Achieving Distributed Training Through MSaaS: Results and Insights

Andreas Krupp, Máté Koch, Benjamin Labas
CAE Germany
Stolberg, Germany
andreas.krupp@cae.com, mate.koch@cae.com, benjamin.labas@cae.com

Jay Freeman, David Bisaccia
CAE USA
Orlando, FL
jay.freeman@caemilusa.com, david.bissacia@caemilusa.com

ABSTRACT

The modern battlefield is more interconnected than ever before, as evident with the conflict in Ukraine. Furthermore, the diversity in systems and the complexity of achieving coherent Command & Control (C2) is on the rise. To succeed, the need to truly master Multi-Domain Operations (MDO) for integrated deterrence has been widely recognized among Allied and NATO member nations. The key to MDO is to ensure a seamless collaboration and integration between units across all domains and echelons to achieve operational dominance against near-peer adversaries. Modelling & Simulation (M&S) is critical to enabling large-scale distributed training. Freeman, J., et al. (2024) presented our initial implementation of Modelling & Simulation as a Service (MSaaS). We extend and apply it to set up a distributed training use case utilizing a heterogeneous simulation environment across multiple nations and continents.

In line with the goals of Allied and NATO member nations, we create an ecosystem of components, composed using MSaaS concepts, to create distributed Live, Virtual and Constructive (LVC) training systems for various use cases. In our approach, all key services like composition, repository, messaging middleware, security and simulation services as well as front-end services are provided as containers and virtual machines. These are deployed both in commodity cloud infrastructures and on-premise. Virtual and live assets, located in Europe and North America, connect to this common synthetic environment by custom mediation services. Then various compositions and use cases are tested for feasibility and performance. Preliminary results are presented and are promising that distributed training using MSaaS concepts can be matured enough to really optimize training for MDO challenges. In this paper, we provide results and insights to promote further work in this area and to begin defining more detailed MSaaS concepts and standards.

ABOUT THE AUTHORS

Andreas Krupp works for CAE Germany as Group Leader of the Synthetic Environment group, leading an international team of software developers and architects as well as supporting business development in technical matters. He holds an MSc in Physics and Computer Engineering from Heidelberg University. Email: andreas.krupp@cae.com

Máté Koch works for CAE Germany as Innovation and Technology Lead, managing international teams involved in innovation, R&D and product development. He holds an MSc in Computer Science and Physics from Eötvös Loránd University, Budapest and a MIT xPRO certificate for Technology and Innovation Acceleration. Email: mate.koch@cae.com

Benjamin Labas works for CAE Germany as a Senior Software Engineer in the Synthetic Environment group. He holds a BSc in Computer Science and a MEng in Information Systems Engineering from the University of Applied Sciences Aachen. Email: benjamin.labas@cae.com

Jay Freeman works for CAE USA as a Synthetic Environment Technical Fellow and oversees several projects at CAE. Mr. Freeman previously served as the System and Software Architect for multiple DoD programs. He attended Hobart College for undergraduate studies and the University of Alabama in Huntsville for graduate studies. Email: Jay.Freeman@caemilusa.com

David Bisaccia works for CAE USA as a Software Architect. Mr. Bisaccia has worked on multiple DoD programs during his time at CAE where he served as a Technical Lead. He has a BS in Computer Science from the University of Central Florida. Email: david.bisaccia@caemilusa.com

Achieving Distributed Training Through MSaaS: Results and Insights

Andreas Krupp, Máté Koch, Benjamin Labas
CAE Germany
Stolberg, Germany
andreas.krupp@cae.com, mate.koch@cae.com, benjamin.labas@cae.com

Jay Freeman, David Bisaccia
CAE USA
Orlando, FL
jay.freeman@caemilusa.com, david.bissacia@caemilusa.com

INTRODUCTION

Achieving integrated deterrence is vital to Allied and NATO member nations to prevent and be prepared for a potential near-peer conflict. Key in this endeavor is the successful integration of forces of different echelons, domains and nations into a single cohesive unit. This maximizes the effectiveness of the available resources to ultimately achieve operational dominance against any threat. NATO describes this concept as Multi-Domain Operations (MDO).

While the concept is easy to explain and comprehend, putting it into practice is complicated and time consuming. The focus of this paper is supporting training use cases for individual units as well as larger formations. Everyone must train tactics and procedures on their own and collectively with other units to ensure they can perform in today's vast and complex operating environment. This requires small-scale force integration training as well as large collective training exercises. Furthermore, these need to take place regularly and consistently to ensure force readiness. Live execution is simply too restrictive and expensive due to logistical constraints and range limitations, which in turn presents a strong case for such training and exercises to be executed in distributed Live, Virtual and Constructive (LVC) environments with the extensive use of Modelling & Simulation (M&S).

Unfortunately, the creation of such environments is also plagued with its own challenges. Allied and NATO member nations use a vast amount of different and often proprietary (legacy) LVC assets, which prevents data interoperability and disrupts fair fight conditions in the simulations. Physical distance introduces network latency. Scaling up the system with more participants can overload networks and exacerbate these delays. Specific information, cyber security and classification requirements further complicate the networks and slow down achieving an authorization to connect and operate. Additionally, for the training to be effective, the real threat environment across all domains has to be credibly replicated by integrating a large number of simulation models and associated data.

One solution to mitigate some of these problems is to enforce strict adherence to standards and policies. This was attempted but never fully achieved because globally establishing such standards and policies for all Allied and NATO member nations is unrealistic. A major reason for this is because each nation and service utilizes unique implementations of their preferred standards and policies to quickly respond to operational environment demands. Similarly, maintaining a system through evolving needs and standards is key to keeping any distributed training environment operational in the long-term. But keeping all components updated and in sync all the time is impractical. Furthermore, problems like scaling and exercise orchestration require new technologies and operating concepts.

In prior work by Freeman, J., et al. (2024), we looked to Modular Open Systems Approach (MOSA) concepts and especially Modelling & Simulation as a Service (MSaaS). The NATO Modelling & Simulation Group (MSG) published the MSaaS Reference Architecture (RA) in 2019, providing valuable principles and concepts on how to address a lot of the outlined challenges. In this paper, we will expand on the practical implementation of the RA and explore suitable technologies for the different components in a true distributed mission environment. We utilized our resources, expertise, and experiences to improve and extend our implementation to LVC environments that are representative of those likely to be found in distributed training for MDO, validating the technology in real-world trials.

During this research, technology was developed and testing environments were set up to explore the feasibility and performance of our MSaaS implementation for the mentioned training use cases:

- Force integration training using two full flight simulators connected while retaining their own synchronized constructive environment.
- Collective training using a globally distributed set of virtual simulators connected to a common constructive environment.

Objective performance metrics were collected during various test sessions as well as feedback from testers and developers during setup and operation of the tests. The results and insights are transcribed in the relevant following chapters.

CREATING AN MSAAS ECOSYSTEM FOR DISTRIBUTED TRAINING

Summary of previous efforts

MSaaS is NATO's strategic initiative to modernize and enhance the accessibility, agility, and interoperability of M&S capabilities across the Alliance. Rooted in cloud computing and service-oriented architecture (SOA) principles, MSaaS aims to deliver simulation capabilities as on-demand, scalable services to a wide range of users, including military operators, analysts, and decision-makers. The Allied Framework for MSaaS bundles Operational Concept Documents (OCD), the Technical Reference Architecture (RA) and Standard Operating Procedures (SOP).

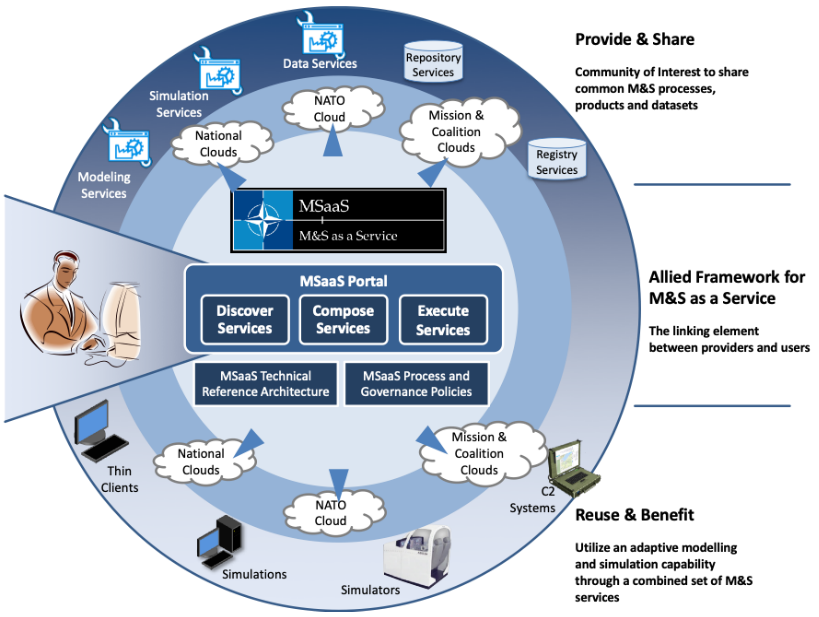
Data Services
Repository Services
Simulation Services
NATO Cloud
Mission & Coalition Clouds
National Clouds
Registry Services
Modeling Services
MSaaS
M&S as a Service
MSaaS Portal
Discover Services
Compose Services
Execute Services
MSaaS Technical Reference Architecture
MSaaS Process and Governance Policies
Thin Clients
National Clouds
Mission & Coalition Clouds
C2 Systems
NATO Cloud
Simulations
Simulators
Provide & Share
Community of Interest to share common M&S processes, products and datasets
Allied Framework for M&S as a Service
The linking element between providers and users
Reuse & Benefit
Utilize an adaptive modelling and simulation capability through a combined set of M&S services

Figure 1: NATO MSaaS concept illustration

The MSaaS RA itself is a layered architecture that consists of a static structure of different types of Architecture Building Blocks (ABB) which can be implemented differently for different purposes and use cases. If adhering to the prescribed Architecture Patterns (AP) and Service Level Agreements (SLA), these ABB implementations can then be composed as required and thus form a dynamic library of simulation components. For more details refer to NATO Science & Technology Organization (2019).

With our prototype we tried to prove the feasibility and applicability of modern commodity cloud and web technologies for M&S and its use cases. The following table lists the key technologies chosen for the respective ABBs.

Table 1: Technologies selected for key ABBs in our MSaaS implementation

MSaaS Architecture Building Blocks (ABBs)	Description	Technologies Employed
Composed Simulation Services	Totality of Simulation Services	vSphere, Kubernetes, VR Forces, CAE GESI, and CAE STRIVE
Core, Communication and SOA Services	Range of functionality enable basic system support	GeoServer, PostgreSQL, KeyCloak, vSphere, Kubernetes, API Gateway, LVC Gateway
M&S Certification Services	Verify compliance to interoperability standards	Apache Kafka + custom connector
M&S Composition Services	Compose and execute simulation services	API Gateway, vSphere, Kubernetes and custom functionality
M&S Mediation Services	Transform/Translate data	Apache Kafka + custom connector
M&S Message-Oriented Middleware (MOM) Services	Efficient and time-coherent exchange of data	Apache Kafka + custom SDK
M&S Registry Services	Metadata for simulation resources	Custom implementations and APIs
M&S Repository Services	Store, retrieve and manage simulation resources	Federated repositories and customer functionality
M&S Security Services	Enforcement of security	Keycloak + custom implementation for Zero Trust
Modeling Applications	Front-end functionality for accessing the Modeling Services	Thin client: OpenLayers & Cesium.js in Angular micro frontend architecture Thick client: SitaWare, CAE Ridge, CAE Prodigy
Modeling Services	Suite of tools needed to construct a Simulation Service	Not implemented yet
Scenario Services	Creation & management of scenarios	Custom C2SIM implementation
Simulation Applications	Front-end functionality enabling users to interact with Simulation Services and Composed Simulation Services	Thin client: OpenLayers & Cesium.js in Angular micro frontend architecture Thick client: SitaWare, CAE Ridge, CAE Prodigy
Simulation Control Services	Logging and input for simulation execution	Custom implementation and APIs
Simulation Services	Individual simulation engines	VR Forces, CAE GESI, and CAE STRIVE

Initial results were very promising and largely validated the choice of technologies. Especially the use of Apache Kafka, an open-source event streaming platform that has found widespread adoption in cloud computing, combined with some M&S-specific additions for messaging middleware and mediation services proved to be quite efficient and scalable to move data between simulation and other services. While also enabling interoperability by providing ways to translate and adapt data as needed between those services. For detailed results and other lessons learnt, please refer to Freeman, J., et al. (2024).

Key enablers for a composable M&S ecosystem

Having a set of validated key technologies is a good first step. But being able to set up large-scale LVC exercises regularly and consistently in day-to-day operations across units, nations and even continents requires additional enablers. Hence the next logical step is to identify and implement those enablers to move from technology demonstrator to a system that enables some level of distributed training.

By nature, for this use case the services of the MSaaS ecosystem need to be deployed and managed in various locations and compute infrastructures. To abstract away this complexity, all services should be as much as possible deployed as containers and virtual machines in managed cloud environments like Kubernetes and OpenShift, which can be on premise or self-hosted, or those provided by Microsoft Azure, Amazon AWS and others. Furthermore, Infrastructure as Code (IaC) should be used to automate setup and config management of the cloud environment and networking. This allows to provision services on the fly, execute multiple exercises at the same time, roll out updates in a controlled and predictable manner and reduces dependency on specific hardware and providers. We followed these principles by having our core services available as prebuilt containers together with deployment scripts and IaC artifacts for OpenShift/Kubernetes platforms.

While it is acceptable for irregular one-off exercises to have personnel manually manage exercises and orchestrate the different simulation systems and services, this is too much effort and too unpredictable for regular operation. This means that registration and orchestration services (simulation control services) need to be implemented. Registration services allow the operators to see which systems and services are available, see their status as well as live performance metrics for monitoring. Orchestration services allow to create different exercises and connect systems and services to these exercises. Additionally, they allow to control the simulation execution and state machine (i.e. start, stop, pause). Both kinds of services were implemented as custom containerized applications with Application Programming Interfaces (API) that cater for both backend and frontend services. These services keep a database of the currently active exercises, all available services, their status and which exercise they take part in. Exercise management and simulation control is also handled through these APIs. Communication with the rest of the system is handled through the messaging middleware. Frontends were developed in the form of web-based and desktop applications that can set up and control exercises as well as show status, metrics and tactical situation.

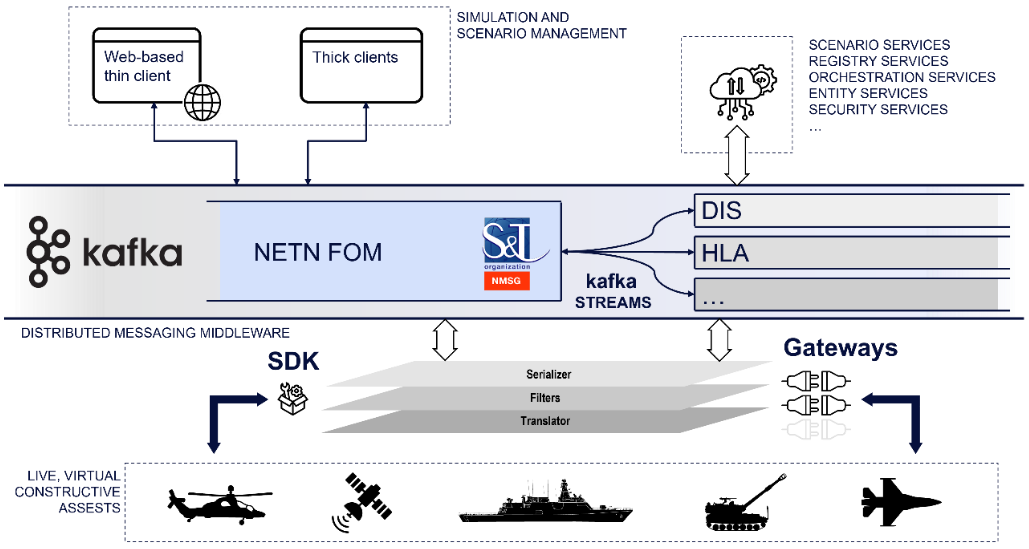

Figure 2: Illustration of our Kafka-based messaging middleware together with systems connected via the SDK or gateways and mediation via Kafka Streams

Since we are likely to deal with several dissimilar simulation systems and services, it is imperative that onboarding those systems and services to the common messaging middleware is as easy and efficient as possible. Thus, excellent documentation, an easy-to-use Software Development Kit (SDK), and the availability of ready-to-go gateway applications for commonly used protocols (e.g. Distributed Interactive Simulation (DIS) or common High-Level Architecture Run-Time Infrastructures (HLA RTI)) are essential. To achieve this, a custom SDK layer was added on top of Apache Kafka. This layer abstracts away much of the low-level complexities of dealing with Kafka and provides higher-level functionality specific to M&S. The user does not need to deal with which communication channels (Kafka topics) to subscribe to for a given exercise. Communication with the orchestration and registration services is taken over by the SDK and many helper functions are provided. Additionally, the SDK was used to create a set of gateway applications, e.g. to take DIS traffic and push it through Kafka and vice versa. To further support developers, an extensive set of documentation including many examples and tutorials was also created.

In a larger distributed federation, it is almost guaranteed that not all simulation systems and services will communicate using the same data model (e.g. HLA Federated Object Model (FOM) or DIS version) or even the same protocols. Hence it is imperative to from the beginning assume that every exercise participant will need some kind of data mediation to properly integrate with the simulation environment. This approach also allows for phased migrations to newer data models and gradual onboarding and upgrading of simulation systems and services. A two-tiered approach was implemented. The first tier is Kafka Streams. This is a mechanism to deploy applications that stream messages from one Kafka topic to another, enabling transformations like filtering, aggregation, data translation or data cleansing on the fly. In our implementation, the NATO Education & Training Network (NETN) FOM is the canonical message format and therefore the standard for services to exchange data. Other output channels (or topics in Kafka speech) are provided via Kafka Streams applications, e.g. to native DIS and other HLA FOMs. The messaging middleware SDK allows users to then receive messages in the desired format, e.g. DIS 7, directly and communicate with other services using the same format without conversion. All transformations are handled centrally and only once. The second tier happens at SDK level, where additional filtering and data transformations can be applied. This is intended for e.g. small tweaks to the message format and further filtering to conform to service limitations or security requirements.

Detailed test setup

To achieve MDO capabilities, Allied and NATO member nations define the need for two different kinds of training. Force Integration Training (FIT) happens on smaller, tactical level with select units from different nations. The focus is to enhance interoperability and coordination on company to battalion level. Examples being Griffin Lightning 25, where helicopter forces gathered in the Baltics, or an exercise between a German logistics unit and a Polish infantry unit. Collective Training (CT) takes this one level further and aims at the larger, strategic level with full units or formations. It often involves multiple entire brigades and divisions of several domains and focuses on combat readiness and mission rehearsal for interregional or global conflict. Here an example is Steadfast Defender 2024 that combined a series of exercises all over Europe to train defense plans for a hypothetical Article 5 response.

Today, both kinds of training are mostly conducted in the real world, with all the planning, logistics and cost overhead this entails. In the future, M&S will need to play a much more prominent role here to lower the barrier of entry and enable more frequent and regular training. To gather insights into the suitability of our developments for these use cases, different test setups were created, relevant scenarios were executed and metrics were collected.

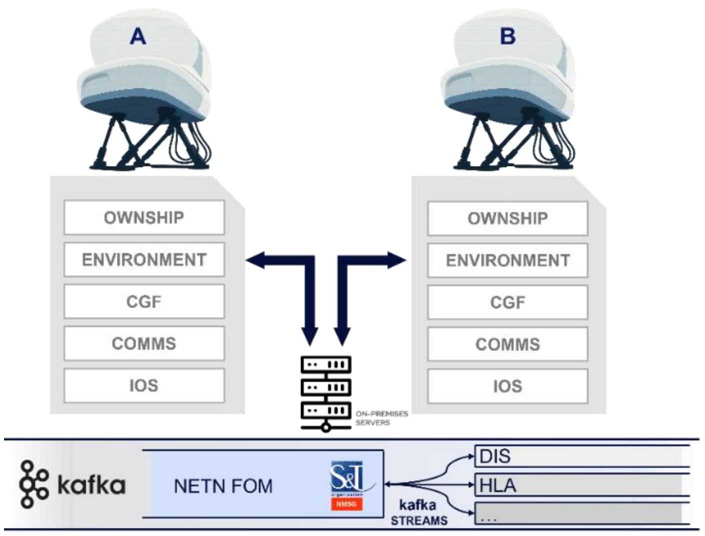

Figure 3: Our test setup for force integration training, connecting two full flight simulators via on-premise infrastructure

Representative for high-fidelity Force Integration Training, two NH-90 Sea Lion full flight helicopter simulators were connected to train formation flying and coordinated sensor use in a tactical environment. The constructive simulation services, including terrain, weather, computer-generated forces (CGF) were provided by CAE's STRIVE constructive simulation environment and hosted with every virtual simulator individually. Synchronization, state information, voice communication and other simulation traffic between the two simulators were exchanged fully via the Kafka-based messaging middleware. Instead of using a gateway approach, the simulation services of each device were connected directly to Kafka by custom connector plugins developed using the messaging middleware SDK. The Kafka infrastructure (brokers) and MSaaS registration and orchestration services were deployed on the virtualization host of one of the simulators. While both simulators were located within the same building, their networks were segregated into their own VLANs with just the Kafka traffic allowed between both systems. This setup could be used e.g. to run training between different helicopter units hosted by one of the participants with otherwise very much standalone systems.

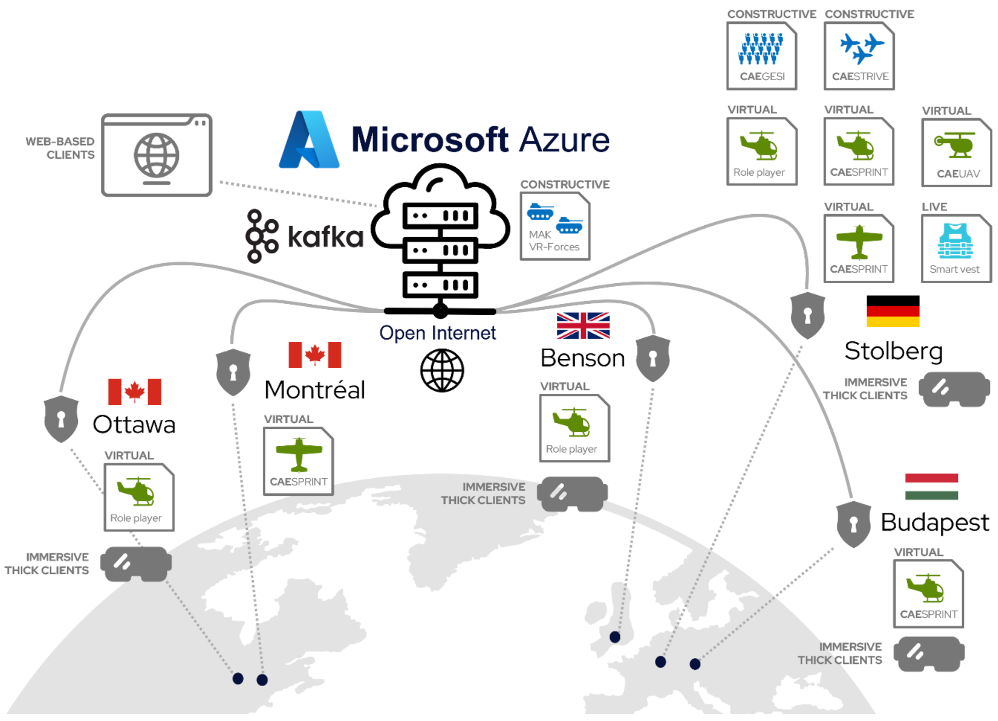

Figure 4: A test bed for collective training with several sites and LVC assets connected via gateways

A Collective Training use case was achieved by a joint global effort, connecting several simulator laboratories together into a single training environment. The laboratories were located in Canada, Germany, United Kingdom and Hungary. Constructive simulation services, the MSaaS backend (security, registration, orchestration, mediation, messaging middleware) as well some of the frontend services (web applications) were hosted centrally in the Microsoft Azure public cloud. For the constructive environment CAE STRIVE was used to provide most of the air picture. It was augmented by MAK VR-Forces and CAE GESI for land and naval entities. Virtual simulators of various types were integrated, among them Virtual Reality (VR) helicopter and fixed-wing simulators based on X-Plane (a commercially available flight simulation software), Mixed Reality (MR) helicopter and fixed-wing simulators based on CAE's Sprint family of devices plus a Remotely Piloted Aircraft System (RPAS) based on CAE's Unmanned Aerial System (UAS) stack. To add a live system, a wearable sensor vest for infantry soldiers was also connected. Network connectivity was provided by VPN connections from each site to the central cloud-hosted services. Various compositions and scenarios were tested, e.g. a search & rescue scenario involving fixed-wings and RPAS providing search capabilities, helicopters providing extraction and rescuers and people in distress on the ground as a mix of constructive and live entities. This setup closely resembles a situation where a central facility provides the simulation environment with multiple different exercise participants connecting remotely to this environment.

PERFORMANCE TEST RESULTS AND TECHNICAL FEASIBILITY

Before looking at more subjective observations, first the objective performance and technical feasibility of our MSaaS implementation can be assessed by analyzing the performance metrics collected during the testing. Using Kafka for messaging middleware and mediation services provides some key advantages over more traditional approaches. Not only allow Kafka Streams to centralize data mediation, but also scaling to very high numbers of LVC participants due to its ability to distribute loads over multiple nodes (brokers) as well as segregation of data into different topics and even parallel processing of data by multiple simulation services (load balancing). All this as very accessible open-source software that is not specific to M&S. But of course, this as well as our SDK layer adds processing and latency overhead. Hence it is crucial to evaluate performance in real-world scenarios.

We start by looking at some data on scaling aspects in terms of message count, data throughput and latency in the Force Integration Training use case that is summarized in the following table:

Table 2: Test results from our force integration training setup for typical message traffic and latency

Force Integration Training **Two NH-90 Sea Lion Full Flight Simulators** **About 80 services (consumers and producers) connected to messaging middleware**			
Test Setup	**Number of Messages [1/s]**	**Data Throughput [kB/s]**	**Latency [ms]**
Two Ownships only with active voice communication	180 ± 13	70 ± 5	
As above, with radio jammers added	350 ± 19	137 ± 7	12 ± 4
As above, with 30 CGF entities added	720 ± 27	281 ± 10	

Several crucial observations can be made from these results. First , by connecting all simulation services of a single simulator directly to the messaging middleware, we add a significant number of producers and consumers per simulator, here around 40. At the same time the base load of a simulator is quite low with only about 200-400 messages per second and around 80-150 kB per second of data being transferred. Per constructive entity we add roughly 20 messages per second and 5 kB per second of data. This is orders of magnitude less than the several hundred thousand messages per second and over 100 MB per second of data that a single Kafka node can handle and stays true even if we assume that a more complex scenario with lots of interactions (sensors, weapons, …) would drive up the message count two- or threefold. Also, latency from message sent by one service via the SDK until it is received by another service seems promisingly low. Testing with our own low-overhead HLA RTI has shown, in the same test setup, a latency of 3 ± 1 milliseconds. Meaning that all the additional processing of the Kafka transport and the other features of the messaging middleware, mediation services and the SDK layer add about 9 milliseconds of latency.

For our Collective Training use case the setup is a bit more complex with all services and simulation systems connecting to cloud-hosted infrastructure. The scaling aspects for simulators and CGF entities remain the same. Additional latency is added by the simple fact of geographical distribution, but also by moving from to a gateway approach instead of direct connection at service level. The latter introduces another 2 ± 1 milliseconds for the additional hop. When using cloud infrastructure, other aspects become important though. Mostly the required compute and memory resources, as those drive the cost of the deployment. A few data points can be found in the table below:

Table 3: Resource usage of the Kafka messaging backbone in our collective training setup

Collective Training **Multiple sites: Germany, Hungary, United Kingdom** **Gateway approach; one consumer and producer per service or device** **Microsoft Azure Container with 8 vCPUs and 16GB RAM**				
Test Setup	**CPU Usage [%]**	**RAM Usage [MB]**	**Number of Messages [1/s]**	**Data Throughput [kB/s]**
Idle	1.73 ± 0.10	396 ± 23	0	0
Single virtual simulator	4.30 ± 0.06	421 ± 24	192 ± 11	75 ± 4
Three virtual simulators	5.85 ± 0.33	449 ± 26	574 ± 33	220 ± 13
As above, with one constructive service (50 CGF entities) added	6.40 ± 0.36	655 ± 38	1546 ± 89	468 ± 27

From these results we can deduce that, at least for our M&S use cases, the number of consumers and producers is driving the CPU and RAM load as opposed to the low number of messages that we send and receive. Even our single node with very modest specifications, a standard Microsoft Azure container with eight vCPUs, is able to run exercises with dozens of live and virtual participants and hundreds of constructive entities. This means that larger exercises and complex setups with multiple parallel exercises, redundant messaging and automatic failovers are very achievable with commodity hardware or cloud resources.

In summary, the presented technical approach is suitable for typical M&S use cases found in the context of MDO training and integrated deterrence, which was validated through testing with real-world systems and real-world scenarios. While our implementation of a messaging middleware introduces measurable additional latency and processing overhead, both are at a very low and acceptable level, dwarfed by network latency in true distributed setups. We could not identify any technical reasons that would prevent the use of this MSaaS implementation in distributed training.

OBSERVATIONS, CONSIDERATIONS AND RECOMMENDATIONS

While onboarding more and more services and setting up the tests, several challenges were encountered and needed to be overcome. These also lead to some operational considerations for distributed training environments.

The SDK and other additions over pure Kafka to create the messaging middleware proved very effective in expediting the onboarding of services and simulation systems. The simulation-specific abstraction layer placed on top of the low-level Kafka functionality provides easy access to the simulation messages, control messages, the state machine and other key functionality in a way that can easily be mapped to most simulation services. Together with guides and examples this made it easy even for new developers to create new integrations within days. It was also possible to tweak and update the core of the messaging middleware with the integrations following along with a simple exchange of the SDK libraries instead of touching the code. Hence, even though this layer comes with a performance penalty in terms of processing and latency overhead, the advantages outweigh these penalties, which are of tolerable magnitude for our use cases. On a more organizational level, for widespread adoption the SDK needs to be available for various programming languages and operating systems. Also, it needs to be placed under strict configuration management in accessible repositories for easy download to make sure developers use official releases.

While doing data mediation centrally via Kafka Streams in theory enables services and simulation systems to just send and receive messages in their native format without any further conversions, it quickly became apparent that further data mediation capabilities were needed at the SDK endpoints. Even though services often promised to fully comply to standards, e.g. DIS 7, the implementation of these standards differ between vendors. Basically no integration was exempt from the need to fix (often smaller) issues with individual data fields or other similar deviations. Thus, a need for the second mediation tier within the SDK was identified. This allowed us to make small adjustments right at the source without creating an undue amount of parallel streams in the central messaging infrastructure.

In our tests we were regularly dealing with more than a dozen LVC systems spread over multiple sites. Having registration and orchestration services plus easily accessible frontends was key in making these tests a routine affair and hence a persistent capability. Being able to see the aggregated connection, health and simulation status of each participant is an important debugging tool. As distributed training systems grow, the overall complexity and chance of failure of individual services and simulation systems increases. Quickly identifying issues and getting hints at how to rectify them greatly accelerated the time from setup start to exercise readiness. Similarly, having orchestration services to control and manage the exercise and its participants enabled a small core team to ready all services and simulation systems remotely from a central location. No need to configure each service or simulation system manually for each exercise, e.g. by entering DIS Exercise IDs or joining a HLA Federation. We therefore consider both registration and orchestration key capabilities for any distributed training system and recommend those to be implemented with priority.

Using Kafka at the core of the messaging middleware adds another piece of central infrastructure that needs to be procured, deployed and maintained. But having such an infrastructure also has key advantages. Most data mediation happens in one location and only once, reducing overall compute requirements and increasing maintainability as complex gateway setups are not necessary anymore. In addition, well-known concepts and implementations for scaling, parallel processing, fault tolerance and automatic failover from the cloud computing space can be repurposed for M&S use cases as the technology baseline is the same. This becomes especially important when distributed training systems grow in the number of participants and CGF entities. Adding to this the relatively moderate compute resource use and easy deployment using commodity container technology, we again consider this approach to be preferable.

SUMMARY AND CONCLUSION

In previous research, the suitability of the MSaaS RA for large-scale LVC exercises and training events was explored by the creation of a technology demonstrator that used cloud-computing concepts and components applied to M&S. We managed to build on this prior work and extend our MSaaS implementation into a composable ecosystem of components that enabled actual distributed LVC use cases in a routine, repeatable manner.

During development and testing, several key elements were identified that drove these successes. Two-tiered mediation services combined with an SDK make composition of a heterogeneous system with components of different vendors considerably easier, hence faster and cheaper. Custom registration and orchestration services address the need to manage and control exercises across locations with less burden on the organizers and on-site staff. Deployment using cloud-native methods reduces effort to set up and maintain the LVC infrastructure.

The most influential decision was, however, to leverage Kafka as basis for our messaging middleware. Compared to more traditional approaches based purely on DIS or HLA, it delivers good performance while also allowing us to employ strategies to create systems that scale efficiently not only with the number of LVC participants but also the number of connected sites and parallel exercises. Being able to draw on a vast repository of knowledge and experts due to its large user base within the cloud computing industry is also a major plus.

Overall, this effort proved that following the guidance provided by the MSaaS RA combined with sensible choices during its implementation provided clear potential to deliver the kind of persistent distributed training capabilities that Allied and NATO member nations seek out as key enablers for MDO in initiatives like NATO Distributed Synthetic Training (DST). This leads to the conclusion that today theoretical concepts, technologies and practical implementations are available to turn these visions into reality. We hope that our work can further promote the use of MSaaS and help guide other implementations.

ACKNOWLEDGEMENTS

Setting up and running the various test setups across multiple sites in North America and Europe would not have been possible without a vast number of supporters that spent countless hours helping deliver the results and insights presented in this paper. We would like to extend our heartfelt thanks to all contributors.

REFERENCES

NATO Science & Technology Organization. (2019). *Modelling and Simulation as a Service. Volume 1: MSaaS Technical Reference Architecture*, NATO Science & Technology Organization, [STO-TR-MSG-136-Part-IV].

Defense Standardization Program Office. (n.d.). Modular Open Systems Approach (MOSA). Defense Standardization Program. https://www.dsp.dla.mil/Programs/MOSA.

Freeman, J., Oster, E., Ivchenko, S., Krupp, A., Koch, M., Bernheim, E. (2024). Leveraging MSaaS Concepts to Enable Mission Environments: Lessons Learned. I/ITSEC 2024 Conference Proceedings.

NATO Science & Technology Organization. (2022). NATO Reference Architecture for Distributed Synthetic Training, NATO Science & Technology Organization, [STANREC 4799].