A Reusable Simulation Environment for Digital Engineering

George H. Gilman
Naval Surface Warfare Center Panama City Division
Panama City, FL
george.gilman@navy.mil

ABSTRACT

Often when a simulation is constructed to represent a developmental system, it is targeted towards a single lifecycle phase of that system and is entirely discarded along with that phase. Whether for the purposes of simulating a separate system or a subsequent lifecycle within the same system, the next simulation carries forward only lessons learned of the previous iterations but little in the way of reusable software components. Each simulation in turn is created as a fresh new endeavor even when all simulations are fundamentally created upon the same common building blocks such as communications, time keeping, data management, coordinate system management, and event handling.

An example of this can be readily seen in the common scenario where a detailed engineering level simulation is used during the early phases of system development and as the system becomes fielded, a training simulation is required. Though the two simulations may require vastly different user interfaces and level of detail in specific models, they still share common fundamental building blocks and have obvious domain knowledge linkages. Even with the shared commonality, however, all too often the entire codebase is discarded and common components rewritten between developments.

The Multi Warfare Simulation Environment (MWSE) was developed at Naval Surface Warfare Center Panama City Division (NSWC PCD) as a reusable simulation environment that can cross the boundaries between software lifecycles and be further utilized to represent multiple disparate systems with minimal effort. At the core, this environment consists of reusable software components designed with industry accepted data and protocol standards in mind coupled with connections for domain specific plugin modules. This plugin concept allows a simulation developer to focus solely upon their own domain area while utilizing a wide range of previously developed libraries along with data and protocol standards via relatively simple API access. The environment facilitates rapid simulation development for new systems as well as selective reuse of lifecycle specific models of common systems.
In this paper, the MWSE development environment will be discussed in both its current state as well as future visions which will allow rapid development and simulation reuse.

ABOUT THE AUTHORS

George H. Gilman is a senior simulation engineer within the Modeling and Simulation Branch of Naval Surface Warfare Center Panama City Division (NSWC PCD). He has more than 28 years of experience in modeling and simulation and the mine warfare area and has been the technical lead on multiple simulation efforts at NSWC PCD. He holds master's degrees in Electrical Engineering and Computer Science from Florida State University.

A Reusable Simulation Environment for Digital Engineering

George H. Gilman
Naval Surface Warfare Center Panama City Division
Panama City, FL
george.gilman@navy.mil

INTRODUCTION

The first of five goals outlined in the Navy's Digital Engineering Strategy is to "Formalize the development, integration, and use of models to inform enterprise and program decision making" (Office of the Deputy Assistant Secretary of Defense for Systems Engineering [DASD(SE)], 2018, p.4). As further elaborated, this goal establishes development and use of models "across the lifecycle" of systems.

Though the terms model and simulation are often used synonymously, they are related but distinctly different. The purpose of a model is to represent a system of interest. It attempts to approximate the salient features of the real system within some threshold of acceptable fidelity. This could be a physical mock up, mathematical equation, diagram, or any number of other artifacts. A simulation in turn is the process of using one or more models to study the dynamic behavior and performance of an actual or theoretical system.

As a simple example, one of the simplest performance models for a mine countermeasures (MCM) system is known as an A/B table. Simply put, given that the sensor comes within range "A" of the object it is attempting to detect, there exists a probability "B" of the detection occurring. While providing a level of insight to the system's overall performance, only through exercise within a simulation can the dynamic effects of this model be realized. For example, through simulation, dynamic effects of sensor variation due to environmental conditions and platform behavior can give a greater insight to the overall system performance even when a system is modeled with an implied environment through a probabilistic data set.

As systems mature, the models representing them mature as well. The Set Based Design (SBD) paradigm being fostered within the defense community defers detailed specifications until trade-offs are more fully understood (Singer, Doerry & Buckley, 2009). In this vein, early models for systems developed under SBD rely upon generally abstract models which become more solidified as the process progresses.

Returning to the MCM example, if a new MCM were being developed under the SBD paradigm, the initial design space would include multiple platforms, both aerial and under water, as well as multiple sensors, including acoustic, magnetic, and LIDAR imaging. Within this design space, modeling at the most simplistic level, such as maximum range, provides metrics that can be evaluated equally across all systems to provide down selection criteria. Later, as system design matures, a more detailed model such as an acoustic propagation model can be developed and provide greater fidelity.

When simulations are developed, all too often a stovepipe approach is taken. At each phase of a system's lifecycle, a new simulation is developed and the old discarded. As previously discussed, models must evolve as a system design matures. However, the facilities required to execute the overall dynamics of the system do not. Models represent specific aspects of the system while the simulation provides for dynamic execution of the models. A simulation environment developed by extracting the common components such as communications, time keeping, data management, coordinate system management, and event handling, while providing hooks for model integration eliminates the need for stovepipe development. As a system is developed, the simulation can remain relatively unchanged while providing modularity for model maturity.

Along with providing system lifecycle support, a common simulation environment with the ability to "plug in" models also allow simulation development in multiple domains. Though a simulation involving a ground assault appears on the surface to be vastly different from an MCM mission, fundamentally what is required in the core of the simulation

remains. The passage of time is tracked and systems communicate with one another, move, interact with the environment and other entities, receive input from sensors and external stimuli and employ behavior. The models employed by the two simulations are widely different as the systems in each instance have different physical properties, behavior, and performance metrics. However, even with vastly different models, great commonality exists in the simulations.

NSWC PCD has developed a number of well pedigreed as well as lesser known simulations related to mine warfare (MIW), in both the mining and mine countermeasures (MCM) arenas. As with our peers, previously each new development carried forward additional experience and lessons learned but precious little code reuse. The goal in the development of the MWSE was to develop a single simulation development environment under which a variety of models are hosted to produce multiple simulation products.

DESIGN

The fundamental design decisions for MWSE focus on reuse, maintainability, and employing open standards. While not all choices are clear cut, a simulation framework was developed under C++ using Qt. C++ is a widely accepted language used across multiple platforms and standard compilers available from multiple vendors, reducing dependence upon vendor support. While Qt is a vendor specific product, it is also broadly used and provides great flexibility in operating system abstraction and creating cross platform software. Even with wide acceptance, the conscious decision to minimize Qt usage was made to focus it on areas only where its implementation is most valuable. Therefore, Qt is used extensively in the graphical user interface (GUI) development and for abstracting database access but not in the back-end simulation engine.

Enabling cross simulation communications is accomplished through AMIE (Naval Air Warfare Training Systems Division, 2016). This coincides with our developmental goals of developing to open standards as AMIE supports HLA, TENA, DIS, and other external simulation communication standards.

To provide reuse of the simulation, a plugin framework was developed. While the common simulation components are constant across installations, the plugin framework allows the insertion and removal of models as desired. This allows the developer to focus on developing models of interest rather than being burdened by developing the underlying simulation components.

Such a plugin framework provides the capability of establishing lifecycle specific solutions. One plugin can be developed for a system with very basic, low fidelity series of models during the initial concept phase. Later, as the system matures, the low fidelity models can be easily replaced either in whole or in parts by other plugins with higher fidelity modeling capability.

As seen in Figure 1, this plugin capability facilitates domain specific simulation development as well. The MWSE core components, including graphical user interfaces for scenario editing, network capabilities, mathematical libraries, and other foundational simulation components that the simulation developer can simply use off-the-shelf without the need for development. Instead, the developer can focus only the models specific to the domain subject of interest. Addition of the domain models creates a new simulation configuration or instance able to exercise the implemented models to determine dynamic system behavior.

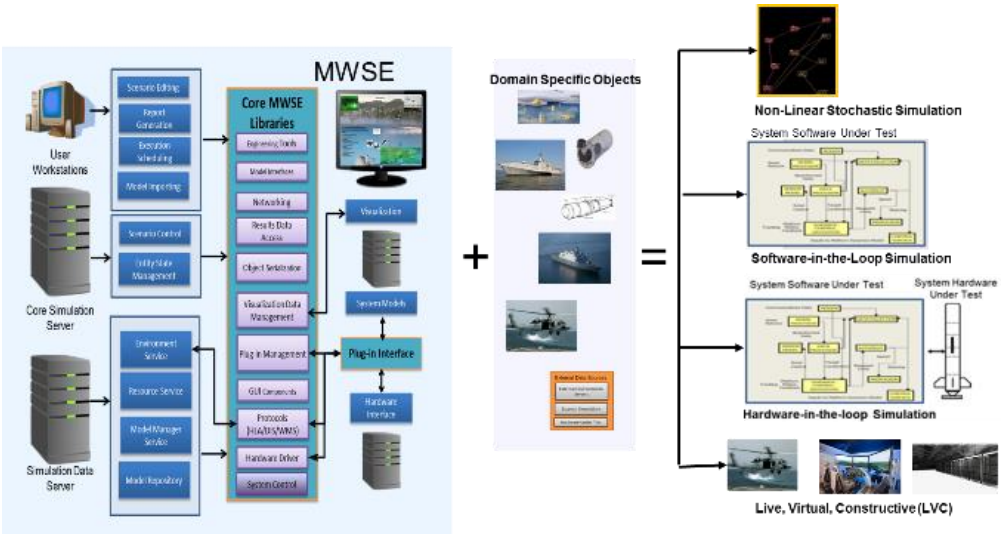

Figure 1. Combining Common Elements with Domain Specific Solutions to Create Individual Configurations

As an example, one domain specific solution may be developed for MCM operations. Within the simulation development environment, a variety of MCM platforms, tow bodies, unmanned underwater vehicles (UUVs), and other MCM related assets will be readily available for interaction with newly created simulation entities. However, another domain solution can exist for an aerial mine laying simulation. In this development, the aerial developer is not interested in creating interactions with MCM systems. Without the MCM domain plugin loaded, focus can be maintained on creating the behaviors and models pertinent to the aerial domain without distraction from MCM related components.

In both domains, reuse of the fundamental concepts provided by the simulation environment such as data management and time keeping simplifies development, while keeping each configuration from being distracted with behaviors and interactions in the other domain.

For plugin development, the simulation developer is provided a series of interfaces to which the plugin must adhere. As well, mechanisms are in place for the developer to advertise his capability to the simulation. For example, the developer can designate how a newly created entity can be deployed, whether as a single entity, within a grouping, or not at all. The developer also has the ability to use common simulation-based data editors for setting parameters or provide custom graphical interfaces to do the same. Primarily, these advertisements are used for providing the scenario editor with information on entity initialization as well as providing entity discovery to other entities for interaction management.

The plugin framework employed by MWSE provides for a better user experience with domain separation as well. As with providing the developer the ability to focus on a single domain, the end user is provided similar focus. Through addition and removal of plugins, the end user is presented in the GUI with only systems of interest to analysis needs.

COMMON COMPONENTS

Though the primary reuse and flexibility of the simulation is derived from the plugin framework described above, the environment would be of limited benefit without providing common facilities. It is through these capabilities that

rapid simulation development is achieved. A diagram of the software components can be seen in Figure 2, with descriptions below.

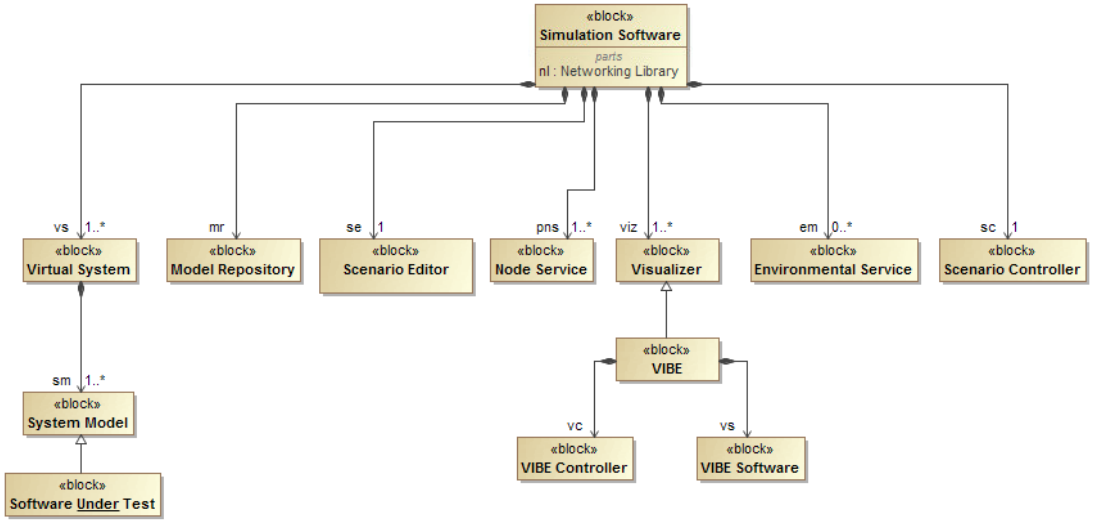

Figure 2. MWSE Software Components

Virtual System: This is the abstract concept for domain specific modeling. Interfaces and constructs are provided for incorporating domain specific models into a virtual system and supports purely virtual system models as well as software-in-the-loop (SWIL) and hardware-in-the-loop (HWIL) capabilities.

Model Repository: The purpose of this repository is to provide version-controlled simulation models upon request. The models will exist as plugins and are serialized to the repository. When an execution node has need for a specific model which it does not already maintain a copy of, a request will be made to the model repository server and the model reconstituted on the execution node. As well as providing a level of simplicity when installing models on multiple nodes, this capability also will provide for specific versions of a model to be executed, as designated by the user during scenario creation.

Scenario Editor: The purpose of the scenario editor is to provide a graphical interface for scenario creation. Entities can be created within the scene based upon which plugins are installed. If a plugin does not exist or is disabled, entities advertised within that plugin are not present within the editor, allowing for entire domains to be added and removed. The editor also provides graphical editing capabilities for initializing system performance data, as advertised within the plugin. This includes data management for the object instances within the scenario.
Node Service: The purpose of a node service is to provide simulation scalability. By spreading the processing required to model multiple systems and subsystems across processing resources, simulations with larger resource requirements can be executed. The node service is the software component which resides on each hardware execution resource and provides initialization and communication of system models.

VIBE: The Virtual Integrated Battlespace Environment provides a real-time Unity based 3D display capability for the simulation. The visualization provides important simulation feedback to the end user and can be displayed on a variety of hardware, including a single monitor, virtual reality headset such as the Oculus Rift, or multi-display CAVE environment. Though considered a core component of MWSE, all object state data displayed in the visualization is transferred over the network and retains only a very loose coupling with MWSE and has the ability to connect similarly to other simulation environments.

Environmental Service: MWSE hosts a central repository for environmental data containing a number of environmental models. The concept is to store the data in a model agnostic format with open standards. The environmental data service will then provide services for data extraction upon request. As a model requires environmental data, it can make a request from the environmental data service. The environmental service can then provide data in the model-requested format from either static tables or dynamically varying environmental models.

Through these mechanisms, models can receive data in a variety of formats without understanding the underlying structure.

Scenario Controller: The purpose of this component is to maintain centralized control during execution of a scenario. Scenario execution is initiated on the controller. It then determines, based on a variety of factors, how to load balance the work for the execution nodes. The scenario controller is also responsible for maintaining the current state of the simulation. This includes time and event management as well as knowing the state of all objects executing within the scenario.

As well as the primary software components, a number of facilities are also available to the end user. These consist in part of a series of math libraries, including unit conversions and coordinate system transformations, data logging, and report generation capabilities.

CURRENT STATE

The MWSE is currently under development at NSWC PCD. Though the system is not complete to design as outlined in this document, many of the MWSE components are developed and in place. The plugin framework, VIBE visualization, math libraries and networking capabilities are all developed and tested. Primarily the components still being developed are concerned with the distributed computing aspects. The simulation currently executes as a standalone application. Over the next year, distributed computing will be implemented with the addition of the node service and environmental data services. Lack of a distributed capability currently diminishes the ability to facilitate HWIL simulation, but SWIL capabilities are currently being developed.

SIMULATION INSTANCES

Although not fully realized, the MWSE has already been utilized to develop two fully realized simulation instances. The Naval Mine Warfare Simulation (NMWS) was developed in the mid-1990s as the Navy's primary mine countermeasures (MCM) analysis platform. NMWS is a full system-of-systems (SoS) simulation providing a wide range of MCM analysis capability. NMWS was redeveloped to eliminate a number of shortcomings and information assurance (IA) related issues with legacy NMWS's development tools. This redevelopment effort was the inaugural effort for MWSE. The current version of NMWS has passed accreditation efforts and is awaiting full accreditation status from PMS 420.

Seabed Warfare Simulation (SWS) was developed to fill the analytical role of advanced mining systems. It provides the ability to simulate future mining concepts such as distributed sensor fields and communication aspects. SWS was the second effort developed under MWSE and while the MWSE environment took a number of years to develop in support of NMWS, the primary capabilities of SWS were developed in a number of months. The rapid development was made possible by MWSE's ability to allow focus on developing only the domain specific models related to SWS while providing a large number of previously developed common components. Multiple mining domain specific models were developed under a single plugin to the MWSE for SWS.

SWS capabilities were demonstrated during a Live, Virtual, Constructive (LVC) exercise in the summer of 2018. The live components were exercised off the coast of Florida and included the deployment of both fully autonomous and human controlled assets. State information from the live assets was relayed via communication links to a shore based facility where virtual and constructive components executed roles within the exercise. The VIBE visualization provided a common 3D depiction of the scenario both as it was exercised live, as well as during subsequent replay events.

Along with the primary products developed under MWSE, additional capabilities have been demonstrated without being fully fleshed out. In quick turnaround efforts lasting of only a week or two each, capabilities for autonomous vehicle swarming and beach assault scenarios were examined.

CONCLUSION

As outlined in the DoD's Digital Engineering Strategy, systems should be modeled throughout their developmental lifecycles. Simulation provides the dynamic view of models and provides additional insight to system performance, but all too often the simulations are developed with a stovepipe approach and discarded along the development path. Within this paper, the concept of focusing on a single, reusable simulation environment removes the developmental focus from simulation to the domain specific modeling. Such an environment has been designed with MWSE. Though not fully implemented to design, MWSE has been utilized for the rapid development of a number of simulations and shown to have great value.

REFERENCES

Naval Air Warfare Training Systems Division, (2016). *Architecture Management Integration Environment (AMIE)*. Retrieved October 1, 2018, from http://www.navair.navy.mil/nawctsd/pdf/2016-AMIE.pdf

Office of the Deputy Assistant Secretary of Defense for Systems Engineering, (2018). *U.S. Department of Defense, Digital Engineering Strategy*. Washington, DC. Retrieved October 1, 2018, from https://www.acq.osd.mil/se/docs/2018-DES.pdf

Singer, D., Doerry, N., Buckley, M. (2009). What Is Set-Based Design? *Naval Engineers Journal, 121(4)*, 31-43.