DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION **DSTO**

Constructing an Infrastructure to Facilitate Electronic Support Modelling in the Virtual Ship

Shane A. Canney

DSTO-TR-1159

Constructing an Infrastructure to Facilitate Electronic Support Modelling in the Virtual Ship

Shane A. Canney

Electronic Warfare Division
Electronics and Surveillance Research Laboratory

DSTO-TR-1159

ABSTRACT

This document provides a detailed description of the infrastructure that has been constructed to allow Electronic Support (ES) modelling for Virtual Ship applications, together with a discussion of the current Electronic Support Measures (ESM) model. The infrastructure and ESM model, collectively called an ESM federate, has been succesfully integrated into the first scenario of the Virtual Ship, namely the Sensor Data Fusion scenario. The implementation issues that arose throughout the integration process are also addressed to aid future developers.

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION DSTO

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567
© Commonwealth of Australia 2001
AR-011-889
May 2001

Constructing an Infrastructure to Facilitate Electronic Support Modelling in the Virtual Ship

Executive Summary

The Virtual Ship will provide an extensive facility for simulating the operation of warship systems. The Virtual Ship allows individual models to be linked under a common architecture in order to meet the objectives of the particular scenario under consideration. The underlying architecture for the Virtual Ship is the High Level Architecture (HLA). HLA provides a mechanism to connect simulations over a network, exchange information between the integrated models (federates) and coordinate the time advances of each model within the simulation.

The initial scenario chosen as a test-bed for developing and illustrating the concepts of the Virtual Ship was the Sensor Data Fusion scenario. This scenario involved fusing data at the sensor level in a maritime environment where a ship is under attack from an incoming anti-ship missile. The federates used in this scenario include an Electronic Support Measures (ESM) federate.

The ESM federate represents a passive sensor system within the Virtual Ship framework. The ESM federate consists of an interface for communicating with the Run-Time Infrastructure (RTI) as well as an ESM model that primarily provides a generic representation of the track data output from a typical ESM system. The ESM model is only of sufficient fidelity to meet the requirements of the Sensor Data Fusion scenario. The ESM federate has been designed so that all functions within the interface remain separate from the ESM model. The major advantage of this is that once more comprehensive ESM models become available they can readily be evaluated in a "plug and play" fashion without modifying the communication interface, provided the same input/output conditions are supported.

This report provides a detailed description of the infrastructure that has been constructed to allow Electronic Support (ES) modelling for Virtual Ship applications, together with a discussion of the current ESM model. Although the ESM federate was constructed to meet the specifications of the Sensor Data Fusion scenario, it is anticipated that this federate can be reused for other applications of the Virtual Ship. The lessons learned from implementing the ESM federate into the Virtual Ship will also be discussed to aid future developers with the integration process.

Author

Shane Canney
Electronic Warfare Division

Shane Canney completed his PhD in experimental physics at the Flinders University of South Australia in March 1997. The topic of his thesis was "Electron Momentum Spectroscopy of Solids". He joined Electronic Warfare Division (EWD) of DSTO in March 1999 as a Research Scientist in Maritime Systems Group in EWD. His work has focussed on the modelling and simulation of EW systems in support of the Virtual Ship project.

Contents

1. Introduction

The Virtual Ship is a project being led by Maritime Operations Division (MOD) that will provide an extensive facility integrating simulation models of ship systems, thereby creating a virtual representation of a warship. The Virtual Ship allows individual models to be linked under a common architecture in order to meet the objectives of a particular scenario under consideration. The underlying architecture for the Virtual Ship is the High Level Architecture (HLA). HLA provides a mechanism to connect simulations over a network, exchange information between the integrated models and coordinate the time advances of each model within the simulation.

Within the HLA each simulation model is known as a federate and a collection of federates operating together is known as a federation. A Federation Object Model (FOM) describes all the data that is exchanged between federates. The FOM utilises object classes and object interactions to group data on a logical basis. Each federate within a simulation is described by a Simulation Object Model (SOM), which is a subset of the FOM. The transportation of information across the network is provided by the Run Time Infrastructure (RTI). The RTI provides services that enable the management, data exchange and time coordination within a federation. This functional view of HLA is shown in Figure 1.

The initial scenario chosen as a test-bed for developing and illustrating the concepts of the Virtual Ship was the Sensor Data Fusion scenario. This scenario involved fusing data at the sensor level in a maritime environment where a ship is under attack from an incoming anti-ship missile. The federates involved were: Virtual Ship Execution Manager (VSEM), Motion, Radar, Virtual Ship Simulation Display (VSSD), Missile, Electronic Support Measures (ESM), Infra Red Search and Track (IRST), Fusion and Helm. All of the federates listed have been successfully integrated within the Virtual Ship.

The ESM federate represents a passive sensor system within the Virtual Ship framework. The ESM federate consists of an interface for communicating with the RTI as well as a simple ESM model that primarily provides a generic representation of the track data output from a typical ESM system. The ESM model is only of sufficient fidelity to meet the requirements of the Sensor Data Fusion scenario. The ESM federate has been designed so that all functions within the interface remain separate from the ESM model. The major advantage of this is that once more comprehensive ESM models become available they can readily be evaluated in a "plug and play" fashion without modifying the communication interface, provided the same input/output conditions are supported.

This report provides a detailed description of the infrastructure that has been constructed to allow Electronic Support (ES) modelling for Virtual Ship applications, together with a discussion of the current ESM model. Although the ESM federate was constructed to meet the specifications of the Sensor Data Fusion scenario, it is anticipated that this federate can be reused for other applications of the Virtual Ship.

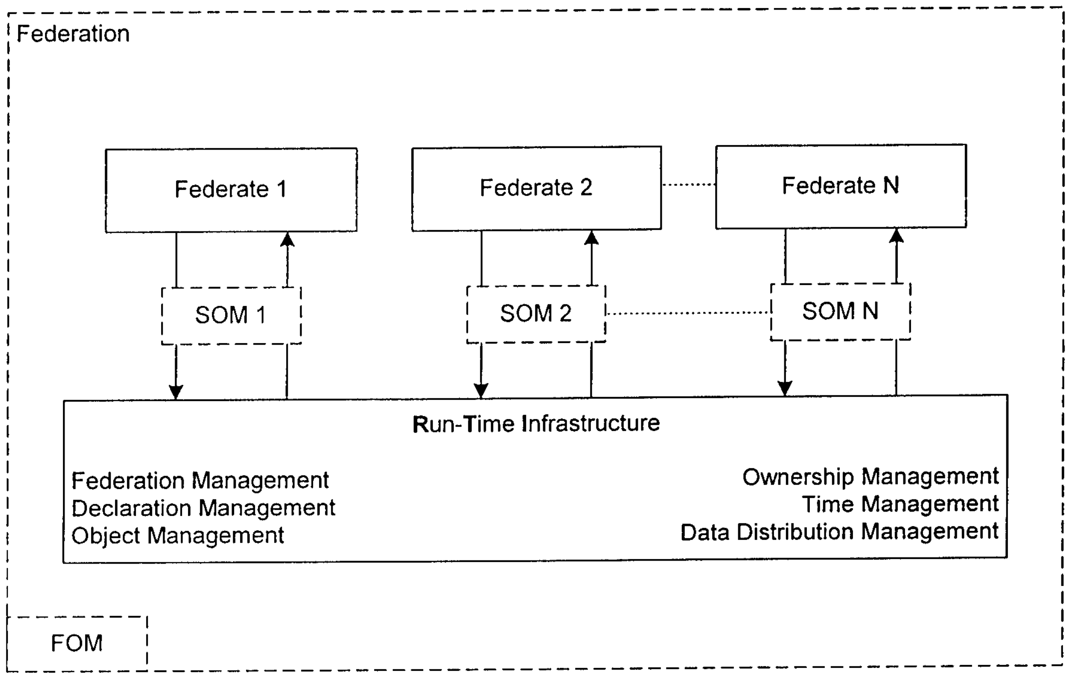


Figure 1: Functional view of High Level Architecture (HLA)

The lessons learned from implementing the ESM federate into the Virtual Ship through the Data Fusion Scenario will also be discussed to aid future developers with the integration process.

2. Virtual Ship Architecture

The Virtual Ship Architecture (VSA) provides the framework for integrating models in a standardised way to meet the objectives of a particular scenario [1]. HLA forms the basis of this architecture, as this is well suited for providing interoperability and reuse capabilities.

2.1 HLA

Using HLA as the underlying architecture for the Virtual Ship provides many advantages over other distributed technologies. The HLA supports distributed simulation and promotes interoperability and reuse of simulation models. As HLA is now the standard for distributed simulation in the US, a large amount of effort is being spent developing the architecture and supporting tools. Also any models developed in the US, and potentially other countries, will be HLA compliant and thus can be readily integrated into the Virtual Ship environment.

Each simulation model is known as a federate within the HLA. The collection of federates operating together is known as a federation. There are three elements that constitute the High Level Architecture. These elements are the interface specification, the Object Model Template (OMT) and the rules. The interface specification defines the interface functions between the RTI and the federates. The OMT provides a common framework for documenting the information exchanged in the federation. The HLA rules outline the steps that must be followed to achieve meaningful interaction of simulation models and to support reuse of the models over different federation applications.

The key components that constitute a federation are the federates and the RTI. The RTI resembles a distributed operating system that provides common services during run-time of an HLA federation. A description of all shared information (objects, attributes, interactions and parameters) essential to a particular federation is represented in a Federation Object Model (FOM). The intrinsic capabilities that an individual federate offers to the federation is documented in a Simulation Object Model (SOM). Both the FOM and SOM must be documented in accordance with the Object Model Template.

2.2 VS-FOM

The Virtual Ship Federation Object Model (VS-FOM) defines all information that can be exchanged amongst federates participating in a Virtual Ship federation. The object class and interaction tables of the VS-FOM V1.1 are reproduced in Tables 1 and 2

Table 1: Object table for VS-FOM.

Class 1	Class 2	Class 3
CompositeEntity (N)	Air (PS)	
	SeaSurface (PS)	
	SubSurface (PS)	
ComponentEntity (N)	CommandAndControlSystem (PS)	
	CountermeasureSystem (PS)	
	NavigationSystem (PS)	
	SensorSystem (N)	ESSystem (PS)
		IRSystem (PS)
		IFFSystem (PS)
		RadarSystem (PS)
		SonarSystem (PS)
	WeaponSystem (PS)	
SensorTask (N)	ESTask (PS)	
	IRTask (PS)	
	IFFTask (PS)	
	RadarTask (PS)	
	SonarTask (PS)	
Track (N)	AbsoluteTrack (N)	AbsoluteSystemTrack (PS)
		AbsoluteFusedTrack (PS)
	RelativeTrack (N)	RelativeSystemTrack (PS)
		RelativeFusedTrack (PS)
		RelativeESTrack (PS)
		RelativeIRTrack (PS)
		RelativeIFFTrack (PS)
		RelativeRadarTrack (PS)
		RelativeSonarTrack (PS)

Table 2: Interaction table for VS-FOM.

Interaction 1	Interaction 2
Execution Management (N)	CreateCompositeEntity (IR)
	CreateComponentEntity (IR)
	SetScenarioDescription (IR)
	SetRandomNumberSeed (IR)
	ExecutionManagementError (IR)
SystemControl (N)	SetPropulsionSystemAttribute (IR)
	AckSetPropulsionSystemAttribute (IR)

respectively. The VS-FOM will continue to evolve to support future Virtual Ship applications.

Each object in the object class table is designated by its publication and subscription capabilities (enclosed in parentheses after the name). There are three designating letters for the object class table. These are described in the High-Level Architecture Object Model Template Specification Version 1.3 [2] as follows:

(P) - Publishable: Indicates that the object can be published by a federate.
(S) - Subscribable: Indicates that a federate is capable of utilising the information that is provided.
(N) - Neither publishable or subscribable: The object class is neither publishable or subscribable by a federate.

It should be noted that an object should only be designated as publishable if subscription to that object is supported as it is meaningless for an object to be published if it is not subscribed to within a federation. Thus the allowed set of publication and subscription capability designations is {S, PS, N}.

Similarly, in the interaction table, designations are given that outline a particular federates capability with respect to given classes of information. The designations as given in the High-Level Architecture Object Model Template Specification Version 1.3 [2] are as follows:

(I) - Initiates: Indicates that a federate is capable of initiating and sending interactions of that type.
(S) - Senses: Indicates that a federate is currently capable of subscribing to and utilising the interaction information.
(R) - Reacts: Indicates that a federate is capable of reacting to the interaction.
(N) - Neither initiates, senses or reacts: Indicates that a federate is not capable of initiating, sending or reacting to this interaction class.

2.3 Execution Management

Execution management is utilised in the Virtual Ship to enable federation executions to occur in a controlled manner and to be repeated if required. The Virtual Ship Execution Manager (VSEM) controls the flow of a federation execution [3]. The VSEM reads in a script file defining the configuration of the federation to be executed. Included in this file is a listing of the participating federates; a listing of the entities to be created and their initial attribute values; the end time for the federation; the number of loops for the federation execution; a random number seed and a scenario descriptor.

The VSEM makes use of synchronisation points to control the flow of execution. It is the responsibility of the managed federates to respond to these synchronization points. Deriving from a supplied class, namely VSEMFederate, satisfies this responsibility [4]. The general flow of an execution-managed federation is shown in Figure 2.

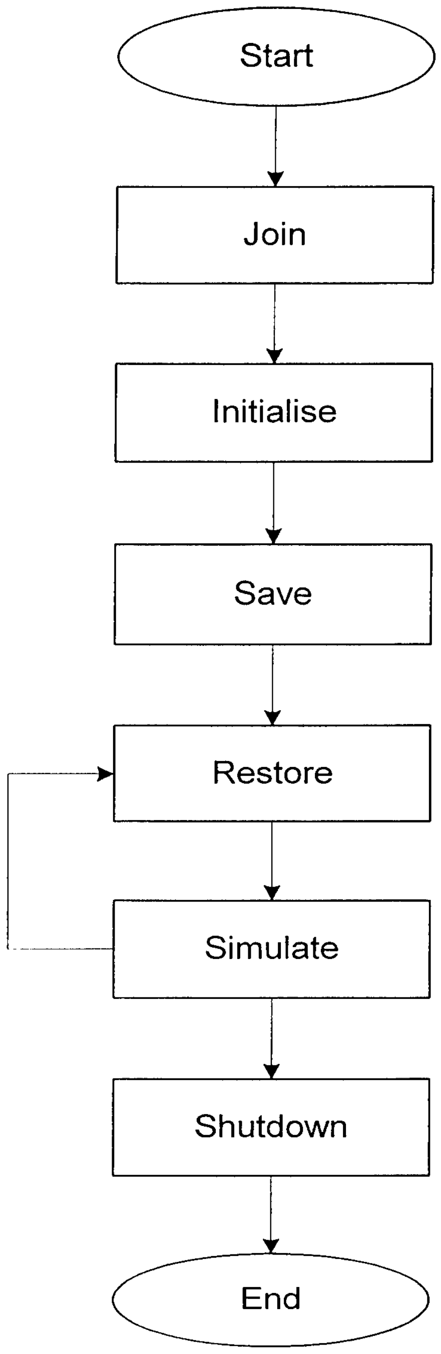


Figure 2: Flow of an execution managed federation execution.

2.4 Coordinate System

The reference coordinate system used in the Virtual Ship is WGS 84 [5]. Reference to this global coordinate system is denoted by G in the Virtual Ship architecture. In the VS-FOM there are objects that have an additional reference coordinate system. CompositeEntity objects have an associated coordinate system denoted by E and ComponentEntity objects have a coordinate system denoted by e. SensorTask objects have an associated coordinate system denoted by S. All three of these coordinate systems are illustrated in Figure 3 where $\boldsymbol{O}_E$ is the origin of the composite entity object, $\boldsymbol{O}_e$ is the origin of the component entity object and $\boldsymbol{O}_S$ is the origin of the sensor task object.

The notation $\boldsymbol{O}_A^B$ is used to define the position vector of these coordinate systems, and thereby the locations of the entities to which they are attached. The subscript A denotes the coordinate system of which this point is the origin and the superscript B denotes the coordinate system with respect to which the coordinates of this point are given. The velocity and acceleration are simply the first and second derivatives of the position. The orientation of the object is specified through the use of Euler angles that define the rotation transformation from the reference coordinate system B to the coordinate system A, $\Omega^{B \to A}$.

3. ESM Federate

The ESM federate represents an electronic support measures sensor system within the Virtual Ship. The primary role of the ESM federate is to detect the radio frequency (RF) emissions of radar systems and provide tracks of that radar to the federation. The ESM federate has been constructed so that only one ESM system can be represented for each executable. If multiple ESM systems are required in a scenario then multiple executables are required, with each having a unique federate name. The data requirements of the ESM federate are documented in a SOM.

3.1 SOM

The ESM federate SOM is a subset of the VS-FOM and details the information that the ESM can provide to the federation (publish) as well as the information that the ESM requires from other federates within a federation (subscribe). The ESM represents an instance of ComponentEntity.SensorSystem.ESMSystem. The ESM federate has been constructed so that the ESM component entity can be attached to either an air or a sea-surface composite entity. If future scenarios require the modelling of an ESM system on a sub-surface entity then the ESM federate will need to be modified to account for this.

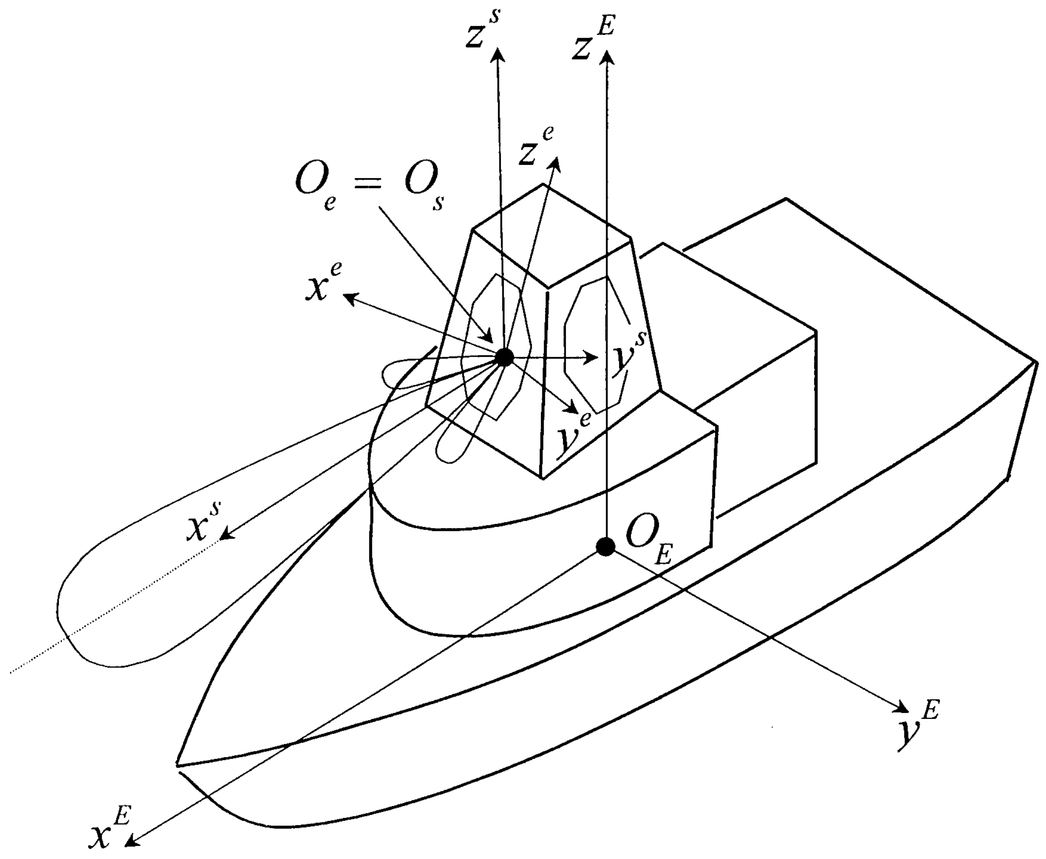


Figure 3: CompositeEntity (E), ComponentEntity (e) and SensorTask (S) coordinate systems within the Virtual Ship Architecture.

The role of the ESM is to detect the signals emanating from all radar systems located on all composite entities other than the ESM parent entity. To enable detection the ESM also needs to acquire knowledge of the tasking of each radar sensor. After detecting a radar system the ESM will commence tracking.

The object table for the ESM SOM outlining the publication and subscription requirements for the ESM federate is reproduced in Table 3. Table 4 shows the ESM SOM interaction table outlining the execution management interactions.

3.2 Response to VSEMFederate

The ESM federate has been designed to participate in execution managed federations and is thus derived from the VSEMFederate. The ESM federate passes control to the VSEMFederate immediately after an ESM federate is constructed within a federation execution. All execution managed federates must respond to callbacks from the VSEMFederate and its associated base classes. The response from the ESM federate to these callbacks is outlined below.

3.2.1 createComponentEntity

Upon receiving the createComponentEntity callback an instance of an ESM object is created. One of the parameters of createComponentEntity is the name of a configuration file containing the parameters for the ESM system being modelled. The ESM parameter values specified in this file are assigned when the ESM is created. Following creation the ESM object is registered with the RTI (registerObjectInstance) so that any other participating federates that have subscribed to the ESMSystem will discover the ESM object. The initial attributes for the ESM sensor system are also provided to the federation using the updateAttributeValues RTI call.

3.2.2 createCompositeEntity

As the ESM federate is only responsible for creating an instance of a component entity this callback has no functionality. The createCompositeEntity function is however a pure virtual function in a derived class and consequently it must be overridden, but remains empty in ESM federate.

3.2.3 simulate

The simulate loop for the ESM federate is shown in Figure 4. The simulate loop continues to iterate the ESM federate in time (by the lookahead value) until the execution manager indicates that the federation has finished. The state of the local ESM model is updated for each iteration. The ESM model is described in detail in Section 4.

Table 3: Object table for the ESM SOM.

Class 1	Class 2	Class 3
CompositeEntity (N)	Air (S)	
	SeaSurface (S)	
ComponentEntity (N)	SensorSystem (N)	ESSystem (P)
		RadarSystem (S)
SensorTask (N)	RadarTask (S)	
	ESTask (P)	
Track (N)	RelativeTrack (N)	RelativeESTrack (P)

Table 4: Interaction table for the ESM SOM.

Interaction 1	Interaction 2
Execution Management (N)	CreateCompositeEntity (S)
	CreateComponentEntity (R)
	SetScenarioDescription (R)
	SetRandomNumberSeed (R)
	ExecutionManagementError (I)

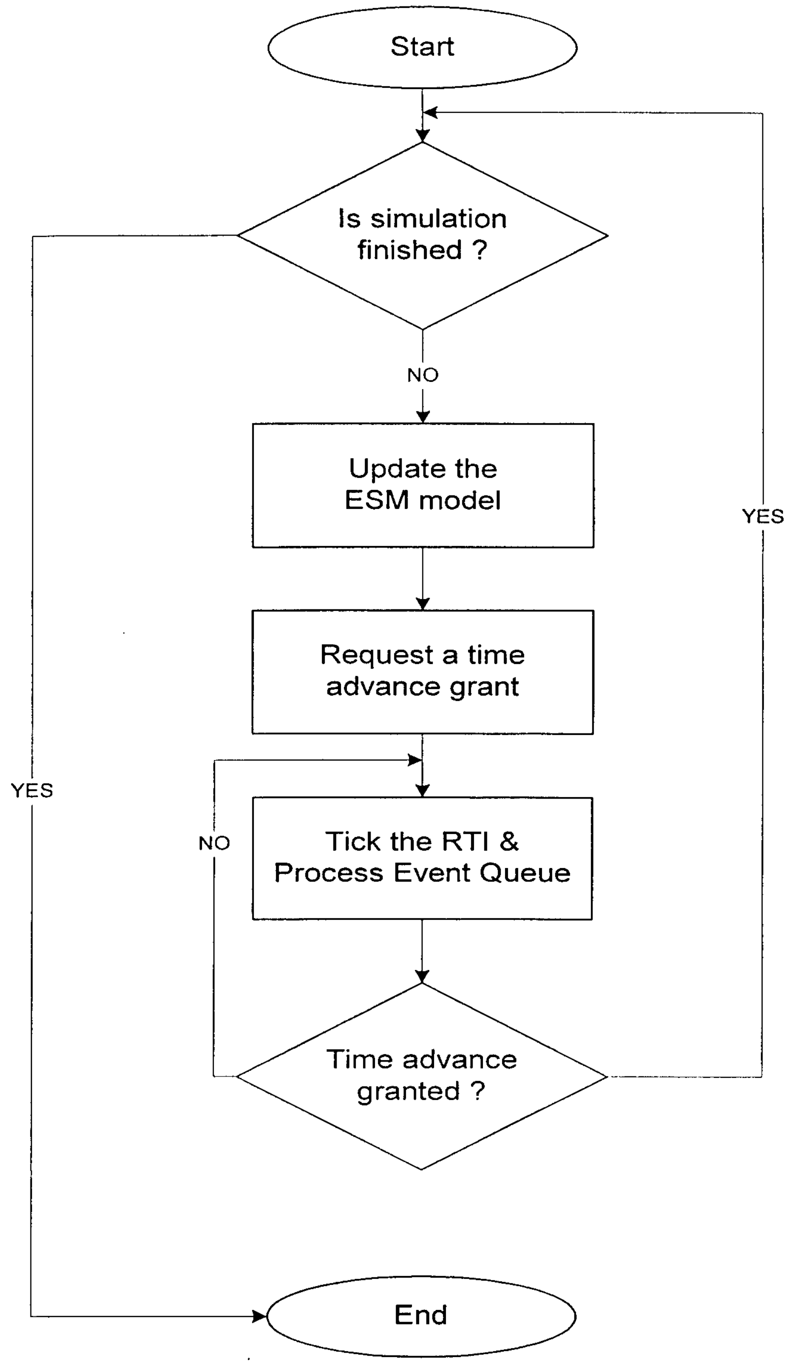


Figure 4: Flow diagram for ESM federate simulate loop.

3.2.4 enableTimeManagement

The ESM federate enables both the Time-Regulation and Time-Constrained time management switches. Enabling time regulation allows the ESM federate to send time-stamped-order (TSO) messages and enabling time constrained allows TSO messages to be received by the ESM federate. A lookahead value of 0.1 seconds is currently used, ensuring that no TSO events will be generated by the ESM federate with time stamp less than the current federate time + 0.1 seconds.

3.2.5 processEvent

This function is used to process all Federate Ambassador callbacks (from the RTI) relevant to the ESM federate. These callbacks are discussed in Section 3.4.

3.3 RTI Ambassador Calls

The HLA interface specification describes the methods by which federates interact with the RTI. The RTI provides six service sets to enable communication and these are: Federation Management, Declaration Management, Object Management, Ownership Management, Time Management and Data Distribution Management. The services that the ESM federate currently utilises to make calls to the RTI (via RTI Ambassador) are outlined in Table 5. If the service is not directly called from the ESMFederate class, but from one of the derived base classes [4] then this is highlighted in the table.

3.4 Response to Federate Ambassador Callbacks

Some of the management service function calls that a federate makes to the RTI will invoke a callback from the RTI (via Federate Ambassador) to the relevant federates. The relevant federates then respond in the appropriate manner. Detailed below is the response of the ESM federate to the callbacks for each of these services.

3.4.1 Federation Management

The VSEMFederate base class handles all federation management functions and thus no response is required by the ESM federate.

3.4.2 Declaration Management

StartRegistrationForObjectClass
Any object classes that are published by the ESM federate and have not been registered are subsequently registered with the RTI. If the object does not yet exist it is registered as soon as it is created. Following registration an immediate receive order (RO) updateAttributeValues for the registered object is sent and for those objects with kinematic attributes, a TSO updateAttributeValues is scheduled for the next iteration of the simulate loop.

Table 5: ESM federate RTI Ambassador calls.

Federation Management
Create Federation Execution (GeneralFederate base class) Destroy Federation Execution (GeneralFederate base class) Join Federation Execution (GeneralFederate base class) Resign Federation Execution (GeneralFederate base class) Synchronization Point Achieved (VSEMFederate base class) Federate Save Begun (VSEMFederate base class) Federate Save Complete (VSEMFederate base class) Federate Restore Complete (VSEMFederate base class)
Declaration Management
Publish Object Class (GeneralFederate base class) Unpublish Object Class (GeneralFederate base class) Publish Interaction Class (GeneralFederate base class) Unpublish Interaction Class (GeneralFederate base class) Subscribe Object Class Attributes (GeneralFederate base class) Unsubscribe Object Class Attributes (GeneralFederate base class) Subscribe Interaction Class (GeneralFederate base class) Unsubscribe Interaction Class (GeneralFederate base class)
Object Management
Register Object Instance Update Attribute Values Delete Object Instance Local Delete Object Instance Request Attribute Value Update
Ownership Management
No services invoked
Time Management
Enable Time Regulation (GeneralFederate base class) Enable Time Constrained (GeneralFederate base class) Time Advance Request Flush Queue Request (VSEMFederate base class) Enable Asynchronous Delivery (VSEMfederate base class) Query LBTS (GeneralFederate base class)
Data Distribution Management
No services invoked

3.4.3 Object Management

ReceiveInteraction
The execution manager is responsible for sending the interactions to the RTI. The VSEMFederate handles the receiving of the SetScenarioDescription and SetRandomNumberSeed interactions. The CreateCompositeEntity and CreateComponentEntity interactions are filtered by the VSEMFederate and the relevant information is passed onto the ESM federate through the functions createCompositeEntity and createComponentEntity.

DiscoverObjectInstance
Any object that is registered with the RTI and that the ESM federate subscribes to will result in this callback. All DiscoverObjectInstance callbacks result in the ESM model creating a new object of the type discovered. The creation of an object is immediately followed by a RequestObjectAttributeValueUpdate call to the RTI to obtain the current attribute values for that object.

ReflectAttributeValues
The ESM federate reflects the attribute values of the objects it subscribes after they have been updated by the publishing federate. Any kinematic attributes that are reflected with a time-stamp less than the current federate time are dead reckoned by the difference between the time-stamp and the current time. The concept of dead reckoning is discussed further in Section 4.1.

ProvideAttributeValueUpdate
Attribute values for objects that are published by the ESM federate can be requested by any other federates at any time during a federation execution. The ESM federate immediately sends out a RO updateAttributeValues containing the requested attribute values for instances of either ComponentEntity.SensorSystem.ESMSystem, Track.RelativeTrack.RelativeESMTrack or SensorTask.ESMTask. The federate time is encoded as a string in the tag for attributes that are updated RO. A TSO updateAttributeValues is also sent the next time through the simulation loop for those objects with kinematic attributes.

RemoveObjectInstance
Once objects are deleted by other federates (DeleteObjectInstance) they are removed from the ESM federate after receiving this callback. If a callback is received to remove a radar object then the associated track for that radar that is currently maintained by the ESM model is subsequently deleted from the federation by the ESM federate.

3.4.4 Ownership Management

No ownership management callbacks are handled by the ESM federate.

3.4.5 Time Management

The VSEMFederate handles all time management callbacks, except RequestRetraction which is not handled by any federate.

4. ESM Model

The ESM model was designed to primarily meet the requirements of the Sensor Data Fusion scenario and to also illustrate the concepts of the Virtual Ship. As this is the initial ES model in the Virtual Ship, a simple ESM model was constructed to provide generic ES track output data and satisfy the objectives of the scenario. It should be stressed here that the current ESM model does not represent a particular ESM system. However, the modelling infrastructure put into place will allow more realistic models to be easily integrated into the Virtual Ship construct once these models become available.

The current ESM model is made up of a number of classes that represent the modelled requirements of an electronic support measures system. The class structure of the ESM model is illustrated in Figure 5. The class structure shown in Figure 5 utilises the Unified Modelling Language (UML). For explanation of the UML notation see Appendix A. The ESM model contains the ESM class and all classes that are composed from it. The ESM model does not communicate directly with the RTI, instead information is passed back and forth through the ESMFederate class. ESMFederate is derived from VSEMFederate as discussed previously. The ESM federate has been designed in this manner to allow ESM models of different levels of fidelity to be used in a "plug and play" fashion.

An ESM object is created after the createComponentEntity callback is received by the ESM federate. There can only be a single instance of an ESM for each ESM federate. The configuration of the ESM system is detailed in the configuration file that is read in when the ESM is constructed. Utilising a configuration file when constructing an ESM object allows different types of ESM systems to be modelled by simply changing the parameter values for each simulation.

A Database is also created when the ESM is constructed. The database is used by the ESM to assist in identifying the detected radar signals. Once the ESM parent is discovered an instance of CompositeEntityESM is created. An instance of the appropriate object class is also created for every other object that is discovered by the ESM federate, i.e. discovery of a radar is followed by the instantiation of a Radar object. The ESM maintains a list for each of these objects to enable access to the members of those classes. Each of these classes has, as a minimum, relevant member variables that represent the attributes documented in the VS-FOM.

It is worth noting here that the philosophy adopted within the Virtual Ship is that it is the responsibility of the sensors to acquire the appropriate knowledge of their targets

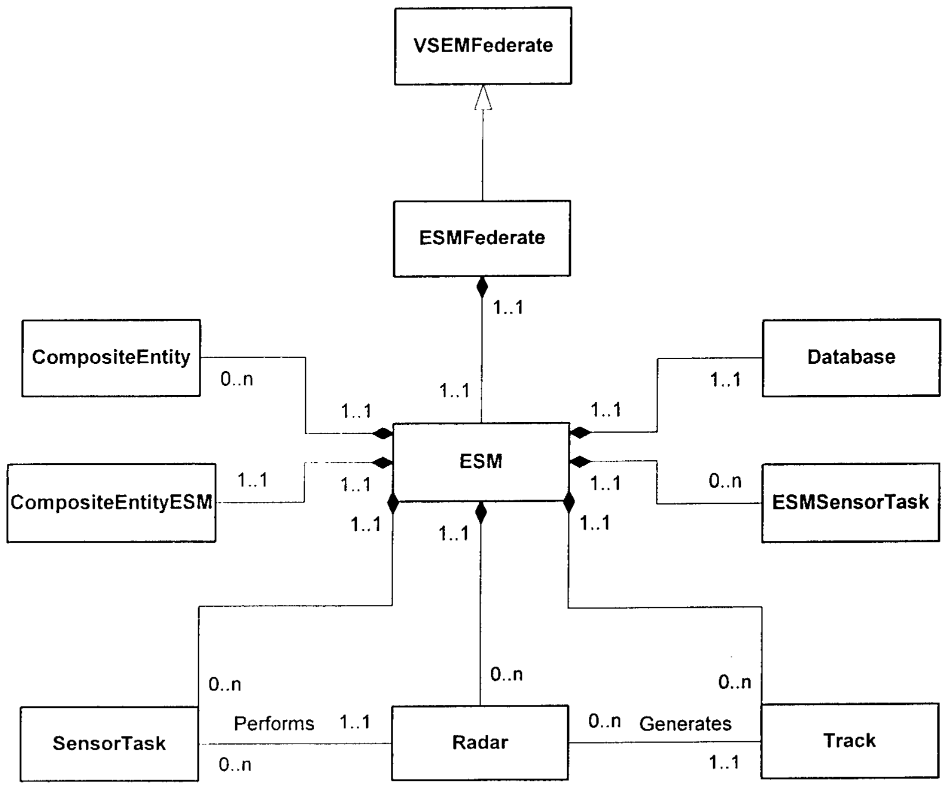


Figure 5: Class diagram for the ESM model.

to be able to compute their response. For the ESM this means determining the operating configuration of all discovered radars. Each radar configuration is dependent on the mode the radar is operating (defined by the radar's sensor task) and is obtained from a look-up table. Thus once the ESM has discovered and reflected the attributes of a radar and its corresponding sensor task, the ComponentName attribute of the radar and the TaskName attribute of the sensor task are used to obtain the radar configuration from a look-up table.

The state of the ESM model is updated for each iteration of the simulate loop of the ESM federate. The update function for the ESM model is shown in the flow diagram in Figure 6. Each of the functions within the update loop for the ESM model is discussed in detail below.

4.1 Dead-reckoning

The ESM is required to dead-reckon all objects within the ESM federate that have kinematic attributes. This involves extrapolating the kinematic attribute values for each of the time steps in between receiving attribute updates for that object. An enumeration of dead-reckoning algorithms is provided in the VS-FOM. These algorithms provide information on those attributes that need to be extrapolated in time and whether the extrapolation should be to first or second order. There are six dead reckoning algorithms in the VS-FOM and all of these are coded within the ESM model. The federate responsible for updating the attributes defines the algorithm to be used by setting the DRAlgorithm attribute for the corresponding object.

The kinematic attribute values for both the ESM parent composite entity and all other composite entities are dead reckoned each time the update function is called. The attribute updates for these objects are provided in the global reference system G and thus the dead-reckoned values are also referenced to G. The other object in the ESM model that contains kinematic attributes is the sensor task. The sensor task defines the orientation of the radar beam at any point in time. Each SensorTask object has a BeamPatternReference attribute that specifies the coordinate system to which the kinematic attributes are referenced. The SensorTask kinematic attributes are dead reckoned in the coordinate system defined by BeamPatternReference each time through the update function for every radar that is operating.

4.2 Range Calculation

The ESM model needs to calculate the range to the radar system in order to determine the signal power received by the ESM system. This calculation requires determining the vector between the origin of the ESM and radar systems. This is illustrated in Figure 7. To determine this vector a transformation of coordinate systems is required. The attributes for both the ESM and radar system component entities (e) are specified with respect to their parent composite entities (E) whilst the attributes for the composite entities (E) are given with respect to the global reference coordinate frame (G). The subscript i denotes coordinate systems associated with the ESM and its parent

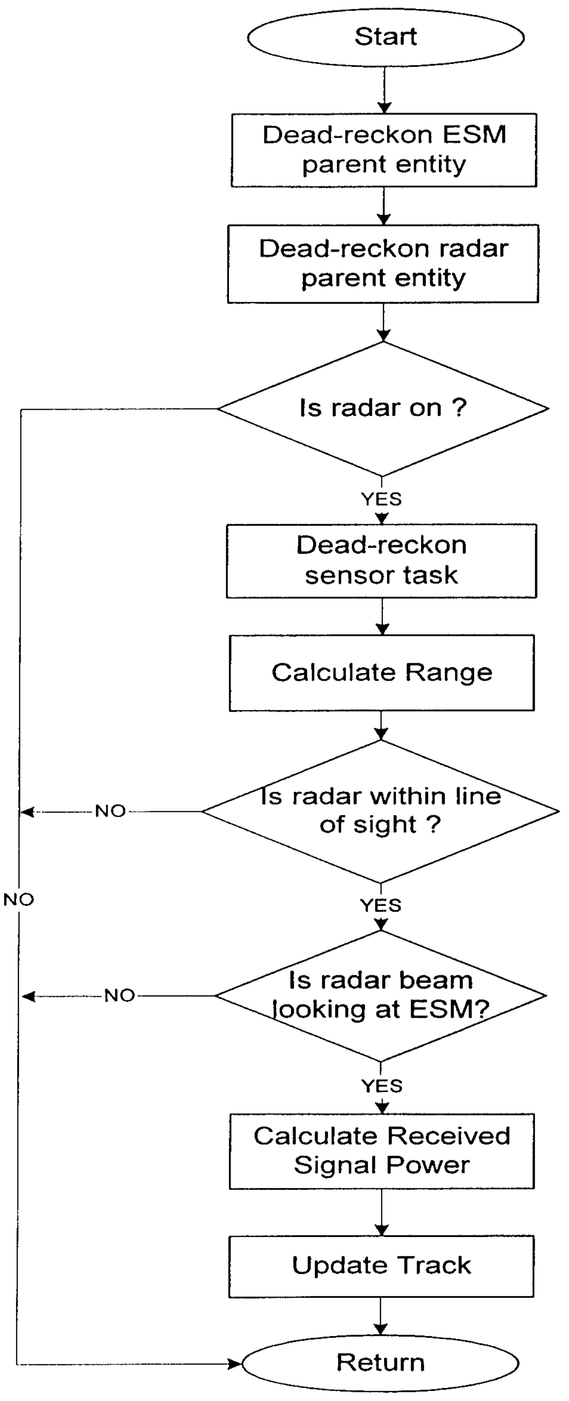

Start
Dead-reckon ESM parent entity
Dead-reckon radar parent entity
Is radar on ?
YES
Dead-reckon sensor task
Calculate Range
Is radar within line of sight ?
NO
YES
NO
Is radar beam looking at ESM?
NO
YES
Calculate Received Signal Power
Update Track
Return

Figure 6: Flow diagram for ESM model update function.

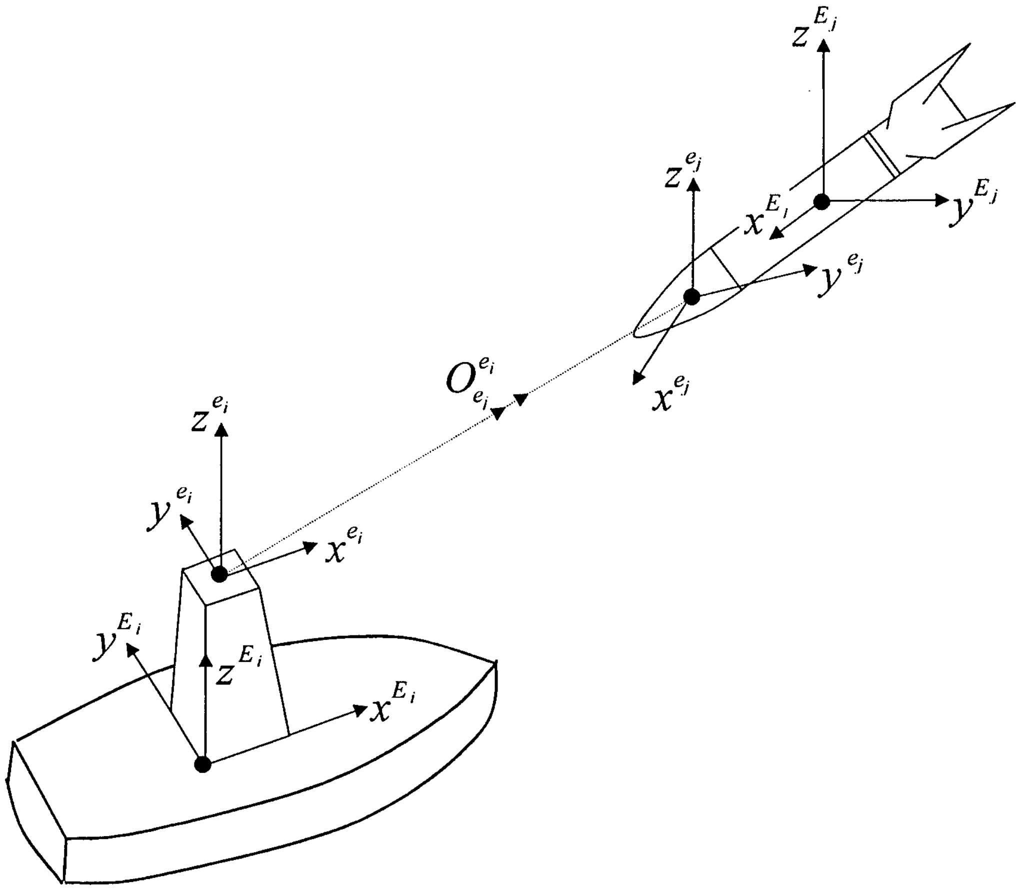


Figure 7: Vector ($\boldsymbol{O}^{e_i}_{e_j}$) required for determining the range between the ESM component entity (e_i) and a radar component entity (e_j).

and the subscript *j* denotes coordinate systems associated with the detected radar and its parent. The following represents the position attributes:

$$\text{Position of ESM system} = \boldsymbol{o}_{e_i}^{E_i}.$$

The position of the ESM system is given by the value of the ESM component entity RelativePosition attribute. This value is set in the script file and is passed to the ESM model via the createComponentEntity function.

$$\text{Position of radar system} = \boldsymbol{o}_{e_j}^{E_j}.$$

The position of the radar system is given by the reflected value of the radar component entity RelativePosition attribute.

$$\text{Position of ESM parent} = \boldsymbol{o}_{E_i}^{G}.$$

The position of the ESM parent is given by the reflected value of the ESM parent composite entity Position attribute.

$$\text{Position of radar parent} = \boldsymbol{o}_{E_j}^{G}.$$

The position of the radar parent is given by the reflected value of the corresponding composite entity Position attribute.

The position of the ESM system in the global reference frame (G) is given by

$$\boldsymbol{o}_{e_i}^{G} = \boldsymbol{o}_{E_i}^{G} + R_{E_i \to G}\boldsymbol{o}_{e_i}^{E_i}$$

and the position of the radar system in the global reference frame is given by

$$\boldsymbol{o}_{e_j}^{G} = \boldsymbol{o}_{E_j}^{G} + R_{E_j \to G}\boldsymbol{o}_{e_j}^{E_j}$$

where $R_{E_{i(j)} \to G}$ defines the rotation matrix to transform from the reference frame of the composite entity (E) to the global reference frame (G) and is given by

$$R_{E_{i(j)} \to G} = \begin{pmatrix} \cos\Omega_y\cos\Omega_z & -\cos\Omega_x\sin\Omega_z + \sin\Omega_x\sin\Omega_y\cos\Omega_z & \sin\Omega_x\sin\Omega_z + \cos\Omega_x\sin\Omega_y\cos\Omega_z \\ \cos\Omega_y\sin\Omega_z & \cos\Omega_x\cos\Omega_z + \sin\Omega_x\sin\Omega_y\sin\Omega_z & -\sin\Omega_x\cos\Omega_z + \cos\Omega_x\sin\Omega_y\sin\Omega_z \\ -\sin\Omega_y & \sin\Omega_x\cos\Omega_y & \cos\Omega_x\cos\Omega_z \end{pmatrix}.$$

The notation Ω represents the Euler angle set that defines this rotation and the values of Ω are given by the reflected CompositeEntity Orientation attribute for both the ESM and radar parent.

The vector joining the origins of the ESM system and radar system is thus given by

$$\boldsymbol{o}^G_{e_j} - \boldsymbol{o}^G_{e_i} = \left[\boldsymbol{o}^G_{E_j} + R_{E_j \to G}\boldsymbol{o}^{E_j}_{e_j} - \boldsymbol{o}^G_{E_i} - R_{E_i \to G}\boldsymbol{o}^{E_i}_{e_i}\right].$$

To determine the range, bearing and elevation between the ESM system and the radar system (in the ESM reference frame) this vector must be premultiplied by the rotation matrix to transform from the global coordinate system G back to the coordinate system of the ESM component entity e_i. Thus we have the following:

$$\boldsymbol{o}^{e_i}_{e_j} = R_{G \to e_i}\left[\boldsymbol{o}^G_{E_j} + R_{E_j \to G}\boldsymbol{o}^{E_j}_{e_j} - \boldsymbol{o}^G_{E_i} - R_{E_i \to G}\boldsymbol{o}^{E_i}_{e_i}\right]$$

$$= R_{E_i \to e_i} R_{G \to E_i}\left[\boldsymbol{o}^G_{E_j} + R_{E_j \to G}\boldsymbol{o}^{E_j}_{e_j} - \boldsymbol{o}^G_{E_i} - R_{E_i \to G}\boldsymbol{o}^{E_i}_{e_i}\right]$$

where $R_{E_i \to e_i}$ defines the rotation matrix to transform from the composite entity reference frame (E) to the component entity reference frame (e) and the Euler angle set that defines this rotation is given by the ESM RelativeOrientation attribute. The rotation matrix to transform from the global frame (G) to the composite entity reference frame (E) is given by $R_{G \to E_i}$ and the Euler angle set that defines this rotation is given by the ESM parent Orientation attribute. Both of these rotation matrices are defined by taking the transpose of $R_{E_{i(j)} \to G}$.

The range between the ESM system and the radar system is then simply

$$\text{Range} = \left|\boldsymbol{o}^{e_i}_{e_j}\right| = \left\{\left(\boldsymbol{o}_x{}^{e_i}_{e_j}\right)^2 + \left(\boldsymbol{o}_y{}^{e_i}_{e_j}\right)^2 + \left(\boldsymbol{o}_z{}^{e_i}_{e_j}\right)^2\right\}^{1/2}.$$

4.3 Line of Sight Calculation

After the range has been determined the next stage in the update function is to determine whether the radar is within line of sight of the ESM receiver. The line-of-sight (LOS) range is calculated using a spherical earth model, taking into account refraction by the atmosphere (approximated as 4/3 times the Earth's radius). The LOS range is given by the following formula:

$$LOS = \sqrt{\left(\frac{4}{3}\right)2R}(\sqrt{h_1} + \sqrt{h_2})$$

where R = radius of the Earth (m)
h_1 = height of ESM above sea-surface (m)
h_2 = height of radar above sea-surface (m)

The above formula assumes that the radius of the Earth is much larger than the height of either system above the Earth's sea-surface. The height of both the ESM and radar systems is determined by transforming the global position of each system (Cartesian coordinates) into a latitude, longitude and height. The method detailing this conversion is given in the coordinate document [5].

4.4 Radar Look Direction

Each radar has an associated sensor task that has attributes that define the orientation of the radar's beam pattern. The beam pattern coordinate system is denoted by S and its origin is assumed to coincide with the origin of the radar component entity associated with the sensor task. The sensor task can thus be used to determine where the radar's beam is directed at any instant in time. To determine if the beam pattern of a radar is directed at the ESM, the vector between the origin of the sensor task coordinate frame and the origin of the ESM system needs to be determined in the S frame. This vector is illustrated in Figure 8 and is given by

$$\boldsymbol{o}_{e_i}^{S} = R_{G\rightarrow S}\left[\boldsymbol{o}_{E_i}^{G} + R_{E_i\rightarrow G}\boldsymbol{o}_{e_i}^{E_i} - \boldsymbol{o}_{E_j}^{G} - R_{E_j\rightarrow G}\boldsymbol{o}_{e_j}^{E_j}\right]$$

where $R_{G\rightarrow S}$ defines the matrix rotation to transform from the global coordinate system G to the sensor task coordinate frame S and is given by the transpose of $R_{E_{i(j)}\rightarrow G}$. The BeamPatternOrientation attribute for the SensorTask object provides the Euler angle set that defines the rotation from the reference system to the S system.

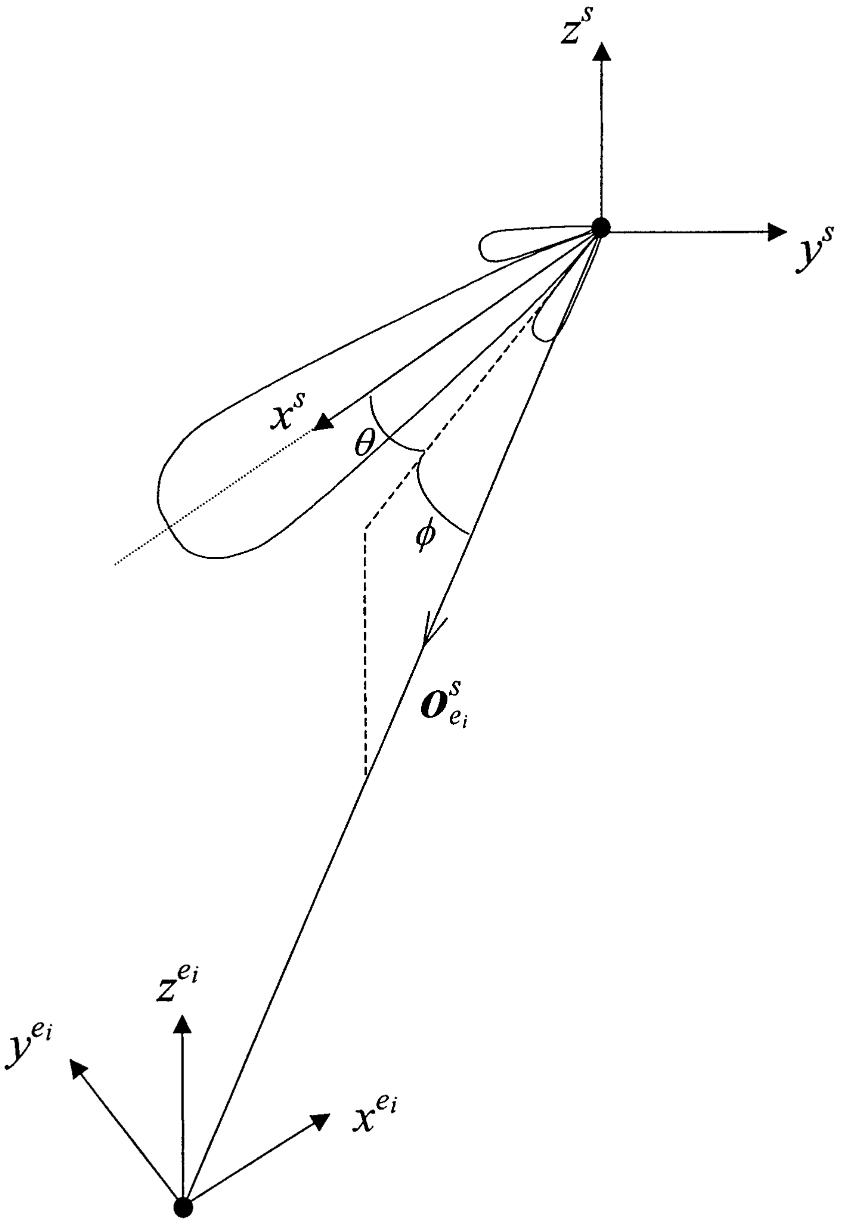


Figure 8: Vector ($\boldsymbol{O}^S_{e_i}$) required for determining the azimuth (θ) and elevation (ϕ) of the ESM component entity (e_i) with respect to the radar sensor task (S).

The reference system is specified by the BeamPatternReference attribute for the SensorTask object. If BeamPatternReference = 1 then the sensor task is defined relative to the G coordinate system and the Euler angle set that determines $R_{G \to S}$ is given by the SensorTask BeamPatternOrientation attribute.

If BeamPatternReference = 2 then the reference system is the LG system and

$$R_{G \to S} = R_{LG \to S} R_{G \to LG}$$

where the Euler angle set that determines $R_{LG \to S}$ is obtained from the SensorTask BeamPatternOrientation attribute and $R_{G \to LG}$ is given by

$$R_{G \to LG} = \begin{pmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\phi\cos\lambda & -\sin\phi\cos\lambda & \cos\phi \\ \cos\phi\cos\lambda & \cos\phi\sin\lambda & \sin\phi \end{pmatrix}$$

and ϕ and λ are the latitude and longitude of the radar parent composite entity. These values can be determined by converting the position attribute (Cartesian coordinates) into latitude, longitude and height using the method outlined in the coordinate document [5].

If BeamPatternReference = 3 then the reference system is the E_j system and

$$R_{G \to S} = R_{E_j \to S} R_{G \to E_j}$$

where the Euler angle set that defines $R_{G \to E_j}$ is given by the orientation attribute of the radar parent composite entity and the Euler angle set that defines $R_{E_j \to S}$ is given by the SensorTask BeamPatternOrientation attribute.

Having determined the vector between the SensorTask origin and the origin of the ESM coordinate system, the azimuth and elevation of the ESM system with respect to the SensorTask needs to be determined. The azimuth (θ) and elevation (ϕ) as shown in Figure 7 are given by the following:

$$\theta = arctan\left(\frac{\boldsymbol{o}^{S}_{ye_i}}{\boldsymbol{o}^{S}_{xe_i}} \right),$$

$$\phi = \arctan\left(\frac{\boldsymbol{O}^{S}_{ze_i}}{\left[\left(\boldsymbol{O}^{S}_{xe_i}\right)^2 + \left(\boldsymbol{O}^{S}_{xe_i}\right)^2 \right]^{1/2}} \right).$$

If the modulus of both the azimuth and elevation is less than or equal to half of the radar 3 dB beam width in those directions then the main lobe of the radar is considered to be looking at the ESM system. It should be noted here that no attempt is made to determine the drop in received power as a function of angle in the main lobe, or to calculate the received signal power from the side lobes. However, having evaluated the azimuth and elevation, it should be straight forward to include side lobes in the calculation as a future modelling requirement.

4.5 Received Signal Power

The signal power received by the ESM is given by the following:

$$S_r = \frac{(ERP) G_r \lambda^2}{(4\pi)^2 R_r^2 L_r}$$

where S_r = received signal power at the ESM (dBm)
(ERP) = effective radiated power of the radar seeker (dBm)
G_r = gain of ESM antenna in the direction of the radar (dB)
λ = wavelength of the radar seeker λ = c/frequency (m)
R_r = range from the radar to the ESM (m)
L_r = losses at the ESM receiver (dB).

The radar signals that can be detected by the ESM are limited by the minimum and maximum radio frequency (RF), pulse width (PW), pulse repetition frequency (PRF) and elevation values of the ESM receiver being modelled. The RF, PW and PRF for each radar are obtained from the look-up table after discovering the radar and it's associated sensor task. The elevation of the radar with respect to the ESM is calculated when determining the range between the ESM and the radar. If either the RF, PW, PRF or elevation of the radar is not within the limits specified for the ESM system, the received signal power is set to zero.

The ESM system being modelled also has a specific dynamic range and sensitivity. If the calculated received signal power is below the ESM sensitivity, the ESM receiver does not detect the signal. At close ranges the received signal power from a radar may be larger than the sensitivity plus the dynamic range, resulting in saturation of the ESM receiver. For this case the received signal power is set to the maximum detectable limit which is simply given by the addition of the ESM sensitivity and dynamic range.

4.6 Tracks

Each time the received signal power from a radar is larger than the sensitivity and less than the sensitivity plus the dynamic range of the ESM system, an ESM detection has occurred. Each ESM detection represents a sequence of pulses (pulse train), with enough pulses to enable identification. After each detection the trackRadar function is called to determine the track conditions for that radar. The ESM instantiates a track for a radar once it has been detected a predetermined number of times (currently 10). Once a track has been created it is immediately registered with the RTI to allow subscribing federates to discover the track object. Each of the track attributes is also set within the trackRadar function, along with a flag indicating that the track attributes need to be updated next time through the simulate loop of the ESM federate.

One of the attributes for the track object is platform classification. The ESM model uses a database (separate to the radar look-up table) to assist in identifying the platform. The RF, PW and PRF parameters for every radar entry in the database are compared to those of the detected radar (obtained from the look-up table) and all matches are provided as possible platform classifications. If no matches are found the radar remains to be tracked, however the platform classification is labelled as unmatched.

For each iteration of the ESM update function the track parameters are also updated. If a radar has not been detected for five seconds the track is assumed lost and is consequently deleted from the ESM federate next time through the simulate loop. If that same radar is detected again (after the track has been deleted) a new track is created. Thus there can only be one track for each radar at any point in time in the simulation. In addition, if the received signal power from a radar saturates the ESM system it is assumed that reliable track attributes cannot be measured and consequently the track is set to invalid. Invalid tracks are still monitored by the ESM, however only the Valid attribute (false) is provided to the federation.

5. Implementation Issues

The ESM federate has been successfully integrated into the first Virtual Ship scenario, namely the Sensor Data Fusion scenario. Implementation of the ESM federate established that the ESM federate correctly interacts with the VSEM and that data is exchanged as specified in the ESM SOM. The ESM federate also correctly acquires information regarding its parent composite entity and all relevant coordinate transformation calculations were performed correctly. There were several implementation issues that needed to be addressed to allow the ESM federate to meet all these criteria. These issues are discussed in detail below.

The ESM federate is written in C++ and compiled as a Win32 console application. Win32 machines use a little-endian architecture to define the byte ordering of multi-byte datatypes. In little-endian architecture the right-most bytes are most significant. However the VSA has adopted the standard of sending out data to the RTI in network byte ordering, which uses a big-endian architecture. Thus the ESM federate must swap the byte order for every multi-byte attribute that is either sent to the RTI or received as a federate Ambassador callback.

After an ESMFederate object instance is created control is passed to VSEMFederate for further processing. At a particular time (defined in the script file) the VSEMFederate then issues a callback to the ESM federate to create a component entity and at this point an instance of an ESM is constructed. Even though all events are time stamped, it is possible for the ESM federate to receive multiple callbacks (either base class or Federate Ambassador) at the same time. Further to this there is no guarantee of the order that these callbacks will be received by the ESM federate. As a result of this it is possible for the ESM federate to discover objects before an instance of an ESM has been created. Since the state of all objects is maintained by the ESM through lists or pointers, the simulation crashes when an attempt is made to link the discovered object to the ESM model. The solution to this problem is for the ESM federate to keep a list of object handles (locally unique identifier) for all objects that are discovered before the ESM is created. Following creation of the ESM this list is traversed and for each object in the list a localDeleteObjectInstance call is made to the RTI. The effect of this is that the RTI acts as if the ESM federate had not received the DiscoverObjectInstance callback and thus reissues the callback for each object. As the ESM has now been created the usual event handling can occur.

In the sensor data fusion scenario the parent entity of the ESM also has a radar attached. As a result a method had to be implemented to ensure that the ESM system did not detect its own ship radar system. All radar systems that are discovered by the ESM federate result in an instance of Radar being created. These radars are maintained in a list in the ESM model and the radar attributes are populated once the ReflectAttributeValues callback for that radar is received. One of these radar attributes defines the composite entity object instance name of the radar parent. If a radar is reflected with the same parent instance name as the ESM system then it is removed from the list to ensure no further processing of the own ship radar.

Another implementation issue that arose relates to the specification of the transportation type and order type of the attributes in the VS-FOM. As the ESM federate is both time regulating and time constrained, TSO messages can be both sent and received by the ESM federate. To ensure that all attributes are both sent and received TSO the order type for these attributes must be set to time-stamp in the VS-FOM and the transportation type must be set to reliable. It was found that for attributes whose transportation type was set to best-effort, occasionally the ESM federate would receive RO messages instead of TSO messages.

6. Future Directions

The ESM federate currently meets all requirements for participating as an execution managed federate within the Sensor Data Fusion scenario. A logical future development is to increase the fidelity of the ESM model as it currently stands. One such development would be to use the radar sensor tasking to model the received signal power levels in the side-lobes of the radar beam pattern. This would require information about the radar beam pattern in both the azimuth and elevation directions. This information is currently not available for the ESM federate. A judgement would be required to determine if modelling the side-lobe power levels was actually beneficial to the scenario under consideration.

In the ESM federate the only propagation loss that is calculated between any radar and the ESM system is the $1 / R^2$ spatial factor. Signal propagation between the ESM and radars certainly needs to be addressed. A ray trace method could possibly be used to determine the propagation path from the radar to the ESM receiver. The current method adopted for the Virtual Ship Architecture is that each sensor performs its own propagation modelling. This method can however lead to inconsistencies as the propagation can be modelled differently by the various sensors. As a result a central propagation federate is currently being proposed for the Virtual Ship. The advent of such a propagation federate would provide an excellent method for modelling additional propagation losses relevant to the ESM federate.

In addition to improving the fidelity of the current ESM model, additional ES models that represent specific ES systems could be integrated into the Virtual Ship using the infrastructure that has already been developed. For these models to provide useful information on the performance of a realistic ES system in the maritime environment some of the issues that need to be considered are receiver architectures, ESM search strategies, the impact of the emitter environment, receiver antenna patterns and ambiguities in the ES threat library.

A service set that is offered by the RTI but not currently implemented by the ESM federate is Data Distribution Management. This service provides a data filtering mechanism and is worth exploring as part of the future developments of the ESM federate. As an example this service could be utilised so that any radar that is operating with a frequency outside the detectable limits of the ESM system will be filtered by the RTI and hence not seen by the ESM federate.

From a broader perspective, the involvement of Electronic Warfare Division (EWD) in the Virtual Ship project should be extended with the inclusion of Electronic Counter Measures (ECM) modelling. The incorporation of a NULKA model into the Virtual Ship would provide one ECM method that would be most beneficial. Other ECM that could be included are jamming and chaff. It needs to be stressed here that these counter measure systems will only benefit the Virtual Ship if other participating federates have sufficient fidelity to model the effect of the ECM on their systems.

7. Acknowledgements

The author would like to acknowledge the contributions from all members of the Virtual Ship group in both the construction of the ESM federate and the preparation of this document. In particular the contributions from Anthony Cramp were instrumental in developing and integrating the ESM federate into the Virtual Ship Architecture. Grant Horsfall, John Best and Anthony Cramp also reviewed the manuscript.

8. References

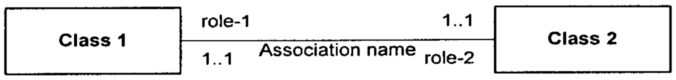
1. Best, J.P. et al. (2000) *Virtual Ship Architecture Description Document Issue 1.00.* DSTO-GD-0257.
2. URL - http://hla.dmso.mil/tech/omtspec.html
3. Cramp, A. and Best, J.P. (2001) *Execution Management in the Virtual Ship Architecture Issue 1.00.* DSTO-GD-0258.
4. *Base Classes and Utilities,* In preparation.
5. Best, J.P. (2000) *Coordinate Usage in the Virtual Ship Architecture Issue 1.00.* DSTO-GD-0260.

9. List of Acronyms

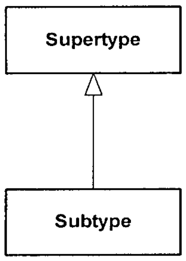
ECM	Electronic Counter Measures
ES	Electronic Support
ESM	Electronic Support Measures
EWD	Electronic Warfare Division
FOM	Federation Object Model
HLA	High Level Architecture
IRST	Infra Red Search and Track
LOS	Line of Sight
MOD	Maritime Operations Division
OMT	Object Model Template
PRF	Pulse Repetition Frequency
PW	Pulse Width
RF	Radio Frequency
RO	Receive Order
RTI	Run Time Infrastructure
SOM	Simulation Object Model
TSO	Time Stamped Order
UML	Unified Modelling Language
VSA	Virtual Ship Architecture
VSEM	Virtual Ship Execution Manager
VS-FOM	Virtual Ship Federation Object Model
VSSD	Virtual Ship Simulation Display

Appendix A: UML Notation

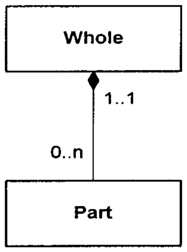
Association



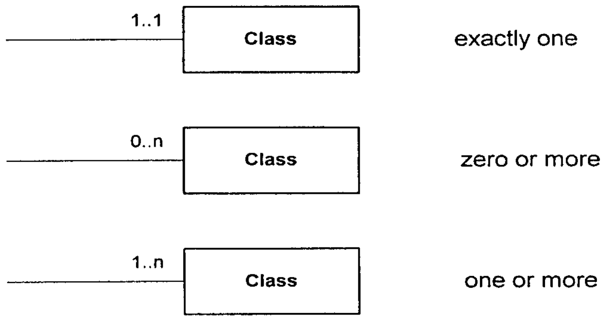
Generalization



Composition



Multiplicity



DISTRIBUTION LIST

Constructing an Infrastructure to Facilitate Electronic Support Modelling in the Virtual Ship

Shane A. Canney

AUSTRALIA

		Number of Copies
1. DEFENCE ORGANISATION		
Task sponsor: DGMD		1
S&T Program		
Chief Defence Scientist	} shared copy	1
FAS Science Policy		
AS Science Corporate Management		
Director General Science Policy Development		
Counsellor Defence Science, London		Doc Data Sheet
Counsellor Defence Science, Washington		Doc Data Sheet
Scientific Adviser to MRDC Thailand		Doc Data Sheet
Scientific Adviser Policy and Command		1
Navy Scientific Adviser		Doc Data Sheet 1 x distribution list
Scientific Adviser - Army		Doc Data Sheet 1 x distribution list
Air Force Scientific Adviser		1
Director Trials		1
Aeronautical and Maritime Research Laboratory		
Director		1
CMOD		1
Dr. John Best (MOD)		1
Anthony Cramp (MOD)		1
Nickolas Luckman (WSD)		1
Dr. Darren Sutton (WSD)		1
Marcus Saunders (WSD)		1
Dr. Bernard Kachoyan (MOD)		1
Electronics and Surveillance Research Laboratory		
Director		Doc Data Sheet 1 x distribution list

EWSTIS	softcopy for accession to EWSTIS Web site
CEWD	Doc Data Sheet
Research Leader, Electronic Warfare Systems	Doc Data Sheet
Research Leader, RF Electronic Warfare	Doc Data Sheet
Head, Maritime Systems	Doc Data Sheet
Head, Aerospace Systems	Doc Data Sheet
Head, Strategic Electronic Warfare	Doc Data Sheet
Head, ES Systems	Doc Data Sheet
Patrick Taliana (EWD)	Doc Data Sheet
Dr. Shane Canney (EWD)	1
Catherine Howard (EWD)	Doc Data Sheet
Dr. Jonathan Legg (SSD)	1
DSTO Library	
Library Fishermans Bend	Doc Data Sheet
Library Maribyrnong	Doc Data Sheet
Library Salisbury	2
Australian Archives	1
Library, MOD, Pyrmont	Doc Data sheet
National Library of Australia	1
Capability Systems Staff	
Director General Land Development	Doc Data Sheet
Director General Aerospace Development	Doc Data Sheet
Knowledge Staff	
Director General Command, Control, Communications and Computers (DGC4)	Doc Data Sheet
Director General Intelligence, Surveillance, Reconnaissance, and Electronic Warfare (DGISREW)R1-3-A142 CANBERRA ACT 2600	Doc Data Sheet
Director General Defence Knowledge Improvement Team (DGDKNIT) R1-5-A165, CANBERRA ACT 2600	Doc Data Sheet
Navy	
SO (SCIENCE), COMAUSNAVSURFGRP, BLD 95, Garden Island, Locked Bag 12, PYRMONT NSW 2009	Doc Data Sheet 1x distribution list

Army

Stuart Schnaars, ABCA Standardisation Officer, Tobruck Barracks, Puckapunyal, 3662	4
SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), MILPO Gallipoli Barracks, Enoggera QLD 4052	Doc Data Sheet
NPOC QWG Engineer NBCD Combat Development Wing, Tobruk Barracks, Puckapunyal, 3662	Doc Data Sheet

Intelligence Program

Manager, Information Centre, DIO	1
DGSTA, Defence Intelligence Organisation	1

Corporate Support Program (libraries)

OIC TRS, Defence Regional Library, Canberra	1
US Defense Technical Information Center,	2
UK Defence Research Information Centre,	2
Canada Defence Scientific Information Service,	1
NZ Defence Information Centre,	1

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy	
Library	1
Head of Aerospace and Mechanical Engineering	1
Serials Section (M list), Deakin University Library, Geelong, 3217	1
Hargrave Library, Monash University	1
Librarian, Flinders University	1

OTHER ORGANISATIONS

NASA (Canberra)	1
AusInfo	1
State Library of South Australia	1
Parliamentary Library, South Australia	1

OUTSIDE AUSTRALIA

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK	1
Library - Exchange Desk, National Institute of Standards and Technology, US	1
National Aerospace Laboratory, Japan	1
National Aerospace Laboratory, Netherlands	1
SPARES (Normally 6 copies)	5

Total number of copies: 55

Total number of hard copies: 54

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE Constructing an Infrastructure to Facilitate Electronic Support Modelling in the Virtual Ship	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) Shane A. Canney	5. CORPORATE AUTHOR Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury SA 5108 Australia

6a. DSTO NUMBER DSTO-TR-1159	6b. AR NUMBER AR-011-889	6c. TYPE OF REPORT Technical Report	7. DOCUMENT DATE May 2001

8. FILE NUMBER U 9505-19-107	9. TASK NUMBER NAV 00/011	10. TASK SPONSOR DGMD	11. NO. OF PAGES 32	12. NO. OF REFERENCES 5

13. URL on WWW http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-1159.pdf	14. RELEASE AUTHORITY Chief, Electronic Warfare Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SA 5108

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Electronic support measures
Virtual reality
Sensor data fusion
High level architecture
Simulation

19. ABSTRACT

This document provides a detailed description of the infrastructure that has been constructed to allow Electronic Support (ES) modelling for Virtual Ship applications, together with a discussion of the current Electronic Support Measures (ESM) model. The infrastructure and ESM model, collectively called an ESM federate, has been succesfully integrated into the first scenario of the Virtual Ship, namely the Sensor Data Fusion scenario. The implementation issues that arose throughout the integration process are also addressed to aid future developers.