Creating a Re-useable knowledge repository for UK MoD CGF Behaviours

Mark Lewis

Centre for Simulation and Analytics
Cranfield University
Defence Academy of the United Kingdom

m.d.lewis2@cranfield.ac.uk

Dan Allison

Discovery Machine Inc. Williamsport, PA 17701, USA

dallison@discoverymachine.com

ABSTRACT

The United Kingdom's Ministry of Defence (MOD) has a large investment in both Computer Generated Forces (CGF) systems and in the supporting data and models that run within them. During the last audit over 20 different CGF were identified in use across the three service domains. A major concern is that the investment in behaviour modelling and data in one area is not easily re-useable due to the fundamental differences in implementation of that data in those systems.

To address this issue, the UK Defence Science and Technology Laboratory (dstl) under the Centre for Defence Enterprise (CDE) set a challenge to investigate the feasibility of achieving methods for capturing models and data within common knowledge repositories for re-use across CGF and common modelling services.

The purpose of this paper is to describe the work of Cranfield University at the Defence Academy to evaluate a new approach for knowledge capture and simulation agnostic execution of CGF Behaviours. It discusses the elements required to define a framework for the UK to develop a Behaviour Repository in the context of informing a possible future MoD CGF Service. This includes a discussion on the software architecture and processes; the lessons learned in the development of a multi-simulation behaviour authoring console; the skills required to populate such a repository and the governance required to provide a MoD CGF Service (including Non-run-time services). It describes the challenges both technical and non-technical in developing such a framework that enables greater re-use of verified and validated data models and behaviours across different simulation systems without being tied to a single CGF supplier.

ABOUT THE AUTHORS

Mark Lewis is a former Royal Navy Air Engineer and now lecturer at the Defence Academy of the United Kingdom. He has over 7 years of experience in research, development and procurement of simulation systems in the UK. For the last 3 years he has been lecturing a variety of military and civilian students on the use of M&S within the military environment. He has worked on the UK Training Transformation Concept Capability Demonstrator and spent 2 ½ years as the M&S lead for the UK MoD Lightening II Program. He has received commendations from the UK MoD Chief of Defence Material and Chief Scientific Advisor. Mark received his MSc in Defence Simulation and Modelling from Cranfield University and a MEng (Hons) in Systems Engineering from Loughborough University.

Dan Allison is the VP of Professional Services for Discovery Machine Inc. He has 25 years experience in system engineering and modelling in DoD SONAR and RADAR systems, military training systems, and commercial medical devices. Recent work focuses on knowledge engineering using the Discovery Machine Knowledge Capture Methodology and building deployable cognitive models into training systems. Mr. Allison earned a BSEE from Penn State and MSEE from Syracuse University and is six sigma certified. He received 2 patents on the topic of deconvolution algorithms and has been an active member of IEEE and INCOSE.

Creating a Re-useable knowledge repository for UK MoD CGF Behaviours

Mark Lewis

Centre for Simulation and Analytics
Cranfield University
Defence Academy of the United Kingdom

m.d.lewis2@cranfield.ac.uk

Dan Allison

Discovery Machine Inc. Williamsport, PA 17701, USA

dallison@discoverymachine.com

INTRODUCTION

The United Kingdom (UK) Strategic Defence and Security Review (SDSR) 2010 stated that the UK Ministry of Defence (MOD) needed to make a step change in training using Off the Shelf (COTS) available both commercially and from government only sources.

The United Kingdom's Ministry of Defence has a large investment in both Computer Generated Forces (CGF) systems and in the supporting data and models that run within them. During the last audit, over 20 different CGF were identified in use across the three service domains. A major concern is that the investment in behaviour modelling and data in one area is not easily re-useable due to the fundamental differences in implementation of that data in those systems.

Initial work undertaken by the Niteworks [1]Training Transformation Concept Capability Demonstrator (TTCCD) on behalf of Capability Joint Training Experimentation and Simulation (CapJTES) in 2012 identified the need for a Simulation Reference Architecture. Niteworks is a partnership between industry and the MOD, including the Defence Science & Technology Laboratory (Dstl). This work by Niteworks proposed an architecture that provided Run-time and Non Run-time services to support MOD Computer Generated Forces (CGF):

> A non-runtime service includes the software development and configuration management of the CGF tool or tools of choice for the UK MOD.
>
> A runtime service can be expressed as "The provision of semi-autonomous representation of civilian population (clutter), red forces (threat generation) and blue forces (flanking and on player unit representation)".

It specifically suggested that a key requirement for a CGF service is to package verified baseline versions that make it easier for users and developers to use and extend the CGF it described this as:

- The provision of semi-autonomous representations of individuals, units and platforms across multiple domains (Land, Air, Surface, Sub-Surface and Space) in order to generate civilian population (clutter), red forces (threat generation) and blue forces (flanking and non-player unit) representations within the simulation environment

To support this step change in training and education the Defence Science and Technology Laboratories (Dstl) as part of the MOD Centre for Defence Enterprise (CDE) launched a Training and Evaluation Challenge in Nov 2012. Within the challenge were a number of calls.

Call 7 was to investigate the feasibility of achieving methods for capturing models and data within common knowledge repositories for re-use across CGF and common modelling services. Dstl described the outcomes they wanted as described below and illustrated in Figure 1

[1] [1] www.niteworks.com UK MOD / Industry Collaboration for Decision Support.

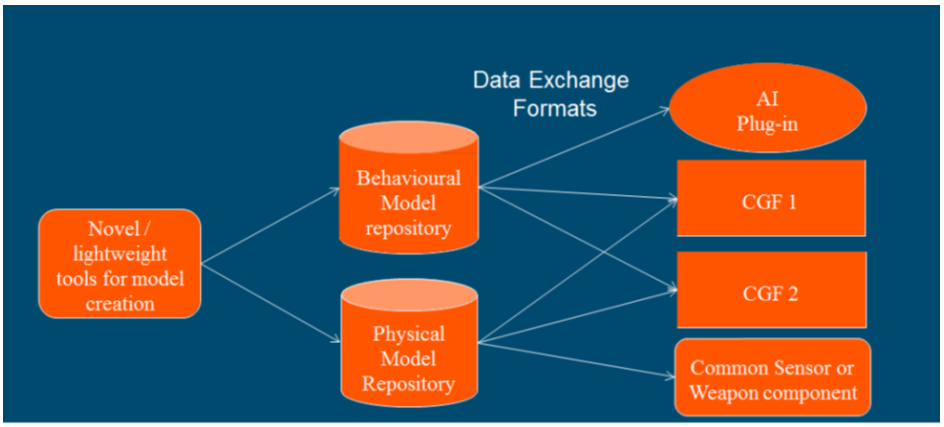

Figure 1. Scope of the Challenge for MoD (Image: Courtesy of dstl)

Dstl was after Novel/lightweight tools for model creation and specification of data parameters (easy to use); with methods for:

- Capturing models & data as re-useable knowledge repositories;
- Populating environments and simulation components with models from common knowledge repositories
- Use of common data exchange formats
- Proposals for physical and/or behavioural modelling approaches

Dstl stated that they didn't want Proprietary toolsets / interfaces with overly complicated approaches that are difficult to use and Run time solutions (i.e., solutions that couldn't be modified without recourse to the vendor). The exploitation path for any solutions was described as needing to provide or inform a common modelling and data repository that provides a central location for capturing knowledge and re-use across defence for simulation, that reduces model and data procurement costs ('buy once') and provides a consistent and coherent approach to use of defence models and data through re-use. (dstl 2012)

The purpose of this paper is to describe the work of Cranfield University at the Defence Academy to evaluate a new approach for knowledge capture and simulation agnostic execution of CGF Behaviours. It discusses the elements required to define a framework for the UK to develop a Behaviour Repository in the context of informing a possible future MoD CGF Service. This includes a discussion on the software architecture and processes; the lessons learned in the development of a multi-simulation behaviour authoring console; the skills required to populate such a repository and the governance required to provide a MoD CGF Service (including Non-run-time services). It describes the challenges both technical and non-technical in developing such a framework that enables greater re-use of verified and validated data models and behaviours across different simulation systems without being tied to a single CGF supplier. Specific recommendations for MOD are provided.

GENERAL APPROACH

A robust process for building CGF behaviours is similar to other software development processes. The process we follow in this work is depicted in Figure 2.

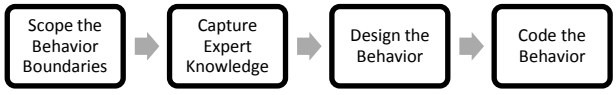

Figure 2. Behaviour Building Process

In scoping the work for the behaviour, we decide the basics of how the behaviour should perform and what the typical environmental conditions would be. We also explore a possible range of conditions and decide a boundary for the expected performance. This includes defining what the starting conditions of the entity would be and what state the entity would be in at the end of the behaviour. Once we understand the scope of the problem, we move on to more detailed knowledge capture.

In the Knowledge capture phase, we choose good subject matter experts (SME) who have both domain knowledge pertaining to the behaviours in mind and experience in training where the behaviours will be used. We used a structured interview process that stimulates the SME to recall tacit knowledge and articulate that knowledge in a way which allows us to understand their decision making process. This knowledge is initially sketched to represent the human decision making process as shown in Figure 3. We find that this type of knowledge representation more readily develops into models as opposed to just recording the interview in a text document.

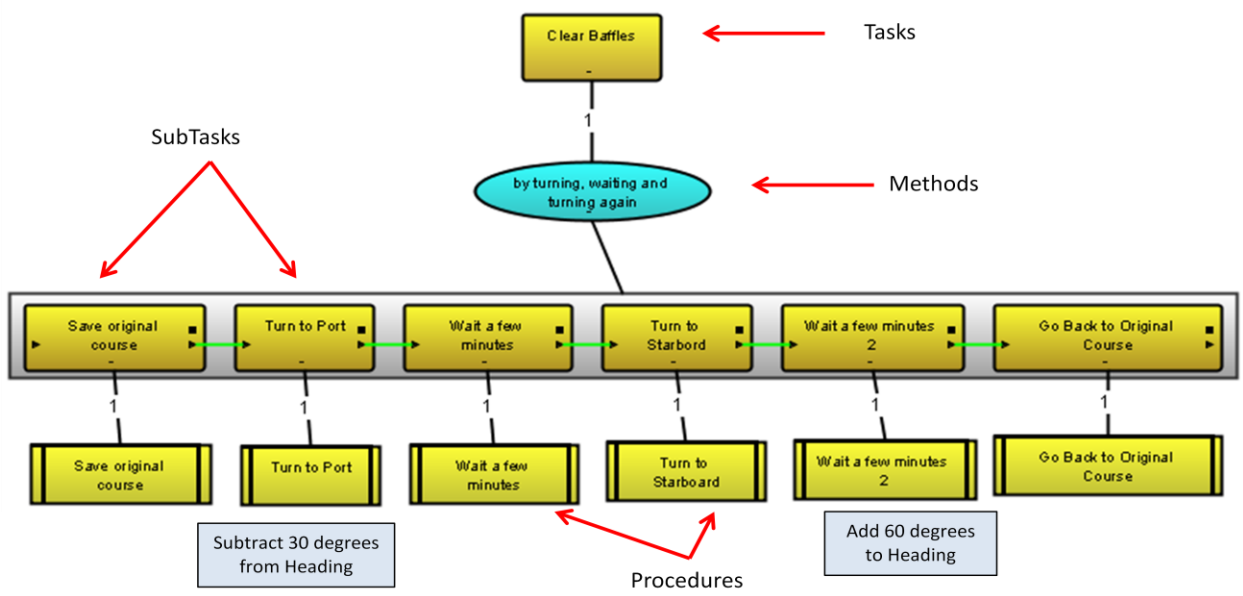

Figure 3. Graphical Representation of an Expert Decision Making Process

Designing the behaviour flows smoothly from the knowledge capture interviews by virtue of the visual language for representing the cognitive process of the SME. Since the knowledge capture is represented in a hierarchical task representation, we can continue with design by decomposing each task in a set of subtasks. At this time, we also consider how the simulation is going to enable the behaviour and what level of fidelity is necessary to achieve the training objectives.

As we continue the task decomposition, we get to the point where the tasks can be implemented by simulation function calls. At this point, we can complete the coding by implementing the correct function calls, providing the correct parameter types and mapping the data throughout the behaviour model. Of course, each of these four steps can be iterative in that as we learn more details, we may have to go back and capture new knowledge and include that in the overall process. Testing is also performed when the behaviour model is coded enough to run within the simulation.

The advantage of this approach is that is also allows the development of behaviour descriptions that can be used to provide baseline behaviour descriptions and documents for other simulations applications or as a means of testing verification and validation of the implementation of a behaviour in another simulation application.

Behaviour Building Blocks

For most practical cases, users want to build a large set of behaviours which would support a variety of training scenarios. Often this set of behaviours is not known ahead of time in enough detail to spec out each one as a separate effort. In our work, we address this need by first identifying a common set of behaviour building blocks and allowing for easy ways to assemble these building blocks into unique behaviours as the need arises. We call these building blocks Basic Level Actions (BLA). Some examples of BLAs for surface ship behaviours are "Move to Location", Turn on Radar", and "Search for OPFOR Sub".

These BLAs can then be used to build a variety of behaviours or missions. Consider the two behaviours below built from this small set of BLAs

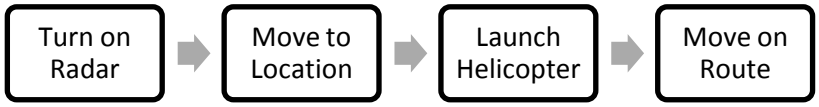

Figure 4. Launch Helo Behaviour

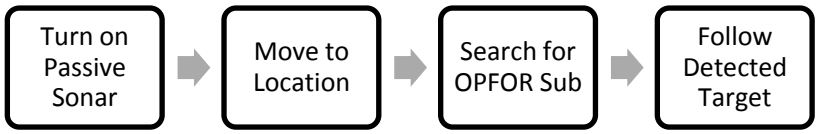

Figure 5. Find and Follow Submarine Behaviour

The two behaviours are quite different, yet each can be authored in a matter of minutes once a set of BLAs are developed. Of course, these are not the complete behaviours, just the tasks for the main mission. Typically these tasks would be embedded in a larger behaviour template which includes a situation awareness process and a reaction process. We have constructed behaviour templates for a few different simulations which can be re-used for a wide variety of mission types.

After several iterations, we arrived at a level of abstraction for the BLAs which proves useful for authoring a variety of missions for different entity types. At one point, we attempted to make the actions very basic and generic so they could be used for any entity type. While interesting, this proved too hard to use to build missions as it put the burden on the mission author to essentially think like a programmer and design a model of a mission. Instead, we decided to identify the major tasks that each entity type would perform in a training simulation. For surface ships, these major categories are as follows.

- Move
- Turn On/Off sensors
- Search
- Follow
- Fire Weapons
- Launch Aircraft

Within each category, we built BLAs that would be useful in automating behaviours for a training simulation. For example, in the Move category, we have these BLA options

- Move to a Location
- Move on a Route
- Move Parallel to a Shipping Route
- Move on a Heading for Specified Time

We have similar breakdowns in the other categories. We find that even this simple set allows for creation of a variety of useful mission types.

Behaviour Engine

One key concept which enables our approach to work is a behaviour engine. This is a separate process which runs the behaviour model outside of the simulation process. The behaviour engine enables another option for controlling characters in the simulation besides the simulation AI. In this research we happened to use the Discovery Machine

Knowledge Service Engines which are integrated with JSAF and VBS2/3 as well as a few other simulations. These engines are capable of reading behaviour models constructed using the Task Method Knowledge (TMK) approach and consoles described by Potts (2010) .

In this approach, the behaviour hierarchy is developed as described above and compiled into a 'gear' file (a serialized Java object). The engine runs the gear and executes the embedded tasks, methods and procedures in a sequential manner as they are laid out in the behaviour hierarchy. Inside the procedures are function calls which the engine executes by calling on the corresponding functions within the simulation. In the original approach described in 2010 Potts and the team developed simulation specific engines that integrate with each individual simulation application so each one knows the correct parameter types to use for each simulation function call.

The main focus of effort to extend this concept is develop a standardized set of simulation function declarations that apply to more than one simulation, in this case both JSAF and VBS2/3. These function declarations are the format that the BLAs use to call the functions. The function implementations, within each engine, contain the specifics that work with each simulation. The overall concept is depicted in Figure 6.

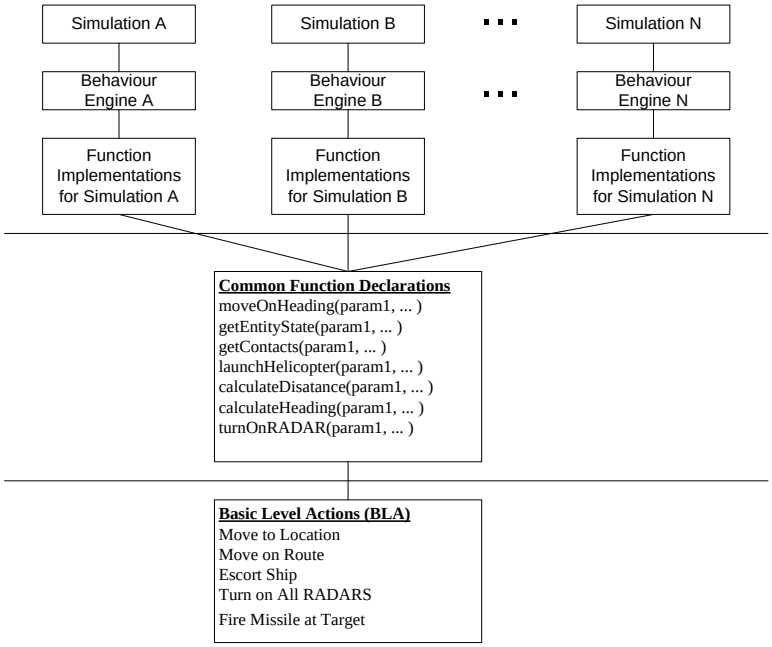

Figure 6. Multi-Sim Concept

With this design, end users can author behaviours by sequentially assembling BLAs within a behaviour template. The BLAs use the functions shown in the middle of Figure 6 to interact with the simulation. Depending on which simulation is used, the function implementations within the corresponding engine talk to the simulation in the correct format.

DETAILED DESCRIPTION

We developed a capability for users to author one behaviour model, using a TMK Console (an application to sketch, develop and compile the behaviour) and a new set of "multi-sim" BLAs, and then compile this behaviour into an execution “Gear” file. This Gear file can then be run in either JSAF or VBS2 (using an updated Knowledge Service Engine(KSE)-VBS2 edition (i.e., VBS2 engine)).

This was achieved by first creating a new layer of simulation integration functions above both the previous JSAF and VBS2 integration functions. This set of higher level functions translates the simulation agnostic inputs and outputs into simulation specific inputs and outputs. We standardized on the JSAF interface for function I/O format so that engine did not change. We also made modifications to the VBS2 KSE to handle the multi-sim function

formats. As an example, the *"GetContacts"* function (whose purpose is to return a list of entity ID's in that class) is defined as follows:

[*Collection of VehicleIdentifierData* contacts] =
GetContacts(*VehicleIdentifierData* MyVehicleID, *VehicleIdentifierData* VehicleID)

where "*VehicleIdentifierData*" is a simple class just containing an entity ID.

Procedures in the behaviour model just call this function as defined here. When that function is executed with the engine for JSAF, the implemented function that the JSAF engine executes calls other functions for "*getDatums", "getLobs", "getEWs",* etc. and combines their outputs into the contacts output for *"GetContacts"*.

When the function is executed by the engine for VBS2, the implemented function on the VBS2 side calls uses the *"nearestObjects"* VBS2 scripting function to construct a list of contacts and generates the output in the format of a collection of *"VehicleIdentifierData"*. This is depicted graphically below.

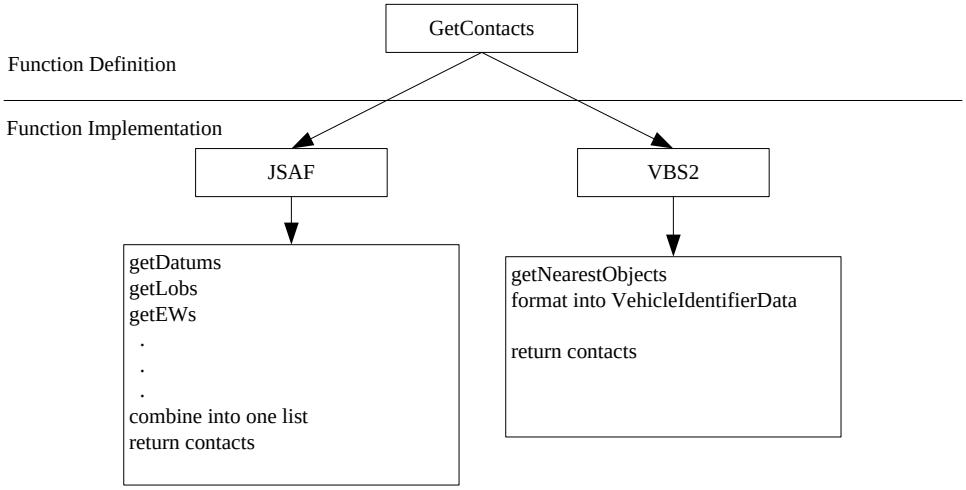

Figure 7. Example of Multi-Sim Functionality Implementation

The multi-sim functions we implemented in this project are limited to simple interactions for surface ships. Although there are only a few functions, end users can still build a variety of automated behaviours for surface ships that run in both JSAF and VBS2. Next step would be to add more functions and a set of functions for another entity type like Rotary Wing Aircraft (RWA) or Fixed Wing Aircraft (FWA). The Console currently includes the following multi-sim functions:

- GetContacts - Returns all entities within 8km of a given entity.
- GetEntityStateData - Return the state of a given entity.
- MoveOnHeading - Moves an entity on a given heading at a given speed.
- Distance - Calculates the distance between two points.
- GetDegreeOffset - Calculates the shortest angle between to degrees.

The development of this multi-sim console highlighted two main challenges for further development.

- The first challenge is in the design of simulation agnostic functions for this console that are general enough to be used by the supported simulation and yet specific enough for the Knowledge Engineer to use them for making useful models.
- The second challenge is then developing the simulation specific implementations that reside within each of the simulation specific Knowledge Service Engines.

Work continued to extend the number of available functions and behaviours that can be created in the multi-sim Console. Specifically, this work results in the ability to build RWA and surface ship behaviours. With this

capability, is it now possible to build the RWA and Surface Ship behaviours and execute them in both VBS3 and USN JSAF scenarios using the same behaviour file.

We added a few new integration functions in order to develop the behaviours required to operationalize RWA behaviours. The current functions available are listed in Table 1.

Table 1. Multi-Sim Behaviour Functions

Function	Return Type	Description
GetContacts(MY_VEHICLE_ID, EntityToGetContactsFor)	Collection of VehicleIdentifierData	Gets the contacts of the given entity
GetEntityStateData(MY_VEHICLE_ID, EntityToGetStateFor)	EntityStateData	Gets the state of the entity
LaunchHelicopter(MY_VEHICLE_ID, motherShipID, heloCallsign)	VehicleIdentifierData	Launches the first available helicopter on the given entity
MoveHelicopterOnHeading(MY_VEHICLE_ID, EntityToMove, direction, speed, altitude)	Void	Moves the helicopter on the specified heading at the specified speed and altitude
MultiSimCalculateHeading(MY_VEHICLE_ID, Location1, Location2)	Real	Calculates the direction from location1 to location 2
MultiSimConvertLatLonToLocation(MY_VEHICLE_ID, LLLocationData)	LocationData	Converts a latitude/longitude into a location data
MultiSimDistance(MY_VEHICLE_ID, Location1, Location2)	Real	Calculates the distance from one location to another
MultiSimGetDegreeOffset(FromDegrees, ToDegrees)	Real	Calculates the smallest radial distance in degrees between two angles
MultiSimTaskEntity(MY_VEHICLE_ID, entityToTask, MissionName, Inputs)	Void	Tasks an entity with the given mission using the given inputs.

These functions resulted in the addition of the following Basic Level Actions (BLAs).

RWA:

- Move on Heading - Moves along the specified heading at the specified speed and altitude. Inputs: Altitude, Heading, Speed Results: None
- Return to Mothership - Causes the helicopter to return to its mother ship location. Inputs: MOTHER_VEHICLE_ID Results: None
- Visually Identify Contact - If a Visual Identification Request is received, the helicopter moves to within visual detection range of a vessel and identifies the type/name of the vessel along with estimated course, speed, and nationality. Inputs: None Results: None

Surface:

- Clear Baffles - A ship clears its baffles by turning to the left for a short time, then back to the right for a short time, then back on course Inputs: None Results: None
- Helmsman Behaviour - a special BLA which allows the user to give voice commands to control ship movement.
- Launch Helicopter - Launches the first available helicopter found on the current vessel and labels it as a subordinate entity of that vessel. Inputs: None. Results: EV_RWA_Subordinate.
- Move on Heading - Moves along the specified heading at the specified speed. Inputs: Heading, Speed Results: None

ONESAF Integration Investigation

An additional task of this research was to conduct an initial evaluation of US DoD OneSAF(One Semi-Automated Forces)[2] using version 6 International, in order to estimate how long it would take to create a behaviour modelling console with OneSAF. At the time of writing this report it is possible to get entity information out of OneSAF but we cannot yet move entities. After some discussion with StackFrame we have been informed there is an updated version of OneSAF and the User Data Gateway UDG that should provide the capability to move entities. (i.e., the UDG does not have the hooks the appropriate OneSAF simulation calls to enable movement). In addition, the current UDG doesn't appear to have capability to get sensor contacts or entity perceptions. Out of this research we

[2] Name of the US PEO-STRI program to develop a single SAF for the US Army. The output application is commonly known as OneSAF.

have been able to effectively control VBS3 entities directly without this capability, although sensor contacts are heavily utilized in maritime and Air behaviours. This work identified has identified two things:

1. While a simulation application may internally have very detailed behaviour characteristics they may not all be available via an API or Interface application.
2. In order to achieve the dstl vision there may be a need for a tiered system of behaviours of low and high fidelity.

This is because not every CGF or simulation application is the same, and capable of implementing the same behaviour attributes. i.e., VBS3 doesn't have any detailed sensor models whereas JSAF and OneSAF do. This tiered system of behaviours could be based on the following:

- **Low fidelity** behaviours would allow the following with a multi-sim console across 2 or more simulation applications:
  - entity state queries
  - basic entity perceptions
  - simple movement options (speed, altitude, and heading changes)
  - simple use of weapons
  - control with voice commands

- **High fidelity** behaviours would allow this same set of functionality, but add the following for one particular simulation:
  - more detailed movement options (like using waypoints in VBS3)
  - detailed entity perceptions using simulation specific sensor models
  - control of individual sensors (like turn RADAR or ESM on/off in JSAF)
  - queries of individual supplies on each entity.
  - detailed use of weapons (like fore missile with appropriate range to hit target).
  - playing entity animations in VBS3

This tiered system would also allow simpler development of multi-security level classifications by changing performance data and or behaviour depending on the classification of that knowledge (i.e., an aircraft may turn slower) without the need to create a completely new behaviour definition from scratch.

Difference from Coalition-Battle Management Language (C-BML or MSDL)

C-BML's scope is to provide a capability to convey orders and commands to live, simulated and robotic forces (moves institute reference). Figure 8 depicts a C-BML concept for a divisional attack, it details tasks to the i.e., Blue-Mech_BDE1 to Attack, and BLUE-AVN-BDE to occupy. C-BML can be used to give the order, but as Figure 8 shows, it's the internal behaviour of the simulation that is then used to achieve the task.

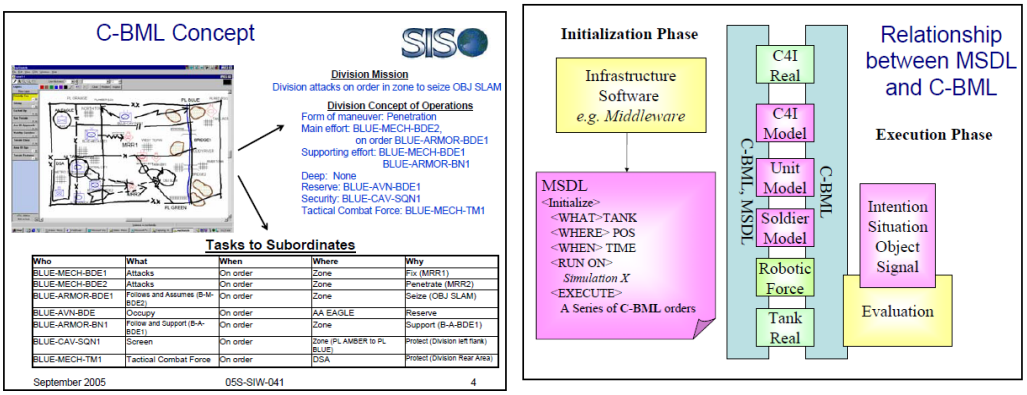

Figure 8. C-BML Concept

DISCUSSION / PROPOSED CONCEPT OF USE

A Concept of Use (CONUSE) for the MOD is that this approach could be used as part of a Defence Training and Education Capability (DTEC)[3] and a UK MOD Defence Simulation Centre (DSC) CGF service (as defined earlier) as a common repository for knowledge capture and behaviour definitions.

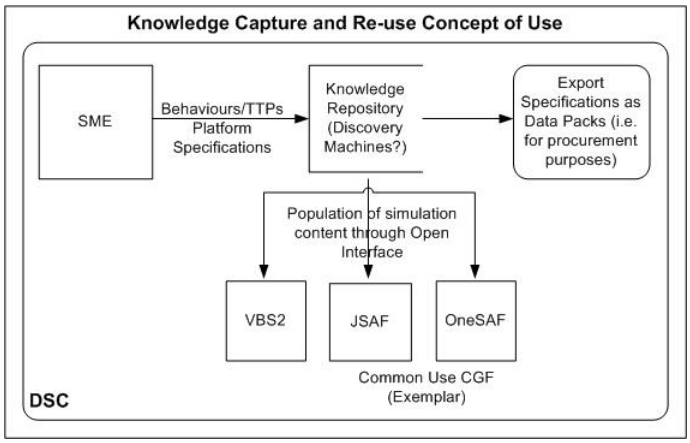

Figure 9. UK MOD CGF Knowledge Capture and Re-use CONUSE (Image Courtesy of dstl)

A generic Task Method Knowledge console could be used to develop a repository of Verified and Validated (V&V) (against UK Doctrine and with SME Input) CGF Behaviour Data Packs for a non-runtime CGF Service. The Knowledge Consoles would have the ability to output these V&V behaviours to pdf and text but could be developed to output in other formats (i.e., xml). Doing this would mean that MOD wouldn't be wholly dependent on one provider of a CGF, Runtime or Non-runtime CGF service in the future.

In order to build this repository under a DTEC DSC the MOD would require access to SME's (probably from the Single Service Warfare Centres) and also to train personnel in the TMK way of capturing behaviours. This would require significant investment in terms of time and access but would create a knowledge repository that can be used for CGF providers. It would require buy in from each of the Front Line Commands and the Defence Concepts and Doctrine Centre. This knowledge repository could then be used in one of a number of ways:

- Providing data packs to CGF providers to develop behaviours.
  - Would allow the MOD to independently V&V behaviours within a CGF application upon delivery.
  - Would require the DSC to re-issue data packs to vendors on change of data within a behaviour.
- Using these Consoles the MOD could create and develop simulation / CGF specific behaviours.
  - These could be developed within a MOD DSC CGF Service.
  - Configuration control would be easier and development of behaviours upon receipt of new data of data especially if only equipment performance changes.
- Developing a cross DTEC Common CGF Behaviour Authoring console.
  - Requires MOD to decided initially on DTEC Common CGF's and then develop a generic console.
  - Requires donor simulation to have a well-defined and functional API.
  - BCT option requires MOD to employ Software Engineers.
    (BCT is the Behavior Creation Tookit - a software development tool which provides software engineers the framework for developing a simulation specific behavior modelling console and engine)
  - Whilst tying MOD to the DMI products for the creation of behaviours would enable DTEC Common GCFs to be changed with only development of the Simulation functions in the DMI console to be

[3] https://www.gov.uk/the-defence-training-and-education-capability-programme

developed. Once done V&V behaviours could be compiled relatively quickly. Would also reduce procurement costs of the CGF if behaviours are already defined.

CONCLUSION

Although the Niteworks TTCD definition of a non-runtime CGF service is adequate at a governance level, in practice further definition should include

> "The capture, management and configuration of knowledge content (behaviours and performance parameters) and the creation of V&V Data Packs for behaviours".

These data packs can then be used to conduct V&V against CGF specific implementation of behaviours. However the level of investment in terms of time, money and personnel access required to create these behaviours is by no means minimal. In order to create a comprehensive cross-domain repository of behaviours a considerable amount of time and effort from within the MoD would be required to work through Doctrine and TTPs. This will require buy-in from all the Front Line Commands but the benefit to Joint and Collective training could be considerable in terms of V&V behaviours across simulation applications.

It is also apparent that whilst a common approach to behaviour representation and execution is possible, like terrain and 3D modelling, the nirvana of creating one behaviour that can interact will all simulation applications is unlikely to be achieved. This is due to the very different nature of each CGF or Simulation Application and its initial design purpose, as highlighted by the differences between JSAF, OneSAF and VBS2/3 in this research. The best one may achieve is a common repository of behaviours that can be centrally managed with different levels of fidelity across applications.

ACKNOWLEDGEMENTS

The authors would like to thank Dstl and the MoD CDE for funding the opportunity to explore this concept.

REFERENCES

Baxter, F (2013) Niteworks Training Transformation CCD Core Stage 3 Report.

Armstrong, S (2102) Niteworks Training Transformation CCD Core Stages 1 and 2 Report.

Lever, A (2012) Niteworks Ground Based Tactical Trainer CCD Final Report (GBTT-06).

Ministry of Defence (2010) Securing Britain in an Age of Uncertainty: The Strategic Defence and Security Review

Dstl (2012) Novel Systems for Defence Training and Evaluation CDE Call

Potts, R. etal (2010) Subject Matter Expert-Driven Behaviour Modeling Within Simulation Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2010 Paper No. 10357

Kriby, J (2006) Formalized Behaviours for Modeling Individual Combatant Weapon Firing in Military Operations in Urban Terrain (MOUT)

MOD (2014) Joint Service Publication 822, Part 6 The Defence Training and Education Capability (DTEC) Rules