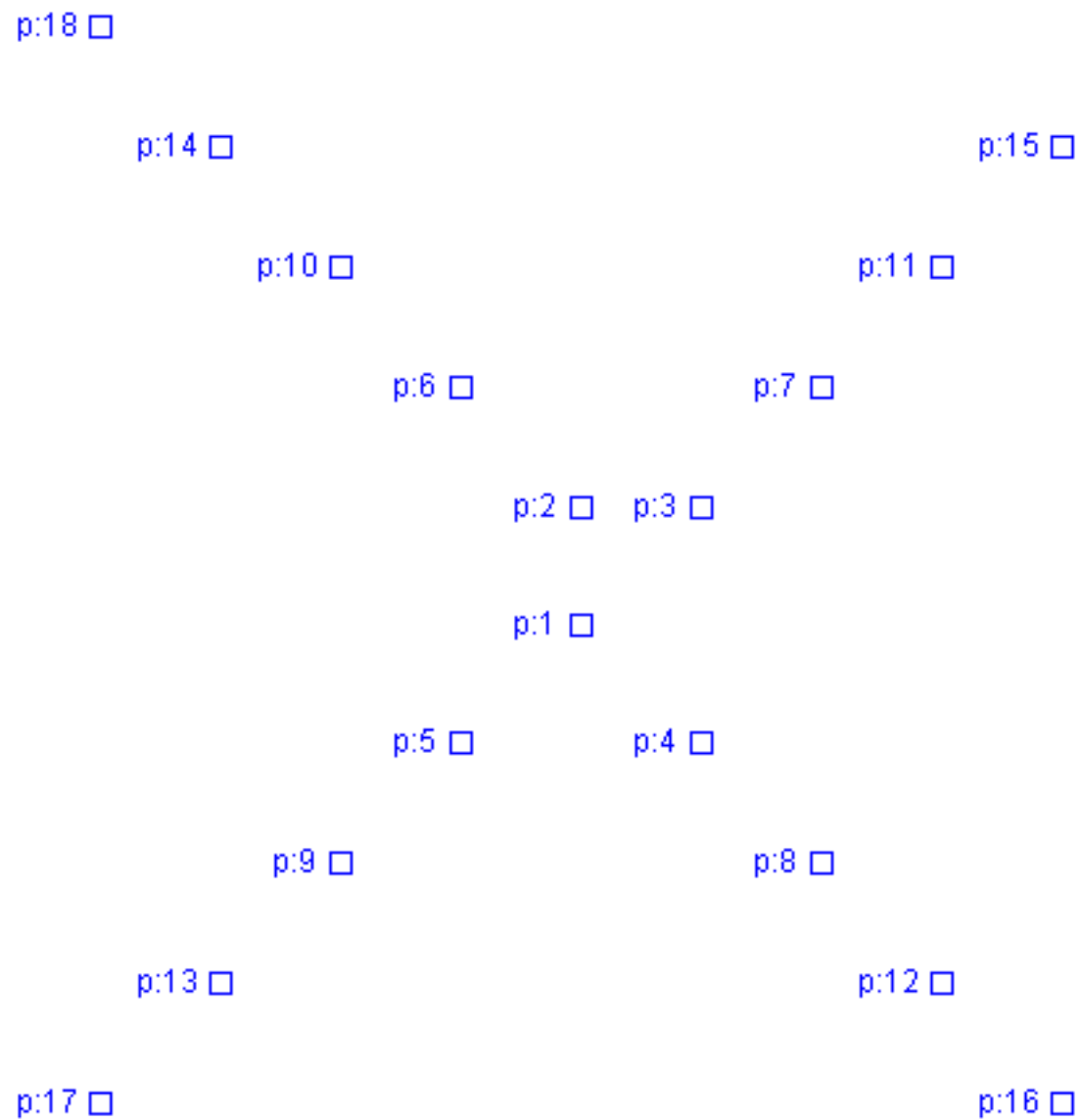
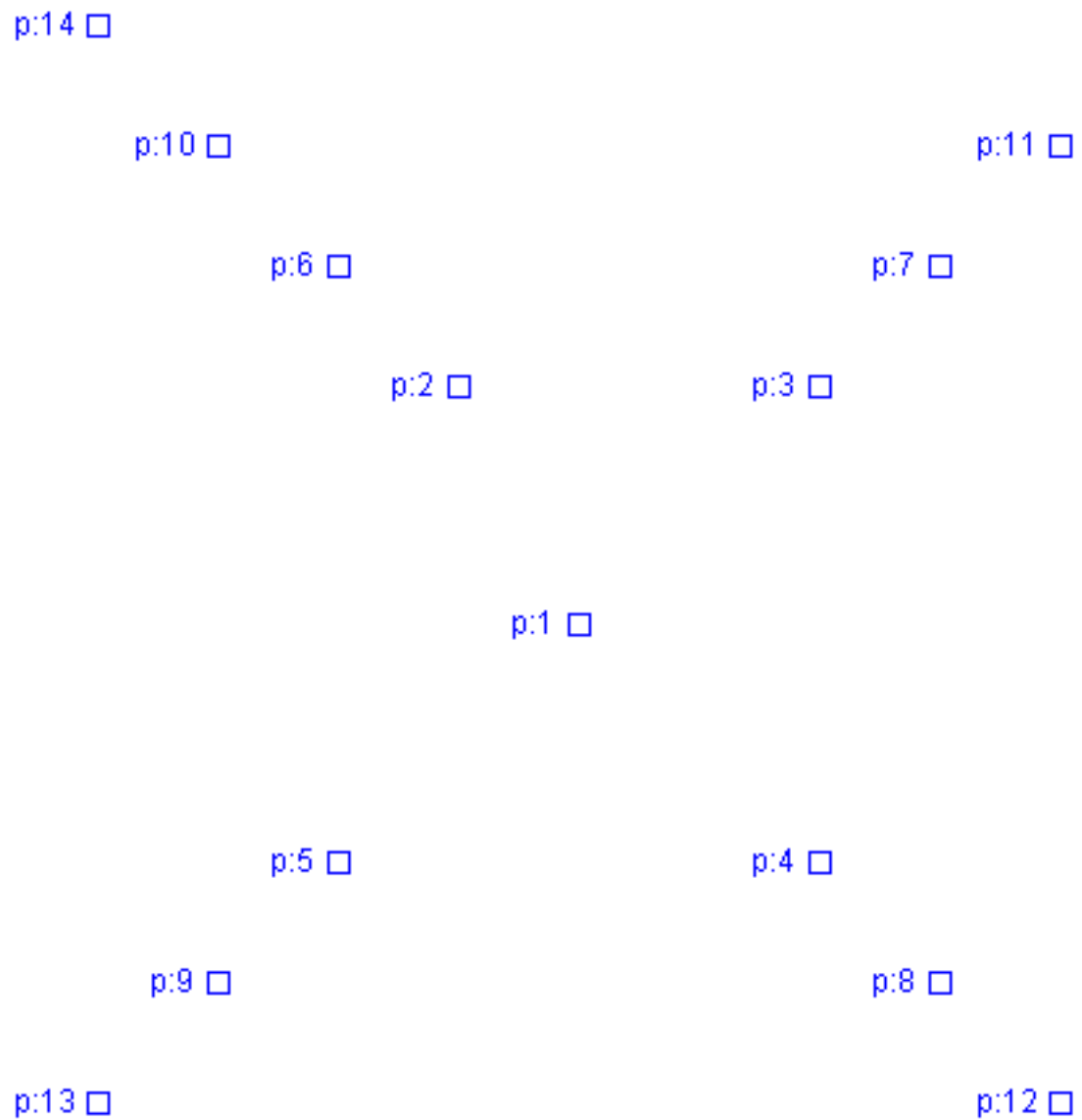
Figure 7.9. Expanding square search with all points attainable

Figure 7.10. Expanding square search with inner points omitted (unattainable)



Incorporation.

8.2. Ladder Search

Author. Jon Walters

Description. An aircraft is required to carry out a ladder search as defined in the Merlin Tacman

Inputs. INPUT - a start point, ladder axis, track spacing, leg length and an optional maximum of legs (default 1000)

Algorithm. Construct a route made of a series of waypoints defining a number of legs. WP1 is start point, WP2 is distance leg length in direction ladder axis - 90°, WP3 is distance track spacing in direction ladder axis from WP2. WP4 is distance leg length in direction ladder axis + 90°. Successive pairs of WP are then calculated in the same fashion, reversing the sign of the 90° modification every time it is made. Each time an even WP is generated increment a number of legs variable. If number of legs is equal to maximum number of legs stop generating route. Then Fly Route based on these WPs with all legs set to "on-top".

Source. Merlin Tacman

Test Case.

- Establish start point (0,0,1000), track spacing 5000, leg length 20000, ladder axis 045°. Set a/c speed to 60 m/s and ROT to 3 deg/sec. Fly Ladder and check that behaves correctly.

Modelling Guidance. Tactical search pattern. This is a geographically fixed search through an area with the start point conventionally fixed at one corner of the area to be searched (the "bottom right" for a northerly ladder axis).

Incorporation. June 2004

9. High level tactical behaviours

Where a participant frequently conducts a pattern of behaviours in a set fashion, these can be represented through high-level behaviours. An example of such a behaviour is a Circular Datum Search

9.1. Circular Datum Search

This search is typically conducted by a helicopter and represents it's response to a submarine detection. The vehicle travels to the datum, lays a circular field of sonar buoys and if it regains contact on the submarine it launches a weapon.

10. Other behaviours

10.1. SSK Recharge

Author. Ian Mayo

Description. SSKs must return to periscope depth in order to run diesel generators to recharge the batteries used for normal propulsion. This process is normally known as snorting. Snorting represents one of the SSKs critically vulnerable states, it is both penetrating the surface and radiating much higher than normal noise levels. A consequence of this vulnerability is that the SSK recharge is a behaviour frequently modelled in ASW tactical development. The SSK snorting vulnerability can lead to complex behaviours.

Inputs.

Min Level	This is battery level at which the SSK will return to the surface to snort (if it is not in contact with any target-types it should be evading)
Safe Level	When the batteries reach this level the SSK will finish snorting
Snort Speed	Whilst snorting the SSK will travel forward at this speed
Evade These	The SSK will cease snorting if it gains contact with any of these categories of target.

Algorithm.

1. Are we in contact with any targets we should evade?
 - a. If we are, we aren't going to snort. Don't execute this behaviour, continue to the next.
 - b. If we're not in contact, check if we are already snorting
 - i. If we are already snorting, check if we have reached our safe snort level
 - A. If we have reached the safe snort level, don't execute this behaviour any further, fall back to the next
 - B. If we haven't reached the safe level, continue snorting
 - ii. If we aren't already snorting, check if we need to snort
 - A. If we don't need to snort, drop out.
 - B. If we do need to snort, head for PD

Source. Original invention

Test Case. Conduct test to verify the following:

- We don't snort when in contact with hostile target
- The correct transition to snort (travel to pd), and correct snort completion (return to original depth)
- The successful termination of snort when hostile target located

Modelling Guidance. An SSK behaviour model may include "Emergency Snort" behaviour at very high priority - but only triggered when the battery level is reaching critically low levels, alternatively the SSK may have an standard snort behaviour at a much lower level - taking the battery back up to full capacity.

Incorporation. Summer 03

Chapter 8. Scenario Control

The modelling guidance recorded so far in this document has all been in support of defining the scenario. The ASSET modelling engine also uses control information to determine how the scenario is run together with how new scenario permutations are defined. Scenario observers define what information is recorded for subsequent analysis during a scenario run, together with under what conditions the scenario should be terminated.

Scenario generation is used within ASSET to combine a scenario template with a control file to produce one or more new scenarios - each based on the template. Variance structures within the control file dictate the ways in which the new permutations are produced.

1. Observers

Observers within ASSET are small, simple software modules which observe an ASSET scenario. Optionally, some observers can trigger the scenario to finish (Referees, such the time observer), and other observers can cause changes in the participant line-up within the scenario. However the normal case is for observers to merely observe scenario events unfolding, either recording information to file at each step or recording summary statistical information on scenario completion.

1.1. Scenario Referees

Scenario referees are observers which are capable of stopping a scenario - either after a particular event/interaction or after a specified time period. Each submits a message explaining the reason for stopping the scenario - messages which may in turn be collated by a batch collator in a summary observer.

1.1.1. Stop on detection observer

This observer stops the scenario once a vessel matching the Category type is detected by any sensor on a vessel matching the watch Category type. Thus, this referee can be used to stop the scenario when a Red SSK is first detected by a Blue Airborne asset.

1.1.2. Stop on proximity observer

This observer stops the scenario if a vessel matching the Category type passes within the specified minimum range of any vessel matching the watch Category type. Thus, this referee can be used to stop the scenario when a Red SSK passes within weapons range of the Blue HVU.

1.1.3. Stop on proximity detection observer

This composite observer stops the scenario once a vessel matching the Category type is detected by any sensor on a vessel matching the watch Category type whilst they are within the specified range. Thus, this referee can be used to stop the scenario when a Red detects a Blue Surface asset whilst within a specified weapons range.

1.1.4. Time observer

This observer stops the scenario after a specified period has elapsed. This is a convenient referee observer that can be used as a fall-back in case scenario events are such that other scenario referees are not triggered.

1.1.5. Remove detected observer

Each time a vessel of the target type is detected by a vessel of the watch type, this observer removes it from the scenario. This is of particular value in a Monte Carlo type scenario run where the use of

multiple participants in one force provides an overall prediction of the behaviour of one participant. In a search problem, removing targets as they are detected focusses the attention of the analyst on the undetected targets (and provides processing efficiencies).

1.2. Recording observers

This set of observers record status information at each scenario step, normally directly to file. They record varying types of information in a number of formats.

1.2.1. CSV Track Observer

This special instance of the Detection Observer records the vessel statuses at each time step, writing the information to file in comma-separated variable format. The comma-separated variable format is directly accessible by Microsoft Excel, which will open the files when they are double clicked. The data recorded to file is x,y in metres, current and demanded course, speed, depth, and the current *activity*.

1.2.2. Debrief Replay Observer

This observer also records vessel status to file, but uses the Replay file format as supported by MWC's Debrief tool. This format tightly constrains the fields recorded, limiting them to location and course, speed, depth. If these fields are sufficient for your analysis then Debrief will normally prove to be the most effective tool for post-event analysis of ASSET runs.

In addition to being able to specify the output filename for your Debrief data, you may indicate what information should be recorded:

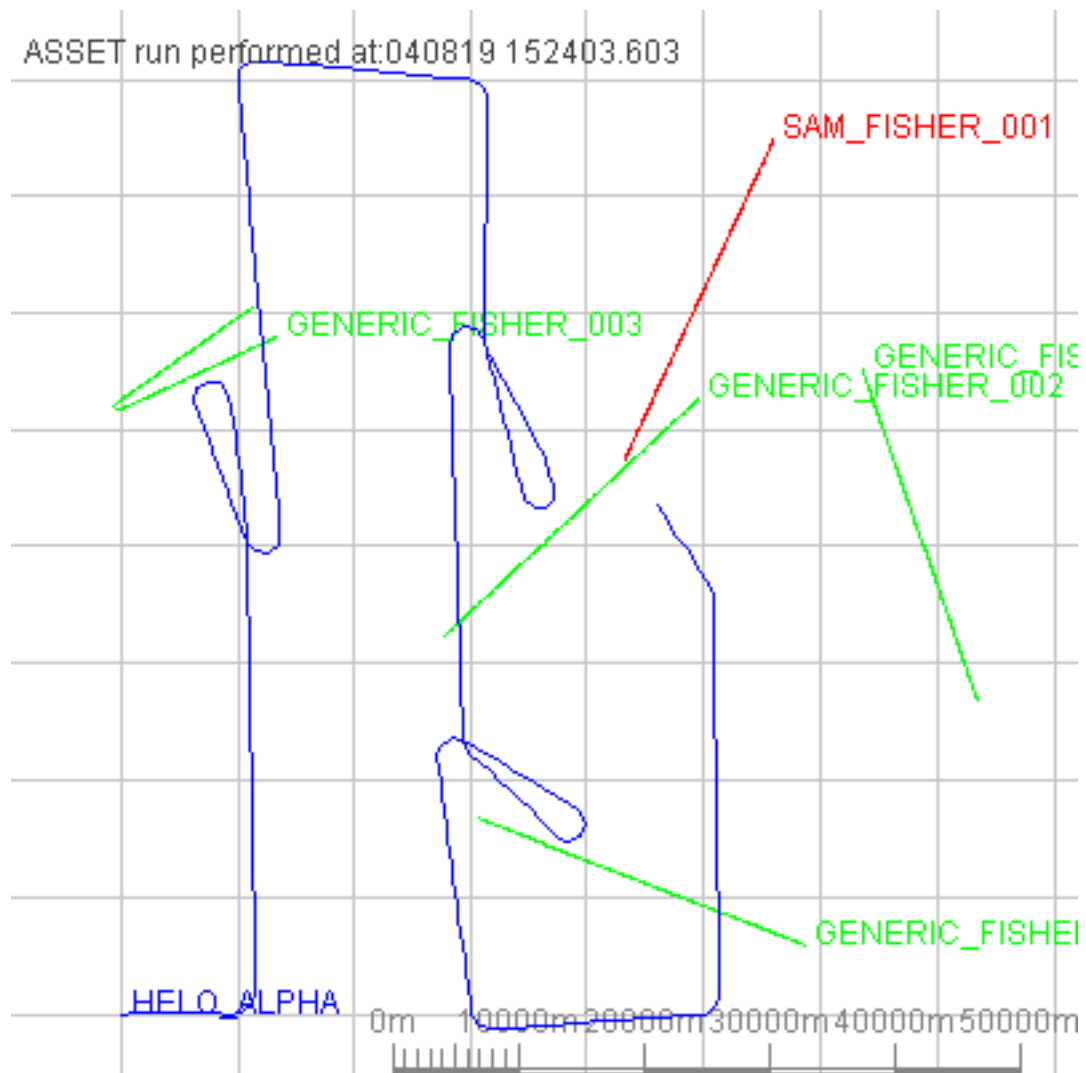
Record Positions	Each time a participant moves, record its new position to Debrief. The position data item contains the correct colour and symbol type for that participant (when known)
Record Decisions	Insert a Debrief narrative entry for each decision made by each participant (where it differs to the last).
Record Detections	Produce a sensor data item for each detection recorded by each participant.

1.2.3. Debrief Deployable Sensor Location Observer

Yes, it's a mouthful. This observer is largely identical to the Debrief Replay Observer, but records the current location of deployable sensors such as helicopter's dipping sonar or a submarine towed array.

1.2.4. Track Plot Observer

This observer records the position of all participants during a scenario run, produces a track plot, and then writes this to disk in png format. The png format may be inserted into Microsoft Word or drag/dropped into Microsoft Internet Explorer. By providing a filename for the track plot image each ASSET run will write it's output to that same file. If that file is already visible in Internet Explorer simply pressing Refresh (F5), will show the updated results plot.

Figure 8.1. Output of track plot observer

1.3. Summary observers

The summary observers monitor events throughout a scenario run, and produce a summary statistic at the end. Some of these observers lend themselves to providing a quantitative record of performance over a large number of scenario runs - drawing the attention of the analyst to the relatively poor or effective performance of individual scenario permutations. Such observers are titled Inter Scenario Observers, and have a pair of flags `Batch Collation` and `Per Case Collation` to indicate whether statistics should be collated across the batch of scenarios, and whether statistics should be collated either against each scenario permutation or against the sum total of scenario runs. See the `Batch Collation` section below for further guidance.

1.3.1. Detection observer

This observer counts how many times any vessel matching the `Category` type is detected by any sensor on a vessel matching the watch `Category` type. Optionally the degree of detection may be specified - only detections reaching or exceeding this level of detection/classification are counted.

1.3.2. Proportion detected observer

This observer counts how many times any vessel matching the `Category` type is detected by any sensor on a vessel matching the watch `Category` type. To do this, the observer determines how

many targets match the target Category, then counts how many of them get detected by a vessel of the watch Category during the run. Using some fantastically advanced numerical processing the algorithm is then able to produce the proportion of targets are detected.

1.3.3. Proportion detected observer

This observer records how close any vessel matching the Category type reaches a vessel matching the watch Category type during the whole scenario run.

1.3.4. Elapsed time observer

This observer records the time elapsed during the scenario run in milliseconds

1.3.5. Final state observer

This observer records the stop reason recorded by the scenario - the cause for the scenario completing. This could be down to an elapsed time (Section 1.1.4, “Time observer” [66]), or particular detection (Section 1.1.1, “Stop on detection observer” [66]).

1.3.6. Time to launch observer

This observer records how the time taken before a participant of the indicated Category is launched (created). Use of the observer can, for example, record how long before the first sono-buoy is dropped or weapon launched.

1.3.7. Batch Collation

The optional Batch Collation data structure is used to control the collation of statistics across a batch of runs. The Batch Collation data structure is only offered for Inter Scenario Observers, and its inclusion indicates that batch collation should be performed. Within the data structure the user is able to specify the Collation Method to control how the collation is performed:

- Count the number of data-values recorded
- Calculate the average of the data-values recorded
- Calculate the average of the data-values recorded
- Record the individual data values
- Provide a list of the individual data values, annotating each with the name of the scenario which recorded it
- Provide a frequency list of data values returned; indicating how many instances of each value were returned from scenarios in the batch

Also, the Per Case Collation indicator controls whether the statistics should be recorded across the whole set of runs, or one result for each permutation of variances, and the only batch processing flag prevents the observer from producing its normal per-scenario file output.

Lastly the optional Filename parameter specifies the results filename to use, and the NoIntraRecording parameter specifies that there should be no intra-scenario recording for the current observer..



Note

In the future the batch collation processing may be changed to export the scenario results directly to an online database.

2. Scenario Generation

Scenario generation is conducted in two ways: ASSET uses control file instructions to generate new instances of a particular participant within a single scenario and also to generate whole new scenarios. Within these two different high level control structures common elements describe how new scenario templates are modified. One or both of the control structures may be used in scenario generation depending on the analysis requirement, with two attributes expressed at the top level:

Output Directory	This attribute specifies the directory where the scenario output files are placed. These files may include copies of the generated scenarios together with results files from any configured observers.
RandomSeed	This seed can be used to initialise the random number generator within ASSET, facilitating repeatable results. When omitted, ASSET will produce different results sets each time a scenario is run (providing there is some element of randomness in the scenario that-is).

2.1. Core scenario generation

Whether ASSET is being instructed to create multiple scenarios or multiple participants it uses the same set of control structures, named variances. The variances are contained in a Variance List structure, and are described in the remainder of this section.

Each variance contains a name and an id attribute. The name attribute is used not processed internally, but used to refer to that variance when outputting scenario results. The id attribute is an XPath expression used to identify which component of the scenario is being modified.

Most of the variances contain an optional attribute which specifies how the random permutations should be generated. If omitted a uniform distribution of new permutations is generated, but the following values are available:

Uniform	Generate random values uniformly across the defined area
Normal	Generate random values with a 3 Standard Deviation gaussian distribution, centred on the mid-point of the defined range. A sample of the output from a normal distribution is shown below.
Normal Constrained	Generate random values with a gaussian distribution, centred on the centre of the defined area, but clipped to the specified range - if a sample is outside the specified range another sample is drawn instead.

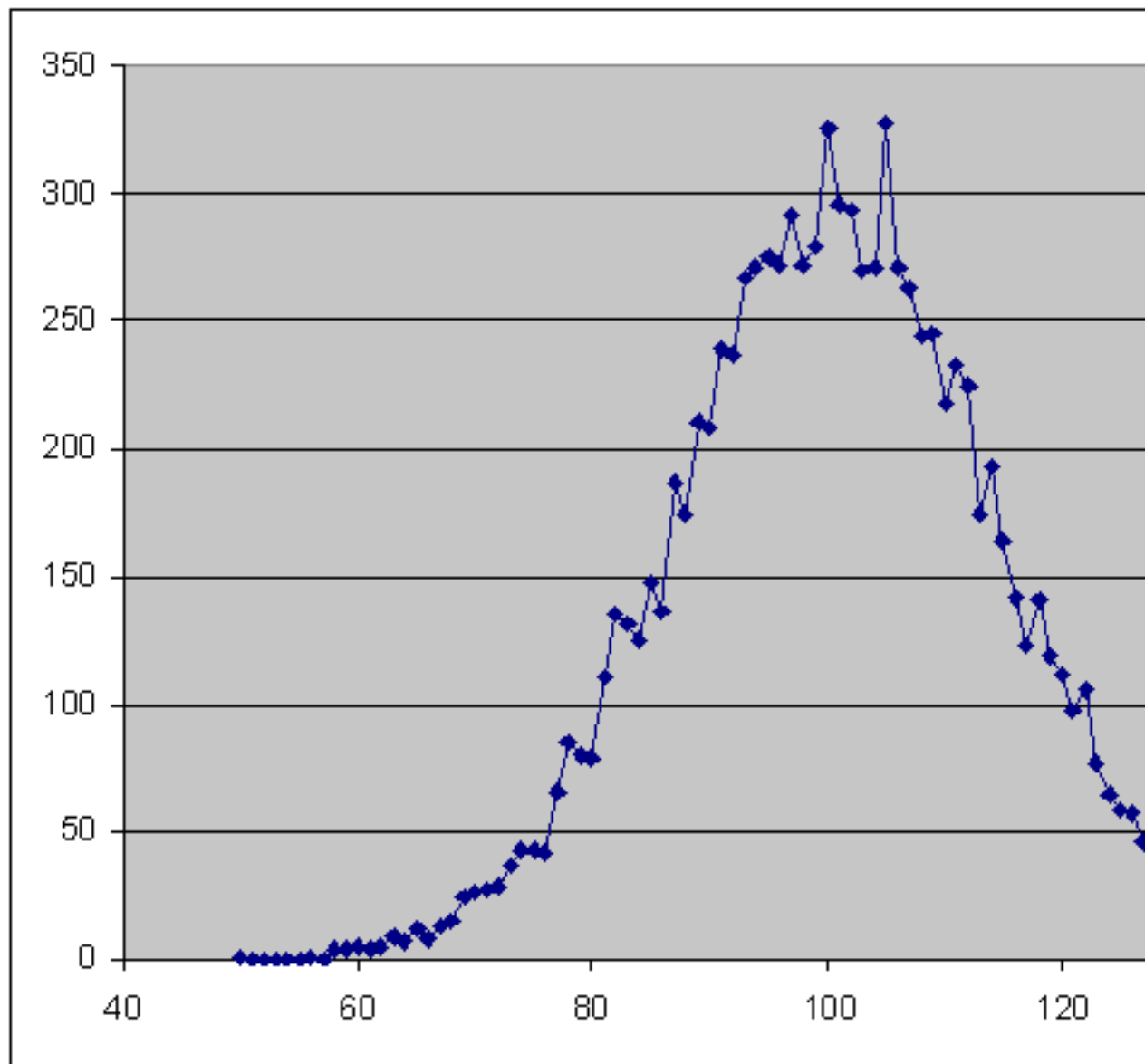
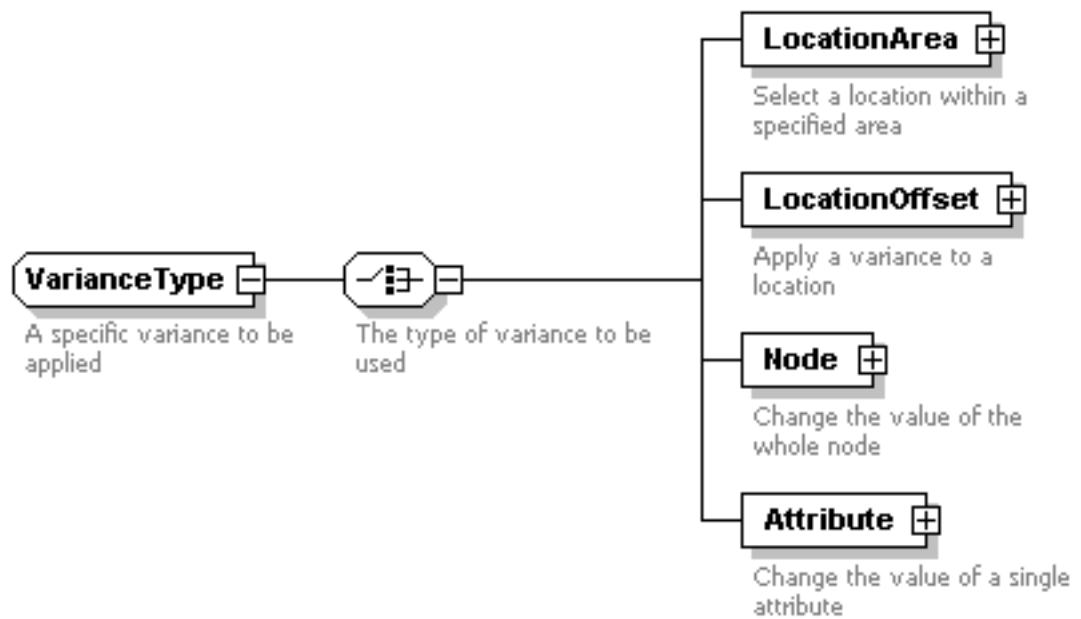
Figure 8.2. Sample of normal distribution (min 60, max 140)

Figure 8.3. General types of variance available



2.1.1. Location Area Variance

This variance specifies an area within which new locations are chosen at random. Locations are distributed either uniformly across the area or with a normal distribution based on the centre of the area.

2.1.2. Location Offset Variance

This variance generates new random locations up to an indicated limit from the subject location, random values generated according to the random model specified.

2.1.3. Node Variance

This variance replaces the contents of the whole referenced Node, using one of a number of supplied Choices.

2.1.4. Attribute Variance

This variance replaces the contents of the specified Attribute, using either a Choice of supplied values or a calculated value within a set Range..

2.1.5. Choice Entry

The Choice structure inserts one of a number of supplied text Snippets, either containing a whole node (when used in a Node Variance) or just the text value to be used in an attribute (when used in an Attribute Variance)

2.1.6. Range Entry

This construct causes values to be calculated within the specified range. Minimum and maximum attribute control the outer range of values created, the format attribute controls the precision (number of decimal places) of values output, and the step attribute allows values to be created at set intervals (whole tenths, units, 5s, 10s, etc). Lastly the number of permutations attribute indicates that while random values should be calculated, a fresh one is not needed each time. Instead, a fixed series of random values are generated, one of which is used each time a random value is requested. The background for this is described below.



Tip

When calculating values within the range, specifying the number of permutations causes that many permutations to be created. Instances from this limited number of permutations are selected in sequence for new random scenarios. When per-case analysis of scenario results is used a continuously random set of values would prevent analysis of grouped results - since every Case Code would be different. Specifying that a restricted number of permutations will be used ensures that groups of matching scenarios can be analysed together.

2.2. Multiple Scenario generation

The multiple scenario generator structure contains a series of attributes containing high level controls over how the scenarios are generated, followed by a series of variances in a variance list. The following scenario generation attributes are used:

Name Template	Whilst the newly generated scenarios files are typically each placed into a new directory it may be necessary to specify a unique name for each one. Provide a filename in this attribute, with %i for a counter, %d for the current DTG and %s for the scenario name.
Max Perms	The maximum number of instances of each permutation to produce. That-is, for each combination of scenario variables, this is the maximum number of scenarios to be produced for it. ¹
Number	The number of new scenarios to produce.

Further guidance in the practice of scenario generation is contained in the Monte Carlo scenario portion of the ASSET tutorial.

2.3. Multiple Participant generation

Whether the user is generating multiple scenarios or just working within a single generated scenario the multiple participant generator generates multiple instances of a single template participant. The multiple participant generator contains a series of sets of participant variances, each creating one or more instances of the specified participant. For each participant variance the following attributes are defined:

Name	This identifies which participant from the scenario template file is being used as a template for these new participants. It identifies the participant using the name attribute of that participant in the scenario file.
Number	This attribute defines how many new instances of that participant should be generated. Thus, in a tactical scenario where you must analyse how many of the target vessels are located you may choose just one new instance of the searcher template and multiple new instances of the target vessel template.
In Parallel Planes	This attribute specifies whether this participant should be processed in a parallel planes mode. Parrallel planes processing causes multiple instances of a participant to run through scenario completely unaware of the others, achieving an efficiency over running through the scenario many times.

In addition to the range of Variances available under multiple scenario generation, a Participant Locator Type offers a convenience way of specifying the range of locations within new participants

will be created. A uniform grid of participants is generated according to the number of participants required.

Chapter 9. Bibliography

References used in support of this modelling guide

Bibliography

[Urick] Robert J Urick. ISBN: 0-9321-62-7. 3rd Edition. Published 1983. by McGraw-Hill, inc. *Principles of underwater Sound*.

[Press] William H. Press. ISBN: 0-521-43108-5. 2nd Edition. Published 1992. by Cambridge University Press. *Numerical recipes in C: the art of scientific computing*.

Appendix 1. ASSET Glossary

1. Glossary

This section contains explanations of terms used within this document

Glossary

Activity	A short phrase describing what a participant is doing in the current time step: such as <i>transitting</i> , or <i>trailling unidentified subsurface contact</i>									
Batch Collation	<p>When an observer is capable of collating results across a batch of runs, a flag is used to indicate whether the user wants this to happen. In addition to specifying that batch collation be activated, the user is also able to indicate the nature of the collation:</p> <ul style="list-style-type: none">• Collect & collate individual results• Calculate sum of individual results• Calculate mean of individual results <p>The user is also able to specify whether this collation is performed on a per-permutation basis or across the total number of runs.</p>									
Blue Force	The Blue Force within ASSET represents the friendly force, whose behaviour can typically be controlled.									
Case Code	Within a multi-scenario batch run where ASSET has produced a scenario using a particular permutation of the user-supplied variances, each particular combination of those variances is termed a Case. A case code is used to uniquely identify each case. Typically this is performed by generating a line of text comprising the name and current value of each variance. Thus the line of text is unique to each permutation. The multi-scenario run may contain multiple instances of each permutation, but statistics may be generated across those permutations by collecting individual scenario performances against their case code.									
Current Status	<p>The current state of a vessel, expressed in terms of course, speed, and location (including depth).</p> <table><tr><td>Time</td><td>The time at which this status was recorded</td></tr><tr><td>Course</td><td>Current vehicle course (expressed in degrees, 0..360)</td></tr><tr><td>Speed</td><td>Current vehicle speed (expressed in knots)</td></tr><tr><td>Location</td><td>Three dimensional location expressed as lat</td></tr></table>		Time	The time at which this status was recorded	Course	Current vehicle course (expressed in degrees, 0..360)	Speed	Current vehicle speed (expressed in knots)	Location	Three dimensional location expressed as lat
Time	The time at which this status was recorded									
Course	Current vehicle course (expressed in degrees, 0..360)									
Speed	Current vehicle speed (expressed in knots)									
Location	Three dimensional location expressed as lat									

		long (degrees), and depth (metres)
	Fuel level	The current fuel level in this vehicle, initially expressed as a figure from 1 to 100 (%) with conversion factors to/from actual data. See the modelling guide (Section 2.3, “Fuel usage” [16]) for more details.
	See the ASSET Source Documentation [http://intranet2/coag/asset/api/ASSET/Participants/Status.html] for more details on the Status object.	
Demanded Status	The demanded state of a vessel, expressed in terms of either individual demanded course, speed and depth, or as a set of waypoints to visit together with demanded speed.. The ASSET Turn Algorithm handles how the vessel achieves this new demanded state. Individual components are specified as follows:	
	Course	The demanded course, expressed in degrees (0..360)
	Speed	The demanded speed, expressed in metres/sec
	Depth	The demanded depth to achieve, expressed in meters
	The higher-level demanded status object consists of a series of waypoints together with a waypoint visiting strategy. The Waypoint Transition model handles the translation between target waypoint lat/long/depth and demanded course/speed/depth.	
Detection	The details of a single detection of a platform by a sensor. The detection contains a varying set of information, since not all sensor produce the same data. Where a sensor isn't able to determine a particular parameter that value is set to null/indeterminable:	
	Bearing	The bearing of the target using this sensor, expressed in degrees
	Speed	The speed of the target determined using this sensor, expressed in knots
	Range	The range to the target from the sensor datum, expressed in yards
	Sensor Location	The 3-dimensional location of the sensor when the detection was performed

	Strength	The strength of the detection, expressed in agreed units for that sensor
	Target Type	The Category (Section 2.1, “Category” [15]) of the target
	Time	The DTG at which the detection was performed.
Green Force	The Green Force within ASSET represents the neutral participants within a scenario, who typically do not interact with Blue and Red participants.	
Monte Carlo	Method of modelling using a large number of model runs to provide a statistical probability of a particular outcome	
Participant	The demanded state of a vessel, expressed in terms of course and speed. The ASSET movement engine handles how the vessel achieves this new demanded state.	
Red Force	The Red Force within ASSET represents the opposing force, against which defensive/offensive tactics are typically constructed.	
Scenario	A specific instance of a synthetic environment containing vessels, sensors, an environment, and time-step instructions.	
Scenario Monitor	An object with an overview of the complete scenario, which is able to handle high-level interactions such as removing destroyed participants following a weapon detonation. The monitor is effectively a <i>glue</i> which connects modelling entities which aren't otherwise aware of each other.	
Sensor	A platform-mounted equipment capable of detecting another platform	
Server	The software process which moves the scenario forward, instructing each platform to move, attempt to detect and other platforms, and make a decision based on that information.	
Variance	In addition to definition of a straight-forward scenario for evaluation using ASSET, data-files can be constructed to include scenario-generation instructions. These instruct ASSET to either insert multiple, subtly different instances of a particular participant (Participant Variance) or to vary one or more attributes of the overall scenario such as weather or start time (Scenario Variance). When the data-file is loaded ASSET processes these scenario generation commands before starting the run.	
XMLSpy	A commercial XML editor providing a data-oriented view of XML data-files. ASSET data-files are strongly structured with little plain text but with most of the information stored as attributes and within the structure itself. There are more text-oriented XML editors on the market, but the word-processor	

type view presented by these does not lend itself to the collation of structured data-files required for ASSET scenario design.

XPath

XPath [<http://www.w3.org/TR/xpath>] is a language for addressing parts of an XML document using a compact non-XML syntax. An XPath tutorial is available on the MWC Intranet. [[../XPathTutorial/General/examples.html](#)]

ASSET User Guide

Updated 2010-01-22 08:51:36

Ian Mayo

ASSET User Guide : Updated 2010-01-22 08:51:36

Ian Mayo

Copyright © 2001,2002

The ASSET User Guide

The ASSET User guide leads you through use of the ASSET Modelling Engine and the standard front-ends supplied with ASSET. The user guide only describes the use of the front ends, it does not supply specific modelling details, this information is contained in the ASSET Modelling Guide . Development and maintenance of the front-ends described in this guide are covered in the ASSET System Documentation

Table of Contents

1. ASSET Engine	1
1. Introduction	1
2. Designing a model run	1
3. Monte Carlo simulation with ASSET	2
2. Command-line ASSET	6
1. Introduction	6
2. Getting hold of ASSET	6
3. Preparing to run ASSET	6
4. Running command-line ASSET	15
3. Command Line Monte-Carlo	18
1. Introduction	18
2. Preparing the data	18
4. Getting to know the ASSET Sensor Model	24
1. Background	24
2. Planning the scenario	24
3. Creating the scenario file (including environment)	24
4. Completing the scenario file	26
5. Creating the control file	31
6. Running through the scenario	32
7. Analysing the results	32
5. Modelling Force Protection	33
1. Introduction	33
2. Define the problem	33
3. Define the environment	33
4. Define the participants, their sensors and behaviours	34
5. Create the scenario file	34
6. Create the control file	36
7. Run through the scenario	39
8. Analyse the scenario results	39
6. ProprietaryWorkbench	40
1. Introduction	40
2. Workbench user interface	40
3. Loading Force Protection scenario	42
4. Moving the scenario forward	45
5. Manipulating the Plot	47
6. Interrogating the scenario	50
7. Eclipse-based Workbench	62
1. Introduction	62
2. Workbench user interface	62
3. Loading Force Protection scenario	63
4. Interrogating the scenario	66
8. Submarine Search	76
1. Introduction	76
2. Preparing the data	77
3. Control file	82
4. Running through the scenario	83
9. Monitor	86
1. Introduction	86
10. Playground	87
11. Factory	88
12. Noise Model	91
1. Requirement	91
2. Tutorial	91
3. Reference	91

Chapter 1. ASSET Engine

1. Introduction

This chapter will introduce you to some of the terms used in ASSET modelling, together with explaining the way in which ASSET is capable of generating multiple scenarios and participants based on template information you have supplied.

Essentially the ASSET modelling engine takes a description of the tactical problem, a set of instructions regarding how to run through the problem and what information to record, and steps forward in time through the scenario until a finish point is reached. It is capable of this without the need for a graphical user interface or user interaction of any kind, though the software can produce graphical plots of vessel tracks either directly (through the TrackPlotObserver) or indirectly (through the use of Debrief data-files).

Model Audit

It is seen as important that ASSET users be able to determine the model build configuration of a particular software version. This could be to aid in reproducing a prior set of results or for recording tools used in a study report.

ASSET uses the CVS (Concurrent Version System) version control application to implement configuration control of source code, documentation and data-files. Consequently a version history is kept of all file changes, including changes to model implementations. Inspection of this version history provides an insight into implementation changes.

All of the text-based results outputs of ASSET include a clear reference to the ASSET build-date - the date when that instance of the software was automatically built. To reproduce a set of results produced using an earlier ASSET build version the source code versions from that date can be extracted and a software build performed.

Further detail regarding ASSET support for Model Audit is contained in the System Guide at Section 1, “Version Control” [13].

2. Designing a model run

The modelling problem is represented in two data-files. A scenario file provides a record of the scenario to be modelled, and the control file indicates to the modelling engine how to run through the scenario, whether multiple permutations of any entities are required, and lastly what non-modelling entities to include (such as an observer which records vessel positions to file).

2.1. Scenario File

The scenario file provides a textual description of the environment to be modelled, and the platforms to model in that environment. It also contains details of the start-time, and the desired model time step.

2.2. Control File

The control file provides information to guide ASSET through how the scenario should be performed - covering the automatic generation of new participants/scenarios, what information to record during the run, and when to stop the run.

2.2.1. Auto generation control commands

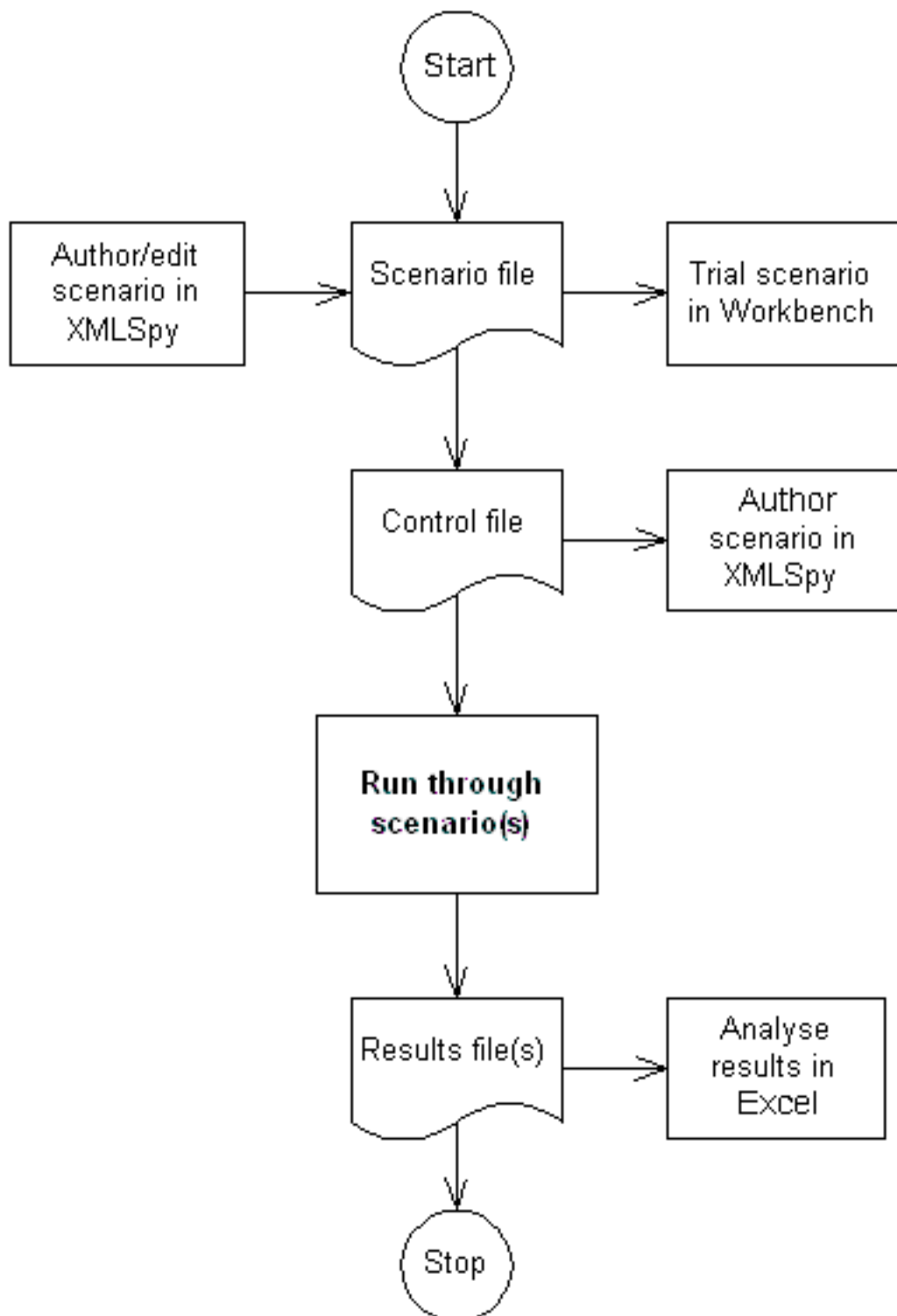
The first block of information in a control file is optional, only required when the ASSET engine is required to automatically generate a number of targets to be used in a *Monte Carlo* type engagement.

2.2.2. Observers

The second type of information found in the Control file is the list of scenario observers. A scenario observer may act as a read-only entity listening out for a particular type of activity within the scenario: such a listener may request that all decisions made by a particular platform be recorded to file. Alternatively a scenario observer may take on a "referee" responsibility, indicating that the scenario should terminate as soon as any red platform makes a valid detection of a particular blue platform. Scenario observers are recorded in the ASSET Source Documentation [<http://intranet2/coag/asset/api/ASSET/Scenario/Observers/CoreObserver.html>], and how they are recorded in the DTD file is recorded in the Schema Documentation [<http://intranet2/coag/asset/help/schemas/ASSET.xsd.html>]

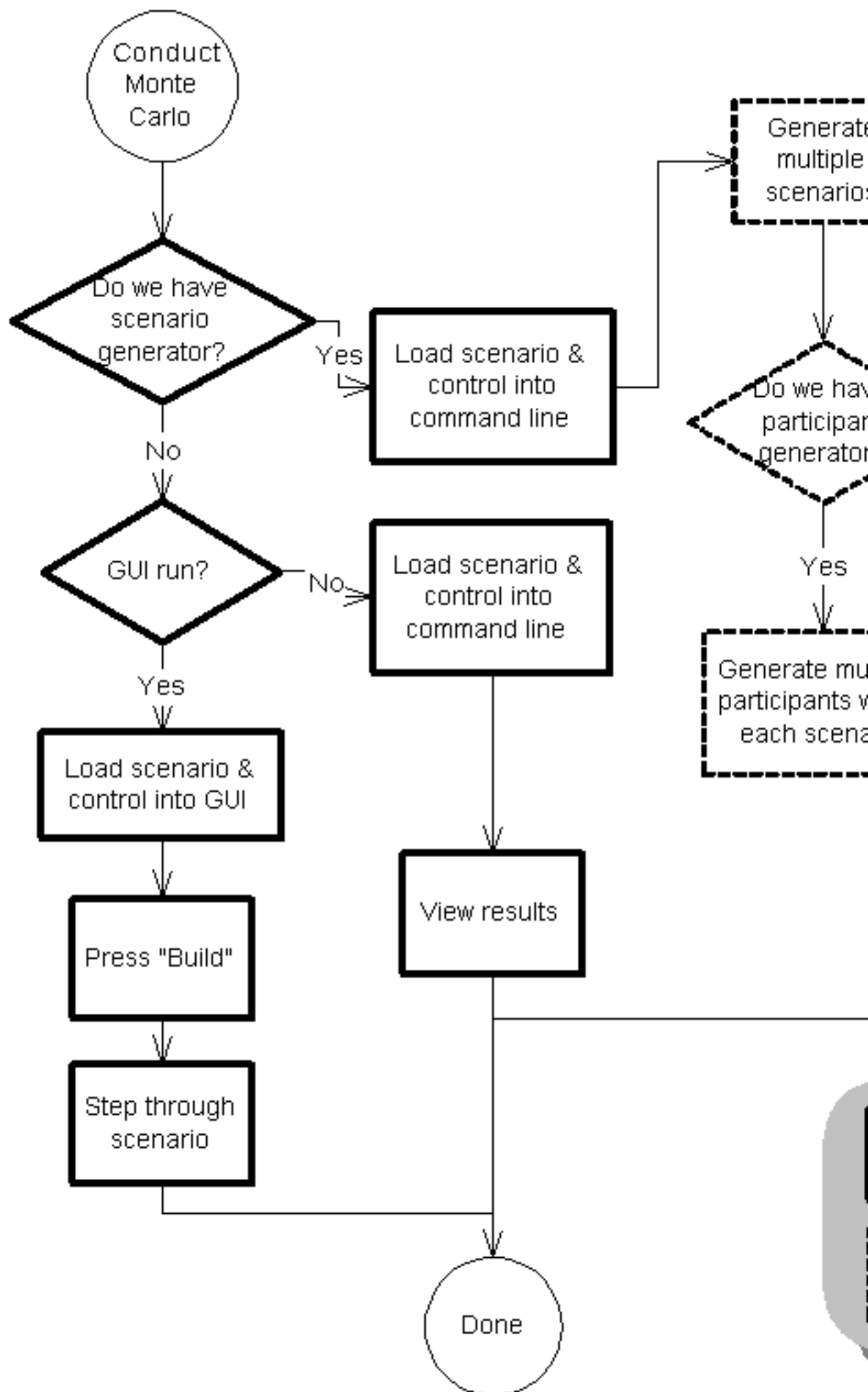
3. Monte Carlo simulation with ASSET

When solving a problem using *Monte Carlo* simulation techniques a number of different strategies for use of ASSET may be applied, some allow for interactive GUI access, others not. The following diagrams illustrate the steps taken:

Figure 1.1. High level Monte Carlo workflow

The analyst first creates a working scenario file (later to be used as template for the new permutations). This scenario file can be loaded into Workbench for initial testing to verify that participants/entities behave as expected. Next the analyst creates the control file which specifies observers to use for the scenario together with scenario generator structures which indicate how multiple Monte Carlo instances are to be created.

The scenario and control files are then processed in either gui or batch mode as shown in the following diagram:



As you can see, the diagram breaks steps down into those undertaken by the operator and those undertaken within the modelling engine itself. Significantly, if multi-scenario Monte Carlo processing is required then it is not possible to interact with ASSET via a gui interface (since we can only see one scenario at a time), though since the control files are all written to disk prior to running, it is quite easy to load the scenario together with one of the control files and run through a *normal* ASSET run - possibly to investigate particular scenarios in detail after viewing their comparative performance.

The batch-mode approach to Monte Carlo modelling is covered in Chapter 2, *Command-line ASSET* [6], whilst the interactive GUI approach is covered in Section 1, “Introduction” [40].

Chapter 2. Command-line ASSET

1. Introduction

The command-line version of ASSET provides a direct way of getting the ASSET modelling engine to run through a scenario file/command file combination. This section of the User Guide will talk you through obtaining ASSET, then preparing and conducting a scenario run.

2. Getting hold of ASSET



Note

For the first two phases of ASSET development, expert users will be receiving and using the software through a more ad-hoc distributions. Users download and uncompress a Zipped archive containing the ASSET software, its libraries and any supporting documents (help-files and schemas). Until the MWC server re-arrangement is complete the downloadable archive will be stored in `nab:\\td_submarines\\info\\asset`. The individual zip files are named according to the build date - obviously the newest one is the one to download and unzip (probably into a directory such as `c:\\asset`).

Right, the first thing you need to do is to get hold of the ASSET software itself, from Applications folder of the NAB. The software is available in two versions, *complete* and *update* - depending upon whether you want a complete fresh installation or just to download the changed files. If you don't already have ASSET installed on your machine, double-click on the *install.exe* file, and ASSET will be installed by default into your `c:\\program files\\ASSET` folder. If you do already have ASSET installed just download the newest update zip from the updates folder - and extract the files into the relevant places.

3. Preparing to run ASSET

So, presuming you have successfully obtained and installed ASSET, we are ready to start.

In order to perform an ASSET simulation, you need to give it a synthetic environment to work with, and tell it what you want doing with that environment. For the purposes of this tutorial we are going to create simple versions of the two files necessary for a command-line ASSET run: the control file and the scenario file. The instructions lead you through creating the files using the *XMLSpy* editor, though any other data-oriented XML editor will do.



Note

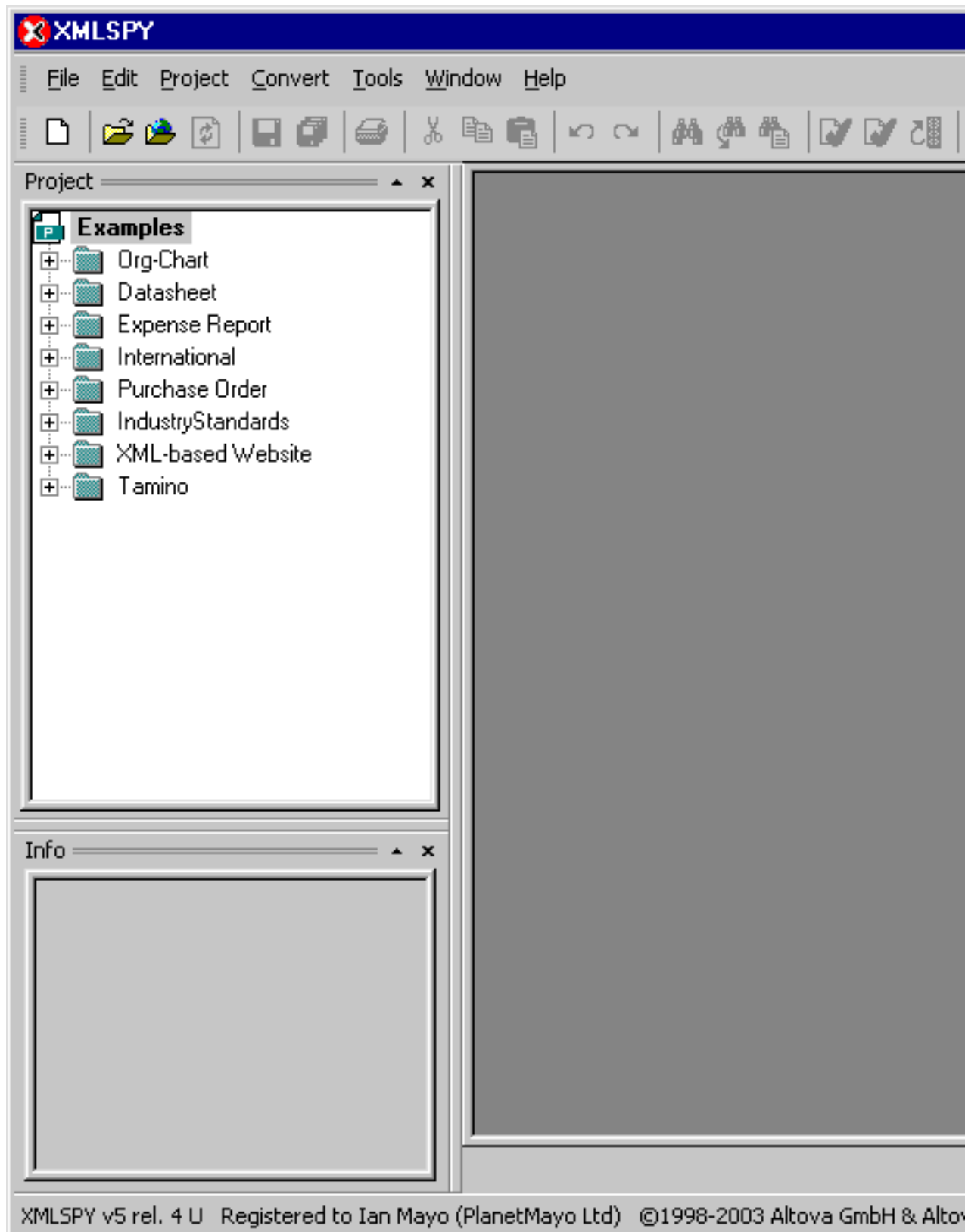
In 2009 ASSET switched to using an internal XML editor. Use of the internal XML editor is very similar to the steps described here, but is covered in detail in the *Getting started with ASSET* cheat-sheet. The cheat-sheet doesn't quite have the breadth that's covered in this document - so this content is worth keeping.

3.1. Creating a scenario file

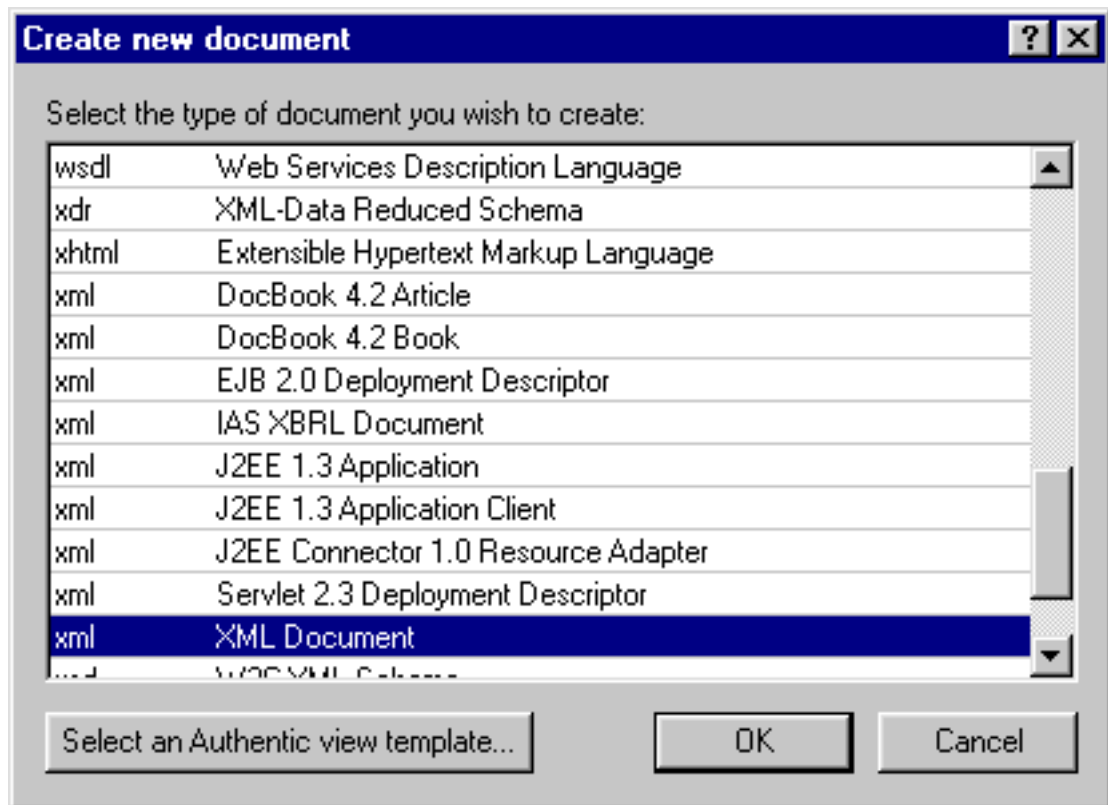
Right, we're going to break down creating the scenario file into two stages: first we define how the whole scenario behaves (the top-level guidance) then we define the participants themselves.

3.1.1. Top-level guidance

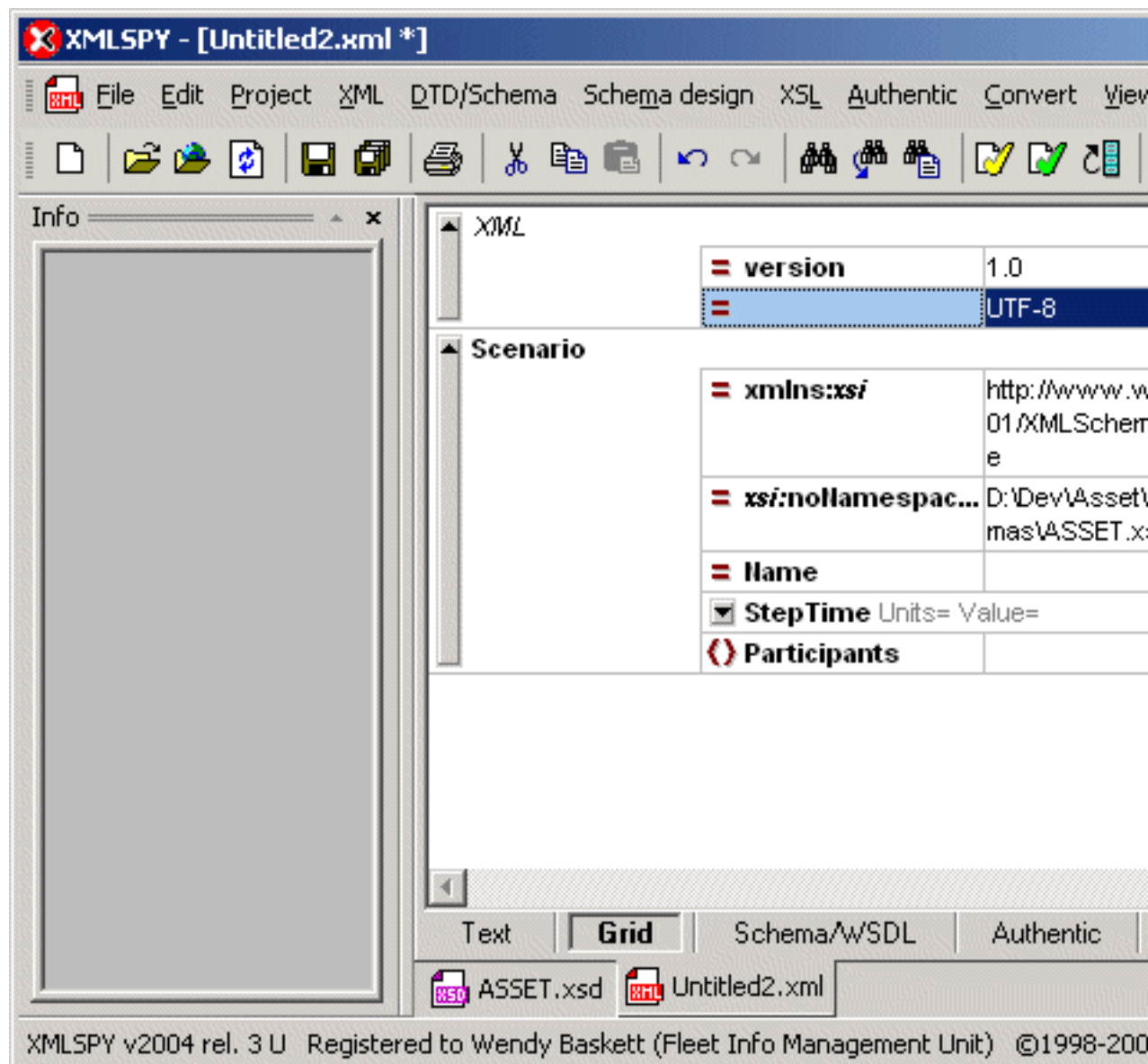
The scenario file is the meatier of the two files, containing a description of both the environment we will be working within and the participants which it contains. So, first you need to open up XMLSpy - which will give you a display like that shown below:

Figure 2.1. XMLSpy opening screenshot

The next step is to open a blank XML document, so select New from the File menu, to get the new file dialog:

Figure 2.2. XML new file dialog

As you can see, there are all sorts of files which can be created, but we're just going to create a plain old XML data-file - so double-click on xml XML Document. The next dialog shown will ask you if you want to assign a DTD or Schema for the data-file. It's the Schema which specifies the structure and form of the scenario file, so yes, we certainly do want to specify a Schema - make sure Schema is selected, and press OK. From the file-selector dialog which appears next, click on the Browse button, navigate to your ASSET installation directory and select ASSET.XSD. You will now be invited to select the root-element, which dictates what type of file you are going to define. Select Scenario.

Figure 2.3. Blank scenario loaded in XMLSPY**Tip**

Now, in your view, you may not have the 3 information windows visible on the right-hand side (Elements, Attributes and Entities). They really help in the creation of a scenario, so make them visible by selecting Entry Helpers from the Window menu.

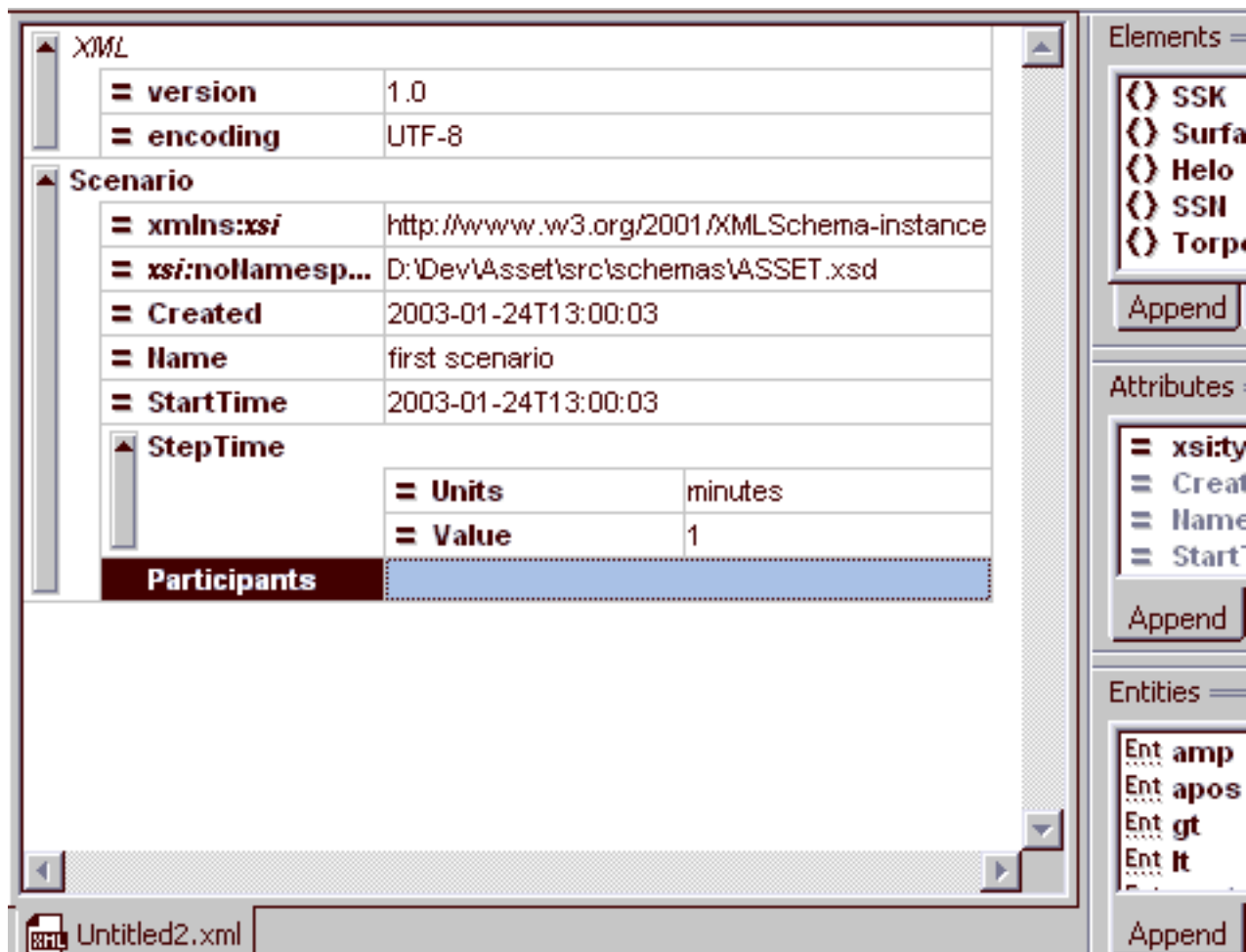
So, as we look at the data visible in the XMLSpy window we can see three sets of information: the XML identifier (which is always present and doesn't relate to our use of XMLSpy), the DOCTYPE specifier of scenario (which tells XMLSpy what we want to build, and what it's made up from, and lastly the scenario itself. Currently the Scenario is empty - it has the preconfigured but unused box titled , Name. Let's fill them in to get started. First type in a Name for the scenario; first scenario should do for this time. With the cursor in the Name box you, if you look at the Append tab of the Attributes toolbox on the right-hand side of XMLSpy you will see a list of optional Scenario attributes. On of these is StartTime. If you single-click on the start-time attribute to select it, the Info window will update to show guidance related to the start-time attribute. Next, double-click on the StartTime attribute in the list, and XMLSpy will add it to the scenario. Into the Start Time box type today's date in the form yyyy-mm-ddThh:mm:ss (such as 2003-01-24T13:00:03). As you select the are typing into the StartTime box, notice

this also provides explanatory information Info window. This explanatory information is available for most of the available information types. Also insert a `Created` item, again giving it today's date. Moving down through the list of core scenario information, the `Step Time` box represents the amount of scenario time we move forward at each step. First click on the down arrow next to `StepTime` to make it's attributes editable. Now double-click to the right of the `Units` attribute and select minutes from the drop-down list. Lastly enter the figure 1 as the value. This will instruct ASSET to move the scenario forward 1 minute at each model step.

3.1.2. Inserting participants

Right, that's the top-level information done. Now we're going to insert our first participant. First put the cursor into the Participants box, then switch the Elements box to show the Add child list. This list shows the types of vehicle which can be inserted as participants (as shown below)

Figure 2.4. List of available participants

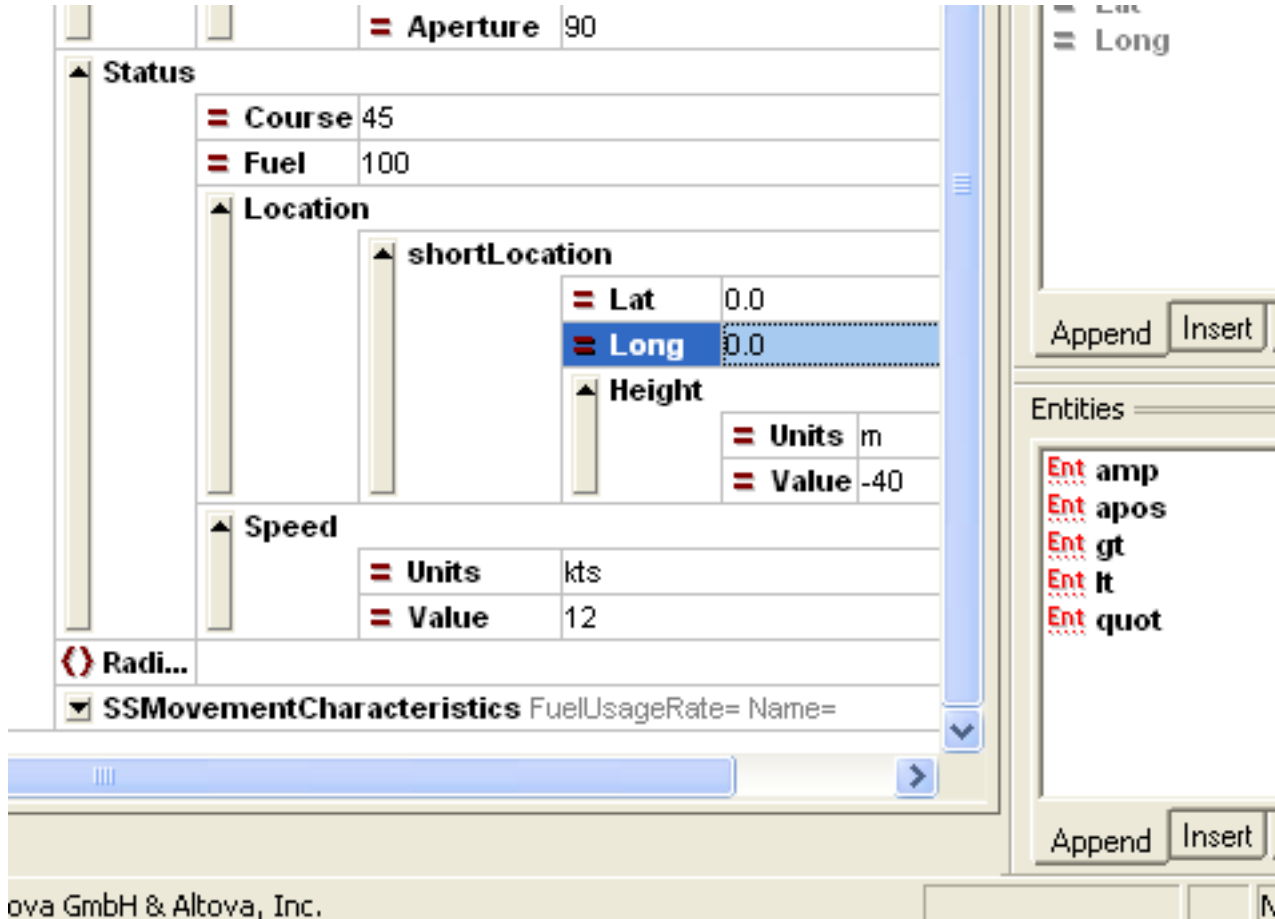


Double-click on the SSK item, and a skeleton SSK will be inserted into the list of participants - pre-populated with it's essential elements. Now let's rattle through its attributes; start off by entering `SSK_1` as the Name. and `0.08` as the ChargeRate. Now drop-down the Category element to show it's constituent types: selecting `Subsurface` for Environment, `Red` for Force and `Submarine` for Type. Now put the cursor into the `SensorFit` element, and select `OpticSensor` from the Add child tab of the Elements toolbox. The optic sensor (periscope in this instance) only has one attribute, its minimum height - the height at which it becomes operable - enter `-20` metres for this.

Next enter the Status information; open up the element's attributes and enter `Course` and `Fuel` values of `45` and `100` respectively. Enter the speed as `12 knots` using the drop-down list of units supplied. We also need to specify where the SSK is starting from, so open up the `Location` object, its `shortLocation` child, and enter `0.0` for the Lat and Long . Height is an optional element

within Location elements - but certainly applicable for a submarine. So, to insert the optional Height attribute, first make sure the cursor is inside the shortLocation element (on either of the Lat or Long elements), and double-click Height from the Insert tab of the Elements toolbox. Set the Height to -40 metres (-ve heights are used to represent depths in ASSET), to look like the next figure.

Figure 2.5. Setting the location for the initial status



Before we move onto the Waterfall element (as described above in Section 3.2, “Waterfall” [38]), we'll fill in the more simple elements. First we have the RadiatedCharacteristics. First click on the RadiatedCharacteristics element, then look at the Add child tab of the Elements toolbox. We can see that so far there are three types of radiated energy defined. Double-click on Broadband to allow us to define the broadband radiated noise characteristics for the SSK. Into the (currently simple) noise model enter a value of 168 for the BaseNoiseLevel. Into the SSMovementCharacteristics element enter the following values for FuelUsageRate Name MinSpeed MaxSpeed, AccelerationRate, DecelerationRate MinHeight MaxHeight, DefaultClimbRate DefaultDiveRate, and TurningCircle: 0.0008, SSK_Performance, 0.1 kts, 16 kts, 1 kt/s, 2 kt/s, -200m, -10m, 1 m/s, 2 m/s, 700m.

Next onto the behaviour of the SSK, which we'll record using the Waterfall element. This goes immediately after the Status element, so first select the Radiated Characteristics element, then double-click Waterfall from the Insert toolbox to place it after Status. The Waterfall behaviour is a container that describes a prioritised series of behaviours for the SSK. Once the new Waterfall element is selected, a large array of child behaviours are available from the Add Child tab of the Elements toolbox/ Clicking on any of them will show a brief explanation in the Info window. We're going to start off simply, with an SSK which just wanders around whilst trying to keep its batteries topped up. So, put the cursor into the Waterfall element, then double-click SSKRecharge from the Add child tab of the Elements toolbox. Set the recharge parameters as Name: : keep topped up, , Min and Safe Level: 10 and 60, leave EvadeThese blank, and finally insert a SnortSpeed of 3 knots.

Next insert the Wander behaviour. Enter `Just_wandering_around`, 12 nm, 14 knots, and -60m as the values for Name, Range, Speed and Height. Fill in a value of 20 kilometres for the Range. We're going to swap the shortLocation for a longLocation. To do this, put the cursor in the shortLocation and double-click longLocation from the Insert tab of the Elements toolbox. Now the longLocation is in our Wander behaviour, delete the shortLocation. Enter a value of 5 minutes North, 5 minutes West (with zero for the other location attributes). The behaviours in the Waterfall should look like the screenshot below:

Figure 2.6. SSK Behaviours

▲ Waterfall			
▲ SSKRecharge			
= Name		keep topped up	
= MinLevel		10	
= SafeLevel		60	
⌘ EvadeThese			
▲ SnortSpeed			
= Units		kts	
= Value		3	
▲ Wander			
= Name		just wandering around	
▲ Range			
= Units		nm	
= Value		12	
▲ Location			
▲ longLocation			
= LatDeg		0	
= LatMin		5	
= LatSec		0	
= LatHem		N	
= LongDeg		0	
= LongMin		5	
= LongSec		0	
= LongHem		W	
▲ Speed			
= Units		kts	
= Value		14	
▲ Height			
= Units		m	
= Value		-60	

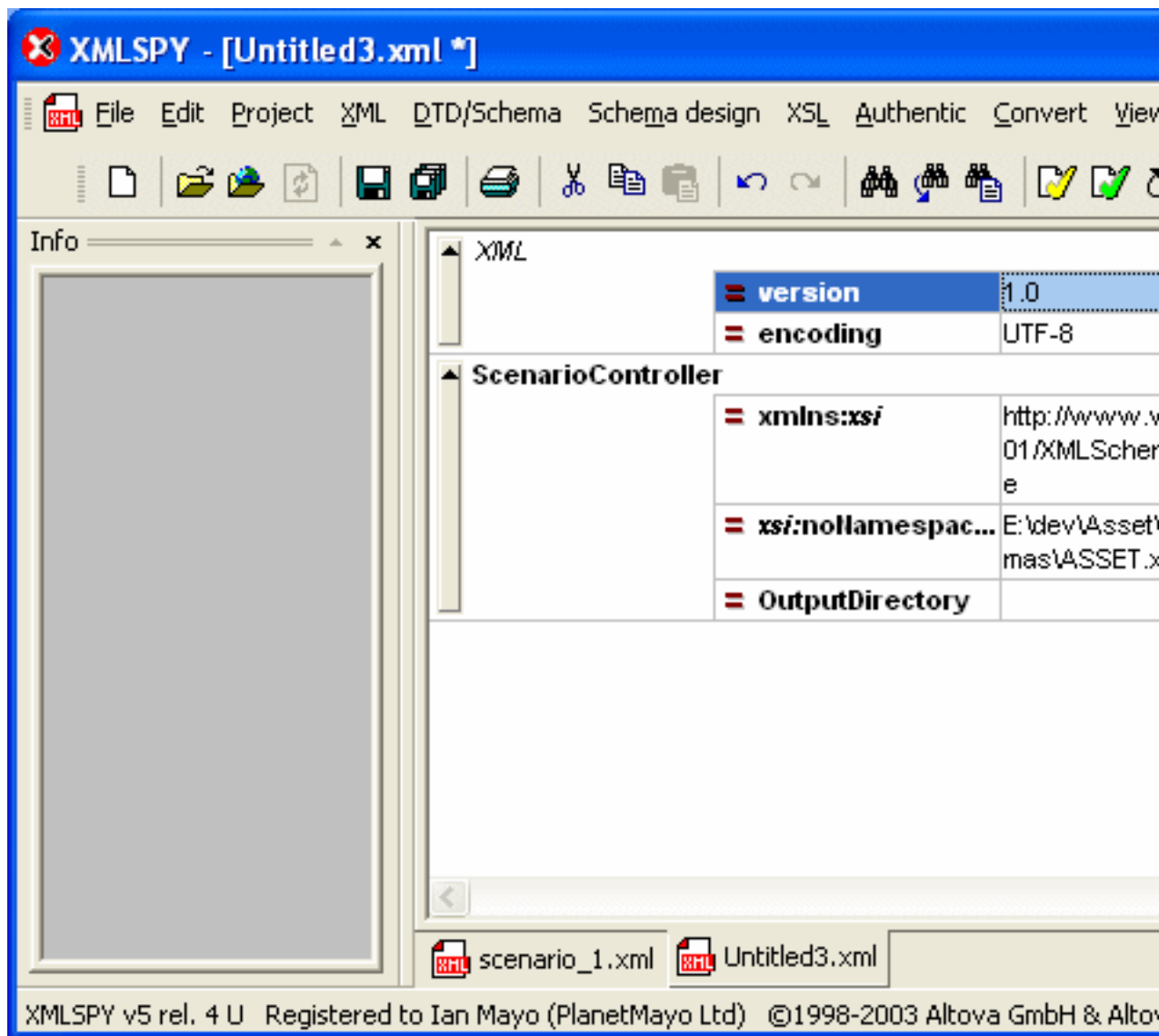
Ok, save the file in your ASSET installation directory as `scenario_1.xml`

If we click on it, then look at the Add child tab of the Elements toolbox we can see the range of behaviours available.

3.2. Creating the Control file

Next we have to create the control file - used to indicate to ASSET what information we want to record to file, and how long we want the scenario to run for. So, first go to XMLSpy, and create a new, blank document. Indicate to XMLSpy that you want to base the document on a schema, and select the ASSET schema, as described above in Section 3.1.1, “Top-level guidance” [7]. From the list of optional root elements, select ScenarioController to give a view like that below.

Figure 2.7. Initial view of control file



Start off by setting the OutputDirectory attribute to the current directory (".") - a single full-stop character).

The scenario controller element we've inserted handles both a list of observers and a set of commands for generating multiple participants/scenarios (as can be seen from the list of applicable child-elements at the top-right of the screenshot). For the purposes of this tutorial however we're just going to add some observers, so place the cursor in the ScenarioController element and then select ObserverList from the Elements toolbox.

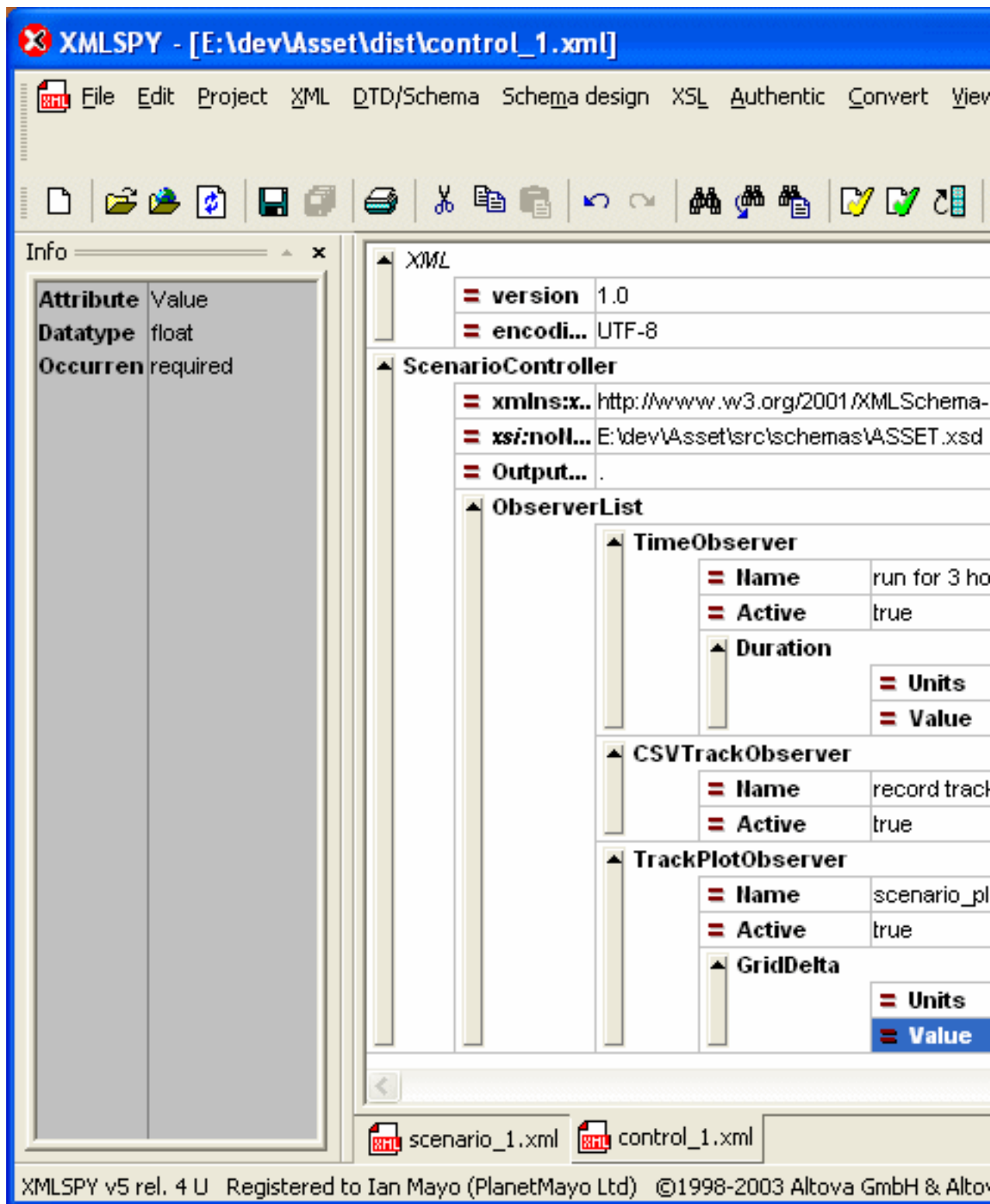
We now need one or more Scenario Observers to record the output of the run. First we add a TimeObserver, which will allow the scenario to run for a specific time period. First put the cursor

into the currently blank `ObserverList`, then select the `Add child` from the `Elements` toolbox. From the list of available observers, double-click `TimeObserver`. Set its active flag to true, and name it `run for three hours`. Lastly set the duration value to 3 hours. Next we will add a `CSVTrackObserver`, which will record the vessel track to a file in comma-separated variable (CSV) format. So, with the cursor on the `TimeObserver`, select `CSVTrackObserver` from the `Append` tab of the `Elements` toolbox. Configure this observer to be active (set active to true), name it `record tracks to csv`, and direct it to place the results files into the current directory, represented by a single full-stop (`.`). Note that the destination directory is an optional attribute - whilst inside the `CSVTrackObserver` select `directory_name` from the `Append` tab of the `Attributes` toolbox.

Lastly we will add an observer to provide us with a graphical track plot of the completed scenario. To do this, put the cursor on to the `CSVTrackObserver` element, then double-click on `TrackPlotObserver` from the `Append` tab of the `Elements` toolbox. Configure the track plot observer with it's name (`scenario_plot`), make it Active, and request a grid delta of 1 nautical mile (`nm`).

The file should now look like that below:

Figure 2.8. Completed control file



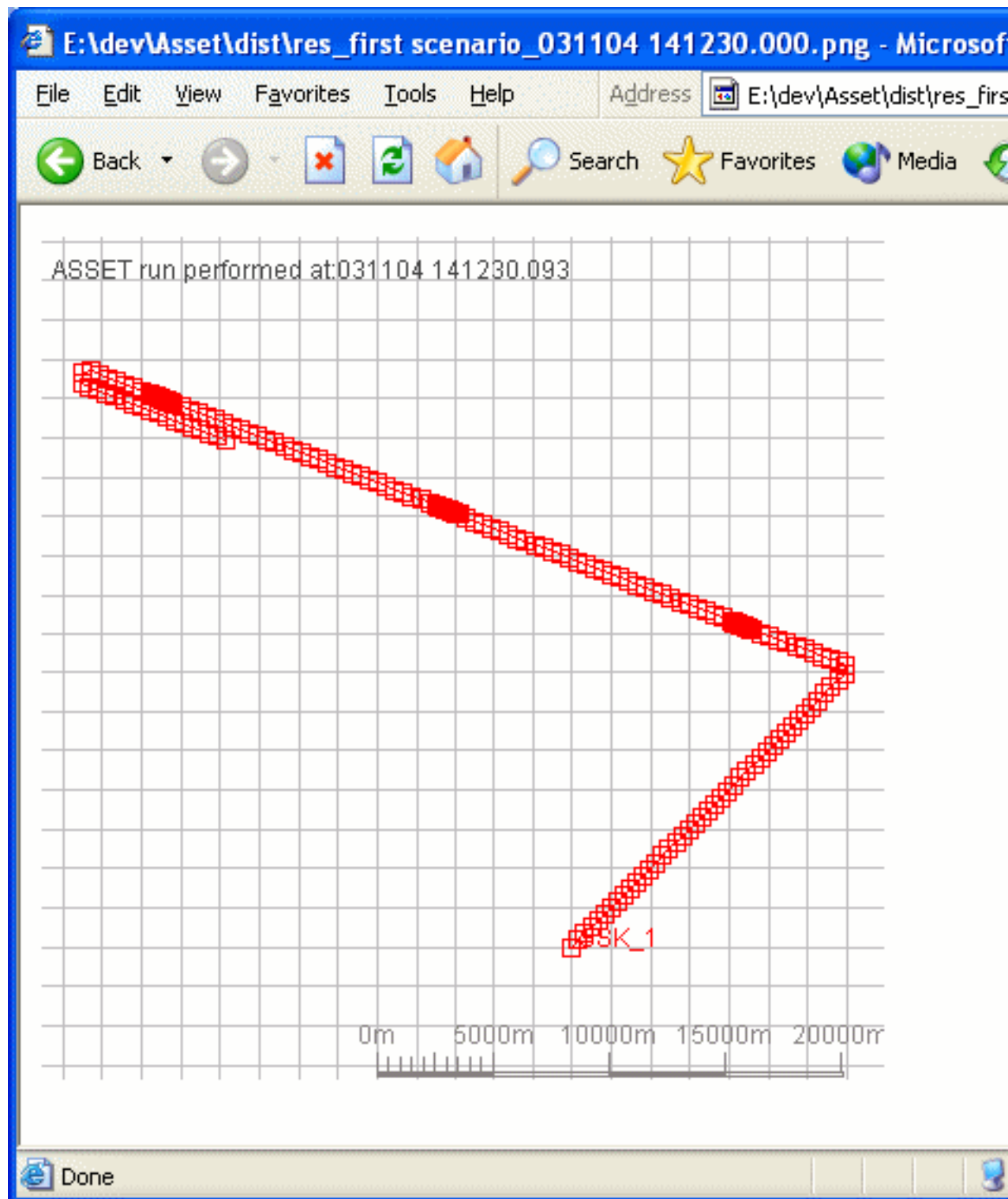
Lastly save the file to disk, placing it into your ASSET installation directory as `control_1.xml`.

4. Running command-line ASSET

So, in our ASSET installation directory we have a scenario file defining an SSK which will wander around a fixed point, and a control file which indicates that the scenario should run for an hour,

with results placed into the current directory. To run this scenario first open a command line ¹, and navigate to the ASSETinstallation directory. Once in this directory enter the following command: “`asset scenario_1 xml control_1.xml`”. You will see Started and Stopped status lines appear, and the command prompt return. Into the current directory you should see that CSV and PNG files has been created - double-click it the CSV to have a look inside. Excel should open up, and show you the columns of data. To view the track plot, open Internet Explorer then drag the PNG file across into it - you should see an image like that shown below.

¹from the Start Menu, select Run. From the dialog which opens type `cmd` into the text box and press Enter. A command line will open.

Figure 2.9. Overview track plot

Viewing the CSV output in Excel will show the periods spent snorting to recharge batteries, but these are also clear on the graphic track plot as track portions shaded solid red - periods when the SSK is travelling more slowly leaving the location markers overlapping one-another.

Chapter 3. Command Line Monte-Carlo

1. Introduction

First read through the use of ASSET in command-line mode for general scenario processing (as described in Chapter 2, *Command-line ASSET* [6]), together with the general guidance in the ASSET approach to Monte Carlo simulation (as described in Section 3, “Monte Carlo simulation with ASSET” [2]). The steps in this portion of the user guide will run through defining, running, then analysing a sample scenario. The scenario will be an extension of the previous scenario.

2. Preparing the data

As with normal ASSET command-line processing two files are required; the scenario file and the control file. The difference from normal processing is that the control file also includes scenario generation instructions. We're going to work with modified versions of the previous files, so using your file manager (such as Windows Explorer) make duplicates of `scenario_1.xml` and `control_1.xml`, naming them as `monte_scenario_1.xml` and `monte_control_1.xml`. Note that currently the location of the ASSET schema needs to be edited prior to running Monte Carlo scenarios.

2.1. Modify the scenario

The modification to be made to the scenario is that we are going to add a helo following a pre-determined search pattern. So, flash up XMLSpy, and load `monte_scenario_1.xml`. Navigate to the Participants element, collapse the existing SSK element, and insert a Helo element from the Append tab of the elements toolbox. Name it `Merlin_A`, mark it as an airborne Helicopter from Blue force (in the category), and add an Optic Sensor. Give the optic sensor a MinHeight of 30 metres. Next set the helo status and movement characteristics as shown in the following screenshot.

Figure 3.1. Overview track plot

▲ Status			
= Course	0		
= Fuel	100		
▲ Location			
▲ shortLocation			
= Lat	0		
= Long	0		
▲ Height			
= Units	m		
= Value	300		
▲ Speed			
= Units	kts		
= Value	80		
▲ HeloMovementCharacteristics			
= FuelUsageR...	0.0		
= Hame	Nil usage		
= DefaultTurn...	3		
▼ MinSpeed	Units=kts Value=1		
▼ MaxSpeed	Units=kts Value=150		
▼ AccelerationRate	Units=kt/s Value=1		
▼ DecelerationRate	Units=kt/s Value=1		
▼ MinHeight	Units=m Value=10		
▼ MaxHeight	Units=m Value=500		
▼ DefaultClimbRate	Units=ft/s Value=5		
▼ DefaultDiveRate	Units=ft/s Value=10		
▼ DefaultClimbSpeed	Units=kts Value=10		
▼ DefaultDiveSpeed	Units=kts Value=20		

Next we will record the radiated characteristics of the helo. Put the cursor into the RadiatedCharacteristics element, and insert an Optic element from the Add Child tab. Mark the XSectArea of the helo as 40, and the height as 4 metres.

Lastly we have to record the helicopter's search plan as a behaviour. Put the cursor into the RadiatedChars element, and insert Waterfall from the Insert tab of the Elements toolbar. Into the Waterfall behaviour insert a TransitWaypoint behaviour. Name it as `plan_alpha`, set it to looping, and select OnTop as the visitor pattern. Set the speed as 80 knots, and instruct the helo to travel through coordinates at: (0.2, 0.6), (0.9, -0.3), (0.9, 0.7) and (0.2, 0.3).

2.2. Modify the control file

Next we're going to modify the control file to instruct ASSET to create a number of SSKs in order that we can examine how they react to our searching helo. Start by opening `monte_control_1.xml` in XMLSpy. Next change the OutputDirectory to `monte_results` so that all of our output data is together. Now we're going to insert a scenario generator element into the scenario controller, so first pit the cursor on the ObserverList element, and select

ScenarioGenerator from the Insert tab of the elements toolbox. Right, set the Filename to results, and then insert a MultiParticipantGenerator from the Insert tab, as below:

Figure 3.2. Initial scenario generation

The screenshot shows the configuration of a ScenarioController. The main window is titled 'ScenarioController' and contains several nested elements:

- ScenarioGenerator**
 - Filename**: results
 - MultiParticipantGenerator**
 - ParticipantVariance**
 - name**: (empty field)
 - number**: (empty field)
 - Variance**: name= id=
- ObserverList**
 - TimeObserver**
 - run for 3 hours**: (empty field)
 - true**: (empty field)
 - Duration**
 - Units**: hours
 - Value**: 3
 - CSVTrackObserver**
 - record tracks to csv**: (empty field)
 - true**: (empty field)
 - TrackPlotObserver**
 - scenario_plot**: (empty field)
 - true**: (empty field)
 - GridDelta**: (empty field)

As you can see, we've been provided with a ParticipantVariance (see Variance in the Glossary) element to start us off - this is the data structure used to define how one particular participant is going to be *cloned* within the scenario. We're going to be making 16 new instances of our SSK, so enter SSK_1 in the Name field and 16 in the Number field.



Note

There's an optional ParticipantVariance parameter which isn't shown in the screenshot. The `inParallelPlanes` attribute indicates to ASSET whether the multiple instances of this participant can see each other or not. The Monte Carlo simulation technique makes use of multiple participants to predicting the probability of a single course of action of a participant using the statistical analysis of many non-interacting instances of that participant. The multiple participants are marked as non-interacting by setting the `inParallelPlanes` attribute to `true`.

Delete the LocationOffset element we've been provided with, and replace it with an Attribute element - indicating that we're going to be varying one of the attributes of the SSK. Name this

variance as `vary the course`, and set the `id` to `Status` to indicate that we're going to be changing an attribute of the `Status` element. Within the `Attribute` element we've already been given a `Range` element by default, so just fill in its details to indicate values from 0 to 360 in 45 unit steps, as below.

Figure 3.3. Variance to apply to the SSK

▲ MultiParticipantGenerator																															
<table border="1"> <tr> <td colspan="2">▲ ParticipantVariance</td> </tr> <tr> <td>= name</td> <td>SSK_1</td> </tr> <tr> <td>= number</td> <td>16</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Variance</td> </tr> <tr> <td>= name</td> <td>vary the course</td> </tr> <tr> <td>= id</td> <td>Status</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Attribute</td> </tr> <tr> <td>= name</td> <td>Course</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table> </td> </tr> </table> </td> </tr> </table> </td> </tr> </table>		▲ ParticipantVariance		= name	SSK_1	= number	16	<table border="1"> <tr> <td colspan="2">▲ Variance</td> </tr> <tr> <td>= name</td> <td>vary the course</td> </tr> <tr> <td>= id</td> <td>Status</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Attribute</td> </tr> <tr> <td>= name</td> <td>Course</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table> </td> </tr> </table> </td> </tr> </table>		▲ Variance		= name	vary the course	= id	Status	<table border="1"> <tr> <td colspan="2">▲ Attribute</td> </tr> <tr> <td>= name</td> <td>Course</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table> </td> </tr> </table>		▲ Attribute		= name	Course	<table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table>		▲ Range		= min	0	= max	360	= step	45
▲ ParticipantVariance																															
= name	SSK_1																														
= number	16																														
<table border="1"> <tr> <td colspan="2">▲ Variance</td> </tr> <tr> <td>= name</td> <td>vary the course</td> </tr> <tr> <td>= id</td> <td>Status</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Attribute</td> </tr> <tr> <td>= name</td> <td>Course</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table> </td> </tr> </table> </td> </tr> </table>		▲ Variance		= name	vary the course	= id	Status	<table border="1"> <tr> <td colspan="2">▲ Attribute</td> </tr> <tr> <td>= name</td> <td>Course</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table> </td> </tr> </table>		▲ Attribute		= name	Course	<table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table>		▲ Range		= min	0	= max	360	= step	45								
▲ Variance																															
= name	vary the course																														
= id	Status																														
<table border="1"> <tr> <td colspan="2">▲ Attribute</td> </tr> <tr> <td>= name</td> <td>Course</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table> </td> </tr> </table>		▲ Attribute		= name	Course	<table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table>		▲ Range		= min	0	= max	360	= step	45																
▲ Attribute																															
= name	Course																														
<table border="1"> <tr> <td colspan="2">▲ Range</td> </tr> <tr> <td>= min</td> <td>0</td> </tr> <tr> <td>= max</td> <td>360</td> </tr> <tr> <td>= step</td> <td>45</td> </tr> </table>		▲ Range		= min	0	= max	360	= step	45																						
▲ Range																															
= min	0																														
= max	360																														
= step	45																														

Note that additional attributes are available for varying how the random courses are generated. This first is `RandomModel`, which specifies which random number distribution used to generate the courses, and the second is `number_permutations`, which specifies that instead of creating 16 random courses, the 16 random courses should be taken from a pre-calculated, smaller number of random courses: 4 for example. Using a smaller number of random variables with per-scenario analysis



Note

ASSET uses a variety of third party libraries to support its functionality, with two separate XML Importer libraries. Unfortunately the importer library used when conducting scenario generation displays its immaturity by requiring that the path to the Schema files be entered as a URI (Uniform Resource Identifier). Consequently the Schema location attributes in our two Monte Carlo data-files needs to be edited. Specifically the DOS-path to the `ASSET.XSD` as currently included in the file needs to be prepended with `file://` and all `\` characters need to be replaced with `/`. A more versatile file importer will be incorporated once a stable, acceptable library comes available. An alternative to including the full URI is to include the `ASSET.XSD` in the current working directory and just use `include ASSET.XSD` as the schema reference without any path details.

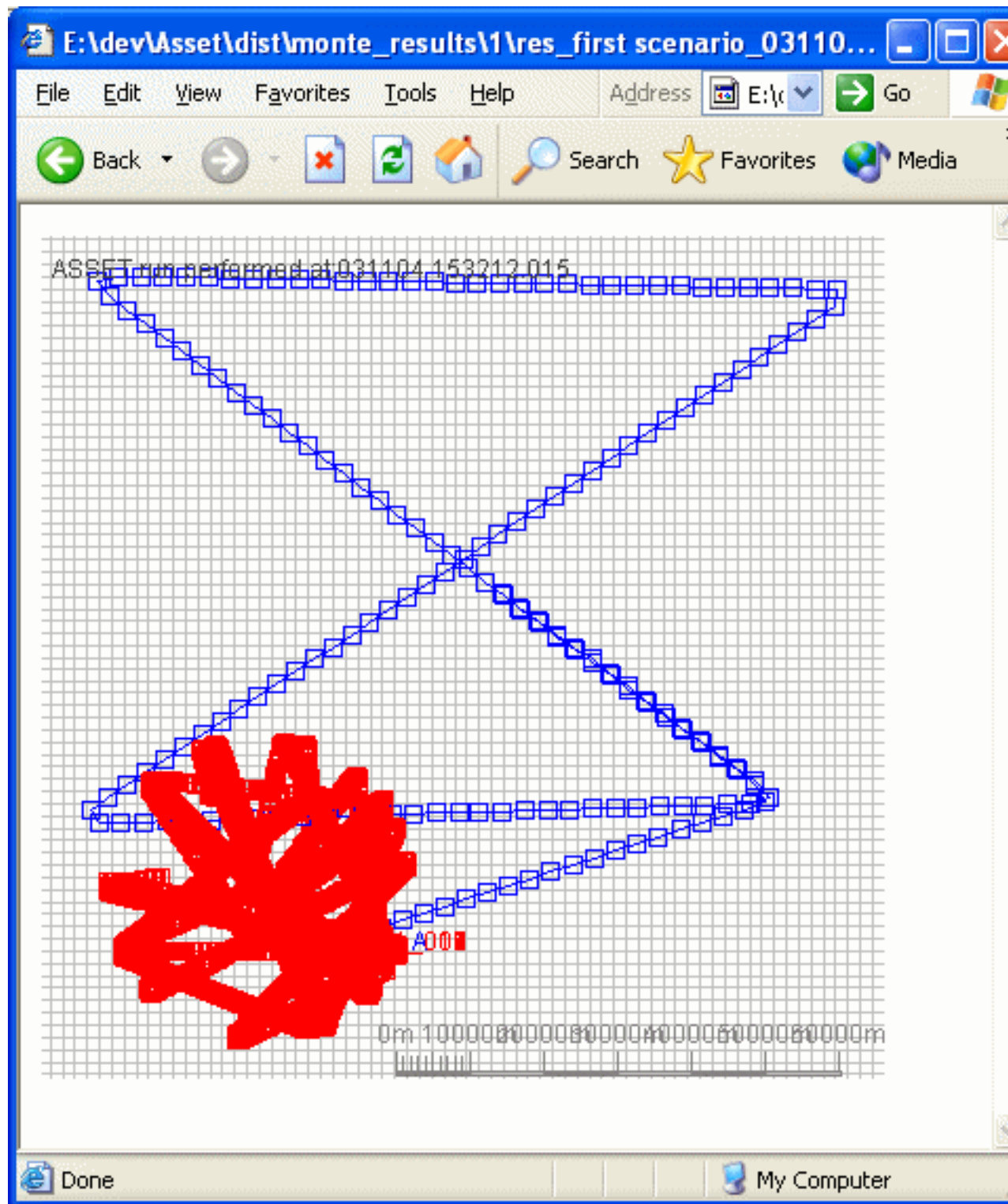
2.3. Running the Monte Carlo scenario

Monte Carlo simulations are started from the command line in the same way as conventional scenarios - so navigate to your installation directory and type the following: `asset monte_scenario_1.xml monte_control_1.xml`.

You should see the progress of the scenario on the command line, and be left with a `monte_results` sub-directory containing a copy of the control file together with a single sub-directory. This sub-directory contains a copy of the generated scenario file (`results_1.xml`) and our output files. If you examine the `results_1.xml` file you will see all 16 of the generated SSKs. Each is almost identical except for having varying unique courses. In the example I produced

the SSKs have initial courses of 90 degrees, 135 degrees, 315 degrees and so on. The PNG plot should look like that shown below:

Figure 3.4. Initial Monte Carlo results



2.4. More Monte Carlo playing

So, we've experimented with creating multiple participants - but this is only half of ASSET's support for Monte Carlo simulation, it can also generate multiple scenarios. To do this, open up

the `Monte_control_1.xml` file again, and insert a `MultiScenarioGenerator` in front of the `MultiParticipantGenerator`. Set the `NameTemplate` to `output` (to name our output files), then set the `Number` to 5 (to indicate that we want to work with 5 copies of the scenario). Insert a `Variance` into the `VarianceList` within which we're going to be varying the helo transit speed, so set the `Name` to `vary transit speed`. We're not directly referring to a participant here as we were in the multiple participant generator, so we have to use the XPath wild-carding syntax to identify the element we're going to adjust. We do this by referring to the status element beneath the parent element named `Merlin_A`. This reads as: `//*[@Name='Merlin_A']//TransitWaypoint/Speed`

Lastly replace the `LocationOffset` with an `Attribute` to indicate that we're going to be changing an attribute, then indicate that we want to use a `Range` of values from 40 to 200 in steps of 20, as below:

Figure 3.5. Scenario generation parameters

▲ MultiScenarioGenerator	
= NameTemplate	output
= Number	5
▲ VarianceList	
▲ Variance	
= name	vary transit speed
= id	//*[@Name='Merlin_A']//TransitWaypoint/Speed
▲ Attribute	
= name	Value
▲ Range	
= min	40
= max	200
= step	20

Now run the scenario again. If you look into the `monte_results` sub-directory you will see a copy of the control file (named `control_file.xml`- gifted!), together with 5 sub-directories. In each is a new version of the scenario (`results_1.xml` for example). Have a quick look at them, you'll see the 16 SSKs in each one, together with the varying transit speeds assigned to the helo.



Note

When conducting a multi-scenario simulation, there is a need to track performance across all scenarios - not just within individual runs. This ability to define inter-scenario observers is described later in Section 3.1, “Inter-scenario observers” [9]. The inter-scenario observer is defined in the control file just like normal observers, but when initialising the scenario ASSET applies special processing to these observers to initialise them prior to running through the individual scenarios.

Chapter 4. Getting to know the ASSET Sensor Model

1. Background

In Autumn 2003 Jon Walters derived an improved above-water sensor model for use in ASSET. Significantly, this algorithm modelled the transition of a target from undetected to through detected and classified to identified - a concept not present in the existing model.

The above-water sensor model utilises lookup tables to determine the environmental and spatial aspects of the sensor, sensor host, and target. Read more about the improved lookup model then run through this tutorial scenario.

2. Planning the scenario

The scenario we plan to model is that of a helicopter investigating a series of fishing vessels, trying to find a particular one. The helicopter first detects the vessels on radar, but since the radar sensor is typically unable to identify a target the helo closes to visual range to actually identify the target. Once the fishing vessel is identified as not being the target of interest the helo continues it's search path.

So, we will be creating a helo and a number of fishing vessels. The helo will be equipped with radar and optic lookup sensors. The helo will conduct a coarse search, investigating all fishing boat contacts to identify the target of interest - one of a number of fishing boats randomly wandering around the area of interest.

3. Creating the scenario file (including environment)

In this section we are going to start the skeleton of the scenario file, and define the environmental lookup tables used by the lookup sensors.

Start off by repeating the steps of the earlier tutorials to create the empty scenario file, naming your scenario with something along the lines of "Lookup sensor tutorial".

Now set the step time to 5 seconds, the scenario start time to 2003-01-24T13:00:03, and the created time to 2004-01-12T13:00:03 (or even the real date if you want).

Now for the scenario, which is inserted before the Participants item. So click on the Participants item, then on the Insert tab of the Elements toolbox. From the list of applicable items double-click on Environment, and it will appear. Now start off with the core environmental data items; `First lookup environment`, 2, and `LIGHT_HAZE`. Next come the lookup sensor-specific environments. The lookup sensor guidance recorded earlier in the modelling guide contains tables of typical values. We'll be starting with the Radar lookup model, taking the values from the earlier tables. Highlight the `AtmosphericAttenuation` attribute and double-click on the `RadarLookupEnvironment` element from the Insert toolbox. Name the Radar environment, and the start populating the `TargetAspectSet` with the Sigma values with the values recorded in the modelling guide. Working from the `Frigate` down, select `Frigate` from the drop-down list for the Type attribute then enter the respective Sigma values for Dead Ahead, Bow, Beam, Quarter, and Astern. Once you've finished the row insert another by clicking on the 1 row counter and then clicking on `TargetAspectDatum` from the Elements toolbox. Insert the CVS values into this row and so on for the other participant types.

In the spirit of having a go at sticking all sorts of data into the scenario we will provide a default Sigma value - to cover instances where we haven't provided a vessel-type (this would normally throw an error as recorded below). To do this click on the `TargetAspectDatum` item under your

RadarLookupEnvironment then from the Add Child tab of the Attributes toolbox double-click on the UnknownType item. Provide a value of 500 for this attribute.

Right, what's next. On to the TargetSeaStateSet. This is collated in a similar way to the TargetAspectSet, entering lists of values indexed against participant type. ASSET only requires data values for sea state one to six, if your scenario requires data in higher sea states they can be inserted from the Append or Insert tabs of the Attributes toolbox.

Now that the radar is in there we'll move onto the visual environment lookup table. The routine here is similar to that for radar. Two of the four sub-tables store vessel-type related data similar those we've just entered, with the other two being simple lists. The vessel-related tables take optional UnknownType attributes as covered earlier. Lastly insert a MADLookupEnvironment - providing sample values for the four participant types covered earlier. As a helping hand, here is the text version of a sample lookup table:

```
<RadarLookupEnvironment
Name="normal radar data"> <TargetAspectSet UnknownType="100">
<TargetAspectDatum Type="FRIGATE" DeadAhead="1000" Bow="3000"
Beam="4000" Quarter="3000" Astern="1000"/> <TargetAspectDatum
Type="CARRIER" DeadAhead="2000" Bow="8000" Beam="10000"
Quarter="8000" Astern="2000"/> <TargetAspectDatum Type="SUBMARINE"
DeadAhead="0.5" Bow="0.5" Beam="0.5" Quarter="0.5" Astern="0.5"/
> <TargetAspectDatum Type="FISHING_VESSEL" DeadAhead="5"
Bow="8" Beam="10" Quarter="8" Astern="5"/> </TargetAspectSet>
<TargetSeaStateSet UnknownType="1"> <TargetSeaStateDatum
Type="FRIGATE" SeaState_0="1" SeaState_1="1" SeaState_2="1"
SeaState_3="1" SeaState_4="1" SeaState_5="0.95" SeaState_6="0.9"/
> <TargetSeaStateDatum Type="CARRIER" SeaState_0="1" SeaState_1="1"
SeaState_2="1" SeaState_3="1" SeaState_4="1" SeaState_5="1"
SeaState_6="0.95"/> <TargetSeaStateDatum Type="SUBMARINE"
SeaState_0="1" SeaState_1="1" SeaState_2="0.8" SeaState_3="0.75"
SeaState_4="0.7" SeaState_5="0.5" SeaState_6="0.3"/>
<TargetSeaStateDatum Type="FISHING_VESSEL" SeaState_0="1"
SeaState_1="1" SeaState_2="0.96" SeaState_3="0.8" SeaState_4="0.75"
SeaState_5="0.7" SeaState_6="0.5"/> </TargetSeaStateSet> </
RadarLookupEnvironment> <VisualLookupEnvironment Name="normal
vis data"> <VisualAttenuationDatum VeryClear="8e-5" Clear="2e-4"
LightHaze="5e-4" Haze="1e-3" Mist="2e-3" Fog="4e-3"/>
<TargetVisibilitySet UnknownType="0.12"> <TargetVisibilityDatum
Type="CARRIER" Visibility="0.2"/> <TargetVisibilityDatum
Type="FRIGATE" Visibility="0.2"/> <TargetVisibilityDatum
Type="SUBMARINE" Visibility="0.12"/> <TargetVisibilityDatum
Type="FISHING_VESSEL" Visibility="0.16"/> </TargetVisibilitySet>
<TargetSeaStateSet UnknownType="1"> <TargetSeaStateDatum
Type="FRIGATE" SeaState_0="1" SeaState_1="1" SeaState_2="1"
SeaState_3="1" SeaState_4="1" SeaState_5="0.95" SeaState_6="0.9"/
> <TargetSeaStateDatum Type="CARRIER" SeaState_0="1" SeaState_1="1"
SeaState_2="1" SeaState_3="1" SeaState_4="1" SeaState_5="1"
SeaState_6="0.95"/> <TargetSeaStateDatum Type="SUBMARINE"
SeaState_0="1" SeaState_1="1" SeaState_2="0.8" SeaState_3="0.75"
SeaState_4="0.7" SeaState_5="0.5" SeaState_6="0.3"/>
<TargetSeaStateDatum Type="FISHING_VESSEL" SeaState_0="1"
SeaState_1="1" SeaState_2="0.9" SeaState_3="0.8" SeaState_4="0.75"
SeaState_5="0.7" SeaState_6="0.5"/> </TargetSeaStateSet>
<LightLevelDatum Daylight="1" Dusk="0.4" MoonlitNight="0.3"
DarkNight="0.05"/> </VisualLookupEnvironment> <MADLookupEnvironment
Name=""> <PredictedRangeSet> <PredictedRangeDatum
Type="SUBMARINE" PredictedRange="1000"/> <PredictedRangeDatum
Type="FISHING_VESSEL" PredictedRange="1200"/> </PredictedRangeSet>
</MADLookupEnvironment>
```




Warning

For most lookup tables ASSET will cease processing if a value is not provided for a particular vessel type (with the exception of MAD sensors where ASSET assumes that the target is not detectable via MAD). Providing a default value allows ASSET processing to continue, albeit with a suboptimal value. In normal use of ASSET you will have a clear idea of what participant types are involved and will create the table accordingly. The fact that ASSET stops when it encounters a participant type which hasn't been represented in the lookup tables acts in your favour here, since it reminds you to enter that data. Still, there may be occasions where a particularly complex scenario requires many participant types and you can't be bothered to populate parts of the lookup table which are insignificant to your tactical problem - fall back on the UnknownType values in these circumstances.

4. Completing the scenario file

Now we will complete the scenario definition.

Next insert the first fishing boat. Put the cursor into the Participants element, select Add child from the Elements tab and double-click on Surface to insert a surface vessel. Fill out the surface vessel name and category to denote it as a fishing boat of *Green* (neutral) force. Give this fishing vessel an id of 55 - so we can use it in our search behaviour later. Also give it an initial location of 25.5 degrees North and 53.5 degrees East. Lastly give it some manoeuvring characteristics; as shown in the screenshot below (note that we haven't had to set any radiated characteristics for the fishing vessel - they aren't needed for the lookup sensor model).

Figure 4.1. First fishing vessel

▲ Surface			
= Name		FISHER_1	
= id		55	
▲ Category			
		= Environm...	SURFACE
		= Force	GREEN
		= Type	FISHING_VESSEL
() SensorFit			
▲ Status			
		= Course	44
		= Fuel	100
		▲ Location	
		▲ shortLocation	
		= Lat	25.5
		= Long	53.5
		▲ Speed	
		= Units	kts
		= Value	6
() RadiatedCharacteristics			
▲ SurfaceMovementCharacteristics			
		= FuelUsag...	0.0001
		= Name	FISHING_CHARS
		▲ MinSpeed	
		= Units	kts
		= Value	0
		▲ MaxSpeed	
		= Units	kts
		= Value	8
		▲ AccelerationRate	
		= Units	kt/s
		= Value	1
		▲ DecelerationRate	
		= Units	kt/s
		= Value	2
		▲ TurningCircle	
		= Units	m
		= Value	20

Next we'll give the fishing vessel a behaviour - wandering around an origin. Put the cursor on RadiatedCharacteristics and the select Waterfall from the Insert elements tab. Name it something

like "fisherman behaviour", set to be Alive, and that it should Stay Alive, then click onto the Add child tab and select Wander. Configure the Wander behaviour so that the fishing vessel wanders out to a limit of 40 miles from 25.5 North, 54 East, at a speed of 6 knots and a height of 0 metres.

Next duplicate the fishing vessel, to create our second vessel. Omit the id (so that ASSET automatically gives it an id), use a new name and a slightly different start location (25.6 degrees North and 53.3 degrees East). Lastly produce another fishing boat with location of 25.4 degrees North and 53.2 degrees East.

That's the fishing boats done. Now move on to the helo. Collapse the third fishing boat, select it, then select Helo from the Append tab of the Elements toolbox. Name the helo as Helo_1. Set the category of the helo, and then it's initial status, as below

Figure 4.2. Helo status

▲ Status	
= Co...	88
= Fuel	100
▲ Location	
▲ shortLocation	
= Lat	25.7
= Long	52.8
▲ Height	
= Units	m
= Value	250
▲ Speed	
= Units	kts
= Value	150

Next give the helo Optic Lookup and Radar Lookup sensors. These sensors are setup using a set of common attributes explained earlier in the Modelling Guide. Configure the Optic sensor as in the following screenshot.

Figure 4.3. Helo Eyesight sensor

▲ OpticLookupSensor		
= Name	Eyesight	
= VDR	0.05	
= MRF	1.05	
= CRF	0.8	
= IRF	0.2	
▲ TBDO		
= Units	seconds	
= Value	10	
▲ CTP		
= Units	seconds	
= Value	20	
▲ ITP		
= Units	seconds	
= Value	30	

Of significance, the helo crew scan the ocean every 10 seconds, after a target is detected the helo must close to 80% of the detection range and wait for 20 seconds to elapse before the target is classified. Next configure the radar sensor as in the following screenshot:

Figure 4.4. Helo Radar sensor

▲ RadarLookupSensor		
= Name	Radar	
= VDR	0.02	
= MRF	1.2	
= CRF	0	
= IRF	0	
= K	20000	
▲ TBDO		
= Units	seconds	
= Value	2	
▲ CTP		
= Units	seconds	
= Value	0	
▲ ITP		
= Units	seconds	
= Value	0	

As you can see from the radar characteristics it's Classification Range Factor and Identification Range Factor are both zero, with the effect that the radar is unable to make the transition from *detected* to either *classified* or *identified*.

Lastly we must set the behaviour of the helo - until we have detection state-aware behaviours we will instruct the helo to wander around the search area but detour to investigate any targets of interest, as in the following screenshot:

Figure 4.5. Helo behaviours

Waterfall

Name	Helo behaviour
StayAlive	true

Trail

Name	Close to identification range
TrailRange	
Units	km
Value	3
AllowableError	
Units	km
Value	1
TargetType	
Type	
Name	FISHING_VESSEL

Wander

Name	Helo wandering
Range	
Units	nm
Value	40
Location	
shortLocation	
Lat	25.8
Long	54
Speed	
Units	kts
Value	150
Height	
Units	m
Value	250

After the behaviours come the final helo characteristics, the radiated energy and movement characteristics. For the purposes of our analysis we will not declare any radiated energy from the helo, but we will supply its movement characteristics, as shown below.

Figure 4.6. Helo characteristics

RadiatedCharacteristics	
HeloMovementCharacteristics	
FuelUsageRate	0.0001
Name	basic helo manoeuvring
DefaultTurnRate	4
MinSpeed	
Units	kts
Value	0
MaxSpeed	
Units	kts
Value	300
AccelerationRate	
Units	kt/s
Value	2
DecelerationRate	
Units	kt/s
Value	5
MinHeight	
Units	m
Value	10
MaxHeight	
Units	m
Value	600
DefaultClimbRate	
Units	ft/s
Value	5
DefaultDiveRate	
Units	ft/s
Value	10
DefaultClimbSpeed	
Units	kts
Value	20
DefaultDiveSpeed	
Units	kts
Value	80

5. Creating the control file

Lastly we just need to create the control file to run through the scenario, much as we did earlier in the tutorial. Configure the control file to run for 6 hours, outputting position to a Debrief Replay file, as follows:

Figure 4.7. Lookup tutorial control file

ScenarioController	
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation	file://D:\Dev\Asset\src\schemas\ASSET.xsd
OutputDirectory	.\
ObserverList	
TimeObserver	
Name	run for 6 hours
Active	true
Duration	
Units	hours
Value	6
DebriefReplayObserver	
Name	Record to rep
directory_name	.\
file_name	res.rep
Active	true
record_detections	false
record_decisions	false
record_positions	false

6. Running through the scenario

So, now we're ready to run through the scenario. Navigate to your ASSET installation directory and type `asset lookup_tutorial_scenario.xml lookup_test_control.xml .` Because we've elected for a 5-second step time it will take a couple of minutes to run, but hang on & it will still appear. ASSET will conduct the run, and leave your output file in this directory.

7. Analysing the results

The only output requested for the run was the Debrief track file. Open Debrief and then load your `res.rep` data file. You will see the three fishermen together with the investigating helo. Experiment with switching record decisions on to find greater detail in the helo behaviour.

Chapter 5. Modelling Force Protection

1. Introduction

This portion of the ASSET tutorial will lead you through the design and creation of ASSET files suitable for analysis in a helicopter-based Force Protection scenario. Modelling and analysing the scenario will be conducted using the following steps:

1. Define problem
2. Define environment
3. Define participants and their sensors & behaviours
4. Create scenario file describing problem
5. Create file to control running of ASSET
6. Run through scenario
7. Analyse scenario results

So, we'll start off by talking through the problem and the analysis requirement then skim through creating the scenario/control files. You won't need a lot of guidance in creating the data files after all of the previous experience in this tutorial.

2. Define the problem

The problem being addressed in this use of ASSET is that of a Merlin HM Mk1 performing force protection duties around ships of a task force. The helo conducts visual identification (without an EO device) of small vessels considered a potential threat to the force. In addition to analysing the search and identification task undertaken by the helo this scenario will additionally consider the effects of an unknown small vessel carrying a surface-to-air missile (SAM) with hostile intentions against the helo. We will need to consider the following:

- Is the likelihood of the helo being shot down related to the time of day, sea state, or atmospheric conditions?
- On average, for a given set of environmental conditions what proportion of the set of small vessels will be identified before the helo comes under fire from the hostile participant.

So we will perform multiple runs through a scenario containing a blue helo, multiple green fishing vessels and a single red fishing vessel. If the helo enters SAM range of the fishing vessel whilst in sight of the fishing vessel we will assume an attack happens and halt the scenario at that point (Section 1.1.3, “Stop on proximity detection observer” [66]). Otherwise each scenario will run for one hour (Section 1.1.4, “Time observer” [66]). For each scenario we may want a track plot (Section 1.2.4, “Track Plot Observer” [67]), and Debrief replay files (Section 1.2.2, “Debrief Replay Observer” [67]). We will also want to keep a record of how many fishing vessels were successfully identified (Section 1.3.1, “Detection observer” [68])

3. Define the environment

This investigation will take place in a rectangular area of around 1600 square nautical miles. The light levels will vary from daylight to moonlit night, the atmospheric attenuation from very clear to fog, and the sea states from 1 through to 6.

4. Define the participants, their sensors and behaviours

The participants modelled will be a single helicopter and multiple fishing vessels.

4.1. Helicopter

The helicopter will have the manoeuvring characteristics used in the lookup scenario example earlier. It will be equipped with radar and optic sensors.

The helicopter behaviour will be comprise a waterfall of tiered behaviours. The lowest-priority behaviour will be to Wander around the area of interest. The next higher priority will be to conduct a ladder-search. Highest of all will be the investigate behaviour, configured such that the helo attempts to Identify each fishing vessel.

Lastly the helo is given its radiated and movement characteristics.

4.2. Aggressive fishing vessel

The first fishing vessel to be described will be the aggressive fishing vessel carrying the SAM. This fishing vessel has an optic sensor and follows a Rectangle Wander (inside its Waterfall container) behaviour through the area of interest.

4.3. Non-aggressive fishing vessel

Multiple instances of the non-aggressive fishing vessel will be created to represent the standard fishing vessels occupying the area of interest. This fishing vessel requires no sensor and only uses the Rectangle Wander behaviour described above.

5. Create the scenario file

Hopefully the past parts of this tutorial, the guidance recorded above, and the analyst's scenario definition file will enable to you construct the scenario. A copy of the scenario is recorded below anyway.

Modelling Force Protection

Example 5.1. Scenario file for Force Protection analysis

```

</Investigate> <LadderSearch Name="First ladder search"
LadderAxis="90"> <StartPoint> <relativeLocation> <North
Units="nm" Value="0"/> <East Units="nm" Value="0"/> <Height
Units="ft" Value="100"/> </StartPoint>
<TrackSpacing Units="nm" Value="10"/> <LegLength Units="nm"
Value="40"/> </LadderSearch> <Wander Name="Stay in patrol area">
<relativeLocation> <relativeLocation> <relativeLocation> <relativeLocation>
<North Units="nm" Value="0"/> <East Units="nm" Value="0"/>
</relativeLocation> </Location> <Speed Units="kts"
Value="40"/> <Height Units="nm" Value="0"/> </Wander>
</Waterfall> <RadiatedCharacteristics/>
<HeloMovementCharacteristics FuelUsageRate="0.0001" Name="RAW
CHARACTERISTICS" DefaultTurnRate="3"> <MinSpeed Units="m/s"
Value="0"/> <MaxSpeed Units="m/s" Value="200"/> <AccelerationRate
Units="m/s/s" Value="4"/> <DecelerationRate Units="m/s/s" Value="2"/>
<MinHeight Units="m" Value="10"/> <MaxHeight Units="m"
Value="300"/> <DefaultClimbRate Units="m/s" Value="6"/>
<DefaultDiveRate Units="m/s" Value="40"/> <DefaultClimbSpeed
Units="m/s" Value="40"/> <DefaultDiveSpeed Units="m/s" Value="60"/>
</HeloMovementCharacteristics> </Helo> <Surface
Name="SAM_FISHER" MonteCarloTarget="false"> <Category
Environment="SURFACE" Force="RED" Type="FISHING_VESSEL"/> <SensorFit>
<OpticLookupSensor Name="Eyesight" VDR="0.16" MRF="1.05" CRF="0.8"
id="444"> <TBDO Units="seconds" Value="10"/> <CTP Units="seconds"
Value="20"/> <ITP Units="seconds" Value="20"/>
</OpticLookupSensor> </SensorFit> <Status Course="22"
Fuel="22"> <Location> <relativeLocation> <North Units="nm"
Value="5"/> <East Units="nm" Value="5"/> </relativeLocation>
</Location> <Speed Units="m/s" Value="16"/> </Status>
<Waterfall Name="Do some stuff" IsActive="true" >
<RectangleWander Name="just wander around my area"> <Area>
<TopLeft> <relativeLocation> <North Units="nm" Value="40"/>
<East Units="nm" Value="40"/> </relativeLocation> </TopLeft>
<BottomRight> <relativeLocation> <North Units="nm"
Value="0"/> <East Units="nm" Value="0"/> </relativeLocation>
</BottomRight> </Area> <Speed Units="m/s" Value="6"/>
<Height Units="m" Value="0"/> </RectangleWander> </Waterfall>
<RadiatedCharacteristics/> <SurfaceMovementCharacteristics
FuelUsageRate="0.006" Name="2"> <MinSpeed Units="m/s" Value="2"/>
<MaxSpeed Units="m/s" Value="18"/> <AccelerationRate Units="m/s/s"
Value="12"/> <DecelerationRate Units="m/s/s" Value="12"/>
<TurningCircle Units="m" Value="600"/>
</SurfaceMovementCharacteristics> </Surface> <Surface
Name="GENERIC_FISHER" MonteCarloTarget="false"> <Category
Environment="SURFACE" Force="GREEN" Type="FISHING_VESSEL"/>
<SensorFit/> <Status Course="22" Fuel="22"> <Location>
<relativeLocation> <North Units="nm" Value="4"/> <East
Units="nm" Value="3"/> </relativeLocation> </Location> <Speed
Units="m/s" Value="16"/> </Status> <Waterfall Name="Fishing vessel
behaviour" IsActive="true" > <RectangleWander Name="just
wander around my area"> <Area> <TopLeft>
<relativeLocation> <North Units="nm" Value="40"/> <East
Units="nm" Value="40"/> </relativeLocation> </TopLeft>
<BottomRight> <relativeLocation> <North Units="nm"
Value="0"/> <East Units="nm" Value="0"/> </relativeLocation>
</BottomRight> </Area> <Speed Units="m/s" Value="6"/>
<Height Units="m" Value="0"/> </RectangleWander> </Waterfall>
<RadiatedCharacteristics/> <SurfaceMovementCharacteristics
FuelUsageRate="0.006" Name="2"> <MinSpeed Units="m/s" Value="2"/>
<MaxSpeed Units="m/s" Value="18"/> <AccelerationRate Units="m/s/s"
Value="12"/> <DecelerationRate Units="m/s/s" Value="12"/>
<TurningCircle Units="m" Value="600"/>
</SurfaceMovementCharacteristics> </Surface> </Participants>
</Scenario>

```

6. Create the control file

The control file is more challenging for this scenario than previous ones, we've a fairly complex sensitivity analysis to support, together with producing a wider-than-previous range of participants. Its production is broken down into the three parts handling generating the multiple scenarios, generating the multiple random fishing vessels within each scenario, and specifying the observers which will control and observe the running scenarios.

6.1. Multi Scenario Generator

The multi-scenario generator is configured to produce the required number of scenarios, with an indicated maximum number of permutations of each scenario variance. This max number of permutations is roughly calculated as the number of scenario runs requested divided by the product of the number of permutations allowed by the variances. After this header information the set of variances themselves are recorded. According to the analyst's problem statement there are variances to be defined: for atmospheric attenuation, sea state and light level. Each of these is implemented as a Choice variance where we supply a list of optional values to choose; though note that the Range variance could have been used for the sea-state variable.

6.2. Multi Participant Generator

The multi-participant generator is used to both give some randomness to the fishing vessel start positions and to create the series of fishing vessels representing the dense fishing area. First we will define the participant variance for the SAM-carrying fisherman. We provide the name of the scenario-file participant we are going to vary (`SAM_FISHER` in this case), and then indicate how many permutations of this participant are required (1). We then vary the initial location, course and speed. Next we specify the participant variance for the non-SAM carrying fishing vessel. Again we name the scenario participant we are varying (`SAM_FISHER`), and indicate how many permutations we require (49 to make a total number of 50 fishing vessels). Again we vary the initial location, course and speed.

6.3. Observers

Lastly we specify the observers for the scenario, comprising controlling observers and recording observers.

6.3.1. Controlling Observers

These observers stop the scenario when a particular condition occurs. The two stop conditions for this analysis are the time limit (see Section 1.1.4, "Time observer" [66]) and the aircraft being shot down. We model the aircraft being shot down through stopping the scenario when the SAM-carrying fishing vessel is in visual contact with the helo and within weapons range (see Section 1.1.3, "Stop on proximity detection observer" [66]).

6.3.2. Recording Observers

There are two thrusts to recording the output of this scenario analysis: to obtain an understanding of the whole batch of runs, and to be able to drill-down into the detail of particular runs. Working in reverse order, we retain individual run results by specifying either an observer producing a track file in Debrief replay format or one producing a single graphic image.

The batch recorders used for this analysis are specified such that the analysis Measures of Effectiveness (*whether the aircraft was shot down, and how many targets were identified*) may be supported. The Final State observer is configured to record per-case results summarised as a frequency list - indicating to us how many scenarios of each permutation ended with helo-splashing versus time out. Secondly we use a Detection Observer configured to record how many fishing vessels were identified by the helo. Per case processing is switched on again, so that our output results are matched against the variables of interest.

The final control file is supplied below.

Example 5.2. Control file for Force Protection analysis

```

<MultiParticipantGenerator> <ParticipantVariance name="SAM_FI
number="1"> <Variance name="SAM_North" id="Status/Location//North">
  <Attribute name="Value"> <Range min="0" max="40" step="0.01"/>
  </Attribute> </Variance> <Variance name="SAM_East"
id="Status/Location//East"> <Attribute name="Value"> <Range min="0"
max="40" step="0.01"/> </Attribute> </Variance> <Variance
name="SAM_Speed" id="Status"> <Attribute name="Cour
<Range min="0" max="360" step="1"/> </Attribute> </Variance>
<Variance name="SAM_wander_corner" id="/RectangleWander/Area/"
<LocationArea RandomModel="Uniform"> <WorldArea> <TopLeft>
<relativeLocation> <North Units="nm" Value="41"/> <East
Units="nm" Value="39"/> </relativeLocation> </TopLeft>
<BottomRight> <relativeLocation> <North Units="nm"
Value="39"/> <East Units="nm" Value="41"/> </relativeLocation>
</BottomRight> </WorldArea> </LocationArea> </Variance>
</ParticipantVariance> <ParticipantVariance name="GENERIC_FIS
number="4"> <Variance name="GENERIC_NORTH" id="Status/Speed">
  <Attribute name="Value"> <Range min="3" max="8" step="0.5"/>
  </Attribute> </Variance> <Variance name="GENERIC_NORTH"
id="/Status/Location/relativeLocation/North"> <Attribute name="Value">
  <Range min="0" max="40" step="0.01"/> </Attribute>
  </Variance> <Variance name="GENERIC_EAST"
id="/Status/Location/relativeLocation/East"> <Attribute name="Value">
  <Range min="0" max="40" step="0.01"/> </Attribute>
  </Variance> <Variance name="GENERIC_COURSE" id="Status">
  <Attribute name="Course"> <Range min="0" max="360" step="1"/>
  </Attribute> </Variance> </ParticipantVariance>
</MultiParticipantGenerator> </ScenarioGenerator>
<ObserverList> <TimeObserver Name="1 Hour Scenario" Active="t
<Duration Units="hours" Value="2.5"/> </TimeObserver>
<StopOnProximityDetectionObserver Name="HELO detected within
Active="true"> <Target> <TargetType> <Type
Name="HELICOPTER"/> </TargetType> </Target> <Watch>
  <TargetType> <Type Name="FISHING_VESSEL"/> <Type Name="RED"/>
  </TargetType> </Watch> <Range Units="nm" Value="2.5"/>
  </StopOnProximityDetectionObserver> <TrackPlotObserver Name="
track plot" Active="true" show_positions="false"> <GridDelta Units="nm"
Value="5"/> </TrackPlotObserver> <DebriefReplayObserver Name="keep
tracks" Active="false" record_decisions="false" record_detections="false"
record_positions="true" file_name="all_tracks"/> <DebriefReplayObserver
Name="watch helo decisions" Active="false" record_decisions="true"
record_detections="false" record_positions="false"
file_name="helo_decisions"> <SubjectToTrack> <Type
Name="HELICOPTER"/> </SubjectToTrack> </DebriefReplayObserver>
  <DetectionObserver Name="per-case inter-scenario detections a
Active="true" DetectionLevel="Identified"> <Target> <TargetType>
  <Type Name="FISHING_VESSEL"/> <Type Name="GREEN"/>
  </TargetType> </Target> <Watch> <TargetType> <Type
Name="HELICOPTER"/> </TargetType> </Watch> <BatchCollator
Active="true" PerCase="true" OnlyBatchReporting="true"
CollationMethod="AVERAGE"/> </DetectionObserver>
  <FinalStateObserver Name="Why did they stop" Active="true">
  <BatchCollator Active="true" PerCase="true" CollationMethod="
OnlyBatchReporting="true"/> </FinalStateObserver>
  <ElapsedTimeObserver Name="How long did each case run for - a
they called" Active="true"> <BatchCollator Active="true" PerCase="true"
CollationMethod="ITEMIZED_LIST" OnlyBatchReporting="true"/>
  </ElapsedTimeObserver> <ElapsedTimeObserver Name="How long di
for - and what where they called" Active="false"> <BatchCollator
Active="true" PerCase="false" CollationMethod="ITEMIZED_LIST"
OnlyBatchReporting="true"/> </ElapsedTimeObserver>
</ObserverList> </ScenarioController>

```

7. Run through the scenario

Ok, run through the scenario by typing `ASSET.BAT scenario_file.xml control_file.xml`. ASSET will now generate, metamorphosize, and store to disk the new scenario permutations. It will run through them, and finally output the results of the inter-scenario observers.



Tip

When running through the Force Protection with scenario with large number of scenario (500) and participant (49) permutations ASSET did trip over with a Memory limit exceeded warning during the scenario generation stage. Currently the scenarios are all generated in memory, and the volume used can exceed the ASSET default (256Mb). Increasing the maximum memory allocated to 512Mb prevented the error occurring, with memory consumption dropping to normal levels once ASSET started executing the scenarios. The memory limit is increased by editing the `ASSET.BAT` batch file to change `-Xmx256m` to `-Xmx512M`

8. Analyse the scenario results

Hey, I'm not the analyst. Here goes anyway:

1. Examine the inter-scenario observer output files, attempting to identify patterns of results across the range of permutations of variables of interest.
2. Where you're not aware of the cause for a particular pattern in the data view load one of that permutation's track plot images into Internet Explorer and try to recognise what happened. If all doesn't come clear load the data-file into Debrief and step through the scenario. Optionally you can re-run the scenario requesting successively more detailed recording.
3. If you don't need to run through the full set of permutations to investigate a particular pattern, create a control-file and run this itself within ASSET against a particular scenario file generated for the permutation of interest.
4. Obtain a report for last time you did this work. Change the date and re-order the conclusions. Submit to the staffing loop...

Chapter 6. Proprietary Workbench

1. Introduction

The ASSET Workbench is a custom graphical front-end to the ASSET modelling engine. Scenario (and occasionally Control) files are loaded into the Workbench, and then you step forward through the scenario. Graphical dialogs are provided to allow you to drill-down into the detailed attributes, settings, and behaviours of the scenario participants - allowing (to a limited degree) interactive scenario design and modification to be performed. The Workbench application makes extensive re-use of components developed for MWC's Debrief [www.debrief.info] application - particularly with regard to the graphical plot, property editing, and core units (distance, locations, areas).

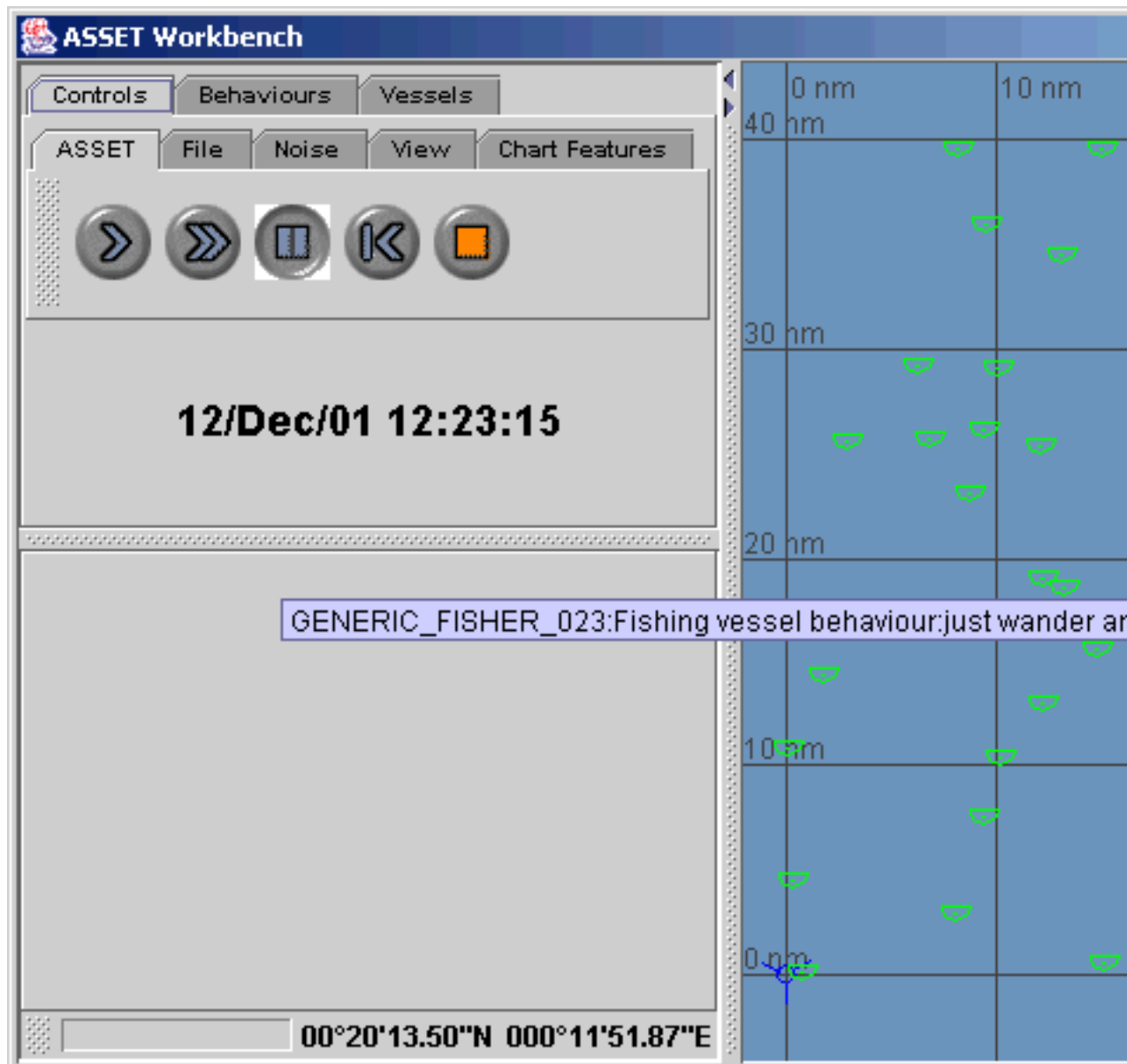


Note

The custom ASSET workbench was superseded in 2009 by the updated Eclipse-based workbench that integrates Debrief UI components with extensive Eclipse capabilities.

2. Workbench user interface

The following diagram highlights the main areas of the Workbench user interface:

Figure 6.1. Screenshot of Workbench application

Toolbar

In the top-left hand corner of the Workbench interface is the toolbar. This comprises a series of toolbars, one of which (Controls) contains its own series of toolbars. Lower sections provide more detail on individual toolbar buttons. In addition to the Controls toolbar, the Behaviours and Vessels toolbars provide a library of behaviours and vessels suitable for inclusion in a scenario. Drag a vessel onto the Plot to create and position it, drag a Behaviour onto the Waterfall editor to add that behaviour to the current vessel.

Tote

The tote is located immediately below the toolbar, and displays the current model time.

Properties Window

Below the Tote is the properties window, a holder for a wide range of editor panels - from the Layer Manager which shows all current data stored, through to the vessel status viewer which shows the course, speed, depth (demanded and current) for the selected participant, together with providing access to the editable attributes of that participant - all in the tabbed properties window.

Plot

The right-hand side of the screen is dedicated to the Plot itself, a 2-dimensional snapshot view of the scenario. Double-clicking on a participant's icon will open that participant in the properties window, and the Chart Features sub-toolbar of the Controls toolbar allows graphic items to be added to the plot (contours, grids, scales, annotations).

3. Loading Force Protection scenario

In this section we'll re-use the Force Protection scenario described above, but gain a feel for the interaction of the participants by graphically viewing the evolving scenario.

So, double-click on `workbench.exe` to open the Workbench. Almost immediately the ASSET Splash Screen (below) will display (significantly showing the build date of the current version in the lower right-hand corner), and a few seconds later the Workbench itself will be visible.

Figure 6.2. Workbench splash screen

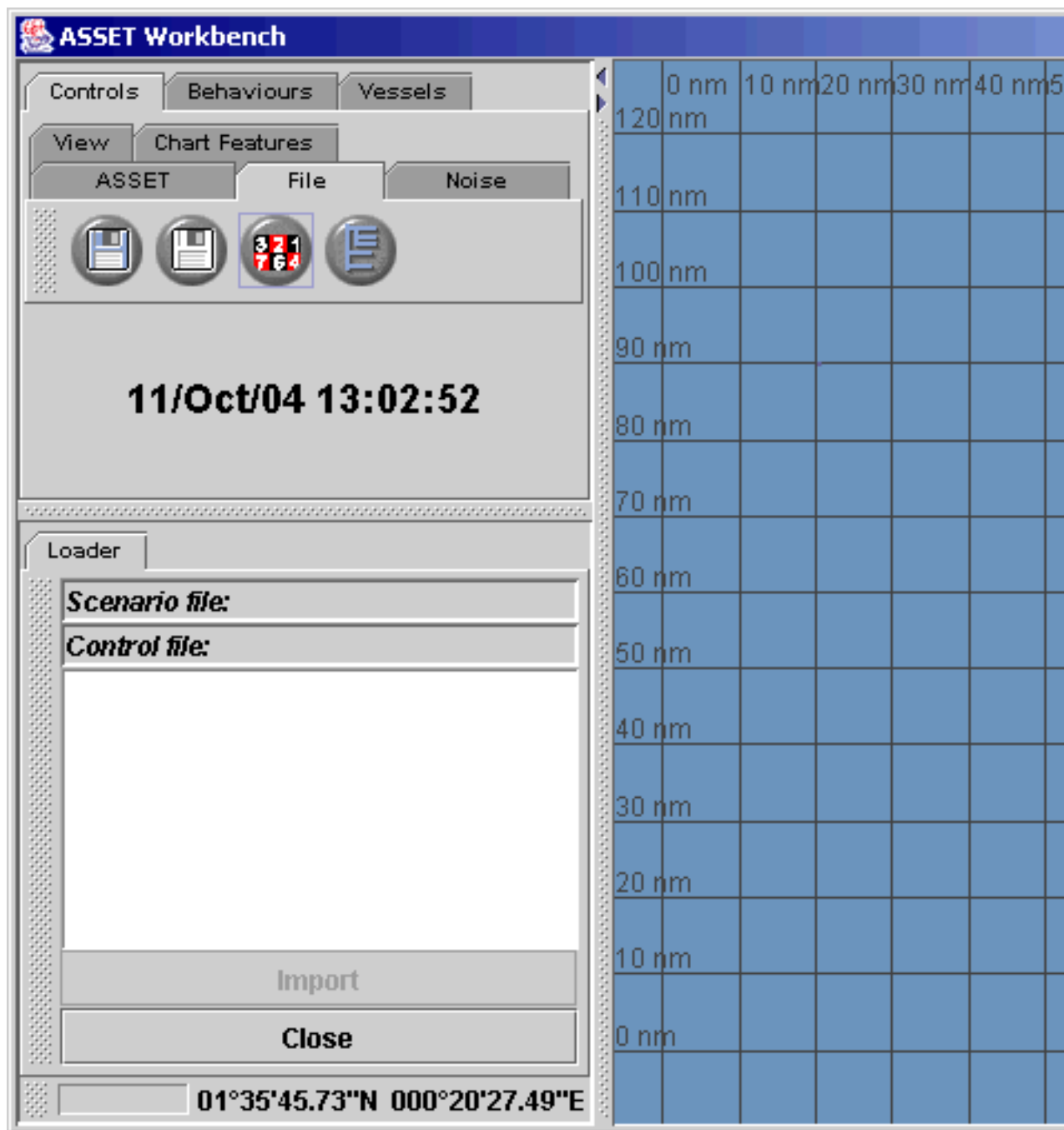


The Workbench is going to generate a number of targets on our behalf in this run, so start by



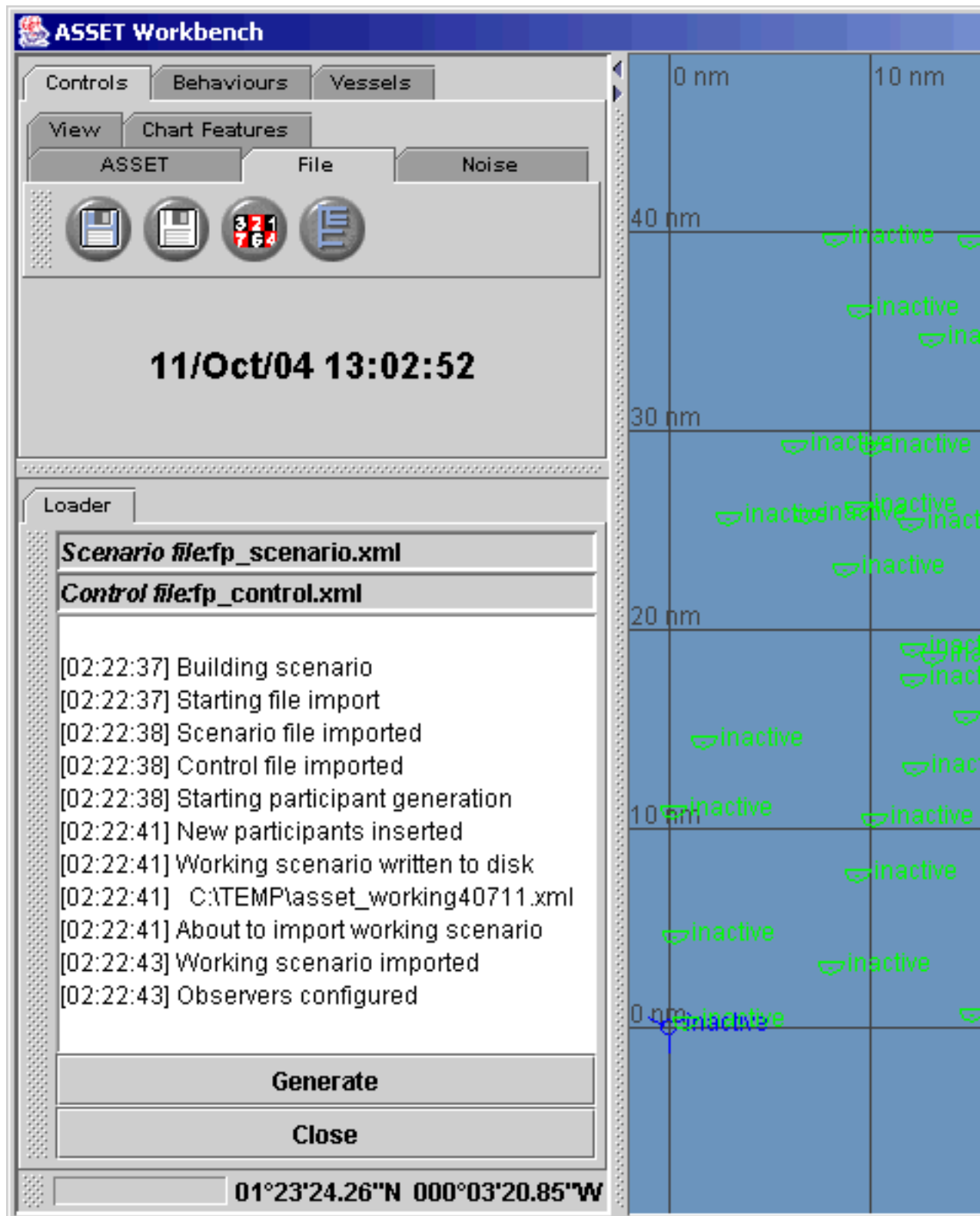
opening the Monte Carlo loader by clicking on the button from the File sub-toolbar of the Controls toolbar. The Monte Carlo loader will open in the Properties Window, as shown below.

Figure 6.3. Monte Carlo file loader



As you can see, the Monte Carlo loader has labels indicating the Scenario and Control files, beneath this is a reporting window, and beneath this are the Import and Close buttons. The reporting window is provided to give you feedback during the potentially lengthy import/participant generation process. So, to start off with, drag and drop the scenario you created for the Force Protection exercise on to the *Scenario file* label. You will see the label get updated with the new filename, and the Import button become enabled. To just load this scenario as-is, you would click on the Import button, but we are just using the scenario as a template - wishing to create multiple instances of the Green friendly fishing vessel. So, drag the Force Protection control file onto the *Control file* label. As you do this, the Import button will change to Generate, acknowledging that you no longer just want to import the scenario, but wish to run it through the Multiple Participant Generator. So, press the Generate button and you will see a scrolling set of messages grow in the reporting window (below), followed by the new participants appearing on the Plot.

Figure 6.4. Participant generation complete








Tip

Some of the steps performed during multiple participant generation can take 10s of seconds - if nothing else the scrolling messages provide you with an indication that yes, something is happening.

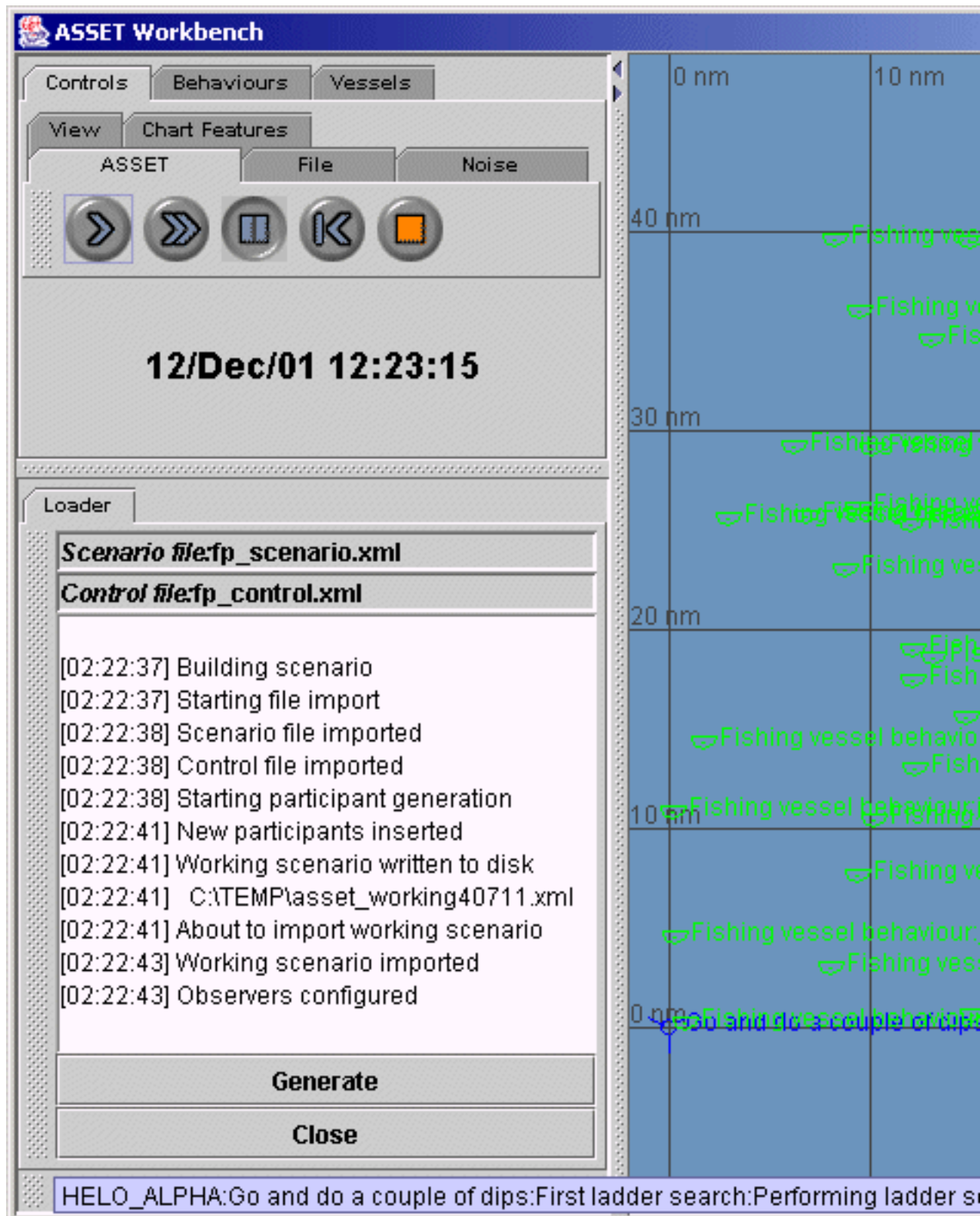
4. Moving the scenario forward

Now that our data is loaded, and we can see the dozens of participants on the plot, it's time to move forward. From the Controls toolbar, select the ASSET sub-toolbar. The five buttons on this toolbar operate as follows:

Step		perform a single step forward in the scenario, according to the scenario time step
Run		run the scenario forward automatically, pausing briefly according to the application time step configured
Pause		suspend the automatic progression of the scenario
Restart		restart the scenario - returning all participants to their original location and state
Exit		exit the ASSET Workbench

In the startup state as shown above, we can see all of the target vessels, but they all have the label *inactive* placed alongside them. The default plotting standard for participants is to represent them as a symbol annotated with a description of their current behaviour. Since the scenario hasn't moved forward yet ASSET does not know what each participant is doing. Pressing the Step button once will move the scenario forward one step, and when redrawn the plot will correctly show the participant states. [WorkbenchWithStates.gif](#)

Figure 6.5. States shown after single step









The high number of participants shown does make it difficult to view all of their states, but we will tidy that up later. If you press the Step button a few more times you will see the movement of the participants - with the faster-moving helo visible at the bottom-left of the plot. Now try clicking the Run button, and see the scenario move forward in front of you - with the time being updated on the

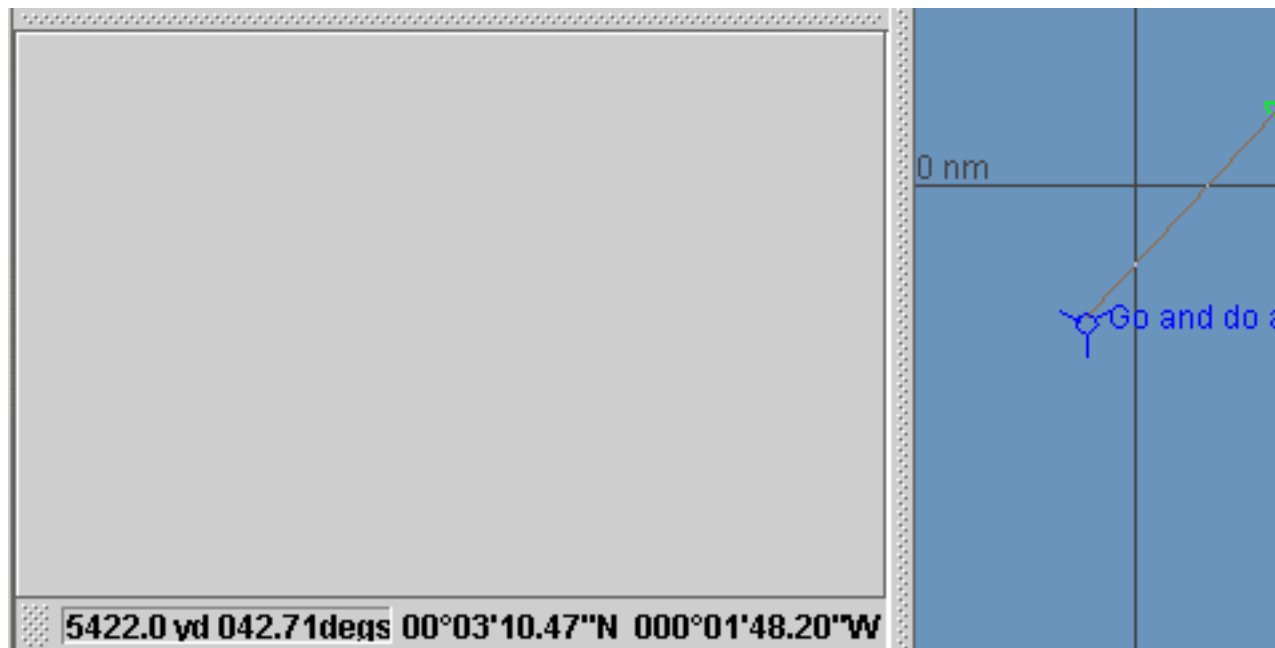
Tote. Now Pause the scenario to give us time to think. That's better. Also close the Monte Carlo Loader using its Close button to make more space in the properties window.

5. Manipulating the Plot

So, before we start drilling down into the data in front of us, we're just going to tidy the plot. First select the View sub-toolbar of the Controls toolbar. This toolbar contains the following controls:






Refresh		Redraw the plot
Fit to window		Zoom out (or in) to ensure that all participants are displayed in as much detail as possible
Pan		Left-click on the plot and drag to move around the view
Range/Bearing		Left-click on the plot and drag to measure range and bearing between two points - the results are shown at the bottom left of the Workbench window
Zoom in		Left-click on the plot and drag to select an area to zoom in on
Zoom Out		Click on this button to zoom the view out to cover approximately twice the current area

Start by clicking on the zoom-in button. If it's not already depressed it now will be. Now drag the cursor over the bottom left-hand quadrant of the plot containing the searching helicopter (or other quadrant if you have moved forward far enough). You should now see many fewer participants, and be able to read their status messages more clearly. Also experiment with clicking on the Range-Bearing tool and measuring the range from the Helo to the nearest fishing vessel - as below:

Figure 6.6. Measuring Range and Bearing on the plot

Now click on the Pan tool and drag the mouse on the Plot - this will move your view of the plot. Next, click on the zoom-out button to zoom out on the plot, and lastly click on the Fit to Window to zoom out to view the full dataset.

Next, we will briefly examine the Chart Features toolbar. This toolbar provides a set of non-tactical graphic items which may be added to the plot:

- | | | |
|--------------------|---|--|
| Scale |  | Place a scale on the plot - with either automatic or manual limits |
| Grid |  | Place a user-configured grid on the plot. World-scale units (degrees, minutes) have a Greenwich Meridian/Equator origin, whereas local area units (miles, metres) have an origin at the bottom-left of the plot. |
| VPF Layers |  | Place a configurable set of VPF layers on the plot. (see Note [48]) |
| Coastline |  | Because of storage constraints ASSET is distributed by default with a coarse global vectored coastline - suitable for providing the user with a very rough feel of their location. |
| ETOPO Gridded Data |  | Show global gridded depth data on the plot, optionally with contours. (see Note [49]) |

Try clicking on the Grid or Scale button to add either to the plot. On clicking each one it's attributes will become editable in the Properties Window.

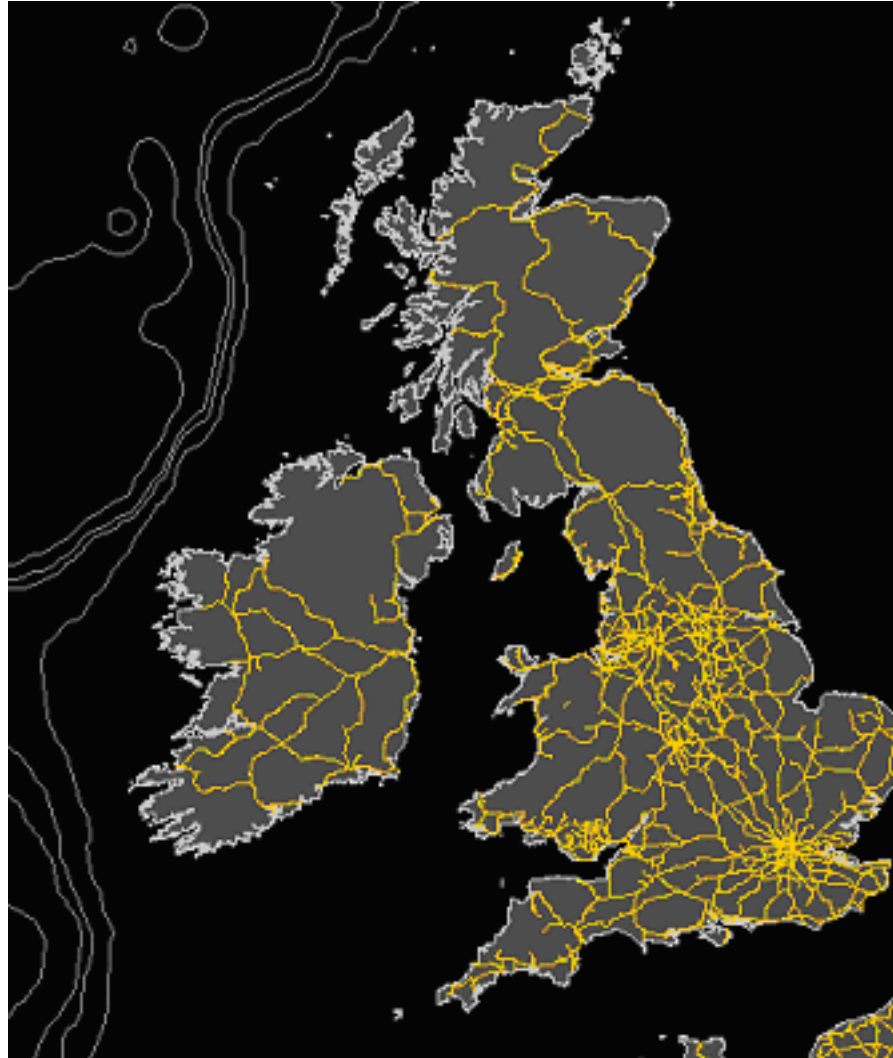


Note

The VPF dataset (Vector Product Format) is a set of vectored data showing physical and human features. Significantly it contains global depth contours at 200, 600, 1000.

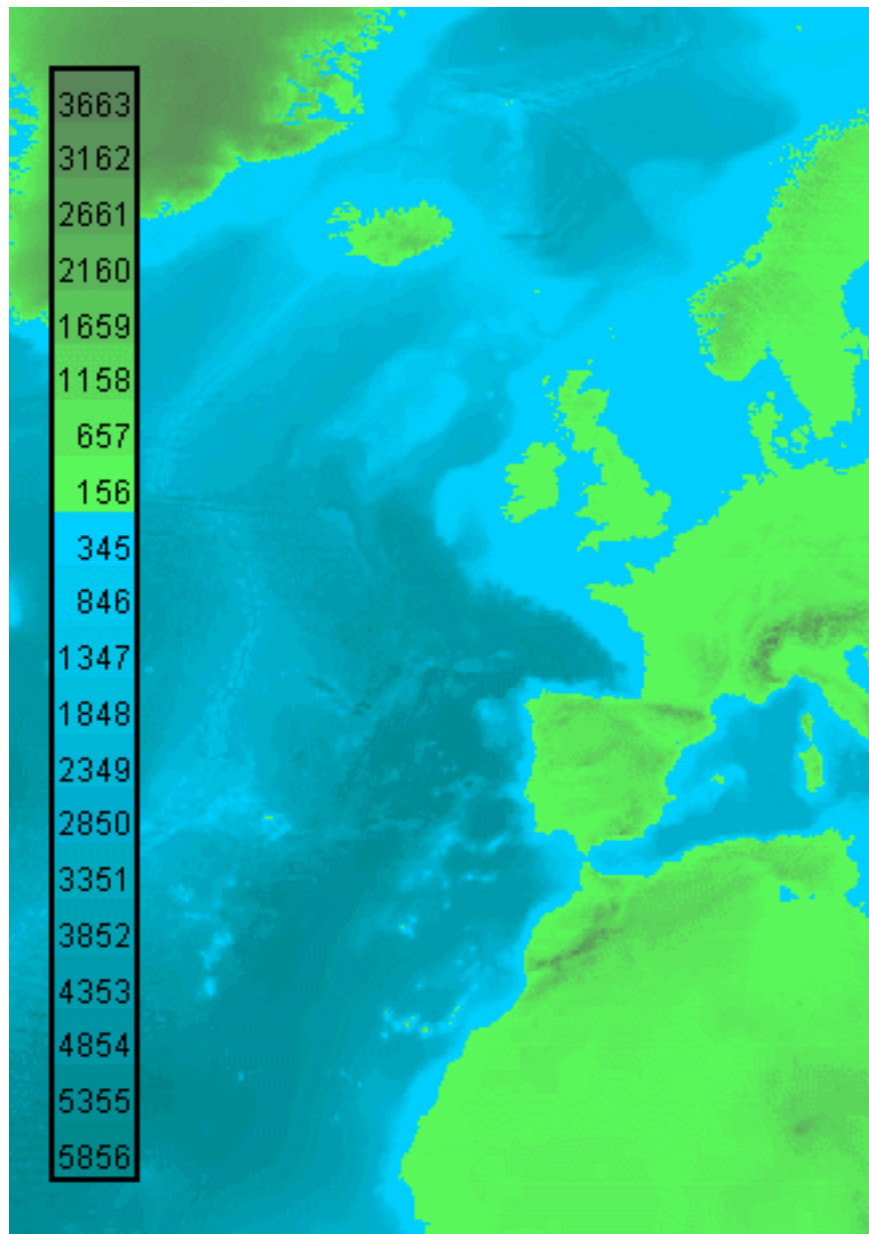
2000, 4000, 6000, and 8000 meter intervals. It also contains political boundaries together with significant roads and railways. The unclassified dataset is obtained from the US NIMA organisation, arriving on 4 CD-ROMS.

Figure 6.7. Sample of VPF data



Note

ETOPO is a global gridded depth/height database at 2 minute intervals. ETOPO is also obtained from NIMA, as well as within the full ASSET installation set.

Figure 6.8. Sample of ETOPO data

6. Interrogating the scenario

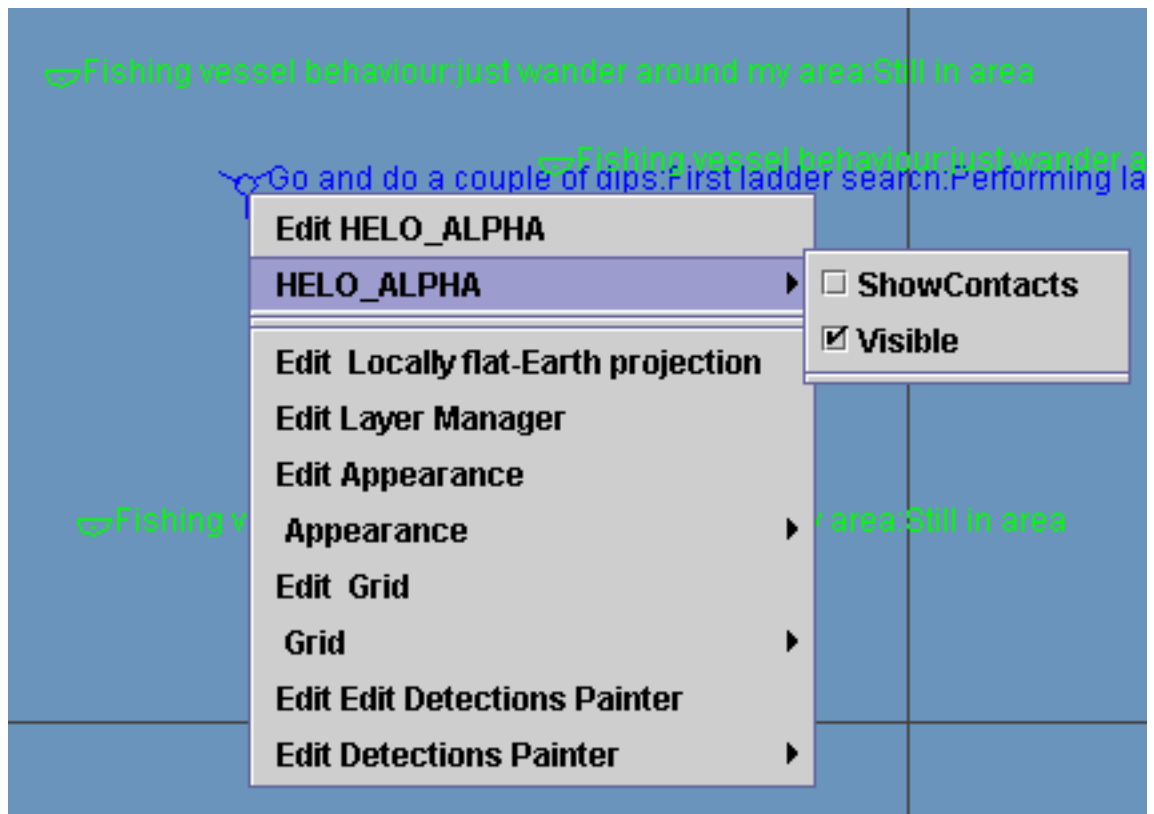
In addition to editing the plot within the Workbench we are also able to drill down into the content of the scenario. Two methods are typically used to start interrogating the scenario - double-clicking on the plot and using the Layer Manager.

6.1. Double-clicking the plot

The Workbench should currently be showing a zoomed in portion of the total tactical area, possibly with one or more scales or grids added from the previous step. If you're not looking at a view like this, just perform a Fit to Window, then zoom in to cover around 25% of the tactical area ensuring that you have included our Blue searcher. Hover the mouse over the blue helo and a tooltip will appear detailing the name of the participant, its current hierarchy of behaviours and current status. In the sample scenario loaded for the tutorial the tooltip text was: `HELO_ALPHA:Go` and do a couple of dips: `First ladder search:Performing ladder search:Heading`

for Point:2 21 degs 72 m_sec 152m. As you hover the mouse around the plot you will see the tooltip update to reflect the status of the nearest participant. If you right-click on an empty region of sea a popup-menu will appear. This menu contains options and commands used to change the format/presentation of the plot - attributes not directly related to any scenario participants. Alternatively, if you right-click on one of the participants then you will see the general commands at the bottom of the menu together with any specific to the current participant at the top (as shown below). The general trend of the popup menu is that each editable item has an entry titled "Edit xxx" which will open the property editor for that window, followed by a cascaded child menu containing directly editable attributes of that item. In the screenshot below we can see that the *ShowContact* attribute of HELO_ALPHA can be ticked (to switch the functionality on) if desired.

Figure 6.9. Sample of popup menu in Workbench



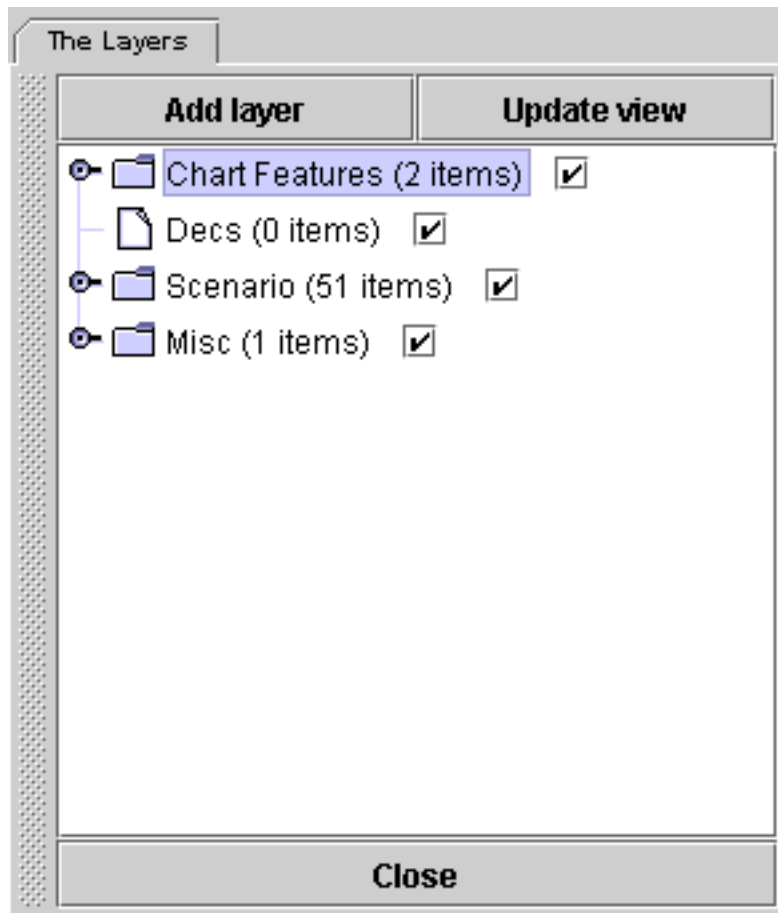
If you single-click anywhere on the plot then the popup menu will disappear. Finally, double-click on the blue helo, and its status editor will appear in the properties window.

6.2. Layer Manager

An alternative way of accessing the current scenario is through the Layer Manager. This window provides a tree-view of the currently loaded data. Open the Layer Manager by clicking on the Show



Layer Manager icon () from the File toolbar.

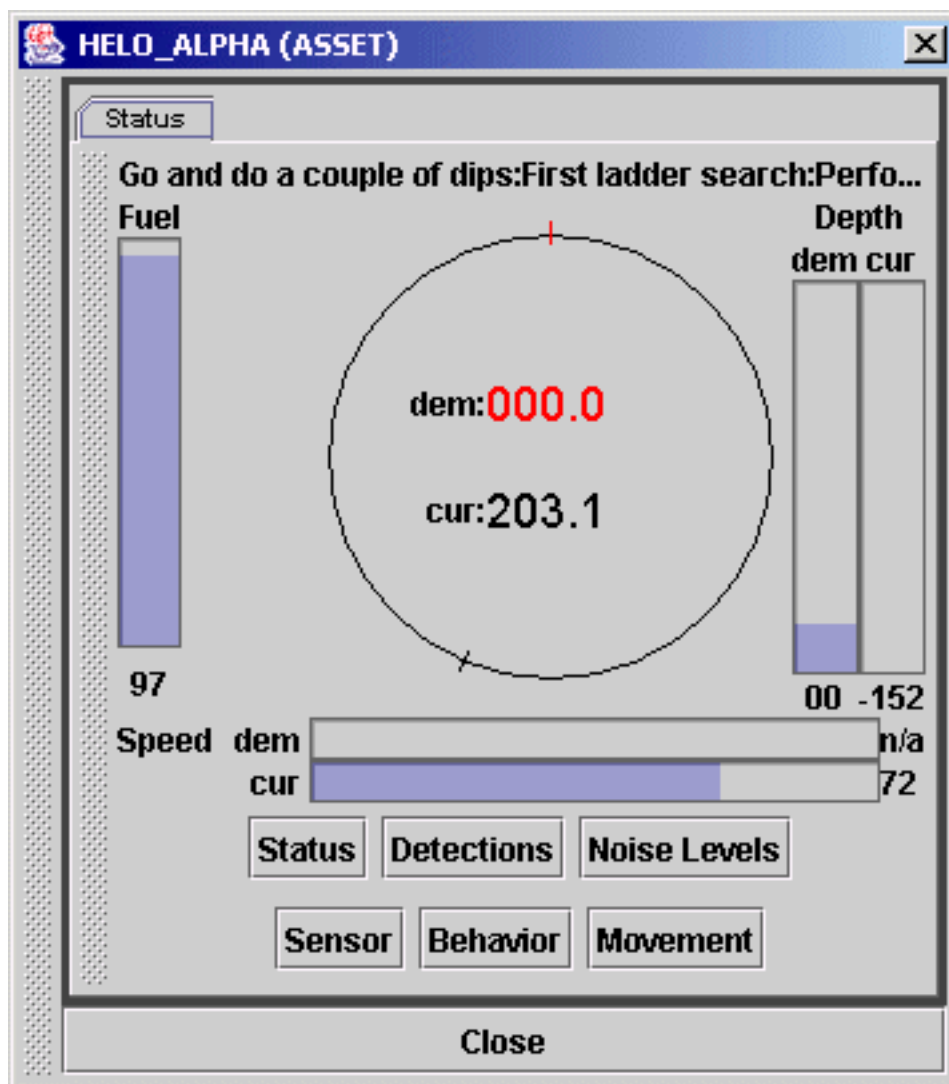
Figure 6.10. Workbench Layer Manager

In its startup view the Layer Manager provides a top-level view of the data collated into groups. In the example above the Chart Features group contains two items. Clicking on the key symbol to the left of the folder icon will open up that group and allow you to set the child items on/off or double-click to edit them. Of significance to this example the Scenario group contains 51 items (participants). Right-clicking on the Scenario group provides access to the Scenario formatting options. Open the cascading menu titled Edit Scenario and clear the tick box labelled Show Activity. At last all of the vessel activity labels have disappeared.

Back to the Layer Manager, opening the Scenario group will show all of the participants listed by name. In the example scenario loaded the blue helo HELO_ALPHA is listed first. Right-clicking on each item provides the popup menu seen earlier, and double-clicking on each one opens it in the vessel status editor.

6.3. Vessel Editor

So, at last we've got to the Vessel Editor itself.

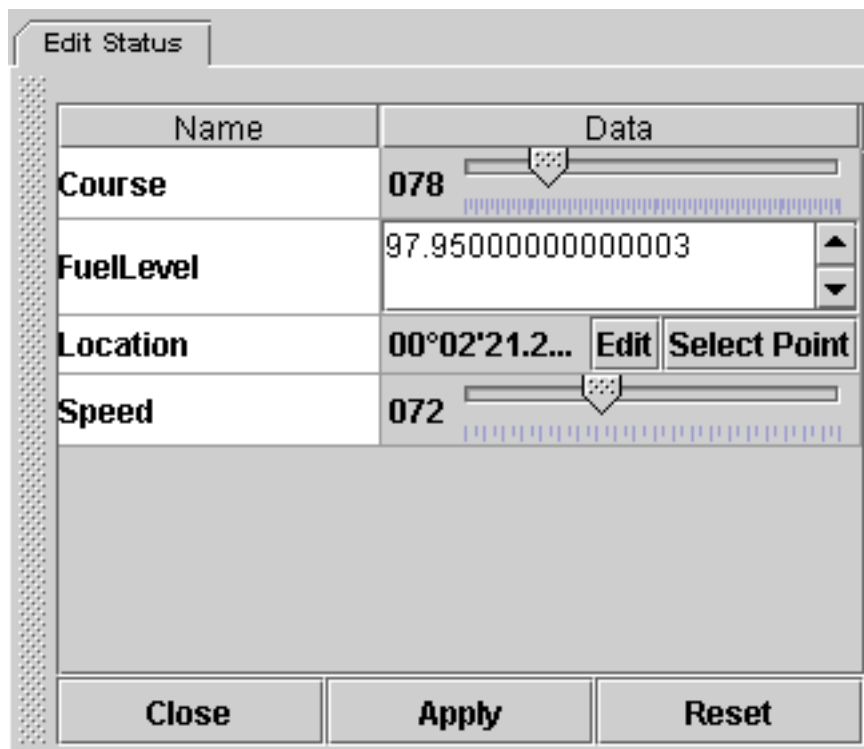
Figure 6.11. Vessel Editor

The editor itself is split into three zones. At the top is a line of text recording the current activity of the participant. In the centre are a series of graphic controls indicating the current and demanded vessel states (course, speed, depth, fuel level where applicable), and at the bottom are a series of buttons used to open further editors/monitors as follows.

Status	to view/modify the current vessel status
Detections	to view a waterfall display of sensor detections (where bearing is provided by the sensor)
Noise Levels	to view/modify the current radiated noise levels used in underwater detection modelling
Sensor	To view the detection components of in-contact sensors
Behaviour	To view/modify the participant behaviour by drilling down through the Waterfall of child behaviours
Movement	To view/modify the movement characteristics specific to this participant

6.3.1. Status Editor

Figure 6.12. Vessel Status Editor



The Vessel Status Editor dialog box features a tabbed interface with the 'Edit Status' tab selected. It contains a table with two columns: 'Name' and 'Data'. The table lists four vessel parameters: Course, FuelLevel, Location, and Speed. Course and Speed are represented by sliders with numerical readouts (078 and 072 respectively). FuelLevel is a text input field showing 97.95000000000003. Location is a text input field showing 00°02'21.2... with 'Edit' and 'Select Point' buttons. Below the table is a large empty rectangular area. At the bottom are three buttons: Close, Apply, and Reset.

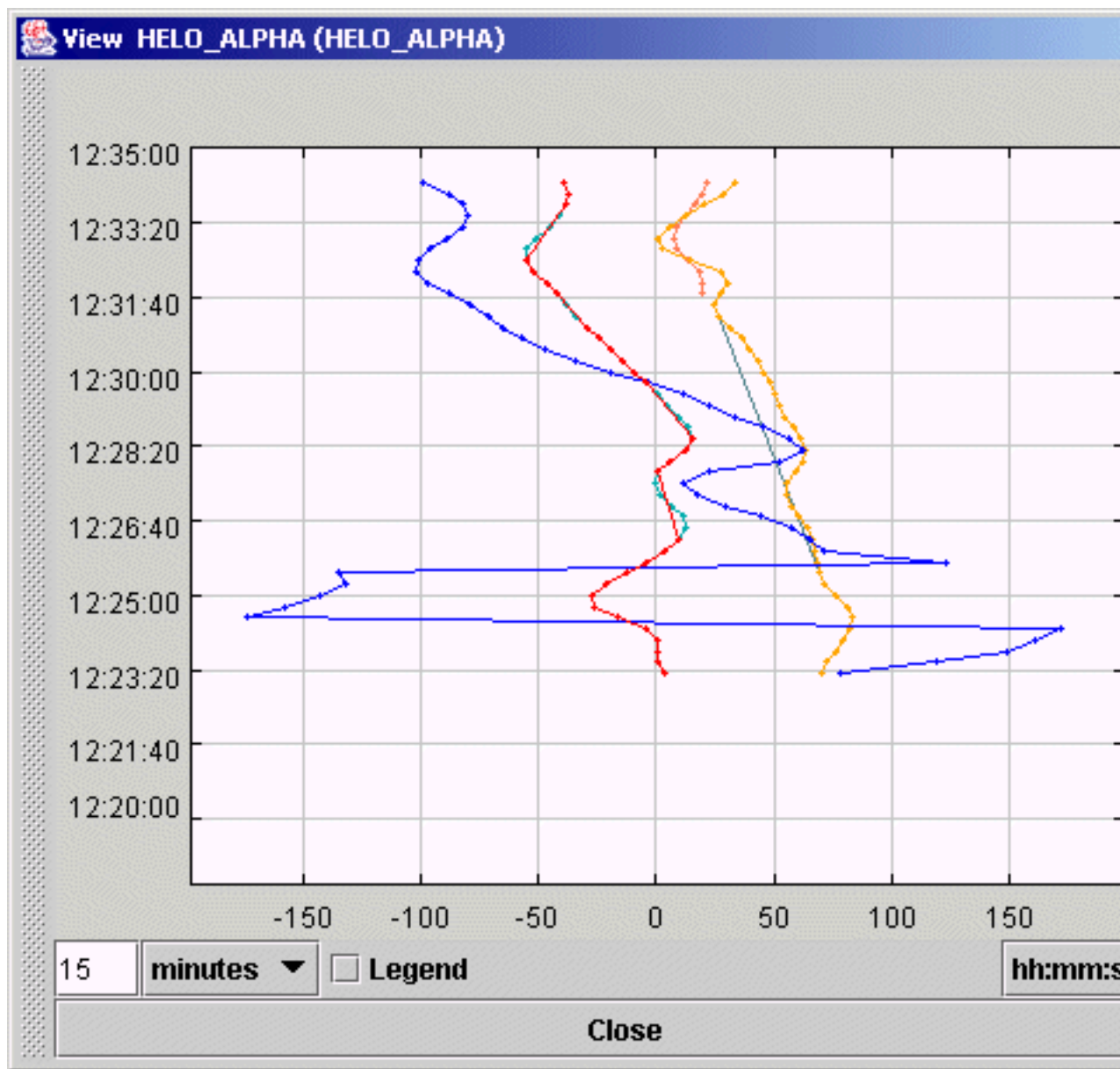
Name	Data
Course	078
FuelLevel	97.95000000000003
Location	00°02'21.2... Edit Select Point
Speed	072

Close Apply Reset

This panel shows the current (not demanded) course, speed, location and fuel level for this participant. Each of the values can be edited to change the vessel status - including moving it to some other location on the plot.

6.3.2. Detection Viewer

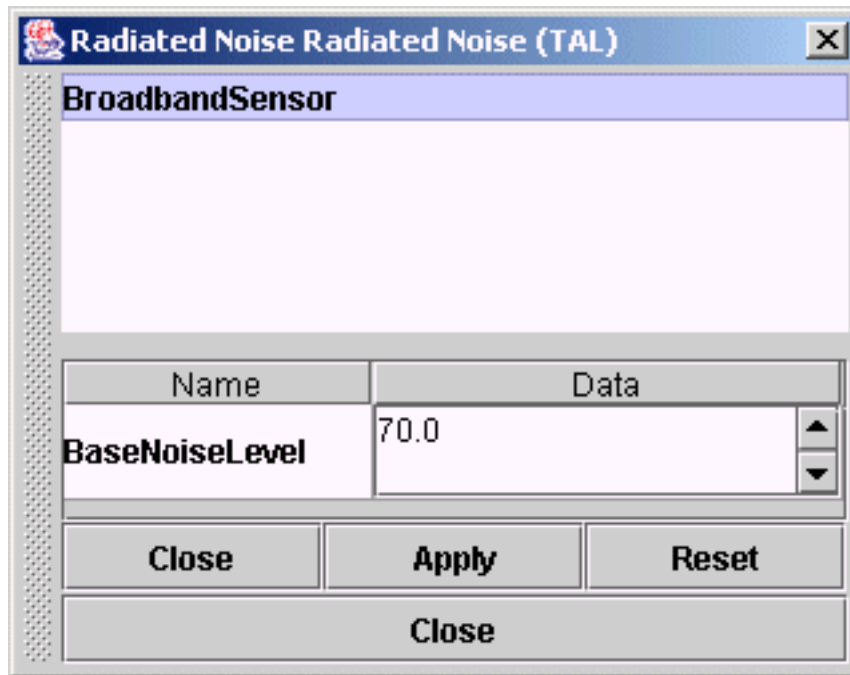
Figure 6.13. Sensor Detection Viewer



This viewer provides a scrolling waterfall of detection bearings. Whilst not all sensors provide bearing information, detections from those that do are shown in this panel. At the foot of the plot are controls to alter the amount of detection history shown, whether or not to show a legend, and the time format.

6.3.3. Radiated Noise Editor

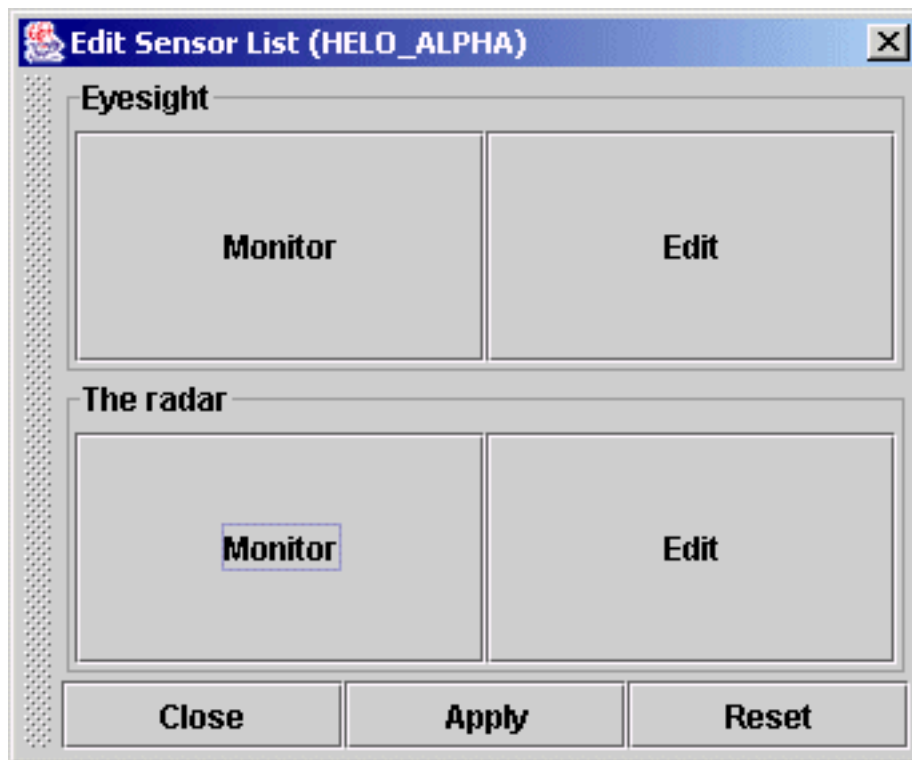
Figure 6.14. Radiated Noise Editor



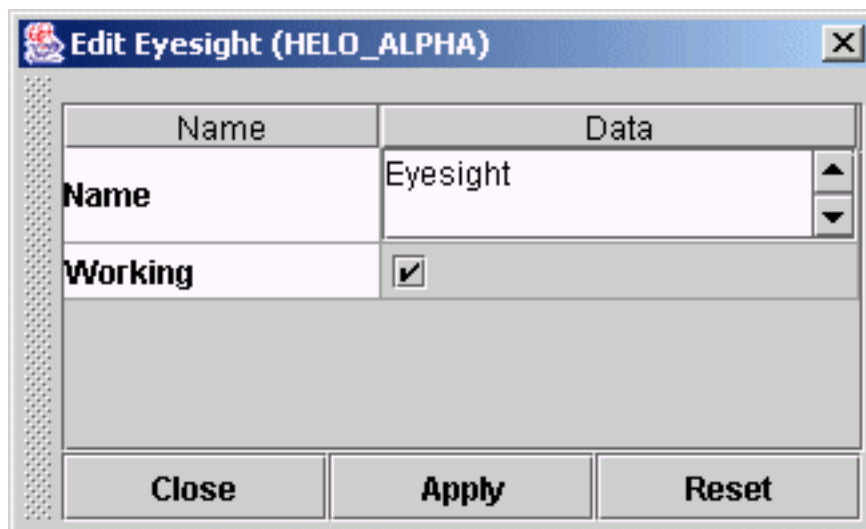
When utilising underwater detection modelling each participant must be provided with radiated noise characteristics, as used within the acoustic detection algorithms. This editor shows a list of the radiated noise mediums supported by the current participant, and if any are double-clicked the attributes of that medium are opened in an editor in the bottom half of the panel.

6.3.4. Sensor Monitor

The sensor monitor is build from two panels. The first provides a list of sensors for the current participant. For each sensor you are offered the choice of viewing its performance , viewing a plot of current contacts, or editing the sensor attributes.

Figure 6.15. Sensor Monitor Overview

Clicking on the Edit button will open the edit dialog. Different sensors have different editable attributes, but all share the core attributes of Name and Working - where obviously clearing the tick against the Working attribute disables that sensor.

Figure 6.16. Sensor Editor

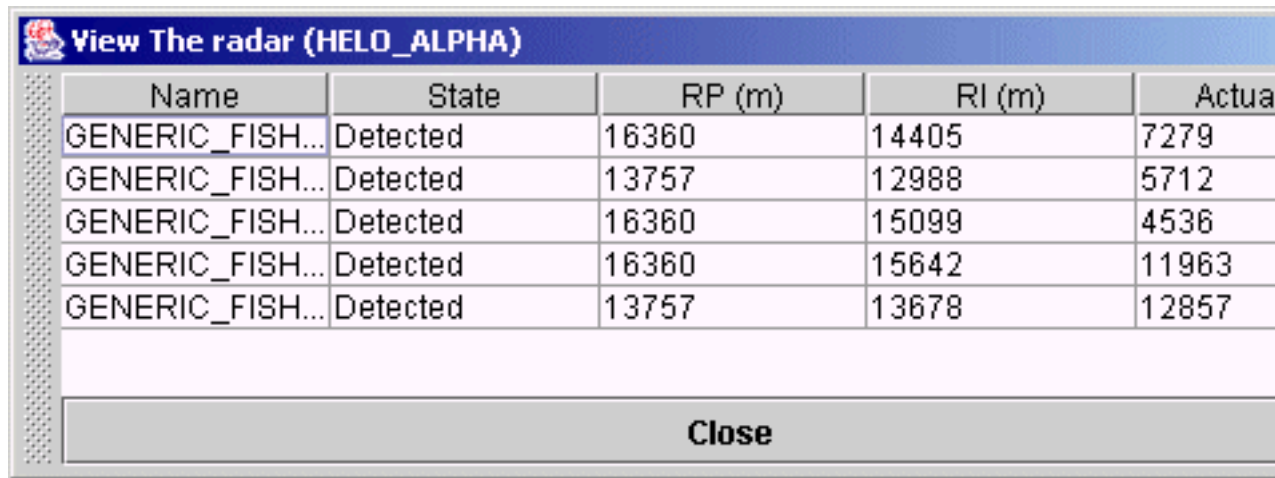
Alternatively, clicking on the Monitor tab opens a table which indicates the components of the detection algorithm for each in-contact target.

**Tip**

On first opening the table resizes to accommodate the current number of in-contact targets. If the table is opened when the current participant holds no targets it will be

of minimal size, and when in contact the targets may not be visible. Simple resize the table to view them.

Figure 6.17. Sensor Monitor Detail



The screenshot shows a window titled "View The radar (HELO_ALPHA)". Inside the window is a table with five columns: "Name", "State", "RP (m)", "RI (m)", and "Actua". There are five rows of data, all with "Detected" in the "State" column. The "Name" column contains the text "GENERIC_FISH...". The "RP (m)" column contains values 16360, 13757, 16360, 16360, and 13757. The "RI (m)" column contains values 14405, 12988, 15099, 15642, and 13678. The "Actua" column contains values 7279, 5712, 4536, 11963, and 12857. Below the table is a "Close" button.

Name	State	RP (m)	RI (m)	Actua
GENERIC_FISH...	Detected	16360	14405	7279
GENERIC_FISH...	Detected	13757	12988	5712
GENERIC_FISH...	Detected	16360	15099	4536
GENERIC_FISH...	Detected	16360	15642	11963
GENERIC_FISH...	Detected	13757	13678	12857

Close

Two types of detection modelling (underwater acoustic and lookup) are supported within ASSET, each based on a different core algorithm. The Sensor Monitor is able to present the components of either modelling mechanism, dynamically updated as the scenario moves forward.

Lastly, ASSET is able to provide a scrolling waterfall plot showing the detections for a particular sensor - similar to the plot provided for detections across a participant's full sensor fit.

6.3.5. Waterfall Editor

The Waterfall Editor is discussed further below.

6.3.6. Movement Editor

Each participant has its own set of manoeuvring characteristics, with different classes of participants having different types of characteristic available (fishing vessels do not have climb rates for instance). The movement editor allows you to edit those characteristics, changing acceleration, fuel usage rate, speed limits, etc.

Figure 6.18. Movement Editor

Name	Data
AccelRate	4 m/s/s ▼
DefaultCl...	6 m/s ▼
DefaultDi...	40 m/s ▼
FuelUsag...	1.0E-4 ▲▼
MaxHeight	300 m ▼
MaxSpeed	200 m/s ▼
Name	RAW CHARACTERISTICS ▲▼

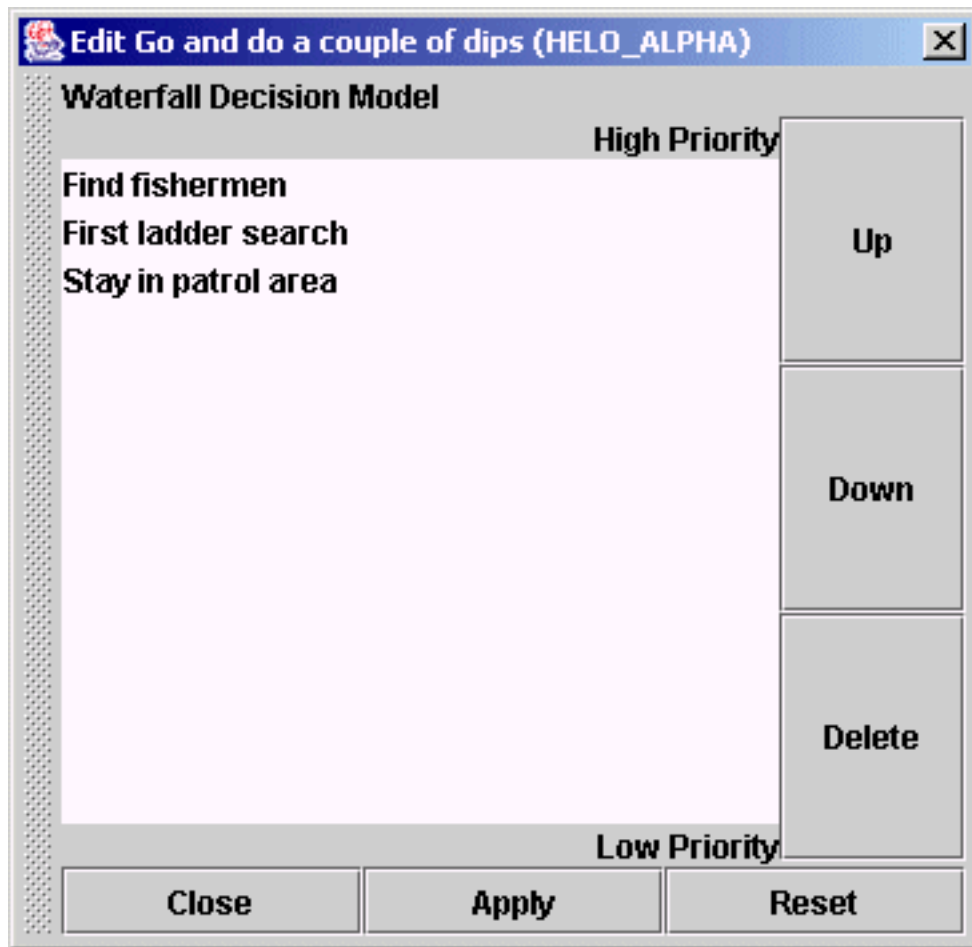
Close Apply Reset

6.4. Waterfall Editor

Design of participant behaviours forms a significant part of solving tactical problems using ASSET. The highest level behaviour within participants is the Waterfall, which contains a cascading list of child behaviours - if a higher level behaviour isn't applicable then control will cascade down to lower levels until one takes effect.

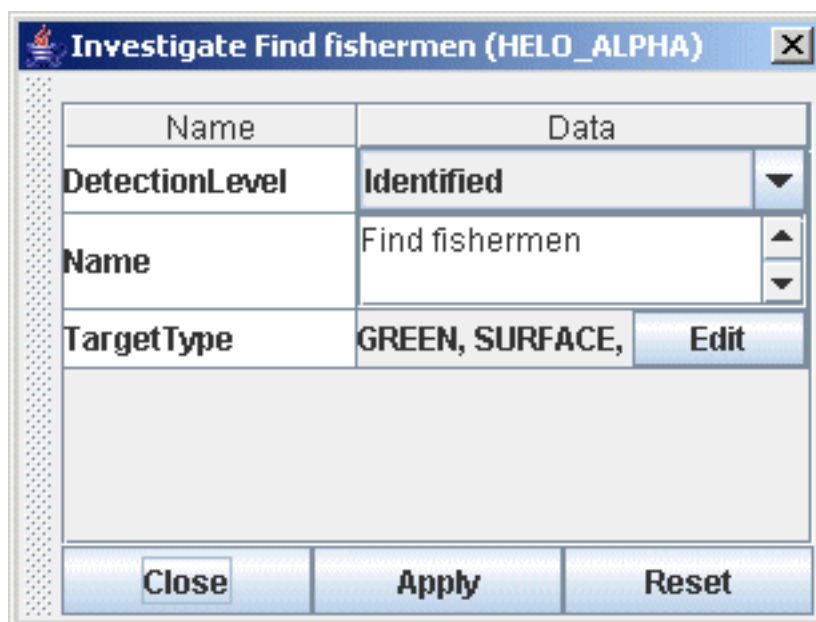
This cascade of behaviours is presented in the Waterfall editor. Within the initial panel (shown below) behaviours can be moved up and down the priority order, and double-clicking on any opens up their editor.

Figure 6.19. Waterfall Editor



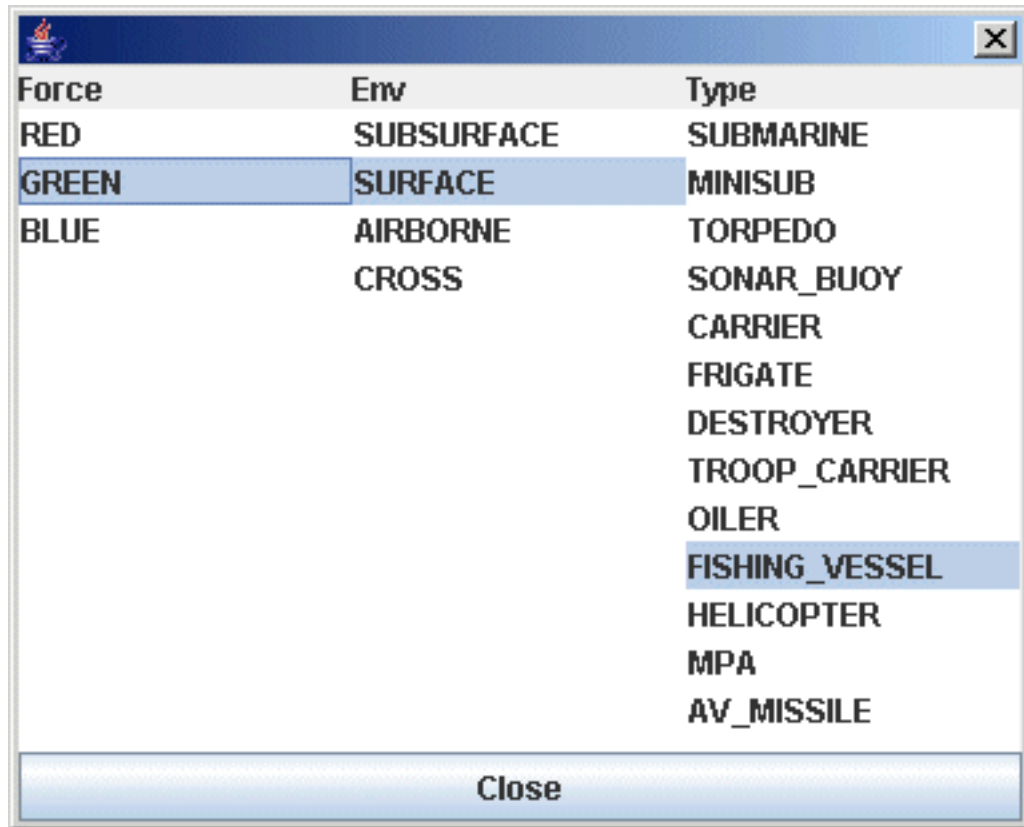
The editor for most child behaviours is a property editor looking similar to the property editor shown below, though some types of attribute are specific to behaviours - such as the category editor shown further down.

Figure 6.20. Behaviour Editor



Categories provide a useful way of making a body of vessels the subject of a behaviour without having to resort to individual filenames and/or id numbers. To identify a type of target as a subject of a behaviour select one or more categories from the lists provided (not all lists have to be represented). Thus, to indicate that a behaviour (such as Investigate) applies to all friendly fishing vessels select GREEN force and FISHING_VESSEL as the type. Alternatively, to indicate that the behaviour applies to all unarmed shipping also select the OILER and TROOP_CARRIER categories.

Figure 6.21. Category Editor



Chapter 7. Eclipse-based Workbench

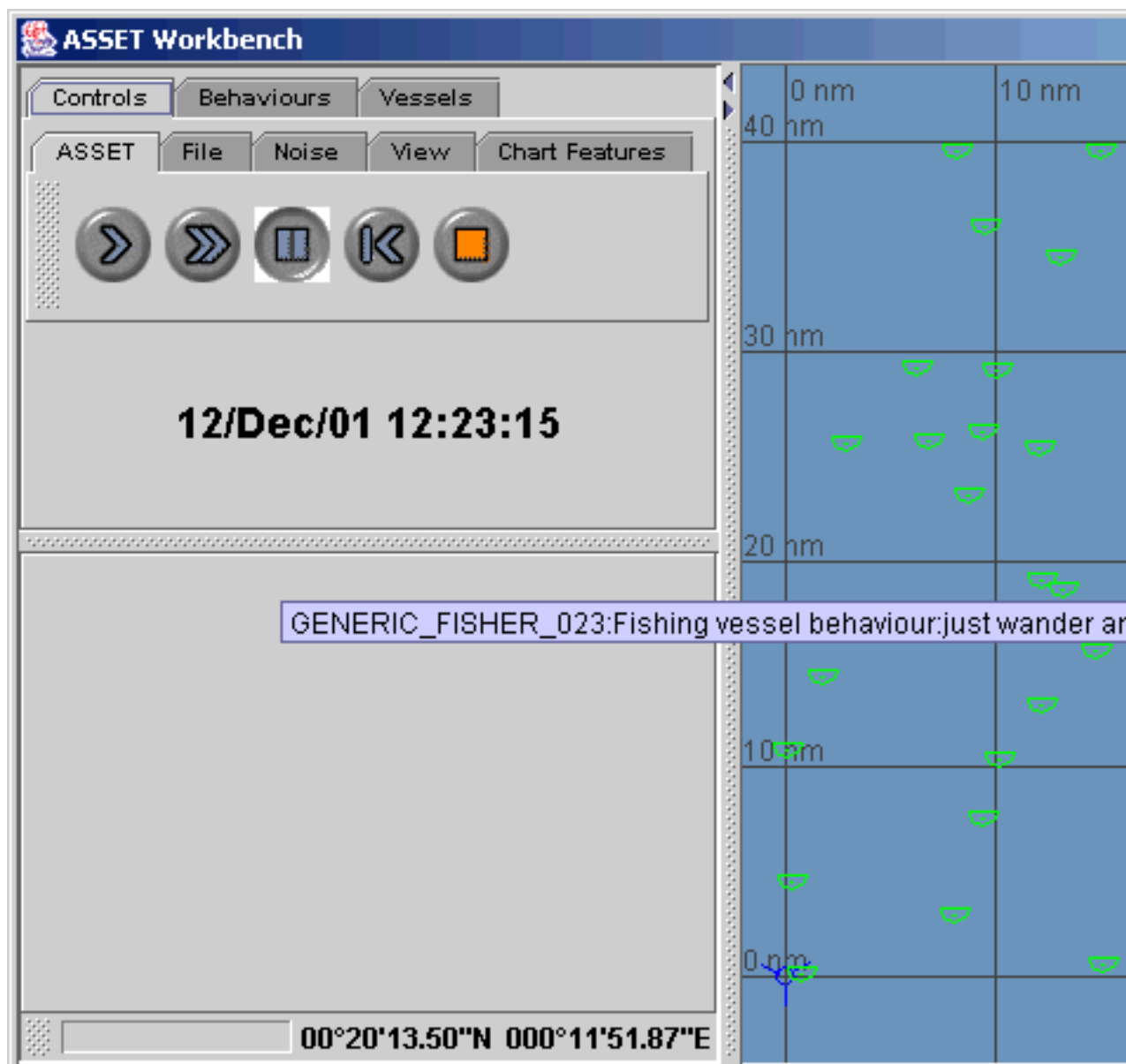
1. Introduction

The ASSET Workbench is a graphical front-end to the ASSET modelling engine. Scenario (and occasionally Control) files are loaded into the Workbench, and then you step forward through the scenario. Graphical dialogs are provided to allow you to drill-down into the detailed attributes, settings, and behaviours of the scenario participants - allowing (to a limited degree) interactive scenario design and modification to be performed. The Workbench application makes extensive re-use of components developed for MWC's Debrief [www.debrief.info] application - particularly with regard to the graphical plot, property editing, and core units (distance, locations, areas).

2. Workbench user interface

The following diagram highlights the main areas of the Workbench user interface:

Figure 7.1. Screenshot of Workbench application



Toolbar	In the top-left hand corner of the Workbench interface is the toolbar. This comprises a series of toolbars, one of which (Controls) contains its own series of toolbars. Lower sections provide more detail on individual toolbar buttons. In addition to the Controls toolbar, the Behaviours and Vessels toolbars provide a library of behaviours and vessels suitable for inclusion in a scenario. Drag a vessel onto the Plot to create and position it, drag a Behaviour onto the Waterfall editor to add that behaviour to the current vessel.
Tote	The tote is located immediately below the toolbar, and displays the current model time.
Properties Window	Below the Tote is the properties window, a holder for a wide range of editor panels - from the Layer Manager which shows all current data stored, through to the vessel status viewer which shows the course, speed, depth (demanded and current) for the selected participant, together with providing access to the editable attributes of that participant - all in the tabbed properties window.
Plot	The right-hand side of the screen is dedicated to the Plot itself, a 2-dimensional snapshot view of the scenario. Double-clicking on a participant's icon will open that participant in the properties window, and the Chart Features sub-toolbar of the Controls toolbar allows graphic items to be added to the plot (contours, grids, scales, annotations).

3. Loading Force Protection scenario

In this section we'll re-use the Force Protection scenario described above, but gain a feel for the interaction of the participants by graphically viewing the evolving scenario.

So, double-click on `workbench.exe` to open the Workbench. Almost immediately the ASSET Splash Screen (below) will display (significantly showing the build date of the current version in the lower right-hand corner), and a few seconds later the Workbench itself will be visible.

Figure 7.2. Workbench splash screen



The Workbench is going to generate a number of targets on our behalf in this run, so start by


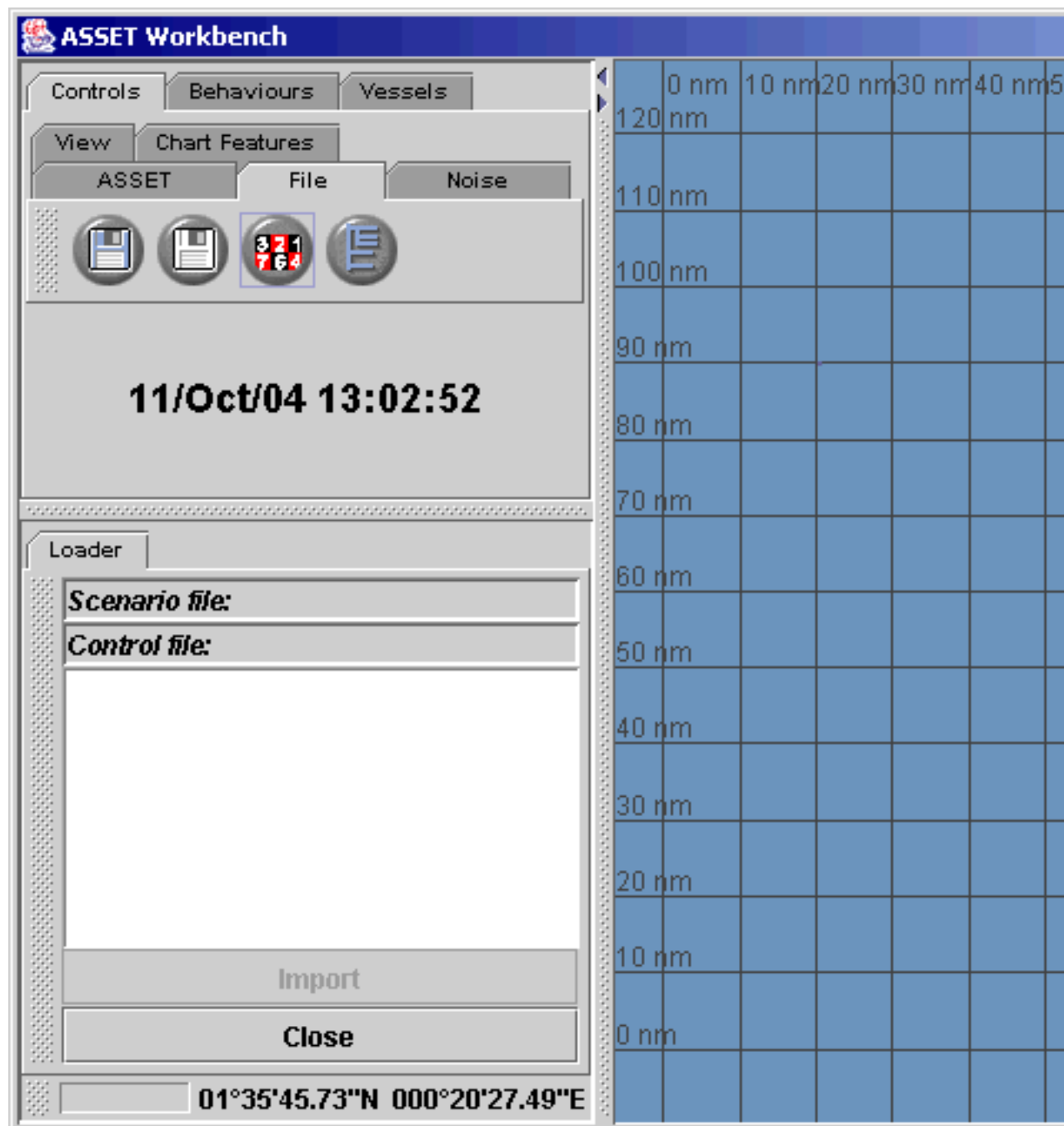
opening the Monte Carlo loader by clicking on the  button from the File sub-toolbar of the Controls toolbar. The Monte Carlo loader will open in the Properties Window, as shown below.

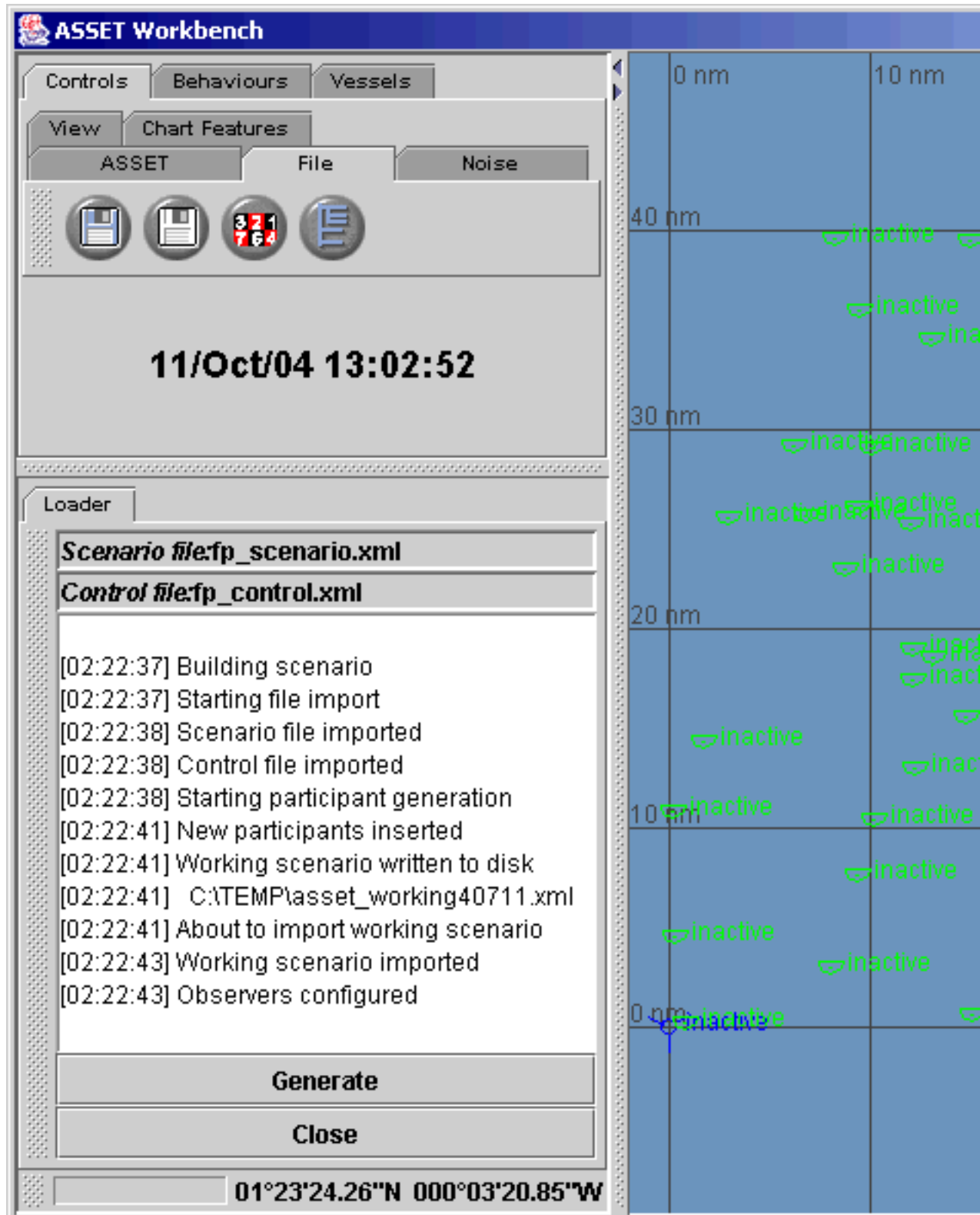
Figure 7.3. Monte Carlo file loader



As you can see, the Monte Carlo loader has labels indicating the Scenario and Control files, beneath this is a reporting window, and beneath this are the Import and Close buttons. The reporting window is provided to give you feedback during the potentially lengthy import/participant generation process. So, to start off with, drag and drop the scenario you created for the Force Protection exercise on to the *Scenario file* label. You will see the label get updated with the new filename, and the Import button become enabled. To just load this scenario as-is, you would click on the Import button, but we are just using the scenario as a template - wishing to create multiple instances of the

Green friendly fishing vessel. So, drag the Force Protection control file onto the *Control file:* label. As you do this, the Import button will change to Generate, acknowledging that you no longer just want to import the scenario, but wish to run it through the Multiple Participant Generator. So, press the Generate button and you will see a scrolling set of messages grow in the reporting window (below), followed by the new participants appearing on the Plot.

Figure 7.4. Participant generation complete



**Tip**

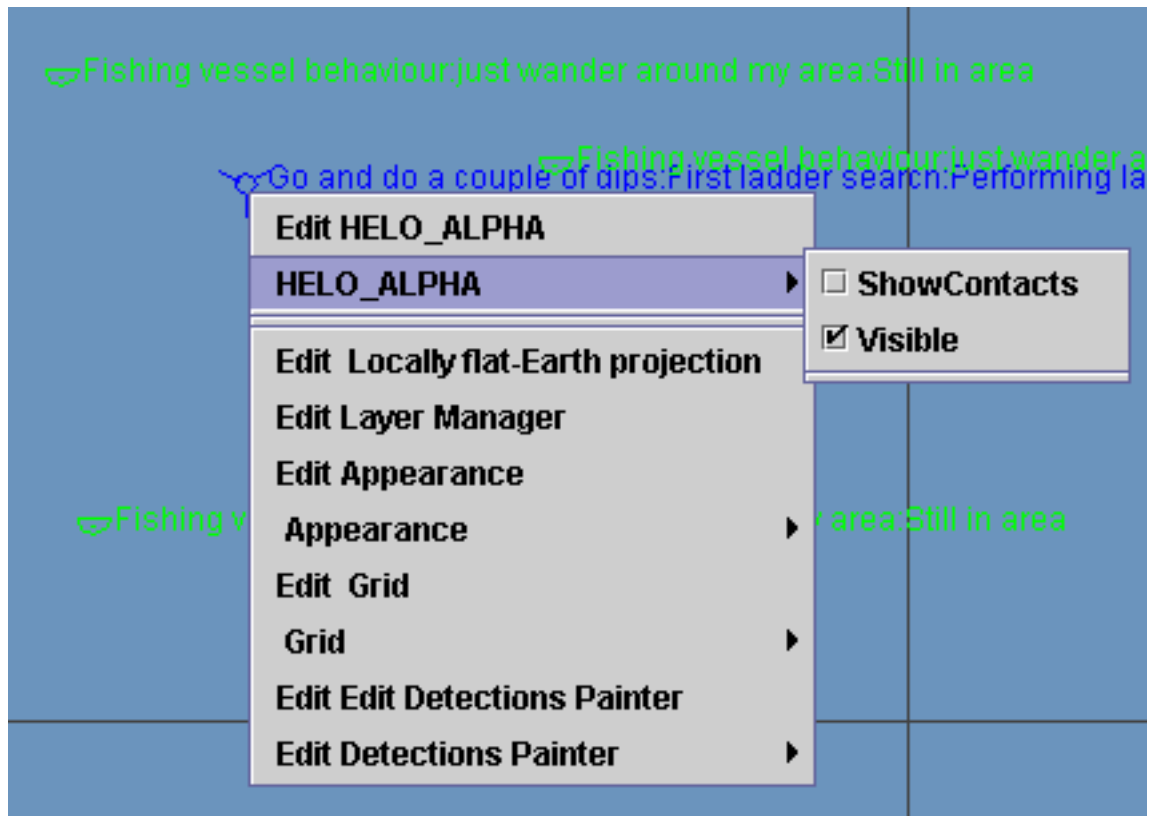
Some of the steps performed during multiple participant generation can take 10s of seconds - if nothing else the scrolling messages provide you with an indication that yes, something is happening.

4. Interrogating the scenario

In addition to editing the plot within the Workbench we are also able to drill down into the content of the scenario. Two methods are typically used to start interrogating the scenario - double-clicking on the plot and using the Layer Manager.

4.1. Double-clicking the plot

The Workbench should currently be showing a zoomed in portion of the total tactical area, possibly with one or more scales or grids added from the previous step. If you're not looking at a view like this, just perform a Fit to Window, then zoom in to cover around 25% of the tactical area ensuring that you have included our Blue searcher. Hover the mouse over the blue helo and a tooltip will appear detailing the name of the participant, its current hierarchy of behaviours and current status. In the sample scenario loaded for the tutorial the tooltip text was: `HELO_ALPHA:Go and do a couple of dips:First ladder search:Performing ladder search:Heading for Point:2 21 degs 72 m_sec 152m`. As you hover the mouse around the plot you will see the tooltip update to reflect the status of the nearest participant. If you right-click on an empty region of sea a popup-menu will appear. This menu contains options and commands used to change the format/presentation of the plot - attributes not directly related to any scenario participants. Alternatively, if you right-click on one of the participants then you will see the general commands at the bottom of the menu together with any specific to the current participant at the top (as shown below). The general trend of the popup menu is that each editable item has an entry titled "Edit xxx" which will open the property editor for that window, followed by a cascaded child menu containing directly editable attributes of that item. In the screenshot below we can see that the *ShowContact* attribute of HELO_ALPHA can be ticked (to switch the functionality on) if desired.

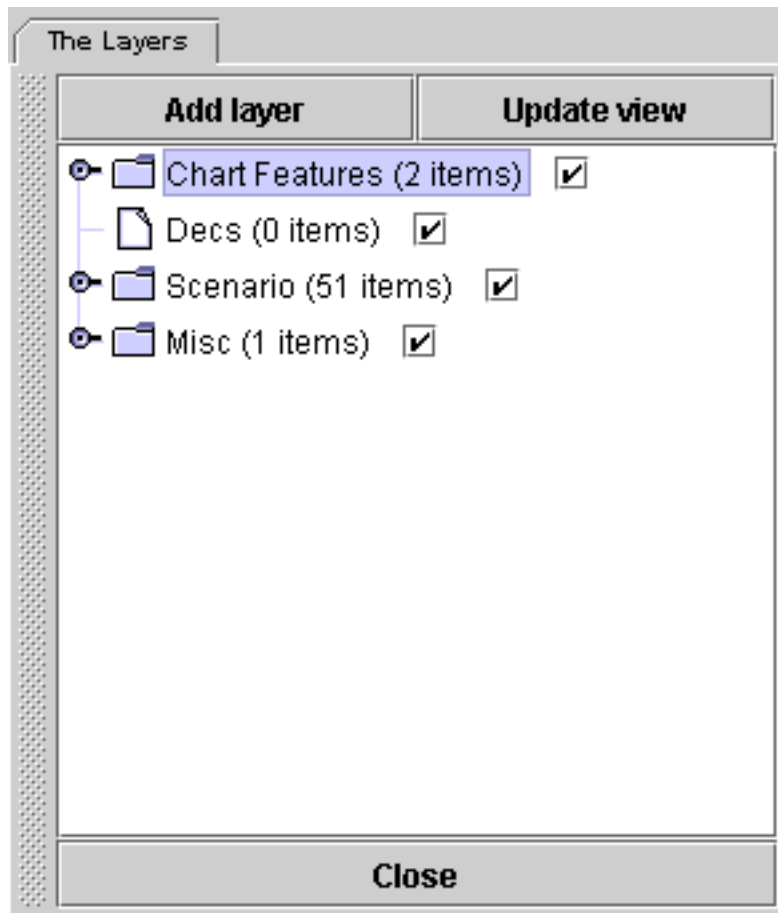
Figure 7.5. Sample of popup menu in Workbench

If you single-click anywhere on the plot then the popup menu will disappear. Finally, double-click on the blue helo, and its status editor will appear in the properties window.

4.2. Layer Manager

An alternative way of accessing the current scenario is through the Layer Manager. This window provides a tree-view of the currently loaded data. Open the Layer Manager by clicking on the Show

Layer Manager icon () from the File toolbar.

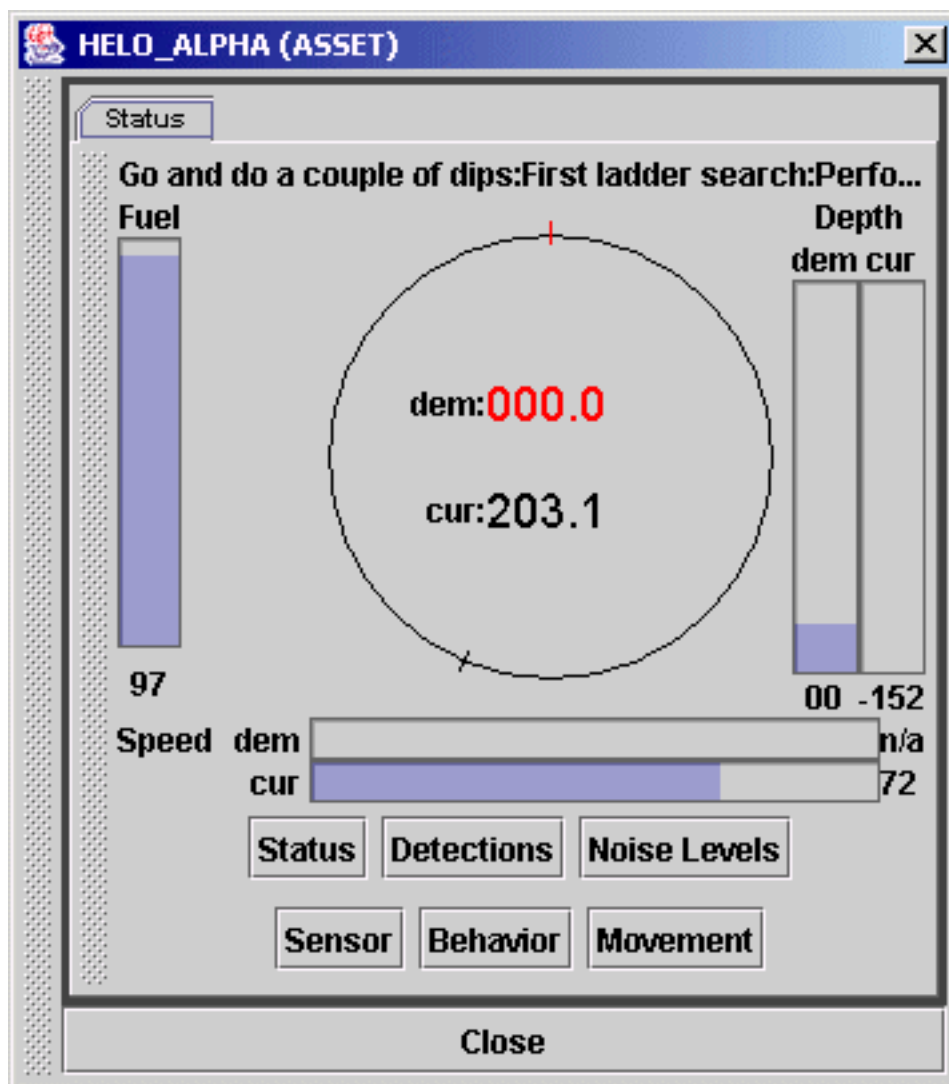
Figure 7.6. Workbench Layer Manager

In its startup view the Layer Manager provides a top-level view of the data collated into groups. In the example above the Chart Features group contains two items. Clicking on the key symbol to the left of the folder icon will open up that group and allow you to set the child items on/off or double-click to edit them. Of significance to this example the Scenario group contains 51 items (participants). Right-clicking on the Scenario group provides access to the Scenario formatting options. Open the cascading menu titled Edit Scenario and clear the tick box labelled Show Activity. At last all of the vessel activity labels have disappeared.

Back to the Layer Manager, opening the Scenario group will show all of the participants listed by name. In the example scenario loaded the blue helo HELO_ALPHA is listed first. Right-clicking on each item provides the popup menu seen earlier, and double-clicking on each one opens it in the vessel status editor.

4.3. Vessel Editor

So, at last we've go to the Vessel Editor itself.

Figure 7.7. Vessel Editor

The editor itself is split into three zones. At the top is a line of text recording the current activity of the participant. In the centre are a series of graphic controls indicating the current and demanded vessel states (course, speed, depth, fuel level where applicable), and at the bottom are a series of buttons used to open further editors/monitors as follows.

Status	to view/modify the current vessel status
Detections	to view a waterfall display of sensor detections (where bearing is provided by the sensor)
Noise Levels	to view/modify the current radiated noise levels used in underwater detection modelling
Sensor	To view the detection components of in-contact sensors
Behaviour	To view/modify the participant behaviour by drilling down through the Waterfall of child behaviours
Movement	To view/modify the movement characteristics specific to this participant

4.3.1. Status Editor

Figure 7.8. Vessel Status Editor

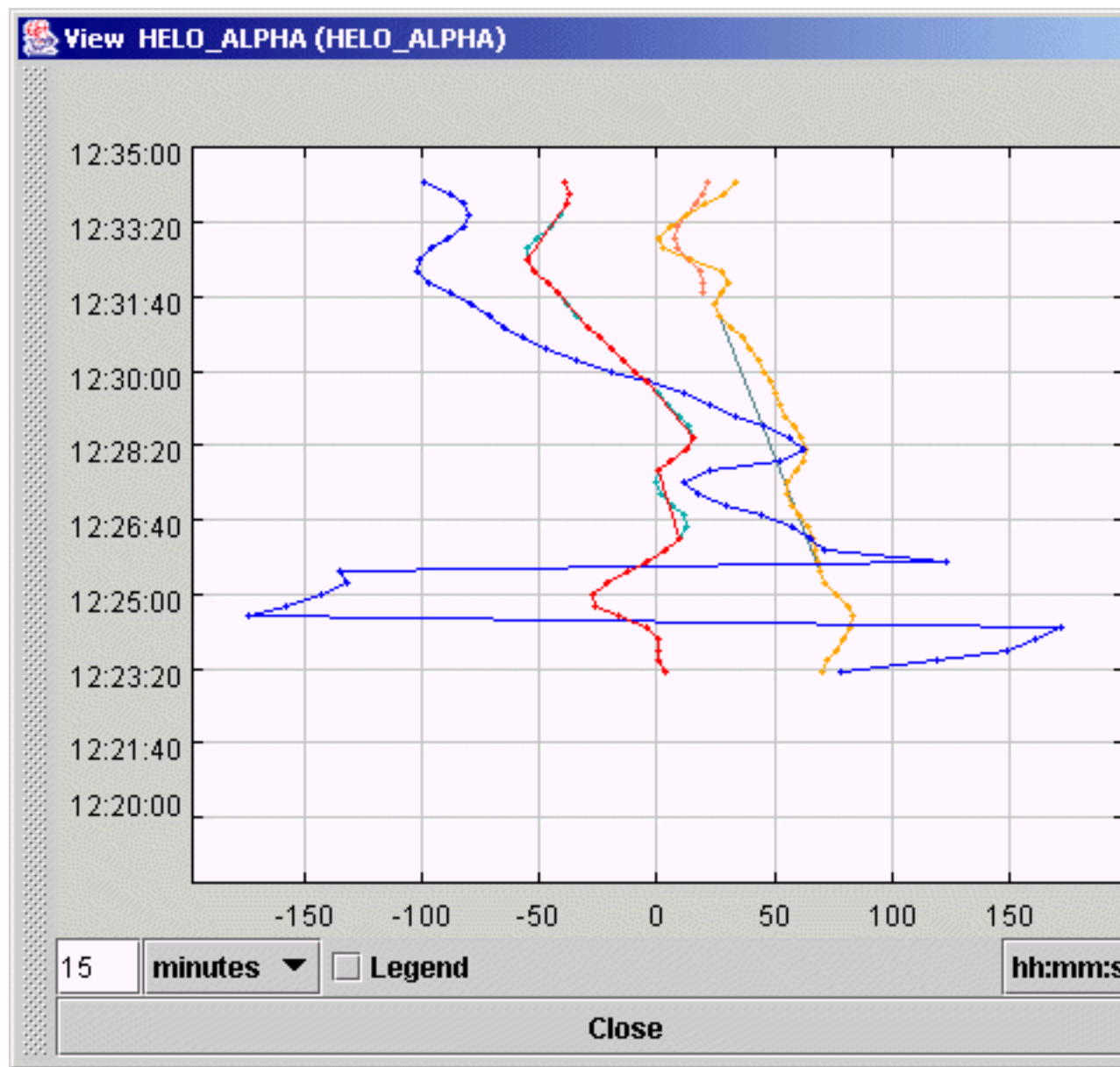
Name	Data
Course	078
FuelLevel	97.950000000000003
Location	00°02'21.2... Edit Select Point
Speed	072

Close Apply Reset

This panel shows the current (not demanded) course, speed, location and fuel level for this participant. Each of the values can be edited to change the vessel status - including moving it to some other location on the plot.

4.3.2. Detection Viewer

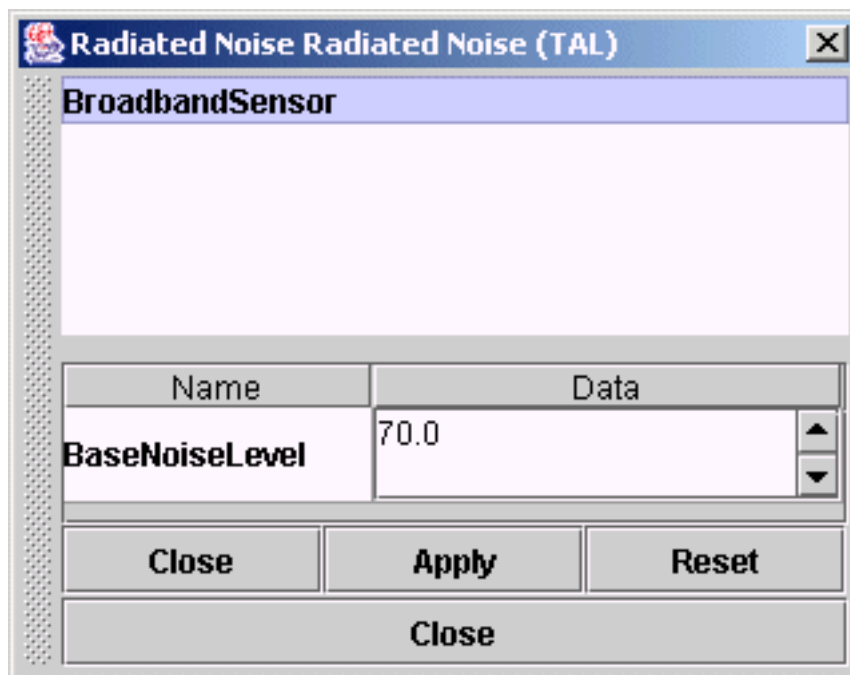
Figure 7.9. Sensor Detection Viewer



This viewer provides a scrolling waterfall of detection bearings. Whilst not all sensors provide bearing information, detections from those that do are shown in this panel. At the foot of the plot are controls to alter the amount of detection history shown, whether or not to show a legend, and the time format.

4.3.3. Radiated Noise Editor

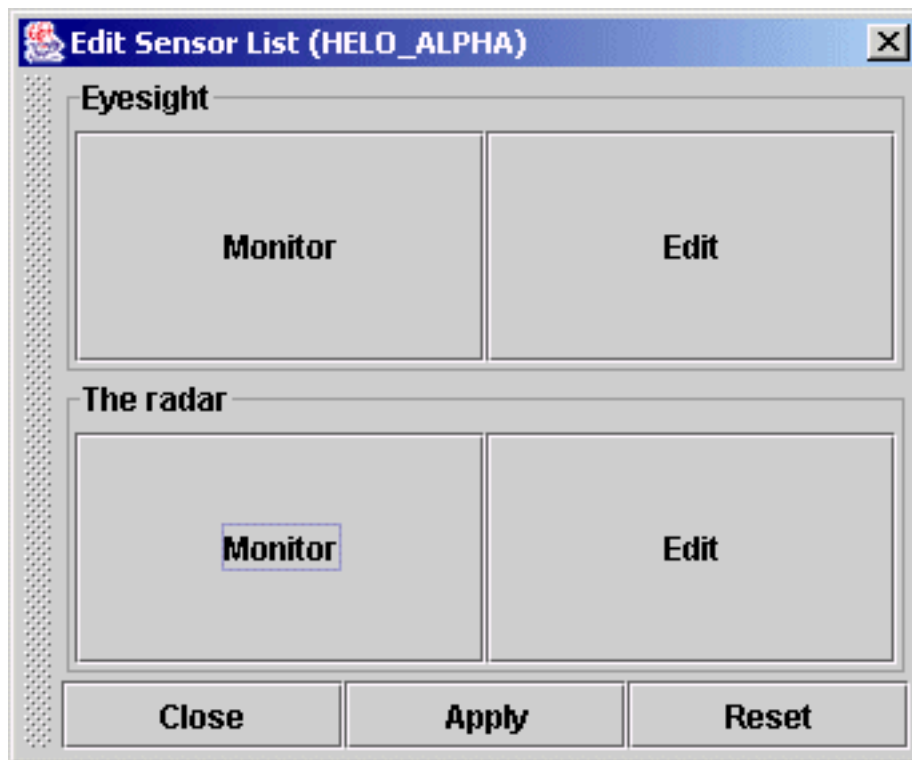
Figure 7.10. Radiated Noise Editor



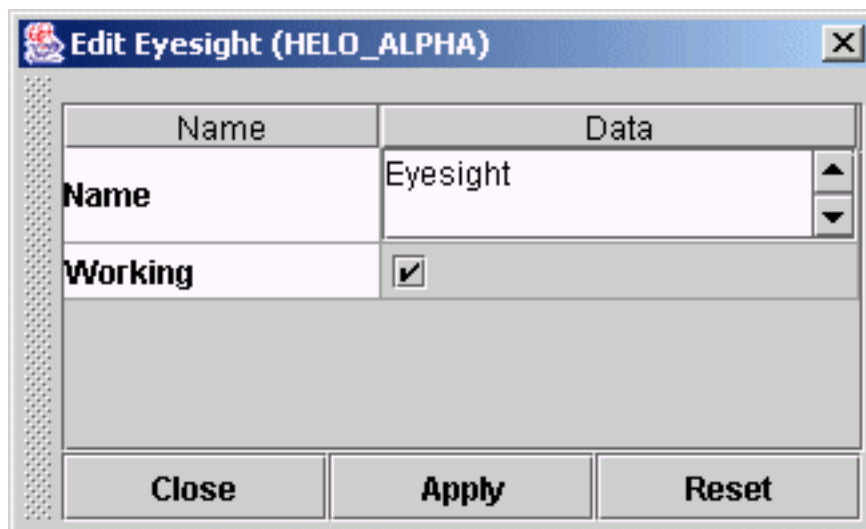
When utilising underwater detection modelling each participant must be provided with radiated noise characteristics, as used within the acoustic detection algorithms. This editor shows a list of the radiated noise mediums supported by the current participant, and if any are double-clicked the attributes of that medium are opened in an editor in the bottom half of the panel.

4.3.4. Sensor Monitor

The sensor monitor is build from two panels. The first provides a list of sensors for the current participant. For each sensor you are offered the choice of viewing its performance , viewing a plot of current contacts, or editing the sensor attributes.

Figure 7.11. Sensor Monitor Overview

Clicking on the Edit button will open the edit dialog. Different sensors have different editable attributes, but all share the core attributes of Name and Working - where obviously clearing the tick against the Working attribute disables that sensor.

Figure 7.12. Sensor Editor

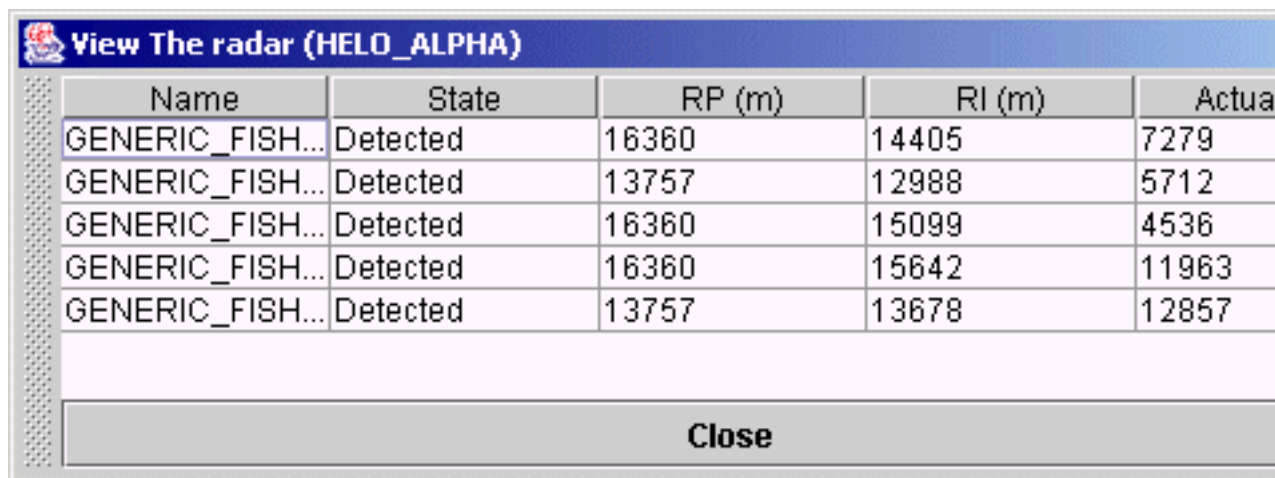
Alternatively, clicking on the Monitor tab opens a table which indicates the components of the detection algorithm for each in-contact target.

**Tip**

On first opening the table resizes to accommodate the current number of in-contact targets. If the table is opened when the current participant holds no targets it will be

of minimal size, and when in contact the targets may not be visible. Simple resize the table to view them.

Figure 7.13. Sensor Monitor Detail



The screenshot shows a window titled "View The radar (HELO_ALPHA)". Inside the window is a table with five columns: "Name", "State", "RP (m)", "RI (m)", and "Actua". There are five rows of data, all with "Detected" in the "State" column. The "Name" column contains the text "GENERIC_FISH...". The "RP (m)" column contains values 16360, 13757, 16360, 16360, and 13757. The "RI (m)" column contains values 14405, 12988, 15099, 15642, and 13678. The "Actua" column contains values 7279, 5712, 4536, 11963, and 12857. Below the table is a "Close" button.

Name	State	RP (m)	RI (m)	Actua
GENERIC_FISH...	Detected	16360	14405	7279
GENERIC_FISH...	Detected	13757	12988	5712
GENERIC_FISH...	Detected	16360	15099	4536
GENERIC_FISH...	Detected	16360	15642	11963
GENERIC_FISH...	Detected	13757	13678	12857

Close

Two types of detection modelling (underwater acoustic and lookup) are supported within ASSET, each based on a different core algorithm. The Sensor Monitor is able to present the components of either modelling mechanism, dynamically updated as the scenario moves forward.

Lastly, ASSET is able to provide a scrolling waterfall plot showing the detections for a particular sensor - similar to the plot provided for detections across a participant's full sensor fit.

4.3.5. Waterfall Editor

The Waterfall Editor is discussed further below.

4.3.6. Movement Editor

Each participant has its own set of manoeuvring characteristics, with different classes of participants having different types of characteristic available (fishing vessels do not have climb rates for instance). The movement editor allows you to edit those characteristics, changing acceleration, fuel usage rate, speed limits, etc.

Figure 7.14. Movement Editor

Name	Data
AccelRate	4 m/s/s
DefaultCL...	6 m/s
DefaultDi...	40 m/s
FuelUsag...	1.0E-4
MaxHeight	300 m
MaxSpeed	200 m/s
Name	RAW CHARACTERISTICS

Close Apply Reset

Chapter 8. Submarine Search

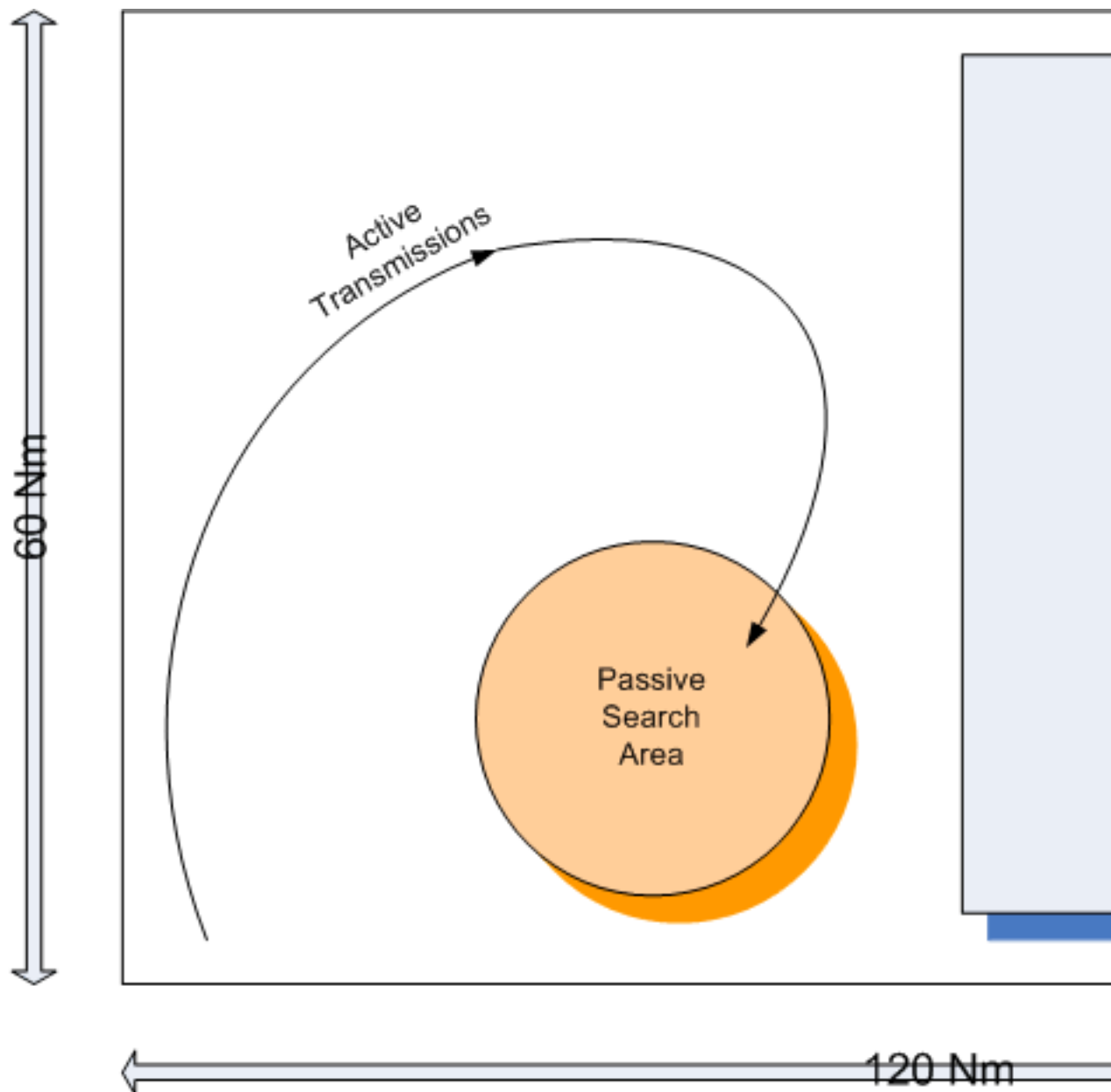
1. Introduction

The example scenarios presented so far largely focus on helicopter based tactics - with interactions dominated by detections from above-water sensors. This section of the tutorial will switch to the other significant area of detection modelling within ASSET, focussing on underwater acoustic sensors.

The tactical problem to be considered entails an SSN performing a series of movements and active sonar transmissions (as shown in the diagram below) with the intention of influencing an opposing force SSK. Airborne participants will provide radar coverage, though this will be modelled at a level above the MPAs themselves; through a detection performance observer.

The analysis of the problem will include variation in the SSN's prescribed search plan and the behaviour (aggressive vs evasive) of the SSK. We will start off by performing a series of command-line Monte Carlo simulations to work through a large number of permutations, then further investigate scenarios of interest using the Workbench GUI environment.

Since the SSN in this scenario is not responsive to the SSKs, we are able to use parallel planes processing - our scenario will contain the single blue searcher but many red ssks all operating in parallel planes unaware of each other. This will both bring performance advantages, and allow the multiple SSK permutations to be monitored within a single Workbench session.

Figure 8.1. Overview of submarine search problem

2. Preparing the data

2.1. Environment

Start off by generating the scenario file, inserting the desired environment at the start (note that the environmental tables used for lookup sensors are not required here - no lookup sensors are required for the scenario). I suggest a low sea-state and relatively clear atmosphere to start with to encourage interaction.

2.2. Participants

2.2.1. SSK

Next come with the participants, starting with the SSK. Give it a name (obviously), something like *Target* should do. Next comes its charge rate, I found 0.005 gave realistic enough results. Lastly

indicate that the SSK is a Monte Carlo Target - to receive special parallel planes processing. Then comes the Category, which will indicate that this is a SUBSURFACE, RED, SUBMARINE.

The sensor fit represents the next data to record, for our example we'll equip the SSK with Broadband Sensor and Active Intercept sensors (giving the broadband sensor an aperture of 130 degrees). For the initial status of the SSK place it using a Relative Location at (0nm,0nm,-60m), with a course of 0, fuel level of 100, and speed of 10 kts. Before we take on the complex task of defining the SSK behaviour we'll finish off the simpler components. So, give the SSK a RadiatedCharacteristics item, and insert SSKBroadband radiated noise dataset. Specify a Base Noise Level of 70, and a Snort Noise Level of 120.

As a reminder, the SSK has two broadband radiated noise levels: a quiet one for when running deep on main motor (battery-powered) and a noisier one for when snorting at periscope depth.

Lastly indicate the submarine's movement characteristics, using the SSMovementCharacteristics structure. I used the following values:

```
<SSMovementCharacteristics FuelUsageRate="0.0002" Name="SSN Mov
<MinSpeed Units="kts" Value="1"/>
<MaxSpeed Units="kts" Value="12"/>
<AccelerationRate Units="kt/s" Value="1"/>
<DecelerationRate Units="kt/s" Value="1"/>
<MinHeight Units="m" Value="-300"/>
<MaxHeight Units="m" Value="-18"/>
<DefaultClimbRate Units="ft/s" Value="1"/>
<DefaultDiveRate Units="ft/s" Value="1"/>
<TurningCircle Units="m" Value="600"/>
</SSMovementCharacteristics>
```

The last item to be entered for the SSK is its behaviour. In support of our analysis objectives we're going to supply the SSK with two alternate behaviours. Our Monte Carlo processing can then provide target instances demonstrating either an Aggressive or Evasive posture. The two high level behaviours are contained in a Switch container. So, start off by inserting a Switch object immediately below the SSK status. Give the switch a name, and indicate the default behaviour to be executed (1 will do). Each of the two behaviours will be contained in a Waterfall structure. We'll start off with the Evasive behaviour. So insert the Waterfall structure, and give it a name of something like Evasive. Into the Waterfall we will insert the hierarchical SSK evade behaviour, starting with the highest priority. In this instance we will make the SSK maintaining it's batteries at a safe level the highest priority behaviour, so insert SSKRecharge from the Elements tab. Name it as Emergency Snort, set the minimum level as 5 and the safe level as 15, and indicate that the SSK will snort at a speed of 4 knots.

So, when the SSK battery level reaches dangerously low levels it will conduct a snort however hostile its current environment may be. Obviously it won't conduct a very long snort - just enough to keep the SSK operational.

The next component of the SSK evasive behaviour will cover the evasion itself. Beneath the SSK Recharge component insert an Evade component, naming it Avoid SSN. Configure this behaviour to continue evading for a period of 30 minutes, at a depth of 80m (that's a height of -80m by the way), and at a speed of 6 knots. Lastly, indicate that the SSK will be evading all Blue targets. To specify Blue force participants as targets, put the cursor into the TargetType element, and double-click Type from the Add child tab of the Elements toolbox. Within the Type element select BLUE from the drop-down list.

Beneath the Evade behaviour insert another SSK Recharge behaviour, to represent normal snort behaviour. This snort behaviour will be conducted whenever the battery level falls below 30%, on which the SSK will attempt to snort back up to 100%, though not when in contact with any blue targets. Set the EvadeThese type to BLUE as you did with the Evade behaviour above.

The lowest-level component of the Evasive SSK behaviour is the general patrol behaviour. This is represented through a Rectangle Wander element, inserted from the Add child tab of the Elements toolbox, giving a sensible name, a speed of 6 knots, and a height of -80m (representing a depth of 80m). By default the top left and bottom right corners are specified using Short-Location elements (which take lat and long expressed as decimal degrees). We will be specifying them using Relative-Location elements, which express the location as ranges from the origin. Start to replace the existing elements by selecting the first shortLocation, then double-click on relativeLocation from the Insert tab of the elements toolbar. Now you can delete the existing shortLocation item. And repeat this process for the other 3 corner references (no, it isn't as long-winded in real-life as when written down). Set the top-left coordinates to 60nm North, 0nm East, and bottom-right to 0nm North, 120Nm West. And that's the SSK evasive behaviour complete.

Next comes the SSK aggressive behaviour. Now, my definition of this behaviour is very similar to the evasive behaviour. So start off by making a duplicate of the previous Evasive Waterfall component, renaming it to Aggressive. Within this component both of the SSK Recharge and the Rectangle Wander behaviours remain unchanged, but we will exchange the Evade behaviour for an Intercept one. Delete the Evade behaviour first, then insert an Intercept behaviour from the Insert tab of the Elements toolbox. Configure the Intercept behaviour to allow speed changes and specify that the SSK will attempt to intercept Blue submarines¹. And that is the SSK complete.

Next comes the SSN. Start off by giving it a category of Blue, Subsurface, Submarine. Then provide it with a sensor-fit comprising Active Broadband Sensor (source level:148, aperture:130, working: false²). Also provide a standard Broadband Sensor (aperture:130), and Narrowband sensor (steady time 10 minutes). For the SSN initial status give it a full fuel load³, and indicate that it is heading North at 10 knots. Make it's initial location 1/2 Nm North & East of the origin, at a depth of 60m (using a relativeLocation).

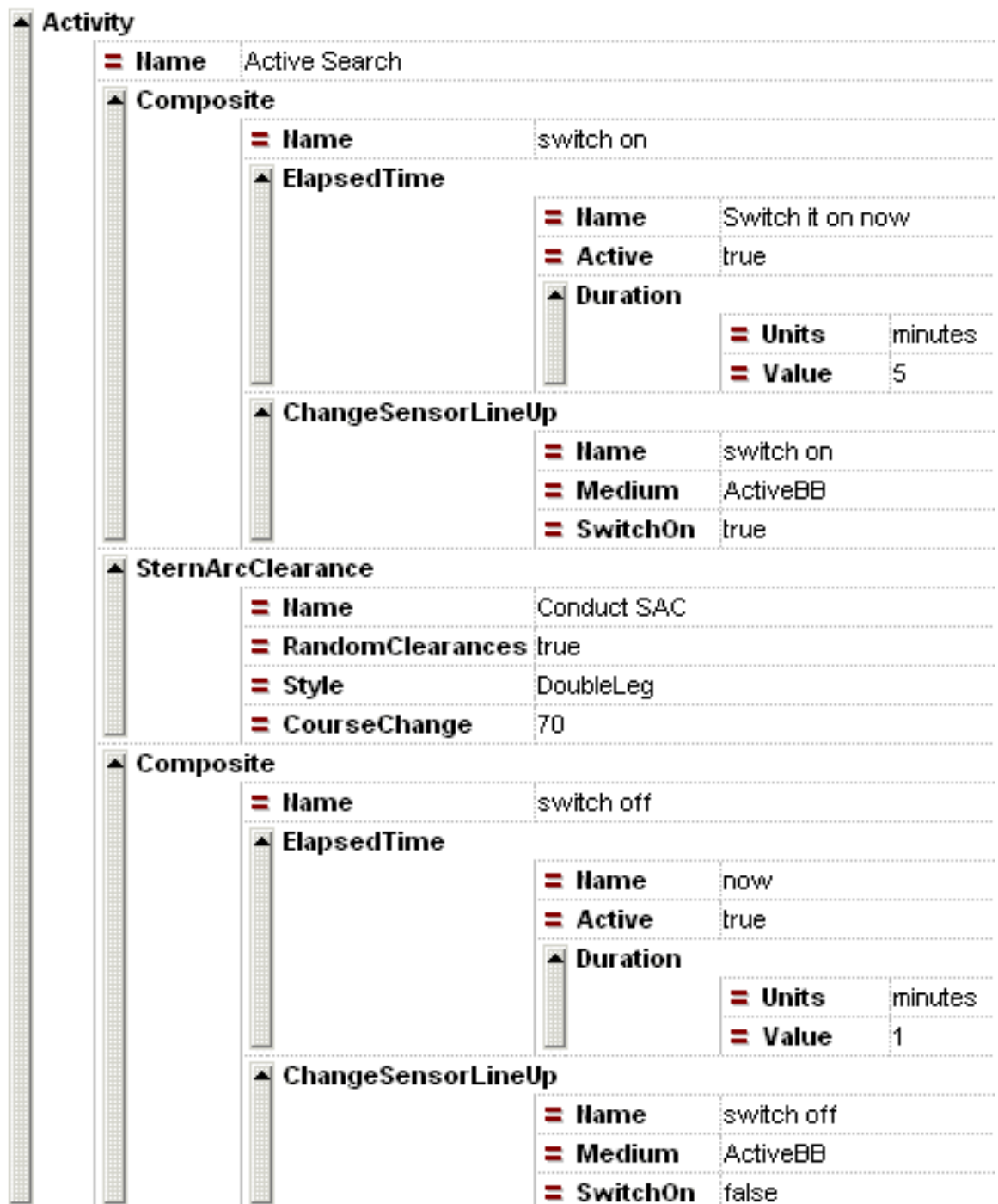
And now on to the SSN behaviour. Insert a Waterfall element beneath the existing status element, naming it something like SSN Search. The first item in the ssn search is a Working Transit, which allows a participant to conduct a transit whilst making regular stops to perform a particular activity. So, insert the Working Transit behaviour into the SSN's parent Waterfall behaviour, configuring it with a name ("sweeping active transit"), number of stops to make (8), indicate that the SSN should not repeat the transit on completion (Looping:false), and to transit at 12 knots. Define the path to travel through as passing through the following sets of coordinates expressed as relative locations in North, East nautical miles: (0,0) (30,10) (50,25) (55,40). Into the Working Transit insert the following behaviour (activity) performed at 8 points during the transit:

¹Sharp-eyed techno-geek readers will have noticed that we could have represented these two behaviours complete. We could have created a single SSK behaviour containing the two recharge and single rectangle wander elements, with the Switch between the two recharges. This Switch could then have contained the two alternate Evade and Intercept behaviours. Whilst this structure would represent our two behaviours in a more compact form, it could ultimately hinder the amount of changes which could be made between the two - having two completely separate behaviour waterfalls provides the greatest room for changes.

²we are setting the working attribute to false to indicate that the sonar isn't transmitting at the start of the scenario - other elements of the SSN behaviour set working to true to indicate that the sensor is transmitting on active.

³yes, fuel level is irrelevant for an SSN in ASSET, it is provided for completeness

Figure 8.2. Submarine active transmissions



After the sweeping curve of active transmissions (as shown in the initial diagram) the submarine moves into its passive search area, so insert a Wander behaviour representing the passive search. Configure the Wander behaviour with a range of 20 nautical miles, centre the behaviour on the relative location 15nm North, 40Nm East, at a depth of 100m and speed of 12 knots. Next for the SSN comes its definition of radiated noise - set the broadband base noise level to be 70 decibels. Finally configure the SSN movement parameters, as shown below:

```
<SSMovementCharacteristics FuelUsageRate="0" Name="SSN Movement">
  <MinSpeed Units="kts" Value="1"/>
  <MaxSpeed Units="kts" Value="28"/>
  <AccelerationRate Units="kt/s" Value="1"/>
  <DecelerationRate Units="kt/s" Value="1"/>
```

```
<MinHeight Units="m" Value="-300"/>
<MaxHeight Units="m" Value="-18"/>
<DefaultClimbRate Units="ft/s" Value="1"/>
<DefaultDiveRate Units="ft/s" Value="1"/>
<TurningCircle Units="m" Value="600"/>
</SSMovementCharacteristics>
```

. And that's the end of the SSN definition, now onto the MPS definition.

Start off by inserting a Fixed Wing participant to the scenario, naming it MPA - or similar. Then configure the MPA as Airborne, Blue, MPA. Provide the MPA with a radar configured as below:

Figure 8.3. MPA Radar parameters

The screenshot shows the 'SensorFit' window with a 'RadarLookupSensor' sub-window. The parameters are as follows:

RadarLookupSensor		
Name	Searchwater	
VDR	1.05	
MRF	2	
CRF	0	
IRF	0	
K	12	
TBDO		
Units	seconds	
Value	6	
CTP		
Units	minutes	
Value	0	
ITP		
Units	minutes	
Value	0	

Then go on to set the initial status of the MPA; full fuel tanks, located at a position 30nm North, 90nm east, travelling on 060 degrees at 90 knots. The MPA behaviour model is quite simple for this scenario, performing a rectangle wander within the corners 60Nm North, 60Nm East, 0Nm North, 120Nm East, travelling at 110 knots at an altitude of 2000 metres. Lastly enter the Fixed Wing Movement Characteristics :

```
<FixedWingMovementCharacteristics FuelUsageRate="0.00001" Name="MPA Flight" Def
  <MinSpeed Units="kts" Value="70"/>
  <MaxSpeed Units="kts" Value="260"/>
  <AccelerationRate Units="kt/s" Value="5"/>
  <DecelerationRate Units="kt/s" Value="7"/>
  <MinHeight Units="m" Value="1000"/>
  <MaxHeight Units="m" Value="6000"/>
  <DefaultClimbRate Units="ft/s" Value="3"/>
  <DefaultDiveRate Units="ft/s" Value="10"/>
  <DefaultClimbSpeed Units="kts" Value="90"/>
  <DefaultDiveSpeed Units="kts" Value="120"/>
</FixedWingMovementCharacteristics>
```

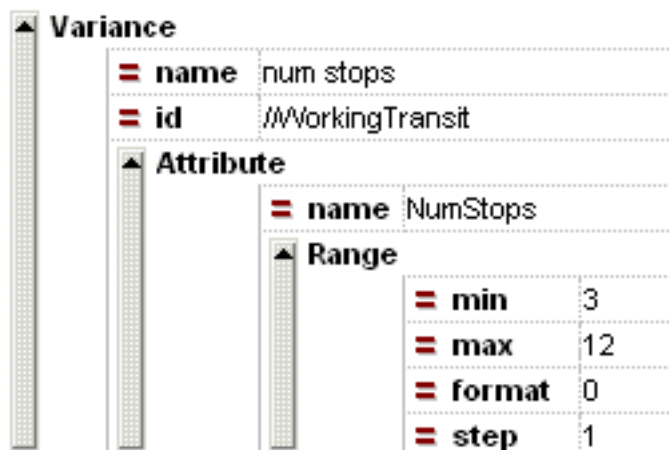

And that completes the MPA description, after which we move on to the control file in the next section.

3. Control file

The control file for this problem will comprise scenario generation instructions (to allow us to run through a batch of simulations), participant generation instructions (to allow us to simultaneously model a number of ssk targets running in parallel planes), and observer definitions used to record analysis outputs and to terminate the scenario.

Start off by creating the empty control file, as we did earlier. Set the output directory to somewhere tidy like `c:\asset\sm_output`. Insert your scenario generator, inserting a `MultiScenarioGenerator` into it. Within the multi-scenario generator set the name template to something useful like `sm_search`, which will become the prefix for the automatically generated scenarios. Also set the number of scenarios to be generated - start off with 10 for now. Also into the Multi Scenario Generator element we have to specify what we want ASSET to vary. For this tactical problem we will experiment with how frequently the searching SSN conducts active transmissions during the sweeping curve, by varying the number of stops on the Working Transit. Do this by inserting a single `Variance` element, itself containing an `Attribute` element into Multi-scenario generator variances list. The list may have sample elements already, just ditch them if there are. Now configure this variance to indicate that it is the `WorkingTransit` element that we are going to modify, the `NumStops` is the name of the attribute to modify within it, and that we want a range of values between 3 and 12 provided (to represent the number of times the submarine will stop).

Figure 8.4. Variance in submarine search



Next we move onto the participant generation element. Start off by inserting the `MultiParticipantGenerator` into the scenario generator. Configure this by indicating that we want 100 targets to be generated, and that the participant name begin with TGT. Note that we don't have to set the `inParallelPlanes` attribute here, since it is already defined (`MonteCarloTarget = true`) for the SSK in the scenario file. We do, however, have to specify the variances to apply to the multiple SSKs. This is a two-stage process, first setting the general attributes to vary, and then specifying where ASSET should place the SSKs. The first variance to insert will vary the speed at which the SSK patrols - so the name is `SSK_Speed`, the id is `/RectangleWander/Speed`⁴, and we are varying the `Value` attribute between 3 and

⁴Which means:

/ At the end of the participant identifier (which gets prefixed to the id before use) there is a trailing slash. Adding this second trailing slash means *descend as far as you like through the description of this participant*

RectangleWander Stop when you get to a RectangleWander behaviour

/Speed Find the Speed element within the RectangleWander behaviour

8 in 0.5 knot steps. Next we are going to vary the initial course. So, add a new variance called `SSK_Course`, and give it the simple id of `Status`. Indicate that we are going to vary the value of the Attribute name `Course` within the vessel status, providing values between 0 and 360 at 1 degree steps.

The next item to vary is more significant to our particular analysis, varying the posture of the SSK. As you no doubt remember we stored the two mutually exclusive SSK behaviours in a Switch construct. The Switch construct contained the index of which behaviour to activate. We are going to use a variance to randomly select one of these behaviours. So, insert a new Variance item titled `SSK Posture` with an XPath id of `Switch`. Next indicate that we are going to vary an Attribute through a `Choice` of values: 1 and 2.

Lastly we will define the area through which we want our series of SSKs to be placed. This is through use of a `ParticipantLocation` construct. Into this we define a `WorldArea` comprising top-left and bottom-right corners. The top-left (expressed in relative distances: `relativeLocation`) is at 60Nm North, 0Nm East and the bottom-right is at 0Nm North and 120Nm East.

After the scenario generation instructions we move on to the `ObserverList`, which indicates which referees and observers we want to monitor the running scenario. The observers we will use will record tracks to file for us, and stop the scenario after a set period. The first track recorder will be the `Track Plot Observer`, used to give us a graphic plot of the tracks on completion. Configure the track plot observer to be `Active`, to show positions, give it a sensible filename, and ask for a grid to be placed over the image on completion (10Nm would seem about right for our scenario size). Also specify that `only_final_positions` should be provided (to avoid receiving a dense mass of target tracks). Next comes an observer used to create Debrief files covering the period: `Debrief Replay Observer`. Indicate that you want this to record decisions and positions, but not detections - just to keep the volume of information down.

To prevent the scenario from running on ad infinitum add a `Time Observer`, configured to stop the scenario after 36 hours. And that completes the control file.

4. Running through the scenario

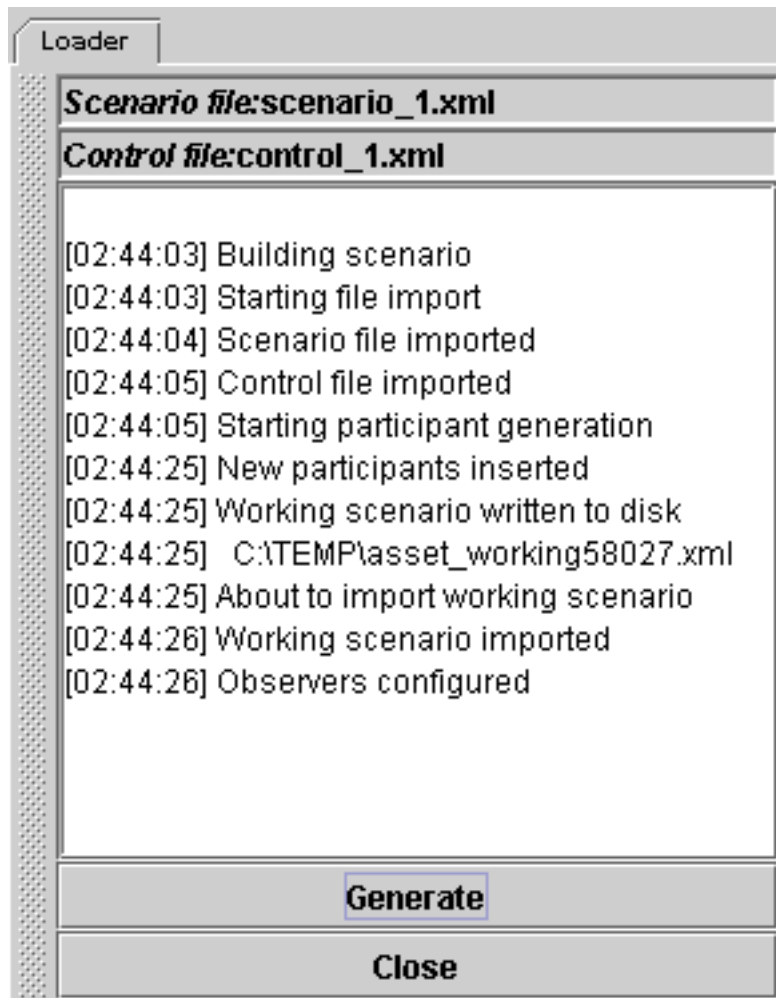
4.1. Quick-look in Workbench

To get some quick feedback on how we've designed the scenario we will first run it through the ASSET Workbench⁵. So, start off by double-clicking on the `workbench.exe` application in your ASSET installation directory.



Once the Workbench has opened, initiate the Monte Carlo Loader form by clicking on the button from the File sub-toolbar of the Controls toolbar. The Monte Carlo loader will open. Into the Scenario file: box drag your submarine search scenario file, and into the Control file: box drag your control file. Next press the Generate button and watch the scenario generation progress reported before your very eyes (as below).

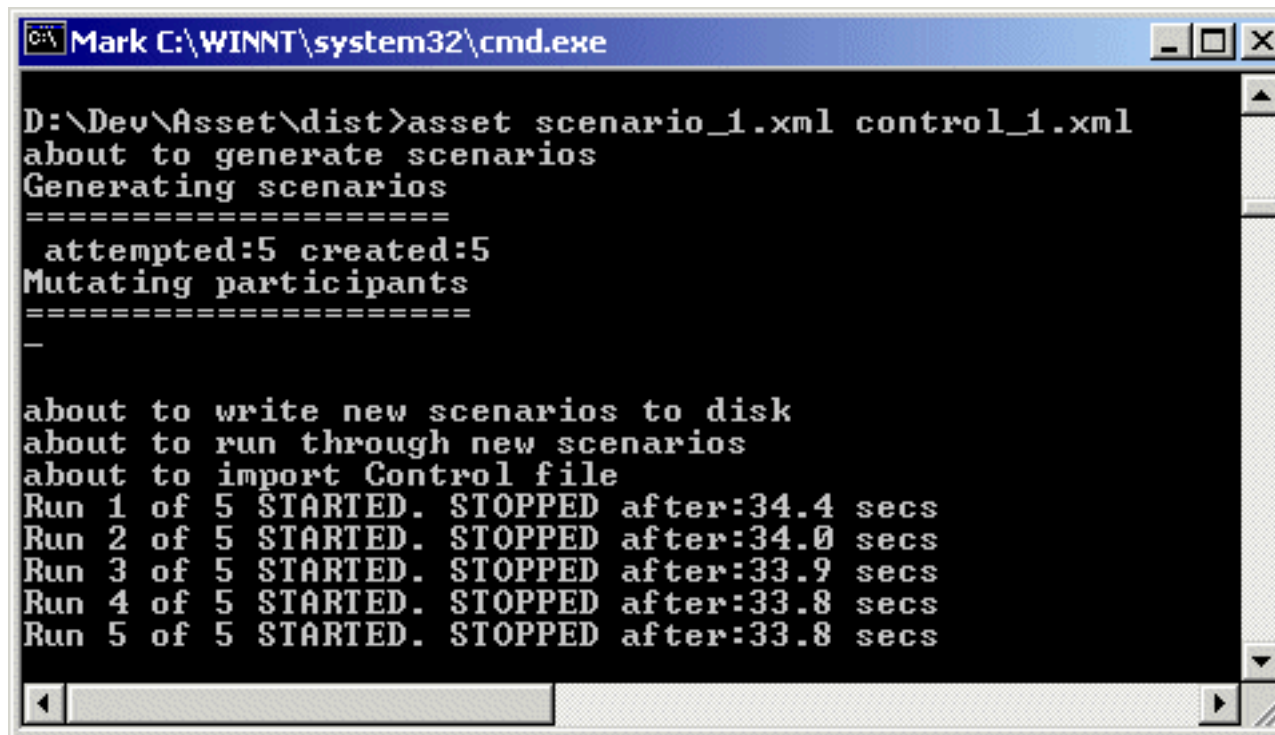
⁵This quick test will verify that our scenario and control files are valid and complete, and also highlight any glaring errors before we switch to batch processing.

Figure 8.5. Scenario generation from Workbench

On the ASSET plot you will also see the blue SSN and MPA, together with the 100 SSK targets. Open up the status panel for one of the SSKs near the SSN so that we can monitor what it's doing. Step forward a couple of times to allow the status to update, then examine what the SSK is currently doing (the line of text at the top of the panel). Due to the names we supplied for the behaviours this should contain a clue about whether this is an evasive or aggressive target. Now run the scenario forward on automatic and when the SSN starts it's all-sector active transmissions (when you see it change direction frequently) the SSK you are looking at should either turn to intercept or turn away. Good, that's the scenario checked. Now for the batch mode processing.

4.2. Batch processing

To perform the batch-mode processing I would copy the scenario and control files into the ASSET installation directory. Then open a command line console, and type `ASSET scenario.xml control.xml`. ASSET should now start processing the scenarios in batch mode, giving you the following results on the console.

Figure 8.6. Monitoring batch processing

```
Mark C:\WINNT\system32\cmd.exe

D:\Dev\Asset\dist>asset scenario_1.xml control_1.xml
about to generate scenarios
Generating scenarios
=====
attempted:5 created:5
Mutating participants
=====
-

about to write new scenarios to disk
about to run through new scenarios
about to import Control file
Run 1 of 5 STARTED. STOPPED after:34.4 secs
Run 2 of 5 STARTED. STOPPED after:34.0 secs
Run 3 of 5 STARTED. STOPPED after:33.9 secs
Run 4 of 5 STARTED. STOPPED after:33.8 secs
Run 5 of 5 STARTED. STOPPED after:33.8 secs
```

Once the processing is complete you can load any of the .rep files into Debrief. Unfortunately the analysis problem under consideration here doesn't lend itself to statistical analysis. One of the best overviews is to quickly flick through some of the track plot images to recognise clusters of ssks (only possible if you set `only_show_final_positions`, else you just have a mush of overlapping tracks).

Chapter 9. Monitor

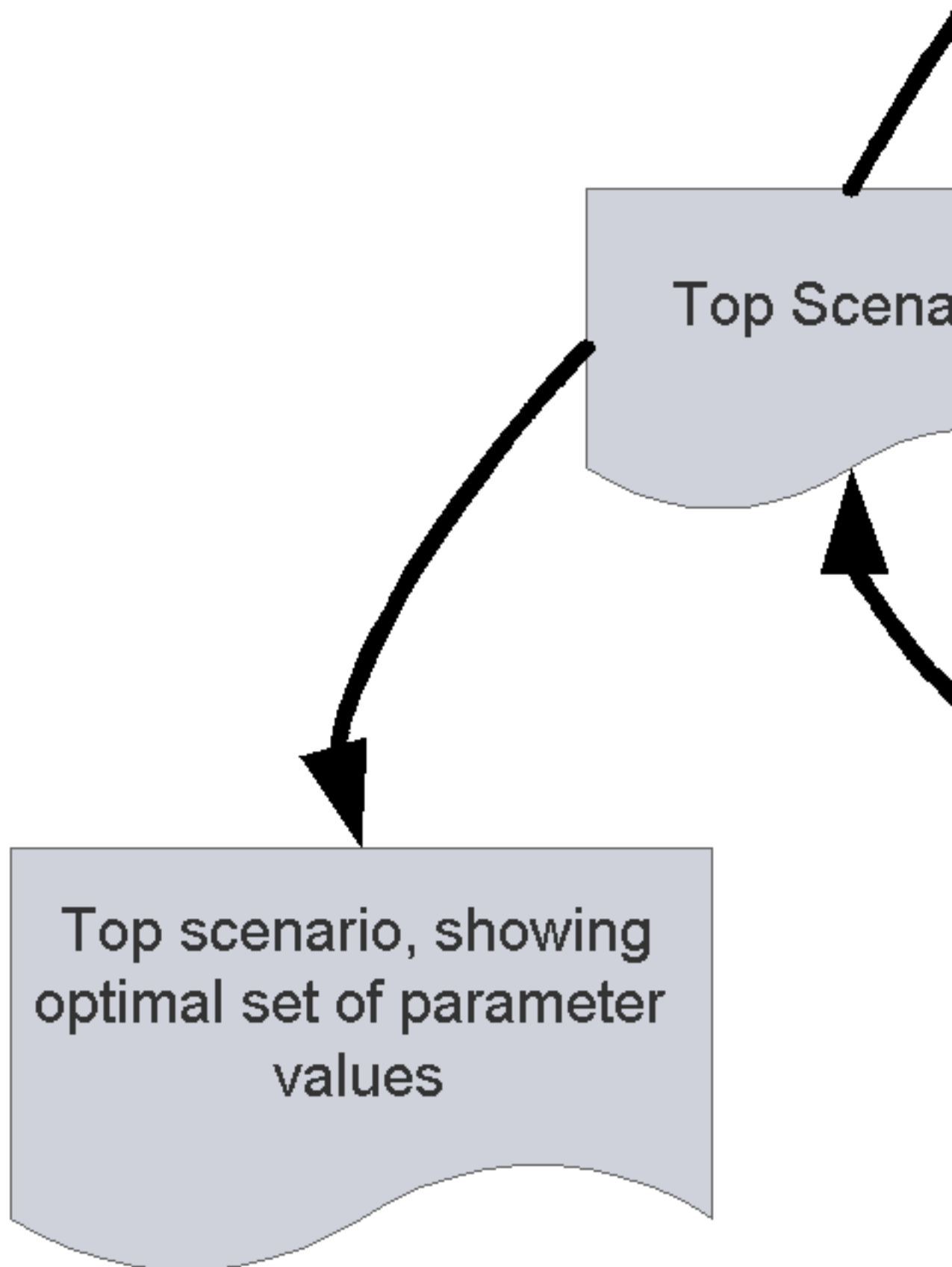
1. Introduction

Chapter 10. Playground

Chapter 11. Factory

The ASSET Tactics Factory is an application used to create initial best-guess tactics for a defined tactical scenario. Whilst there is no expectation that the output of the ASSET Tactics Factory will provide a tactic suitable for direct use at sea, the application is capable of identifying the mathematically optimal¹ tactic for a particular problem. This optimal tactic can then be used as a starting point by the human tactician.

¹Actually, whilst the genetic algorithm employed in the factory can deliver the mathematically optimal solution, it is better at identifying a near-optimal solution in a fraction of the time necessary for an exhaustive search.



As illustrated in Figure 11.1, “Flow of information within ASSET Factory” [89], three sets of data must be loaded prior to running:

Base Scenario	This is the scenario used as a base template for the problem to be analysed
Observers	The list of observers provide two facilities: the provide the score necessary for quantitative analysis of a particular scenario (such as time to weapon release, % area searched), together with scenario control facilities (such as stop after 4 hours running, or stop on weapon release).
Parameters to vary	This is a list describing what to vary within the base scenario in each permutation. Each item identifies what parameter in the base scenario is to be varied, together with instructions in how to produce the new value (such as random integers between 0 and 360).

Chapter 12. Noise Model

The ASSET Noise Model is a subset of the Workbench designed to meet the following requirement:xxxxxxxxxx

1. Requirement

2. Tutorial

lead through, step by step

2.1. Designing a scenario

XML editor, etc

2.2. Opening the ASSET Noise Model

2.3. Loading data

2.4. Viewing noise data

2.5. Interactively changing the scenario

3. Reference

links to parts of this document which define the modelling used, notably: propagation, radiated noise, sensor modelling

ASSET System Documentation

Ian Mayo

ASSET System Documentation

Ian Mayo

Copyright © 2001,2002

The ASSET User Guide

The ASSET System Documentation is an essential tool to the ASSET maintainer or developer, providing an architectural overview in addition to guidance in how to make specific modifications. The ASSET maintainer or developer should be familiar with the application, and know their way around the Modelling Guide and the User Guide.

Table of Contents

1. Architecture	1
1. Model Architecture	1
2. Application Architecture	1
2. Monte Carlo generation	7
1. Introduction	7
2. Multiple participant generation	7
3. Multiple scenario generation	9
3. Typical Tasks	12
1. Modelling Related	12
2. Application Related	12
4. Code Conventions	13
1. Version Control	13

Chapter 1. Architecture

This chapter gives an overview of the architecture of ASSET, both logically (in abstract terms) and physically (naming Java classes).

1. Model Architecture

Interfaces are used extensively within the application to provide modularity. All sensors implement a sensor interface, behaviours implement a behaviour interface, and so on, thus allowing objects to be simply dropped into a parent.

2. Application Architecture

ASSET draws many GUI components from MWC's Debrief application. Debrief is a 2 and 3 dimensional application used for analysis of maritime vessel tracks. Code providing plotting of 2 and 3 data onto a tactical chart, the general graphical user interface, toolbars, property editors and property windows, storing data to XML file format has all been taken from Debrief's Open Source code set.

The source code produced in support of ASSET following the following general guidelines:

Isolate GUI code

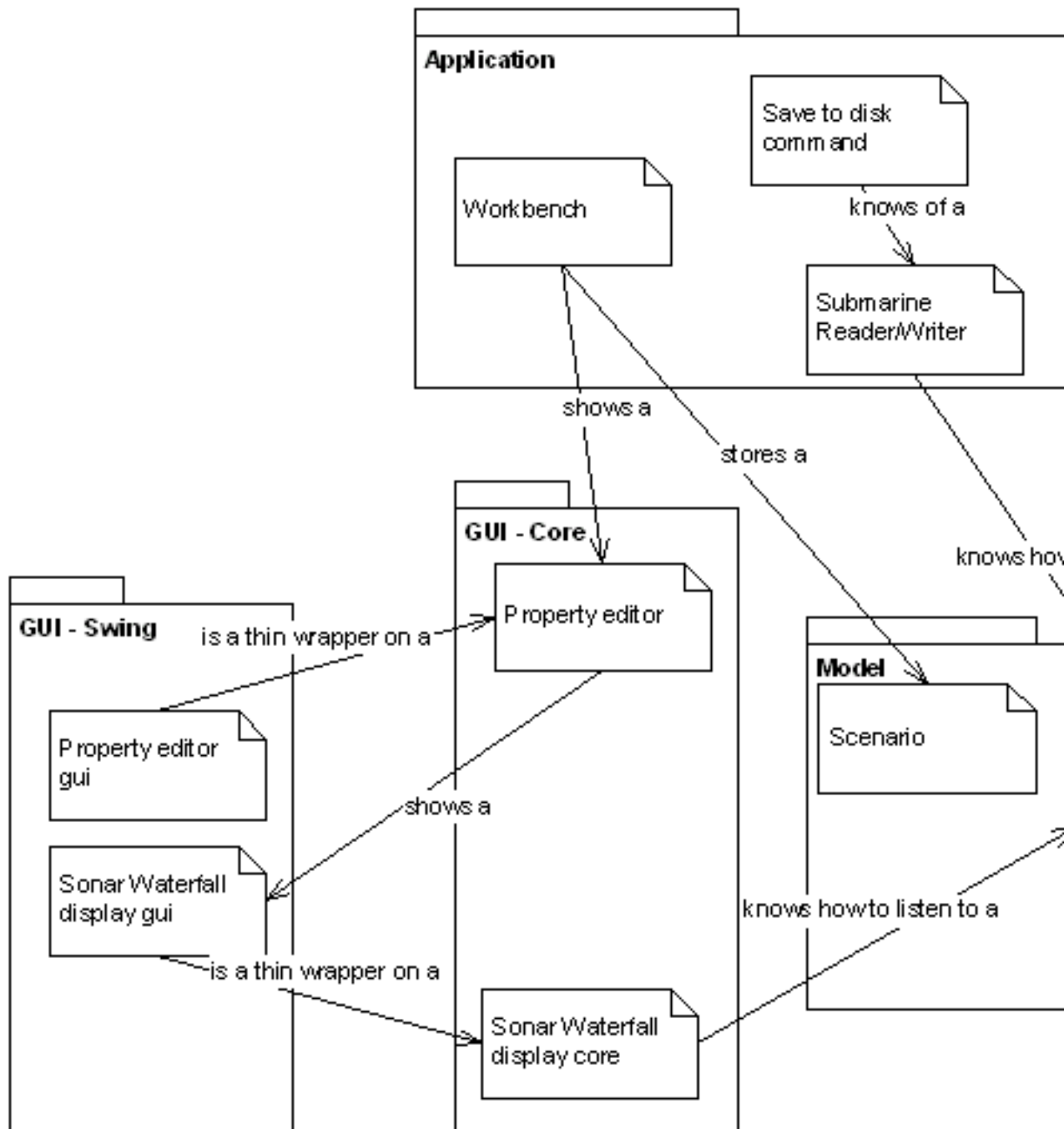
Where possible, GUI code is contained in classes which contain as little non-GUI code as possible - that is all moved into a parent or support class. This is to ease the task of automated testing, which becomes many times more difficult once a real GUI is involved

Isolate modelling code

Code modules which model real-world entities should not know about nor be concerned with application specifics such as GUI or storing to disk

The following diagram illustrates this encapsulation.

Figure 1.1. Example of code encapsulation within ASSET



2.1. Overview

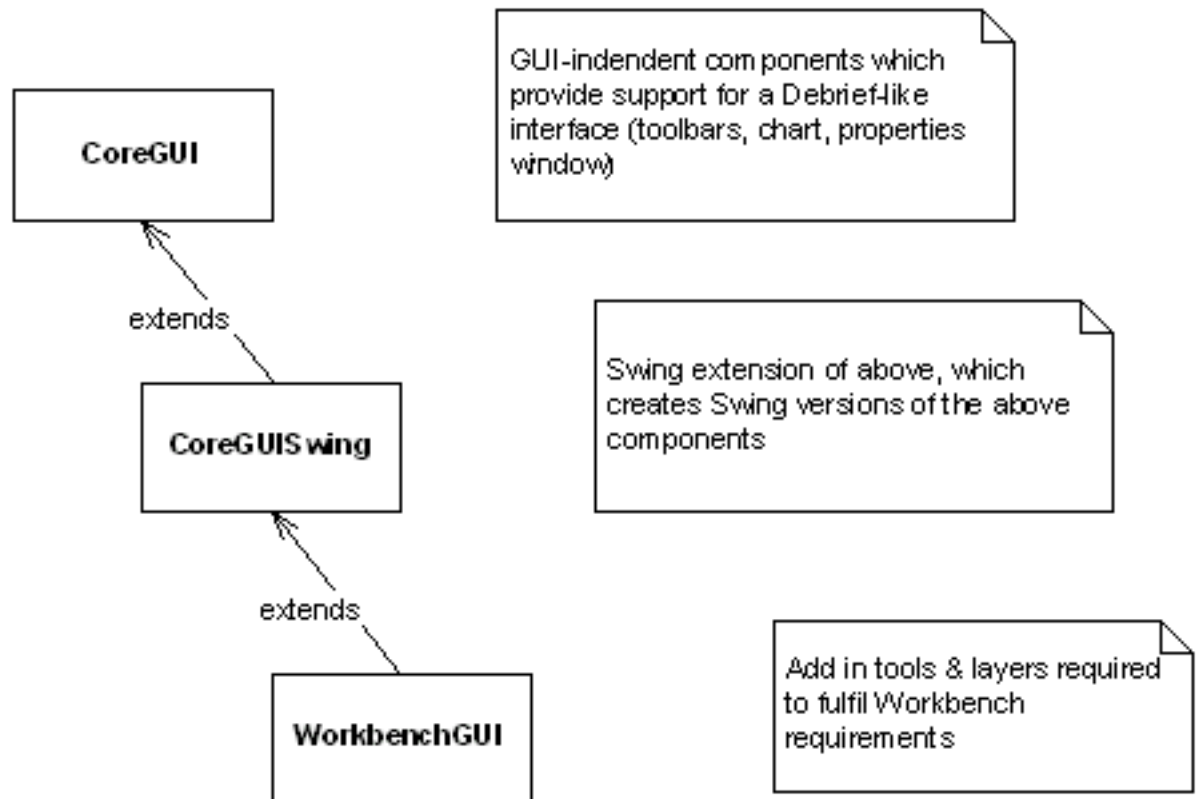
Inheritance is used extensively within the ASSET suite, reducing development times and maintenance costs. Common functionality is moved as far "up" the inheritance tree as possible, both at initial design and during refactoring in subsequent code changes. An example of this inheritance is the ASSET Workbench, which draws first on non-workbench related Java Swing components and then on non-Swing related components (see Figure 1.2, "GUI interface for ASSET Workbench" [3]).

2.2. Workbench

The Workbench application meets a specific set of ASSET requirements, providing a dynamic "gods-eye" view of a scenario, with the user able to edit/review attributes of any entity in the loaded

scenario. The application itself is a collection of ASSET (and Debrief) GUI components, some standard ASSET tools, and a Workbench-specific set of tools and layers.

Figure 1.2. GUI interface for ASSET Workbench



2.3. Server

The Server [<http://intranet2/coag/asset/api/ASSET/Server/CoreServer.html>] is a black-box software component capable of reading in scenario and control files, and moving through one or more scenarios to their conclusion. Essentially it provides the following functionality:

1. Load/save a scenario to/from disk
2. Create new (blank) scenario
3. Retrieve a list of currently loaded scenarios

2.4. Data objects

The real-world entities modelled within ASSET normally comprise the following components:

1. Code module the behaviour of the real-world entity
2. Code module detailing what is user-editable about this entity (speed, behaviour, category, name), and what editors to use
3. Code module supporting XML read/write of this entity, to make it persistent.

2.5. Permanent Storage (XML)

XML is used as the format for permanent storage of ASSET scenarios. The manner in which data is stored to XML format is specified using a number of Document Type Definition (DTD) files. The

DTD files are documented [dtd/index.html] within the ASSET documentation set, and can be read by a number of XML editor applications.

SAX is used to parse the XML files into ASSET, using the manager class at ASSETReaderWriter [http://intranet2/coag/asset/api/ASSET/Util/XML/ASSETReaderWriter.html].

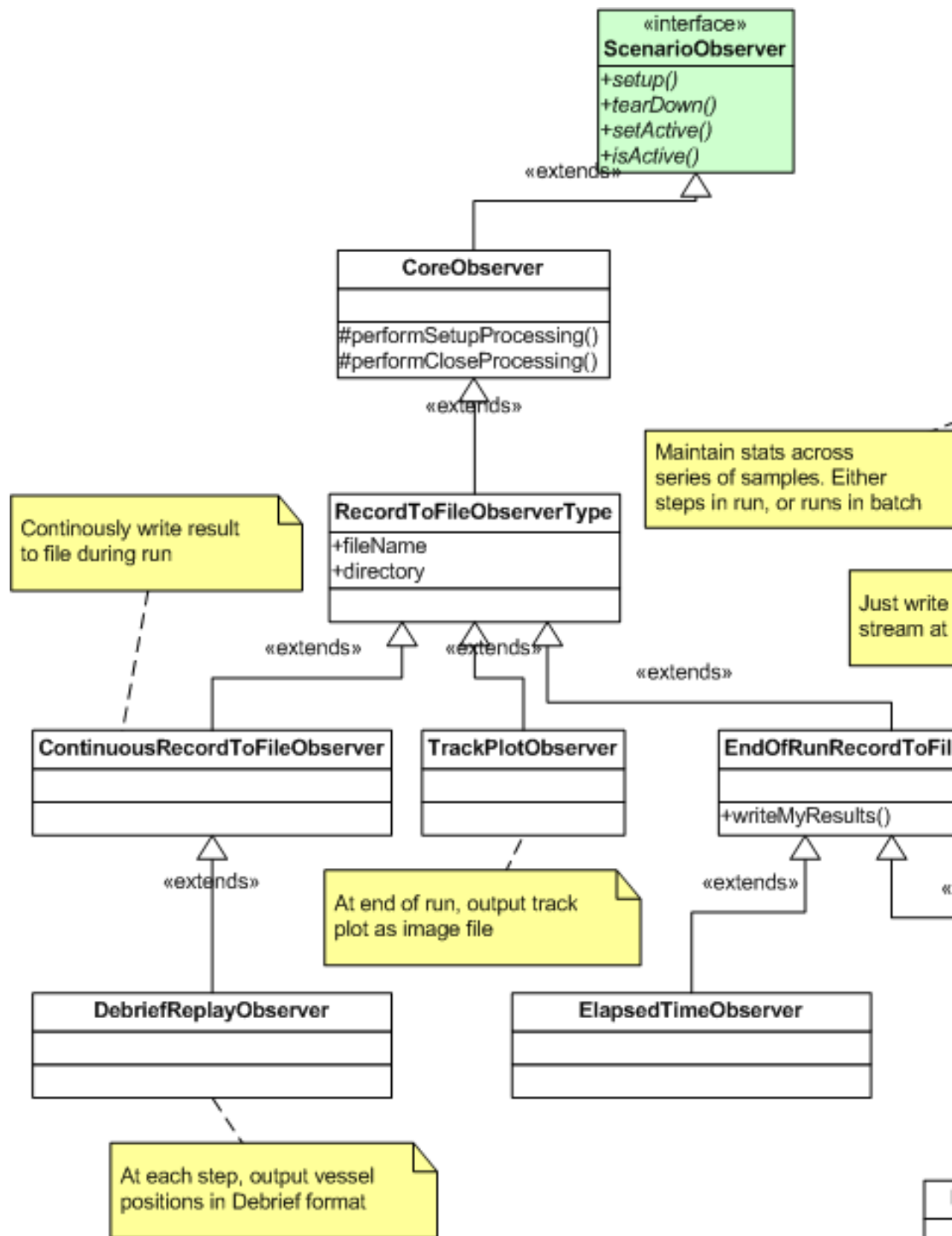
2.6. Communications

The Observer application within ASSET inherently relies on networked communications. Additionally, when the Playground application is running in Network mode it uses networked communications to interact with a remote scenario - either running in another Playground or WorkBench.

2.7. Recording results to file

Quite a complex inheritance tree is used to provide record-to-file functionality, in order to cater for Observers which require to write results to file at the end of a run, during a run, or at the end of batch of runs. The inheritance tree is shown below:

Figure 1.3. Inheritance tree for record-to-file observers



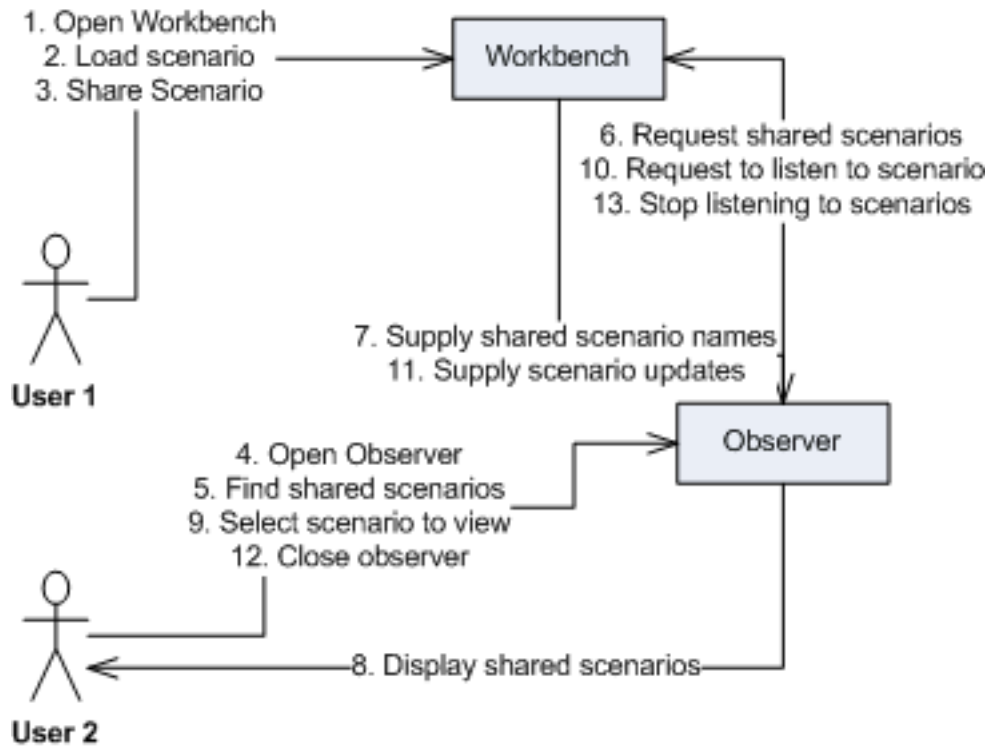
2.8. Network communications

Networked communications within ASSET occur in one of two modes, read-only communications where an observer watches an evolving scenario, or read-write where a remote user interacts with an evolving scenario.

2.8.1. Read-only communications

The control flow of initiating and terminating read-only communications is shown in the figure below.

Figure 1.4. Remote read-only communications



Chapter 2. Monte Carlo generation

1. Introduction

This section covers the component of ASSET used for generating new permutations of an existing scenario. Monte Carlo simulation techniques are used to investigate the probability of a particular outcome by evaluation of multiple permutations of one or more characteristics across multiple scenarios (multiple scenario generation.) together with multiple permutations of a scenario participants within one scenario (multiple participant generation).

One, or both of the generation mechanisms may be usefully employed in Monte Carlo analysis. See the relevant sections for the detail of how each generation technique is implemented, but when both are in use the following is performed:

1. Generate series of new scenarios permutations (in memory)
2. For each scenario permutation, generate multiple participants
3. Store series of scenarios to disk

2. Multiple participant generation

The use of multiple instances of a particular participant in a scenario effectively allows many scenario permutations to be performed at once - provided the required scenario changes are limited to one or more participants of one force. Running through these scenarios uses a special ASSET mode known as Parallel Planes Processing.

This generation mechanism is performed as follows:

1. Load scenario template from disk
2. Load variance file from disk
3. Create new (cloned) copy of scenario
4. Looping through each participant variance:
 - a. Remove the current instance of that participant from the scenario
 - b. Looping through the required number of permutations:
 - i. Create a (cloned) copy of the participant
 - ii. Apply the series of variances to it
 - iii. Insert the new participant back into the scenario

2.1. Target distributions

When creating a series of participants it is common to want to distribute them evenly through an area. This is performed using a `ParticipantLocation` construct within the variance file. The construct contains definitions of the top left and bottom right corners of the area. Into this area an even distribution of targets is produced as follows:

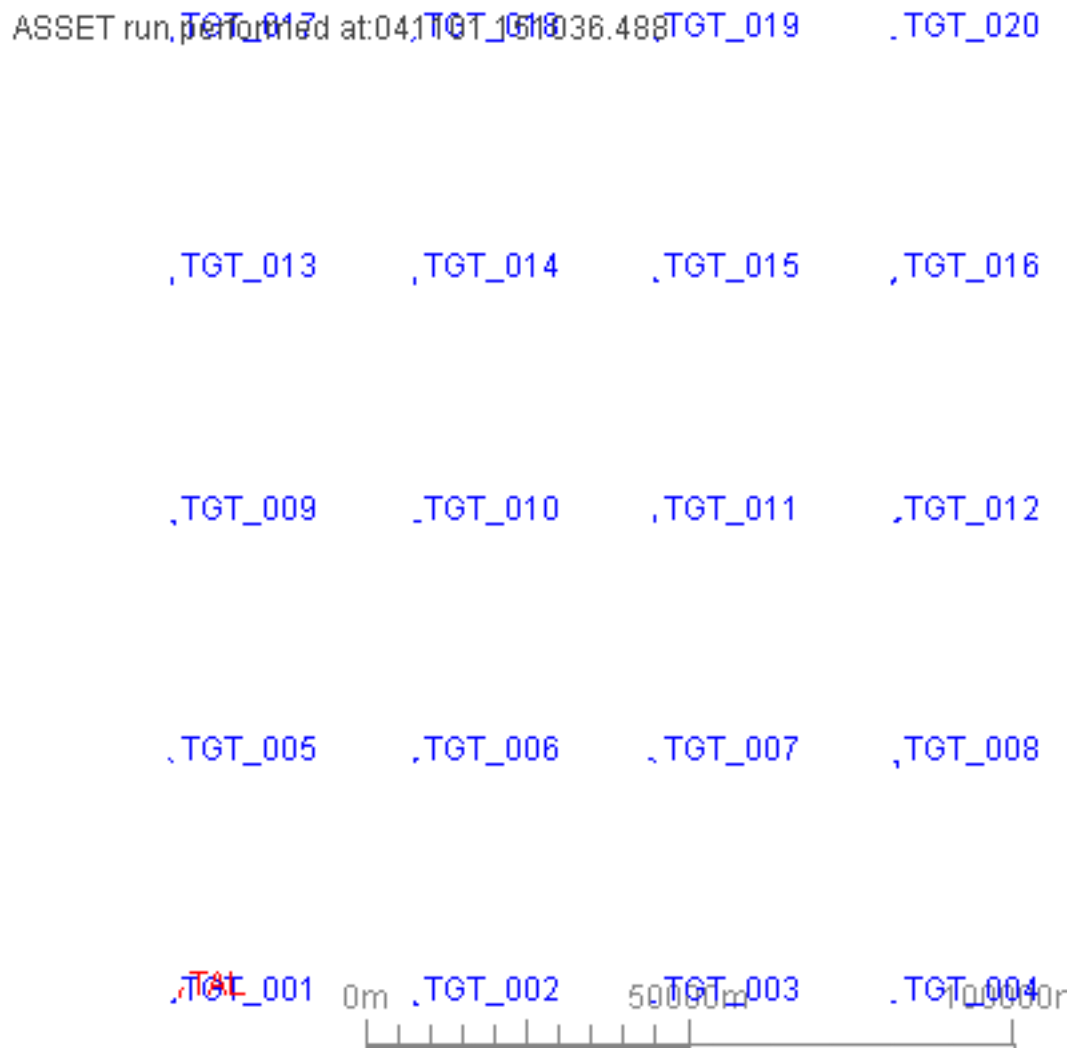
1. Calculate area of distribution (in m in this case)
2. Divide area by number of targets to give target separation
3. Produce point at top-left, then work across top of area producing a new point at each separation until reach right-hand of area
4. Step down to next row, and repeat

2.2. Parallel Planes Processing

Conventional use of ASSET to analyse the effects of varying a set of scenario characteristics requires many model runs. Some of these problems may, however alternately be expressed in such a way that ASSET is able to combine many unique scenario instances into a single run - typically where the participants of one force conduct their behaviour irrespective of the actions of the other force.

For example, we may have a problem where we wish to investigate varying the search speed of a searching participant, aiming to optimise the chances of a detecting a single target on a fixed route (commercial shipping route). We can generate a scenario containing many instances of the searching asset, each containing a different search speed, together with a single target. When running through this scenario we can analyse the comparative performance of the many searchers - all from a single model run. The screenshot below illustrates this. Assume that the red participant (bottom left) is the hostile vessel following a commercial shipping route, and that the blue participants represent a search participant starting the search from a range of start locations. We can run the scenario forward, monitoring for how many of the blue searchers detect the red vessel. The start points which do (or don't) result in a target detection can then be subsequently analysed in greater depth. Here parallel planes processing has given an overview result of many scenario permutations in a single run.

Figure 2.1. Parallel planes layout



As explained earlier, this model use does rely on one force acting as a "clockwork-mouse", acting completely irrespective of the other. Not many scenarios are able to be represented in a way which meets this constraint, but the parallel planes processing certainly offers a worthwhile shortcut to those that do.

3. Multiple scenario generation

Where the *multiple participant generation* trick isn't sufficient, we have to resort to creating complete new permutations of our scenario of interest. Each new scenario permutation is given a unique name (either through sequential numbering or by appending details of the specific variances applied).

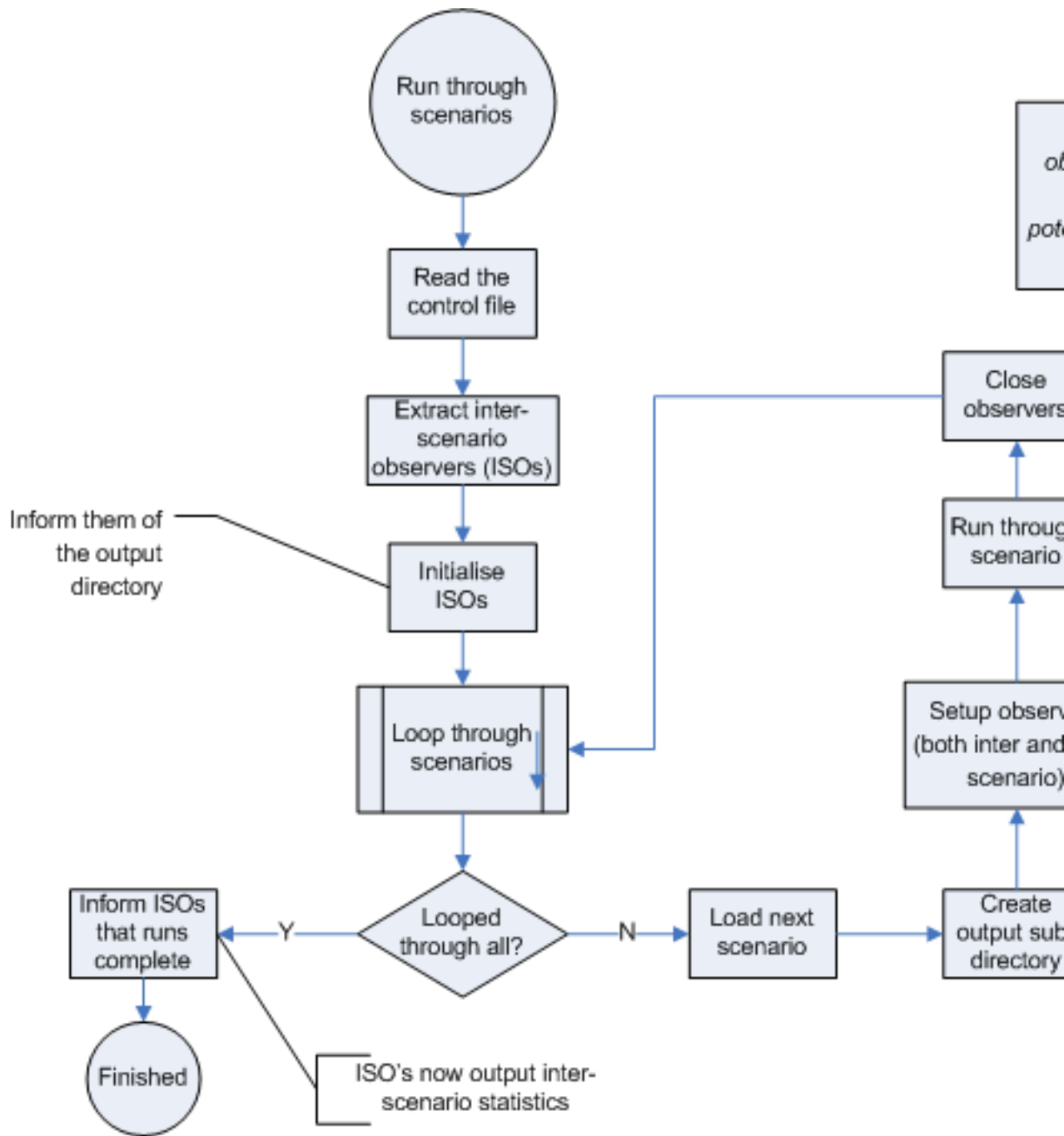
Normally the new scenario permutations are written to disk prior to ASSET working its way through them. Thus the new scenario files can be inspected in an XML viewer/editor or even loaded directly into the Workbench for detailed analysis.

The processes used to generate multiple scenarios is shown in the Monte Carlo Workflow diagram shown in the User Guide at Figure 1.2, "Monte Carlo simulation workflow" [4]

3.1. Inter-scenario observers

Normal scenario observers are created afresh for each scenario run, recording details of events during that scenario. Inter-scenario observers track results across a series of runs, however, and keep their output file open between runs. Their processing is as shown in the figures below:

Figure 2.2. High-level processing for inter-scenario observers



Note

The case code is a significant component within inter-scenario observer processing. This code is unique to each permutation of variances used. Thus an Inter-scenario observer may gather statistics either using the unique scenario name, or against the case-code for that scenario, outputting the collated statistics on batch completion.

An example of an Inter-scenario observer is the DetectionAchievedObserver, which monitors whether a particular target type is detected by a particular host type, maintaining a log which indicates *true* or *false* for all scenarios within the multi-scenario run.



Tip

Hey, if you think that processing inter/intra scenario observers is a handful, you should see the inheritance tree

Chapter 3. Typical Tasks

This section provides an overview of how to conduct typical development/maintenance tasks

1. Modelling Related

1.1. Adding a new vehicle

1.2. Adding a new sensor

1.3. Adding a new behaviour

2. Application Related

2.1. Adding a new graphic plot feature

2.2. Adding a new editor

2.3. Creating a new front-end

2.3.1. Re-using the standard front-end

2.3.2. Creating a fresh front-end

Chapter 4. Code Conventions

The ASSET software uses mostly standard coding conventions similar to most of those available on the Internet: focussing on self-documentation and ease of readability through liberal use of white-space.

1. Version Control

Version history information is included at the top of ASSET source files through use of a CVS template in the javadoc. This version history records the name of the last author and the date last checked-in.

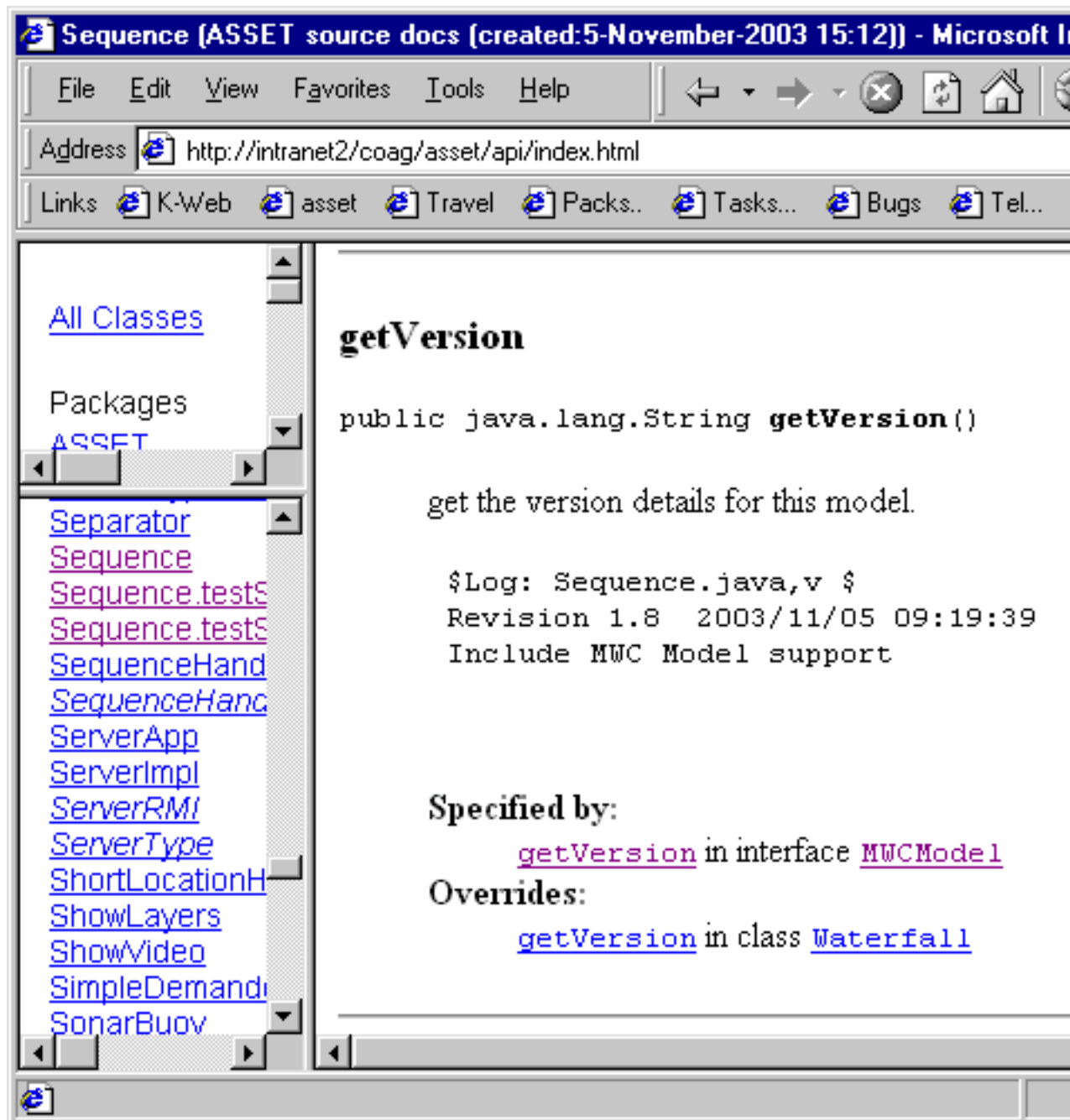
In addition to this, source code files which contain tactical models (such as those recorded in the modelling guide) implement the MWCModel interface - and it's getVersion() method. The body of this method contains a quoted CVS macro which will always store the date last checked-in, and the javadoc comment contains a complete history of check-in comments- providing an insight into how that module has changed.

Figure 4.1. Source code sample containing version details

```
/** get the version details for this model.  
 * <pre>  
 * $Log: Sequence.java,v $  
 * Revision 1.8  2003/11/05 09:19:39  Ian.Mayo  
 * Include MWC Model support  
 *  
 * </pre>  
 */  
public String getVersion()  
{  
    return "$Date: 2003/11/05 09:19:39 $";  
}
```

The modification date can be retrieved programmatically (to indicate the build dates of current software components), and the history can be retrieved using the javadoc compiler as shown below.

Figure 4.2. Javadoc snippet containing version details



Version History

Version History

Table of Contents

1. ASSET Documentation Set	1
----------------------------------	---

Chapter 1. ASSET Documentation Set

This chapter/book contains a change record of the ASSET documentation set. The content contains a number of control characters, since it is produced automatically from the Version Control System.

Last Modified. The document was last modified at: \$Date: 2004/09/07 15:21:00 \$

Current version. This documentation set is currently at version: \$Revision: 1.47 \$

Change notes.

\$Log: asset_docs.xml,v \$

Revision 1.70 2004/11/22 10:38:46 Ian.Mayo
Minor tidying

Revision 1.69 2004/11/18 16:17:24 Ian.Mayo
Remaining SM bits

Revision 1.68 2004/11/17 16:20:14 Ian.Mayo
Now adding the SM Search control file

Revision 1.67 2004/11/16 16:22:45 Ian.Mayo
Lots more bits

Revision 1.66 2004/11/12 16:23:49 Ian.Mayo
Minor new doc bits, including SSK Recharge

Revision 1.65 2004/11/09 10:17:47 Ian.Mayo
Minor extension to SM search definition

Revision 1.64 2004/11/05 15:45:40 Ian.Mayo
Minor extensions to lookup tables, reflect workbench now starting from

Revision 1.63 2004/11/01 16:38:52 Ian.Mayo
More sub-search tutorial

Revision 1.62 2004/10/18 19:47:34 ian
Add tiny bit more on NB

Revision 1.61 2004/10/18

15:11:17 Ian.Mayo Start narrowband sensor docs Revision 1.60 2004/10/15
13:12:35 Ian.Mayo Spell checks from Word Revision 1.58 2004/10/14 15:11:43
Ian.Mayo More vessel editors Revision 1.56 2004/10/13 09:36:37 Ian.Mayo More
Workbench GUI documentation Revision 1.55 2004/10/11 15:22:28 Ian.Mayo Docu
use of Workbench for multi-participant runs Revision 1.54 2004/10/07 14:20:
Ian.Mayo Include guidance in increasing memory Revision 1.53 2004/09/30
14:28:59 Ian.Mayo Remove manual revision list, finish Force Protection sect
of tutorial Revision 1.51 2004/09/27 10:10:58 Ian.Mayo Add sample of gaussi
dist Revision 1.50 2004/09/23 10:49:40 Ian.Mayo Use XML entities rather tha
actual characters Revision 1.49 2004/09/22 10:49:44 Ian.Mayo Replace long
hyphens with normal ones. Revision 1.48 2004/09/09 07:57:08 Ian.Mayo Insert
scenario generation guidance Revision 1.47 2004/09/07 15:21:00 Ian.Mayo Wri
up intercept, delete Avoid Revision 1.46 2004/08/31 07:30:59 Ian.Mayo Remov
stray character Revision 1.45 2004/08/31 07:29:37 Ian.Mayo Switch styleshee
definition - so styles automatically shown when opening in XMetal Revision
2004/08/18 16:39:50 Ian.Mayo Add frequency-list batch processing Revision 1
2004/08/18 15:45:36 Ian.Mayo Tidying some batch store algorithms, observer
which can record Final State Revision 1.39 2004/08/17 13:34:59 Ian.Mayo Upd
past scenarios to match current schema, extend user guide to complete looku

scenario, update screenshots Revision 1.38 2004/08/12 09:23:24 Ian.Mayo Min
extensions to File Observer explanation Revision 1.37 2004/08/11 15:05:42
Ian.Mayo Better inter-scenario recording Revision 1.36 2004/08/11 10:38:21
Ian.Mayo Add guidance on inter-scenario observers Revision 1.35 2004/08/09
13:24:07 Ian.Mayo Add Ladder Search Revision 1.34 2004/08/05 15:23:48 Ian.M
Record investigate behaviour Revision 1.33 2004/05/21 15:24:19 Ian.Mayo Mor
new bits in tutorial/lookup sensors Revision 1.31 2004/02/19 16:40:14 Ian.M
Change location for downloading ASSET Revision 1.30 2004/02/18 09:35:31
Ian.Mayo More tutorial stuff Revision 1.28 2004/02/16 14:02:36 Ian.Mayo Min
change, also first save after XMetal edit Revision 1.27 2004/02/16 13:58:17
Ian.Mayo Add new sensor model Revision 1.26 2003/11/11 11:55:25 Ian.Mayo
Include guidance on setting parallel planes, model audit Revision 1.25
2003/11/07 14:53:35 Ian.Mayo Include model audit Revision 1.24 2003/11/05
13:19:30 Ian.Mayo Correct tutorial mistakes Revision 1.22 2003/09/30 14:24:
Ian.Mayo Remove DTD, since DocBook parser didn't recognise file reference
Revision 1.21 2003/09/30 10:10:57 Ian.Mayo During switch to XMLSpy editing
Revision 1.20 2003/09/29 10:22:22 Ian.Mayo Introduce Monte Carlo generation
Revision 1.19 2003/09/17 14:21:44 Ian.Mayo describe demo of climb/dive Revi
1.18 2003/09/15 09:07:49 Ian.Mayo mod to model authoring process Revision 1
2003/09/09 16:25:19 Ian.Mayo Document observers, put in more on expanding
search Revision 1.15 2003/09/04 15:56:20 Ian.Mayo Lots of putting models in
formal paras, started documenting observers Revision 1.14 2003/09/03 14:26:
Ian.Mayo Added states Revision 1.13 2003/08/28 15:34:22 Ian.Mayo improve
description of time cycle Revision 1.12 2003/08/26 15:09:10 Ian.Mayo More
Merlin, model definition process, rename Chain to Waterfall Revision 1.11
2003/08/14 15:33:56 Ian.Mayo extend modelling documentation Revision 1.9
2003/08/12 10:48:48 Ian.Mayo Third reformat the program listing