Integrating Simulation-Based Training with Future Integration Training Environment (FITE) Assistive Aids

Matthew Franz, William Helfinstine, Matthew LeVan, Dale Moyer, Mark Torpey, Deborah Wilbert
Lockheed Martin
Rotary and Mission Systems
Andover, MA
matthew.franz@lmco.com, bill.helfinstine@lmco.com, matthew.d.levan@lmco.com, dale.moyer@lmco.com, mark.torpey@lmco.com, deborah.wilbert@lmco.com

Ryan Brown
Training and Education Command /
Training Support Center-29 Palms
Twentynine Palms, CA
ryan.s.brown@usmc.mil

ABSTRACT

Integrating legacy and evolving simulation systems complicates planning, creating, and executing training activities. Ensuring interoperability leads to technical problems requiring M&S experts to understand simulation protocols and bridge conceptual gaps between each system's implementation of the simulated world. Even with expertise, designing an interoperable system requires time-consuming and error-prone processes and often results in simplified solutions that bound system capabilities to that of the least-capable component. Focusing on integrating Marine Corps Air and Ground simulators at the Marine Air Ground Task Force Training Command (MAGTFTC) Battle Simulation Center (BSC), the FITE program has created FITEware, a tool suite improving the ability to execute simulation-based training activities using innovative tools and a flexible gateway architecture. Users focus on deciding which system capabilities and types of entities are needed while assistive aids determine the specific details of how each system will model those entities, identify protocol arrangements to link those systems together, and produce the complex configurations needed by simulation systems and gateways to provide fuller interoperable functionality. Further, these configurations can easily be adapted to subsequent similar training events, saving even more time and effort.

This paper details the methods by which this research has eliminated or reduced time-consuming and error-prone aspects of setup through automation and describes how the flexible gateway architecture maintains capabilities across integration. With FITEware in use, MAGTFTC BSC in collaboration with the 29 Palms Training Support Center (TSC) found that building a library of solutions to common simulation interoperability problems has improved their ability to execute historically difficult training activities. Furthermore, BSC and TSC found these tools offer more persistent interoperable training solutions and opportunities to home station units and provides a pathway to achieve similar results for other simulation systems in large-scale, service-level training events.

ABOUT THE AUTHORS

Ryan Brown is the Resource Integration Coordinator for USMC Training and Education Command's Training Support Center aboard the Marine Corps Air Ground Combat Center in 29 Palms, California responsible for the integration of Live, Virtual, and Constructive training resources to enhance Fleet Marine Forces training opportunities. Mr. Brown has supported USMC training efforts aboard MCAGCC for a decade in both industry and government roles through modeling and simulations, training innovation, applied research, and technological integration.

Matt Franz is a Staff Software Engineer at Lockheed Martin and the lead developer of the ONR FITE program. Matt has over fifteen years of experience developing training software and integrating distributed systems across the US Navy, Marine Corps and Air Force. Matt graduated with a B.S. in Computer Engineering from the University of Delaware.

William Helfinstine is a Senior Staff Software Engineer at Lockheed Martin and led the design and implementation of the ONR FITE bridge architecture. Bill has over 27 years of expertise in the modeling and simulation domain supporting a wide range of customers. Bill has extensive HLA experience, and maintains the RTI-s HLA

implementation for the US Navy. Bill received his B.S. in Computer Science from the Massachusetts Institute of Technology.

Matthew LeVan is a Senior Software Engineer at Lockheed Martin with over twenty years of experience working on constructive simulators, during which his principle focus was developing network interoperability solutions. He graduated from Vassar College with a BA in Computer Science. Matt's primary focus on ONR FITE was the design and implementation of the protocols and translations within the FITEware gateway.

Dale Moyer is a Staff Software Engineer at Lockheed Martin who led the ONR FITE Topology Solver focused research. He has over seventeen years of experience developing for constructive simulations on a wide variety of subjects including simulation architecture design, algorithm design and implementation, optimization, sensor and C4I modelling, and autonomous and semi-autonomous behavior design and implementation. He graduated from Worcester Polytechnic Institute with a B.S. in Computer Science.

Mark Torpey is a Lockheed Martin Associate Fellow and Principal Investigator at Lockheed Martin Rotary and Mission Systems (RMS) and is the PI for the ONR FITE Simulation Battlespace Services research. Within RMS's Advanced Simulation Centers, Mark leads the Constructive Simulation and Interoperability Lab, where his research has included distributed simulation, gaming and web technologies, immersive and augmented reality, and modeling and simulation interoperability. Mark received M.S. and B.S. degrees in Computer Science from the University of Massachusetts at Lowell and has over 26 years of experience in modeling and simulation research and development.

Deborah Wilbert is a Senior Staff Engineer at Lockheed Martin who led the ONR FITE enumerations-oriented research. Deborah has over thirty years of experience in the field of distributed simulation with particular interest in the area of graphical interfaces for systems management and configuration. Deborah holds a Bachelor of Science degree in Computer Science from the Massachusetts Institute of Technology.

Integrating Simulation-Based Training with Future Integration Training Environment (FITE) Assistive Aids

Matthew Franz, William Helfinstine, Matthew LeVan, Dale Moyer, Mark Torpey, Deborah Wilbert
Lockheed Martin
Rotary and Mission Systems
Andover, MA
matthew.franz@lmco.com, bill.helfinstine@lmco.com, matthew.d.levan@lmco.com, dale.moyer@lmco.com, mark.torpey@lmco.com, deborah.wilbert@lmco.com

Ryan Brown
Training and Education Command /
Training Support Center-29 Palms
Twentynine Palms, CA
ryan.s.brown@usmc.mil

OVERVIEW

The Future Integrated Training Environment (FITE) program is an Office of Naval Research (ONR) Future Naval Capability (FNC) research activity. The objectives of the program are to enhance interoperability between existing and future United States Marine Corps (USMC) air and ground training simulations. The USMC has many simulation capabilities but most of them were never required or designed to be interoperable with each other, and therefore like many other simulation environments, present interoperability challenges when combined. The research presented here focuses on two specific interoperability challenges – *protocol interoperability* (e.g. are the sims speaking compatible languages and do they understand what each is saying), and *enumerations consistency* (e.g. is an M1A1 tank published by one simulator interpreted as an M1A1 tank in all others). There are many other interoperability challenges, with perhaps the most significant being synthetic environment (to include terrain), and other aspects of the FITE program focus on those other issues. Historically, protocol and enumerations interoperability are solved by a team of Modeling & Simulation (M&S) experts, leveraging a large toolkit of tools, standards, and significant expertise. Events that combine simulations are intentionally designed by experts, and there are well-defined processes (IEEE 1730-2010) that most large-scale environments follow. A desired outcome of this research is a capability that reduces or eliminates the need for significant M&S expertise and contractor support, leveraging assistive aids to help less-expert users configure and manage less-structured training activities.

Processes for configuring data to conduct interoperable exercises between heterogenous simulators are labor intensive, error-prone, and time-consuming (Dvorak, 2019). Working with staff at the Marine Air Ground Task Force Training Command (MAGTFTC) Battle Simulation Center (BSC) and Training and Education Command (TECOM) Training Support Center (TSC) 29 Palms, we identified processes that we believed could be improved with assistive aids. In addition, various shortfalls and concerns were highlighted relating to how one of these events would be configured, with a collection of various gateways and tools, and the strengths and weaknesses of those tools/gateways and even of the simulations themselves. In TALONEX 2-18, for example, MAGTFTC BSC and TSC-29 Palms designed the Tactical Integrated Training Environment (TITE) for command post exercise (CPX), battalion-level staff training. TALONEX 2-18 federated the AH-1 Full Flight Simulator (FFS), UH-1 FFS, and AV-8 FFS across the Aviation Distributed Virtual Training Environment (ADVTE) with Virtual Battlespace 3.9 and the MAGTF Tactical Warfare Simulation (MTWS). TITE used a series of bridges and gateways to integrate Distributed Interactive Simulation (DIS) protocol version 6 and version 7 with High Level Architecture (HLA) versions 1.3 and 1516 utilizing various federation object models (FOM). Furthermore, each federate stimulated its equivalent or associating operational system; most notably, the Command and Control Personal Computer (C2PC), Advanced Field Artillery Tactical Data System (AFATDS), Tactical Handheld System (THSv2), and Common Logistics Command and Control System (CLC2S). The scenario called for approximately (70) 3-D model variations and (100) munition variants. Thus, to accomplish this federation a simulation technician had to manually map every enumeration in every federate to a

The views presented in this paper are solely that of the authors and do not necessarily reflect the opinions of nor endorsement by TECOM, RTPD, TSC, MAGTFTC, or any DoD organization unless otherwise stated in referenced directives or publications.

master scenario enumeration list. Ideally, this list would match SISO standards, but there are disparate protocols and diverse model repositories for each, and often archaic training systems make this process difficult. The manual association of models and mapping proves time-consuming as the simulation technician is chained to each disparate, federate system's organic gateway capability; or, the simulation technician's own technical competency on each disparate system. The status quo requires each federate to conduct its own "parking lot" and procedural testing to ensure credible integration of systems. If, for example, a system's organic gateway can only conduct one-to-one pairing versus a many-to-one or one-to-many pairing, then precise enumerations require manual input to ensure that interoperability. To design and test interoperable training with the tactical fidelity, precise interaction, complexity, and depth in this training event, TALONEX 2-18 demanded (5) personnel commit five weeks to manual model mapping, testing, and rehearsal to complete. Due to the model and enumeration disparity across all federate systems and the labor-intensive process, however, TITE could only use a limited number of munitions and many 3-D models proved non-existent or inconsistent across system repositories. Ultimately, this leads to incomplete or limited training options in federated environments, does not allow for persistent application of many synthetic training designs, and builds an inherent distrust in the stimuli due to the federation "isms". The issue with Marine Corps synthetic training is not demand; rather the design and preparatory work proves more time intensive and exhausting than the Marine Corps' current simulation support contracts can provide on a persistent basis.

In response to the design pain of TALONEX 2-18, MAGTFTC BSC and TSC- 29 Palms attempted a less complex and more centralized approach to TITE events using a VBS-based only scenario through the use of dynamic filtering. Mitigating the entity count load on the VBS machines while keeping a constructive simulation level capability required the implementation of 59 filters across 25 VBS machines and the MAGTF Mobile Flight Rehearsal Simulator (MMFRS) using Joint System Protocol Analyzer (JSPA). While this design provided higher-fidelity to the training audience it lacked the inherent benefits of community-based training system use in the design, and demanded roughly the same amount of preparation time as TALONEX 2-18 due to the intensive nature of the required filters. TALONEX 2-19 only federated VBS 3.9, VBS Fires, and MMFRS with a total entity count of 1,700 across 25 machines; but it took roughly four weeks to ensure the design worked to its purpose, which again, proves unsustainable for persistent interoperable training. **Simply, while other services or organizations can throw support staff or money at these problems, the USMC cannot, and the BSC and TSC are looking for technologies that allow them to be more agile in servicing the training requests they are responsible for supporting in a consistent and persistent manner.**

FITEWARE RESEARCH AND INNOVATIONS

Our research first focused on construction of a next-generation gateway, and tools that enable less-expert users to configure and manage execution of a training activity using the gateway. A traditional protocol bridge often has two significant design characteristics that have some drawbacks. First, they often utilize a single internal data model that provides a superset of all supported external data models, and secondly, they tend to be designed around the concept that a "message in" equates to a "message out" (e.g. stateless to the extent they can be, ignoring protocol differences). While these assumptions allow many design simplifications, a significant disadvantage is they encourage a least common denominator effect that can result in degradation in the translation from one protocol to another. This least common denominator effect in its worst-case results in the entire system of simulations being dumbed down to the capabilities of the least capable component, for each aspect of interoperability (Ceranowicz et al. 2002). The approach we explored doesn't force a single internal data model, but instead retains the state in the native object model of each protocol, and then defines translations on the various forms of the state that convert between them. In this way, the characteristics and idiosyncrasies of one protocol don't affect others, as might happen with a single internal data model. More importantly, the marshalling and control logic that implements each protocol can reason upon the data in the

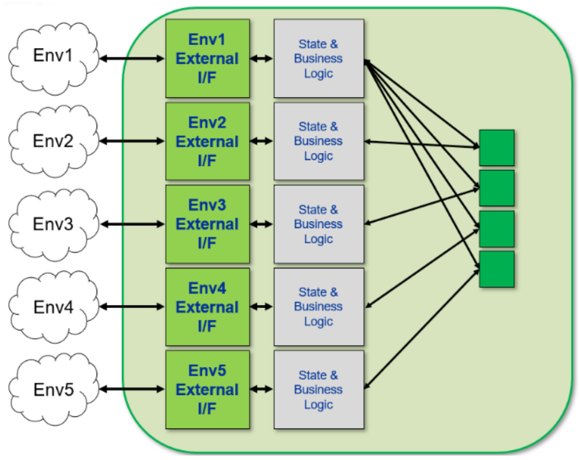


Figure 1: Configuration with a single internal data model that everything converts to/from.

format that it expects in an independent manner, so that subtle effects of complex protocol aspects like dead-reckoning, heartbeating, queries, thresholding, timestamps, etc. can be managed independently and correctly per protocol.

Through pairwise translation between protocols, we attempt to optimize how each pair of protocols are connected. In the extreme, this would require a significant number of translations, which not only requires effort to implement but also to maintain. To combat that, we enable transitive translation, so that only the specific pairs that require optimization need to be built, but any pair of protocols can be bridged as long as there is a path through other defined transitions. As these are in-memory conversions rather than network hops, the extra overhead of transitive translation turns out to be minor. For this research, support for DIS6 and DIS7, and HLA support for RPRv1, RPRv2, JLVC, DVTE-CAN, NTF, and a few others were implemented, along with pairwise translations between most of those protocols.

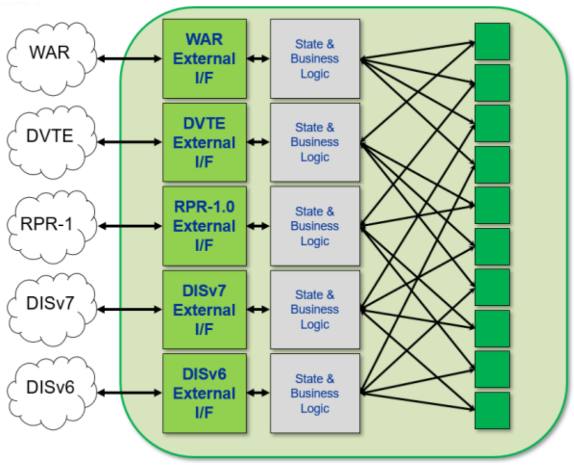


Figure 2: Configuration with every possible translation

The third significant trait of traditional gateways is that they are very complex to use, and therefore they are difficult to configure correctly resulting in poor or catastrophic results (e.g. creating a loop between gateways can cause a catastrophic crash of the entire system). Large scale environments typically have an EXCON (or white cell or tech control) that is staffed by experts who know how to configure these gateways, but less expert users are less likely to do this well. **With this new gateway, we set out to provide an easy-to-use graphical user interface (GUI) with assistive aids that enable users to more efficiently and accurately configure training activities**. The entire capability was designed to work in various types of deployments, ranging from laptop, desktop, or into a server/cloud environment. For example, the gateway can run as a headless daemon if desired, or it can have a terminal connection that provides more-expert users with the ability to inspect and affect the behavior of the bridge. Additionally, the bridge can be packaged up as a set of shared libraries (or DLLs on Windows) that can be linked into another application as middleware – providing that application all of the interoperability features natively. The primary user interface is a web browser, which provides the ability to manage a gateway from anywhere, run multiple GUIs, and simplifies the software installation/distribution process.

FITEware Control

The user interface was designed to be run in a standard web browser which has network accessibility to the FITEware-Control server. The flow of the interface walks a user through the typical steps required to define an executable configuration, supporting a range of user expertise – that is, less-expert users can accept the assistance the tool is providing and move from step to step, while expert users can drill-down to examine, tweak, or override. The goal of this workflow is to address the main interoperability challenges in a way that the tool can then configure the FITEware bridge properly. To facilitate the ability for the tools to make decisions, we analyzed the individual simulations, protocols, enumerations, as well as our historical expertise with constructing large-scale federated simulation environments. This led us to an approach of characterizing the simulation capabilities, the protocols and their nuances, and the capabilities of the bridge we were constructing as metadata that the tools can reason upon. The evolving metadata for protocols and simulations can be imported using the web page and stored in a centralized database on the server. As a user walks through the workflow on the web interface, the tools use this metadata to facilitate decisions about how the simulations will interact and what is required of the bridge, and that information is utilized to create the bridge configuration data.

Protocol Topology Solver

The Protocol Topology Solver is a component that computes an optimal full-connectivity solution between a set of application instances that may speak differing protocols and bridges that translate between those protocols. Once the solution is found, FITEware uses the solution to automatically generate bridge configuration files to implement it. Within the UI, the Build step allows the user to drop and drag applications and view the solver result. The user can then further modify the solution, and between each modification the solver re-runs and generates a new solution with the new user constraints.

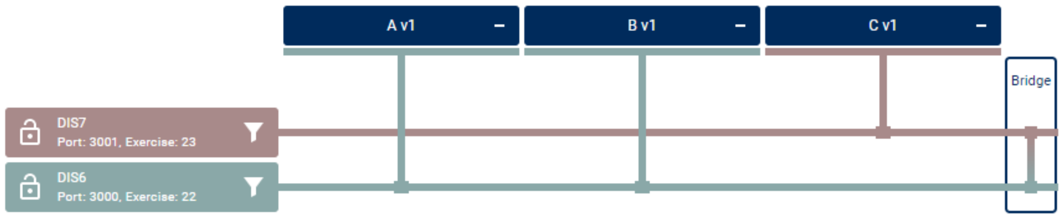


Figure 3: A solution with three applications (A, B, and C). The solver has decided that A and B can interoperate directly (using DIS6), and that the bridge must instantiate a translation between DIS6 and DIS7 to support C.

The Solver is aware of the limitations of each application. For example, two applications that speak the same protocol may still be incompatible due to use of differing dead-reckoning algorithms or enumerations. The solver can generate solutions that bridge the same protocol to translate across these difficulties. Human operators may further constrain the solution by forcing applications to use specific protocols, grouping applications to make them select the same protocol as each other, and forcing certain protocols to be present in the final solution.

Solutions can be quite complex – for example, Figure 4 shows a solution with five instances of four applications. The “E” symbols in the topology indicates that the solution considers enumerations compatibility for that application, which is what is preventing the first three applications from directly interoperating over DIS6. The lock icons on the bottom two protocol instances indicate the user has overridden configuration items for those protocols.

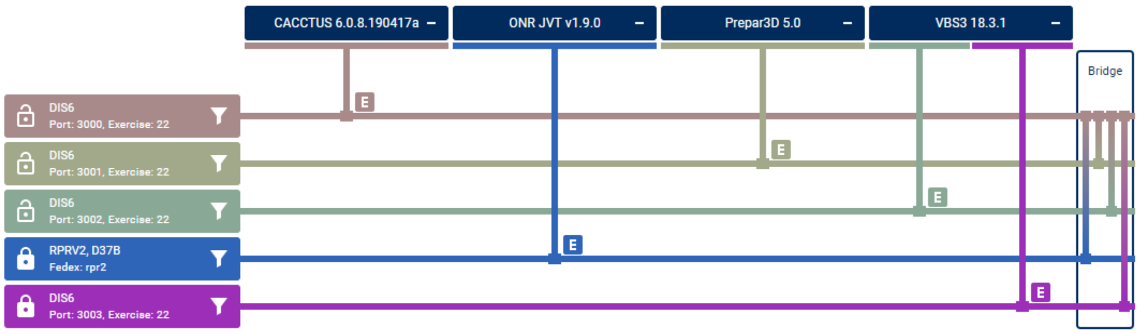


Figure 4: Users add desired simulations from a palette, and the Solver automatically produces a solution that combines them based upon their capabilities and the other simulations in the template, as well as the capabilities of the bridge itself. Users can further modify the problem and the solver will honor those changes, adjusting the rest of the solution as needed.

Generating a connectivity solution is a challenging problem typically solved by human SMEs. The Protocol Topology Solver simplifies and accelerates this process by generating optimal solutions for the human operator. To do so, each application protocol and translation is assigned a cost determining how completely it represents the information being communicated. This generates a cost between each pair of applications communicating. An optimal solution is one where the net cost of protocol selections and translation bridges across all pairs of application instances is minimized.

Figure 5 provides an illustration of solution costing. Application A communicates on the DIS6 protocol with a cost of 1, Application B communicates on DIS6 with a cost of 2, and Application C communicates on DIS7 with a cost of 1. A bridge that translates between DIS6 and DIS7 with a cost of 2 is in use. The total cost of the solution, then, is twelve; three for A to B, four for A to C, and five for B to C.

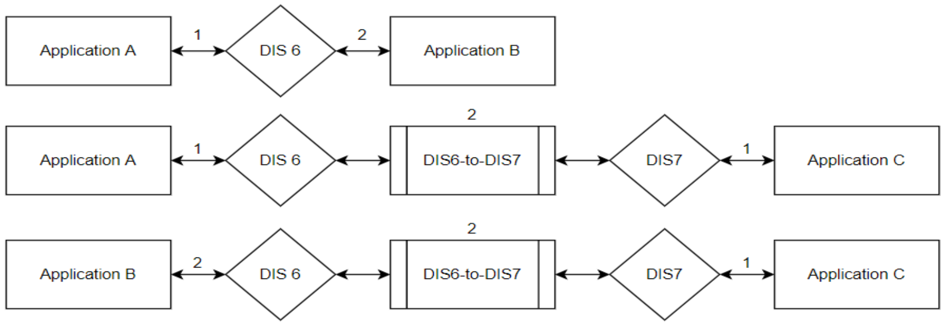


Figure 5: Costing of a Solution.

In addition to being a challenging problem for a human SME, however, connectivity solution generation is expensive computationally. Brute forcing possible protocol selections is approximately an $O(A^N)$ complexity problem where A is the number of applications and N is the number of protocols supported. Additional decision making and computation is required to find the optimal path of bridges between each pair of protocols; an exhaustive search would take time $O(B*P^2)$ where B is the number of translation bridges and P is the number of protocols. The solver must generate solutions rapidly for the user to provide a responsive GUI, so an innovative approach is required to accelerate solution generation.

The bridge pathing problem is the simpler problem of the two to solve, being of a lower order of complexity and operating on the relatively static data set of bridges available to translate between protocols. To resolve this problem, the solver runs Dijkstra's algorithm (Dijkstra, 1959) along each pair of protocols and caches the result. This allows for constant-time lookup of the shortest path between any two problems when generating a solution.

To solve the application protocol selection problem, the Protocol Topology Solver treats it as a pathfinding problem and applies an A* algorithm (Hart, 1968) to search the space. The A* works with partial solutions and assigns a heuristic cost when determining the expense of applications who have not yet been assigned a protocol. Given an application pairing where one or neither application has been assigned a protocol yet, the heuristic assigns a cost as if the unassigned elements chose the protocol that produces the lowest-cost result between those two applications. This ensures that the final solution reached will be optimal, as any actual selections made by the algorithm can produce a cost that is at best identical to the heuristic cost. The initial search node begins in a state where none of the applications have been assigned protocols. The algorithm then selects and removes the lowest-cost node available. An unassigned application is selected from the extracted node, and new nodes are created for each possible protocol selection from the assigned application. This process then iteratively repeats until a solution is found.

In summary, the Protocol Topology Solver allows for rapid, machine-assisted development of simulation interoperability solutions, a problem generally solved manually via human experts. The Solver creates an optimal, fully-connected network between a set of application instances, using protocol translation bridges. This solution is generated in two steps. First, Dijkstra's algorithm is used to determine the optimal path between each pair of protocols. Then, protocols are selected for each application using an A* search with a heuristic that assumes the protocols yet to be selected achieve the theoretical optimal solution where each application was talking to each other application via the best possible protocols. This allows for the rapid generation of the optimal connectivity solution.

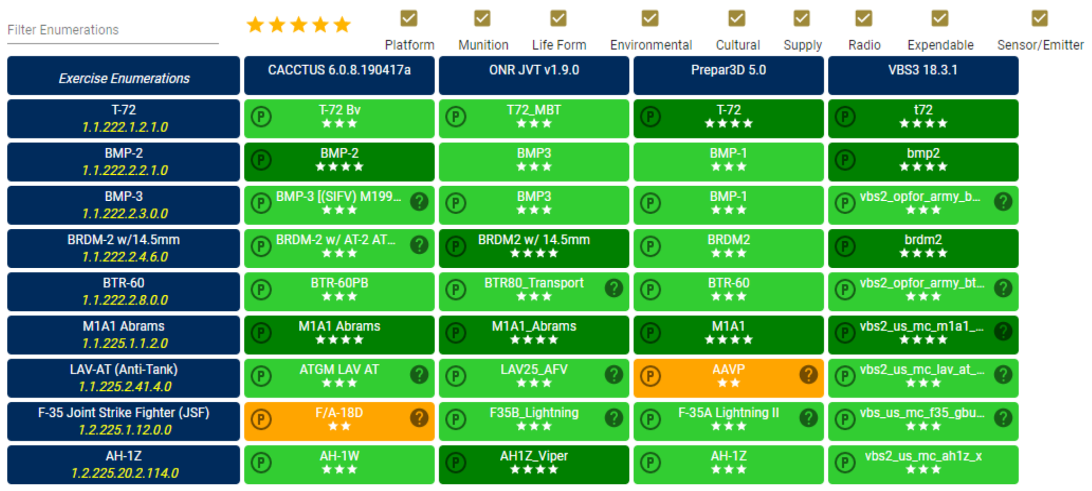
Enumerations SmartMap

One of the other common issues in interoperability between heterogeneous simulators is translating the meaning of enumerations used in each of the participating protocols. A gateway needs to be configured so that it can "map" foreign enumeration values to the closest supported value for each protocol. In particular, entity type enumerations are one of the key challenges that traditionally requires significant manual effort to configure effectively, and there are multiple places that must be configured in a consistent manner (including gateways and each of the simulations and their installed instances). Historically, this is managed using spreadsheets and a lot of manual effort, usually including "parking lot" validation (e.g. each sim publishes a scenario and experts would validate entities received by each of the other simulations match up to what is being published, and then manually setting up best mappings). **We solve this**

by characterizing various implementation details of enumerations as metadata and constructing an assistive aid that implements several heuristics that reason upon that metadata to produce automated best-approximation enumerations mappings, all in a centralized manner.

Our approach to gateway configuration for entity type translation centers around creation of an entity type list for each exercise template, which consists of a selected subset of the entity types imported by "references" (standards). This list is composed by the user by selection from dropdown tree structures of each reference, or re-use from a prior template, or by importing an external CSV spreadsheet. Conceptually, this list of entity types represents the set of entities expected to exist in an exercise that are considered to be valid, and it can include both specific (e.g. "M1A1 w/mine plows") and generic enumerations (e.g. "Tank").

FITEware supports importing two different formats of entity type enumeration references: the hierarchical "SISO" enumerations (Simulation Interoperability Standards Organization, 2020) in XML format and flat spreadsheets (CSV or XLSX). Each entity type in the reference imported is either assigned to an existing record (which is tagged as belonging to multiple references) or if no appropriate entry exists, a new record is created. Each entity type record includes a SISO-style septet assignment, even if septets are not inherently used by the imported reference. The septet provides information to help the enumeration assistive aid to choose the best entity type match. Entity type translation also requires entity type enumerations (which represent models) supported by each simulation. Simulations are also characterized and stored in the metadata. Scripts and import capabilities were written to "scrape" the list of entity type enumerations each simulation externally supports, which ideally includes a SISO-style septet and additional useful information, such as the human readable name for the models.

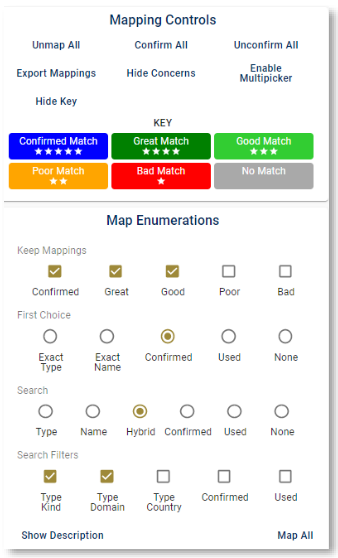


Exercise Enumerations	CACCTUS 6.0.8.190417a	ONR JVT v1.9.0	Prepar3D 5.0	VBS3 18.3.1
T-72 *1.1.222.1.2.1.0*	T-72 Bv ★★★	T72_MBT ★★★	T-72 ★★★★	t72 ★★★★
BMP-2 *1.1.222.2.2.1.0*	BMP-2 ★★★★	BMP3 ★★★	BMP-1 ★★★	bmp2 ★★★★
BMP-3 *1.1.222.2.3.0.0*	BMP-3 [(SIFV) M199... ★★★	BMP3 ★★★	BMP-1 ★★★	vbs2_opfor_army_b... ★★★
BRDM-2 w/14.5mm *1.1.222.2.4.6.0*	BRDM-2 w/ AT-2 AT... ★★★	BRDM2 w/ 14.5mm ★★★★	BRDM2 ★★★	brdm2 ★★★★
BTR-60 *1.1.222.2.8.0.0*	BTR-60PB ★★★	BTR80_Transport ★★★	BTR-60 ★★★	vbs2_opfor_army_bt... ★★★
M1A1 Abrams *1.1.225.1.1.2.0*	M1A1 Abrams ★★★★	M1A1_Abrams ★★★★	M1A1 ★★★★	vbs2_us_mc_m1a1_... ★★★★
LAV-AT (Anti-Tank) *1.1.225.2.41.4.0*	ATGM LAV AT ★★★	LAV25_AFV ★★★	AAVP ★★	vbs2_us_mc_lav_at_... ★★★
F-35 Joint Strike Fighter (JSF) *1.2.225.1.12.0.0*	F/A-18D ★★	F35B_Lightning ★★★	F-35A Lightning II ★★★	vbs_us_mc_f35_gbu... ★★★
AH-1Z *1.2.225.20.2.114.0*	AH-1W ★★★	AH1Z_Viper ★★★★	AH-1Z ★★★	vbs2_us_mc_ah1z_x ★★★

Figure 6: Enumerations mappings are managed in a matrix, with the valid entity types in the first column, and a row per simulation. The cells indicate the sim-specific model that represents a good match for that entity type. The bridge uses this to translate enumerations between the various simulators. Color/star ratings are assessments by the tool expressing quality of the guesses, which can be overridden by the user by clicking on a cell.

The gateway translations between simulation entity types are configured using the Enumeration Matrix (Figure 6) and it's assistive aid (Figure 7). The matrix displays one row for each entity type in the template's entity list and one column for each simulation in the template. From the user's point of view, each cell represents how the simulation will map that row's entity type to a model value, rather than attempting to map each simulation model directly to every other simulation model. The entity type row concept simplifies the task for users and allows them to focus on each simulation's mapping to the entity type, which results in only needing to make **n*m** mappings for n simulators and m entity types instead of **(n * (n-1))*m/2** mappings for pairwise translations, which is important when n is large.

The SmartMap assistive aid performs a configurable heuristic algorithm to suggest a mapping for each selected cell for which it can find a reasonable match. The algorithm by default will first choose mappings that have previously been confirmed as accurate, then perform a weighted hybrid search which factors in both names using a customized fuzzy string search and septet values, and filters those results to only select mappings which match by kind and domain. Which mappings to retain or remap, which mappings to prioritize, the basic search metric, and the exclusionary filters to apply can all be reconfigured and reapplied to (re)suggest mappings for any matrix cells. The hybrid search attempts to use multiple heuristics to cross validate each mapping, since looking at names or the septets alone often results in poor matches due to inconsistent approaches between simulations. To keep this task performant, best name and septet matches are lazily computed on the server by a set of worker threads, so that when the SmartMapper is run, it can operate on cached results in the browser.



Keep Mappings: [✓ Confirmed] [✓ Great] [✓ Good] Poor Bad

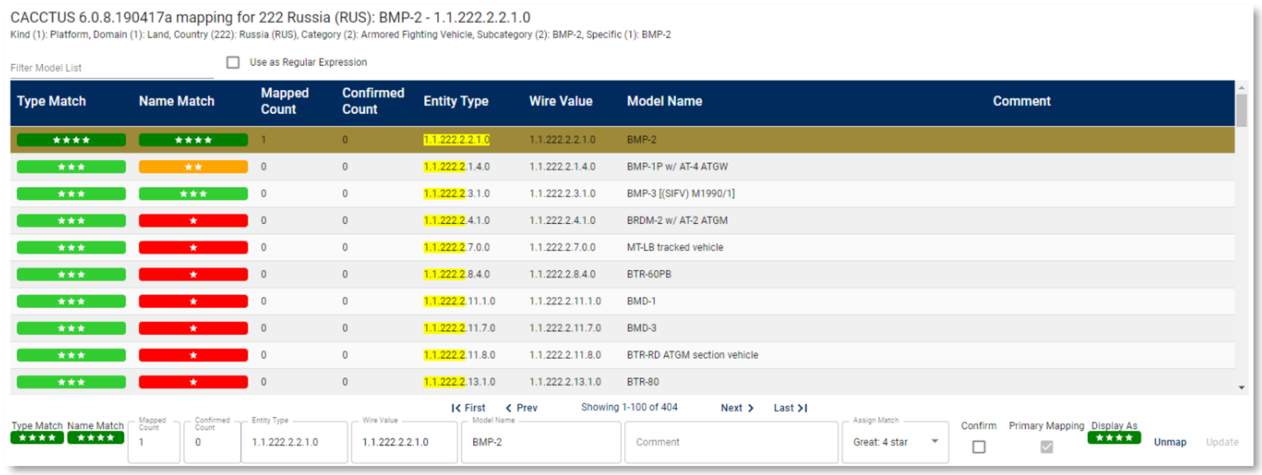
First Choice: Exact Type, Exact Name, [✓ Confirmed], Used, None

Search: Type, Name, [✓ Hybrid], Confirmed, Used, None

Search Filters: [✓ Type Kind] [✓ Type Domain] Type Country, Confirmed, Used

Figure 7: SmartMap Controls

The SmartMap aid rates the quality of each of the mappings it suggests, which is shown both as a color code and a star rating. The algorithm may leave poorly matching cells unassigned and for cells it does assign, it flags mappings that performed poorly by some metric, even if overall the quality seems good. Users can click on any cell in the matrix (Figure 8) to inspect, override, change quality, or express trust in the mappings, which will get utilized in future mapping suggestions.



CACCTUS 6.0.8.190417a mapping for 222 Russia (RUS): BMP-2 - 1.1.222.2.2.1.0

Kind (1): Platform, Domain (1): Land, Country (222): Russia (RUS), Category (2): Armored Fighting Vehicle, Subcategory (2): BMP-2, Specific (1): BMP-2

Type Match	Name Match	Mapped Count	Confirmed Count	Entity Type	Wire Value	Model Name	Comment
★★★★	★★★★	1	0	1.1.222.2.2.1.0	1.1.222.2.2.1.0	BMP-2	
★★★	★★	0	0	1.1.222.2.1.4.0	1.1.222.2.1.4.0	BMP-1P w/ AT-4 ATGW	
★★★	★★★	0	0	1.1.222.2.3.1.0	1.1.222.2.3.1.0	BMP-3 [(SIFV) M1990/1]	
★★★	★	0	0	1.1.222.2.4.1.0	1.1.222.2.4.1.0	BRDM-2 w/ AT-2 ATGM	
★★★	★	0	0	1.1.222.2.7.0.0	1.1.222.2.7.0.0	MT-LB tracked vehicle	
★★★	★	0	0	1.1.222.2.8.4.0	1.1.222.2.8.4.0	BTR-60PB	
★★★	★	0	0	1.1.222.2.11.1.0	1.1.222.2.11.1.0	BMD-1	
★★★	★	0	0	1.1.222.2.11.7.0	1.1.222.2.11.7.0	BMD-3	
★★★	★	0	0	1.1.222.2.11.8.0	1.1.222.2.11.8.0	BTR-RD ATGM section vehicle	
★★★	★	0	0	1.1.222.2.13.1.0	1.1.222.2.13.1.0	BTR-80	

Type Match	Name Match	Mapped Count	Confirmed Count	Entity Type	Wire Value	Model Name	Comment	Assign Match	Confirm	Primary Mapping	Display As
★★★★	★★★★	1	0	1.1.222.2.2.1.0	1.1.222.2.2.1.0	BMP-2	Comment	Great: 4 star	☐	✓	★★★★

Figure 8: Clicking on any cell in the numeration matrix pops up a dialog allowing finer control, override, and trust options. Users can search for a better match, choose a specific enumeration, override the Match Quality, and express trust ("Confirm"). The dialog has several helpers enabling the user to search through the list of models the sim supports.

SISO-REF-010 provides a machine-readable (XML) file of hierarchical "Comprehensive Entity Types" (CET), which provides a strong foundation for characterizing entity types, but it has limitations for identifying similarities in both hierarchical septets and names. One limitation is that the CET isn't truly comprehensive; users may construct valid entity types that are not enumerated in the CET. Users do exploit this flexibility to define their own references and for this reason, FITEware supports importing alternative references in a simple spreadsheet format.

The SISO-REF-010 hierarchical structure is divided into 7 fields (Kind, Domain, Country, Category, Subcategory, Specific, Extra), with the possible values of each field interpreted in the context of the preceding fields. These fields form a useful taxonomy for comparing similarity, with one major quirk. The Country field doesn't really characterize the inherent types of entities and for purposes of identifying entity type similarity, would be better as a separate attribute. As it stands, the Country field allows values to be assigned independently at lower levels. Different countries often, but not always, share at least some of lower level values. As a result, one cannot tell if a septet that contains all the same values except for country are very similar or completely different. The heuristics do factor in the country values, but essentially treat it as a separate attribute. This generally results in better matches when countries share the same lower level values but can produce less desirable septet matches when they differ.

More difficulties arise in using the CET for matching entity type names, particularly when trying to match reference entity types to named simulator models (rather than to other reference entity types). In the hierarchical CET, each node has a description which functions as a name. These node names are not unique (for example, many nodes are labelled as "Other"). While this uniqueness problem could be addressed by concatenating the names of all the septet nodes, this usually results in unwieldy strings that reduce overall similarity. Even individual node names can be unwieldy, for example the "*Patriot Communications Relay Group (CRG), AN/MRC-137 on M927A1 Truck, cargo, XLWB, 5-ton, 6 X 6, w/o winch*" makes it difficult to automatically determine how best to identify a matching model name. In fact, depending on the simulator or the exercise, the best match might focus on the AN/MRC-137 capability or the M927A1 platform.

In order to find good name matches, heuristics first preprocesses the strings in the CET as well as the model names, using both generic (e.g. remove dashes so that "T-72" will match "T72") and specific (e.g. remove "mysim_" prefix) rules. Then a fuzzy string match is performed to get a baseline measurement of similarity which is assigned a high confidence. This similarity value can be too low when names contain a lot of extraneous information, so the heuristics then try to identify important tokens (e.g. military model names) within each string, also factoring in length, whether the letters are uppercase, contains digits, or other characteristics. The heuristics then try to find those tokens or similar tokens, where similarity focuses on the beginning of the token because similar variants generally differ in endings in each name. The system then assigns a match value on that basis, with a lower confidence, depending on the number of tokens involved.

Our heuristics are the result of considerable trial-and-error experimentation, which do a good job in many cases, but do not eliminate the possibility of inaccurate or suboptimal mappings. Ultimately a human must judge which aspects of an entity type make for the best match and, for example, whether country alignment trumps platform details or whether a similar visual profile is more important than weapons capabilities. Therefore, mechanisms exist for the user to evaluate, override, an even express trust in the mappings – which are then factored in by future mapping suggestions.

The FITEware suite of tools automates a large portion of the enumeration mapping process, which has traditionally been an extremely time-consuming and error prone process requiring significant M&S expertise. Importing the simulation metadata in a centralized database eliminates the expertise and effort required to manually search simulation data files or sometimes even capture enumerations by driving or running simulations to see the entity types produced. Additionally, providing suggestions for mappings saves a great deal of monotonous data entry and eliminates the risk of syntax errors or typos. Further, by providing a centralized mechanism for the user to evaluate, override, and validate the results, work done on enumeration mappings in the context of one exercise can be leveraged in future exercises. As a result, the enumerations tools help mitigate the thorny interoperability problem of configuring entity types and their translations in a heterogenous simulation environment.

APPLICATION AT MAGTFTC BSC AND TSC- 29 PALMS

Training Environment

Effective training designs should create an environment that accurately represents conditions within the targeted operational environment (OE); thus good training should provide a structure that best enables units to train to specific mission sets, training tasks, or training objectives within the OE to which those objectives apply. The intent behind any interoperable event, therefore, should aim to incorporate the strengths each individual federate system provides while supplementing the shortcomings in the others.

The current fielded training systems provide high fidelity simulation systems for community specific and individual task training events, but it fails to account for the process and people for which these tasks will execute in a combat environment. A high-fidelity simulation for individual task or community specific training events does not necessarily translate into a high-fidelity training environment that intends to fully replicate the OE. Most Marine Corps communities possess a training system designed specifically for that community, so if integrated properly, could vastly increase the virtual training environments replication of the operational environment. Currently, however, the Marine Corps remains incapable to provide this valuable, synthetic training on a quick and consistent basis because technical intricacies restrict persistent plug-in-play use for interoperability.

FITEware demonstrated a leap forward to solve this issue. Any innovation should attempt to produce higher output from the same input or provide the same output with less input; but FITE produced higher output with less input. Currently, FITEware supports an array of USMC program-of-record training systems and simulation protocols. For example, to design, test, and execute a battalion-level training event using systems from RTPD LVC-TE Increment I (CACCTUS, DVTE, SAVT) would typically take approximately three weeks. To create this federation using FITE took one hour to build and accomplish the same ends. Furthermore, FITE proved timely and persistent in the federation for a typical interoperable event, but also provided greater fidelity in the volume and diversity of models it could map in that time. In a federation test in May of 2021, MAGTFTC BSC and TSC-29 Palms integrated (7) disparate constructive and virtual systems with (443) 3-D and munition model variations across a multi-protocol and multi-domain network using (5) disparate simulation protocols/versions. The test also used FITE's dynamic filtering to control data flow into each respective system/port to ensure or improve system performance and manage data flow. In its entirety, this federation took 3 hours to build, test, and rehearse. Simply, FITE proved it could provide more persistent and timely integration between training systems with higher fidelity in far less time.

FITEware Experience

Many limitations exist for federated training, but the most challenging regards scenario accommodation. Often, exercise designers must adjust training scenarios to accommodate the system federation whereas the system federation should support the desired training scenario and environment. Designers routinely adjust order-of-battle schemes or the intelligence preparation for the battlespace (IPB) to align with the training systems model repository; or they must manually apply an alternative model to accommodate the scenario. This limitation creates unrealistic training scenarios or introduces unrealistic tactical friction during execution. As such, exercise designers must analyze every 3-D model variation within a training scenario then identify which models every system can support. Designers then must determine alternative, yet different models to supplement the model gaps or change the training scenario to accommodate the common model repository for the federation. Once identified, simulation technicians must map each simulation protocol to match those existent or alternative models to each other; and contingent on the technician's system knowledge or the systems internal bridge or gateway capability determines how effective that federated training provides. The larger the scenario or the more individual systems you add to the federation, the more complex and time consuming the process; and bluntly, the less capable the federation becomes. To exasperate the problem, model repositories and training systems do not keep pace with the evolving OE that Marines and Sailors train to or the evolving industry standards that connects those systems. A recent model gap analysis conducted by Training Support Center 29 Palms exposed that all the high priority models (derived from each Marine Expeditionary Force and MAGTFTC training agencies) were marginally existent in the Marine Corps' most used virtual training system. Another example, the TALONEX 2-18 federation was only 30% capable of supporting the initially proposed scenario models. FITEware cannot fix this, however, FITEware quickly organizes, analyzes, and identifies those gaps within the scenario and allows the designer to quickly associate and assign model alternatives across all federates. FITEware's ability to automate this intensive and unavoidable process saves an immense amount of time and energy that supports higher quality training with more opportunity to train to complex problems.

User Intuition and Trust

A primary goal for FITE is to reduce or eliminate the need for significant M&S expertise and contractor support by leveraging assistive aids to help less-expert users configure and manage complex interoperable training events. To that purpose, FITE must possess an appropriate level of trust and intuition between user and gateway. Through multiple training events and federation tests, we observed FITE possesses an effective graphical user interface (GUI) that produced an overall high intuition with system interaction. FITE effectively leads the user through the federation design process to ensure the user accounts for all essential elements. We did observe, however, that each step required limited explanation as to what that step intends to accomplish and what each graphic represents. A couple features within FITE (e.g. the filtering capabilities and system monitoring) require adjustments to increase system intuition further. First-time users did struggle to understand FITE's conceptual model and architecture. Many approached FITE with the same methodologies that they approached in past federated events and struggled to understand the FITE paradigm in relation to the old.

FITE's innovation, as with any new innovation that shifts paradigms, requires users to understand that innovative approach. FITE does what many other gateways or systems like JSPA do, but it does it in a more efficient albeit different way. The hardest part for a first-time user is getting them to understand how FITE operates differently and getting them to apply it practically. In other words, navigating the program was easy to teach, but teaching how to practically apply it was harder. Using FITE requires users to approach the simulation design differently, so FITE's conceptual model is not intuitive in itself. Mainly, though, like anything new, users did not intuitively trust what FITE claims to do. This is rooted in decades of enumeration, model, and terrain mapping pain. In numerous tests, we observed every time a new system was added to the design or a new model was added to the scenario the users intuitively felt they needed to verify and test it every time; however, as first-time users interacted with FITE more you could see an inherent trust in its capability growing. Simply, time will build that intuitive trust in FITE's methodology and capability.

Current Limitations

FITE in its current state demonstrates incredible upside to progress the Marine Corps' synthetic training capability, but in order for FITE to provide a more holistic solution to the current interoperability woes FITE must expand its current brokering function. In order to replicate a realistic operational environment and provide appropriate stimuli to all respective systems and processes, FITE must bridge and broker to operational systems, respectively. Training systems should stimulate operational systems, and a substantial shortfall within current synthetic training regards the inability or partial ability to properly stimulate those C2 systems. If included and proven effective, then government program offices would no longer need to update each individual C2 plugin on each individual federate system, but rather update FITE so that it brokers to all other systems. Further, exercise monitoring and troubleshooting needs further development to make interactions between FITE, systems, and filters more streamlined. As effective as FITEware is now, FITE does not fix every interoperability problem within every system. At the very least, FITE solves the majority of problems in current interoperability methodologies, but the remainder still requires hands-on, manual attention. Therefore, dynamic and robust monitoring and troubleshooting is essential for timely, sensible, and maximum utility in the program.

REFERENCES

Ceranowicz, A., Torpey, M., Helfinstine, B., Evans, J., Hines, J., (2002), "Reflections on Building the Joint Experimental Federation," 2002. *Proceedings of the Interservice/Industry Training Simulation & Education Conference (I/ITSEC), 322, Orlando, November 2002.*

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.

Dvorak, J., Scrudder, R. Gupton, K., Hellman , K. (2019) "Enabling Joint Synthetic Training Interoperability through Joint Federated Common Data Services", *Journal of Cyber Security and Information Systems 7(3)*

Hart, P., Nilsson, N., & Raphael, B. (1968). *A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2)*, 100–107.

IEEE Standard Group 1278: Distributed Interactive Simulation (Revision 2002), IEEE CS Press

IEEE Standard Group 1516: High Level Architecture (Revision 2000), IEEE CS Press

IEEE Standard 1730-2010: IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP).

Simulation Interoperability Standards Organization (2020). *Reference for Enumerations for Simulation Interoperability, Version 28, document no. SISO-REF-010-2020, 7 May 2020*.