Integrating Multiple HLA Federations for Effective Simulation-Based Evaluations of CPS

Himanshu Neema
Vanderbilt University
Nashville, TN USA
himanshu.neema@vanderbilt.edu

Thomas Roth
National Inst. of Standards & Technology
Gaithersburg, MD, USA
thomas.roth@nist.gov

Chenli Wang
National Inst. of Standards & Technology
Gaithersburg, MD, USA
chenli.wang@nist.gov

Wenqi Wendy Guo
National Inst. of Standards & Technology
Gaithersburg, MD, USA
wenqi.guo@nist.gov

Anirban Bhattacharjee
National Inst. of Standards & Technology
Gaithersburg, MD, USA
anirban.bhattacharjee@nist.gov

***Abstract*—Cyber-Physical Systems (CPS) are complex systems of computational, physical, and human components integrated to achieve some function over one or more networks. The use of distributed simulation, or co-simulation, is one method often used to analyze the behavior and properties of these systems. High-Level Architecture (HLA) is an IEEE co-simulation standard that supports the development and orchestration of distributed simulations. However, a simple HLA federation constructed with the component simulations (i.e., federates) does not satisfy several requirements that arise in real-world use cases such as the shared use of limited physical and computational resources, the need to selectively hide information from participating federates, the creation of reusable federates and federations for supporting configurable shared services, achieving performant distributed simulations, organizing federations across different model types or application concerns, and coordinating federations across organizations with different information technology policies. This paper describes these core requirements that necessitate the use of multiple HLA federations and presents various mechanisms for constructing such integrated HLA federations. An example use case is implemented using a model-based rapid simulation integration framework called the Universal CPS Environment for Federation (UCEF) to illustrate these requirements and demonstrate techniques for integrating multiple HLA federations.**

***Index Terms*—co-simulation, cyber-physical systems, distributed simulation, high-level architecture, modeling and simulation, shared federations**

I. Introduction

Cyber-Physical Systems (CPS) comprise networked computational, physical, and human components engineered for function through integrated physics and logic [1]. CPS include critical infrastructures such as the electric grid, transportation, advanced manufacturing, health care, and others. The co-simulation of CPS often requires the collaboration of domain experts across organizational boundaries, leading to the integration of different organizational policies and proprietary information into an integrated distribution simulation. Furthermore, some CPS are system-of-systems (SoS) that are comprised of multiple subsystems with complex needs for synchronization, consistency, security, performance, and interoperability. For example, a smart city may integrate its electric grid, transportation, and health care systems into one integrated service for its residents.

This study explores various complex requirements that arise when using the High-Level Architecture (HLA) [2] for distributed simulations of CPS. It proposes that the real-world concerns of CPS and SoS can no longer be adequately addressed by using a single HLA federation that considers all of its subsystems (federates) as equal peers. It argues that these situations require separation of the system into multiple federations - each dedicated to simulation of part of the CPS - with defined mechanisms for inter-federation communication for joint coordination and control. The federations may be split based on proprietary information, geographic distribution of their member simulations, time-scale of the processes involved, or some other organizing principle. This study first describes the real-world concerns that may lead to the use of multiple interactive federations, then it presents different implementations of time synchronization and data exchange across federation boundaries. In addition, a use case is implemented using a model-based rapid simulation integration framework called the Universal CPS Environment for Federation (UCEF) [3] [4] to demonstrate these concepts.

HLA was designed to be comprehensive and supportive of distributed simulations, which involve different models of computation, physical systems and processes, and even human-in-the-loop for simulated training exercises. However, developing federates that conform to the standard requires a non-trivial amount of effort. UCEF addresses this challenge using a model-based simulation integration technology that can rapidly synthesize HLA-based distributed simulations. The automatic creation of HLA federations is less error-prone due to the use of defined model semantics and a validated synthesis environment. In addition, these automation capabilities can be extended to create co-simulation configurations that require multiple integrated HLA federations.

Consider the use of distributed simulations to analyze an airport system (see Fig. 1). An airport provides services and infrastructure (e.g., transport, baggage routing, airport security, runway allocation, etc.) to passengers and airlines. The airlines are independent entities that utilize the airport as a shared resource and can be thought of as individual CPS that each requires an HLA federation to represent its functionalities. Joining all of the airlines' subsystems and airport infrastructure together into a single federation is complicated and leads to other drawbacks. For example, it can impact performance due to the large amount of combined network traffic and may not be suitable for information hiding between airlines' subsystems. Alternatively, the use of multiple federations can isolate network traffic and restrict unwanted communication between different airlines. In addition, the use of multiple federations is more intuitive, easier to maintain and manage, and better separates the concerns of the individual federations. However, developing these separate HLA federations and integrating them in a holistic, consistent manner is highly challenging. Here, UCEF's modeling and federation synthesis tools can accelerate and simplify the development process.

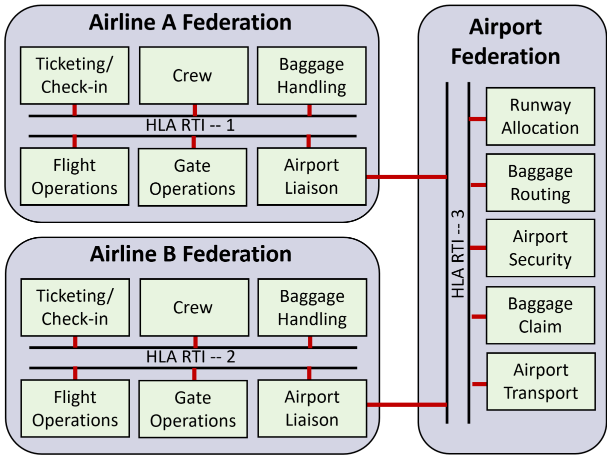

Fig. 1. Using Multiple Federations for an Airport System

The paper is organized as follows: Section II summarizes the related work. Section III enumerates the requirements of real-world use cases that lead to the need for the integration of multiple HLA federations. Section IV identifies various mechanisms to organize and integrate multiple HLA federations. Appendix A illustrates how these mechanisms address the above-mentioned requirements. An example use case is implemented in Section V to demonstrate some of the above requirements and mechanisms. Finally, Section VI concludes the paper and provides limited directions for future research.

II. Related Work

Distributed simulation is widely used in domains where independent subsystems need to be integrated into a larger system, such as in the military. A detailed overview of different technologies for simulation integration for CPS can be found in [5]. Related test beds for cybersecurity research are covered in [6] and [7]. However, many real-world use cases require interconnecting multiple HLA federations for the large-scale simulations, especially simulations over a wide-area network.

Hierarchical Simulation has been used in many different simulation environments. It holds great promise, especially in terms of modeling efficiency and model reuse. A hierarchical simulation-based software architecture is proposed in [8]. An integrated model-driven framework is proposed in [9].

Proxy methods have been used to enable inter-operating federations without modifying the HLA Run-Time Infrastructure (RTI). Federation bridges to wire multiple HLA federations were proposed in [10], [11], and [12]. HLA service management modules to address scalability and security issues was proposed in [12]. Further, [13] provides several bridge topologies (e.g., linear and cyclic inter-federations) and their associated issues and potential solutions. A dynamically adapting bridging federate was proposed in [14] that can change connection paths among bridged federations as systems evolve during the simulation. Further, a forwarder concept [15] was also introduced in the Portico RTI [16] that enables transforming the flat structure of HLA federations into a hierarchical structure where federates are partitioned into different interconnected clusters.

Information hiding is often needed in distributed simulations, but HLA does not provide direct mechanisms for it. A related hierarchical federation architecture based on a hybrid gateway/proxy approach for interoperability between simulation federations was given in [17].

An obvious trade-off in decomposing large federations into multiple smaller, interconnected federations is the overhead of communication. A composed federation framework to achieve a level of balance between information hiding and reducing the communication overhead was given in [18]. It also provides methods to create a secure distributed environment that minimizes critical security concerns with HLA RTIs.

III. Use Cases for Multiple Federations

This section describes several use cases that can benefit from using multiple HLA federations. Each subsection describes one requirement that leads to multiple federations and provides an example application.

A. Shared and Limited Resource

Some resources are limited and cannot be easily reproduced. These include licensed software, prototype hardware, human experts such as surgeons, and entire test facilities. This category represents use cases where multiple independent entities must share access to one of these limited resources. The limitation on availability generates contention among the entities, thus requiring them to collaborate and coordinate. For instance, consider large manufacturing plants located on different floors of a building. The manufacturing operations might be highly complex and require distributed simulation for analysis. At the same time, these plants use shared building infrastructure such as an elevator, a stairwell, and a warehouse. The plants may need to develop a sharing strategy, a priority order, a scheduling mechanism, and an effective method for coordination and control of the shared resource.

B. *Information Hiding*

A distributed simulation can include actors from different organizations with the subsystems for these organizations split across multiple security domains. Some information such as business secrets and classified data may be restricted for exchange between actors in the same security domain. This category represents use cases where a subsystem exchanges sensitive information that must be protected, while at the same time it has a requirement to interact with external entities, which may even extend to geographically distributed Internet of Things (IoT) [19]. For instance, consider a large organization comprised of multiple departments. The low-level information within the department (e.g., staffing, workplace policies) may not be relevant to other departments, while other information must be shared across the organization (e.g., status updates, product flows, process workflow events). Another example is an office which internally may argue over problems and solutions, but choose to present only a unified, filtered, and vetted view (e.g., a report) to external entities.

C. *Model Reuse*

Real-world distributed simulations can be highly complex such that both the models used, and their curated configurations may require a large amount of experimentation, analysis, and even certification. Deviations from the approved configurations can be unacceptable. For instance, military exercises (e.g., pilot flight training, astronaut zero-gravity training) require a well-defined and standardized set of software, hardware, and policies for use. Misconfigured equipment can result in a failed exercise and even injury to the trainees. In distributed simulations, this requires creating reusable federates/federations. The larger distributed simulation is created through composition of these well-curated federates and/or federations. In addition, many reusable federations use a rigid, custom Federation Object Model (FOM) (e.g., Distributed Interactive Simulation (DIS) [20] federates based on Real-time Platform Reference Federation Object Model (RPR-FOM) [21]). Here, techniques to map messages in one federation's FOM to the other are needed (as provided in UCEF).

Another use case involves use of a remote federation. For example, a laboratory that interacts with external systems may require its own curated HLA federation with a fixed FOM. The laboratory services remotely accessed through this curated federation might have access control mechanisms and policies to schedule their use. This necessitates designing reusable, validated federations that can be effectively used by composing them with other integrated federations.

D. *Performance Benefit*

As the size of federations increases, the performance of the distributed simulation tends to decrease significantly. This is mainly due to increased traffic of messages on the HLA RTI, but often also due to sub-optimal RTI implementations [22]. For example, if all the three federations shown in Fig. 1 were combined into a single federation, there would be a large amount of HLA network traffic which could lead to both dropped messages (in case where 'best effort' message delivery is used) and significant network delays. In general, when a federation comprises a large number of federates, a performance benefit in terms of reduced number of messages on the HLA bus could be achieved by *grouping* related HLA federates into a separate federation.

It is also possible that different federates in a federation use different logical time resolutions (i.e., step-size). Here, if certain federates are slow in computing their behavior in every step or they require a smaller step-size, processing bottlenecks can arise in the integrated simulation. In time-constrained HLA simulations, these federates with a very small step-size can trigger recalculation of state variables in fast computing federates as well as in the RTI. This leads to an increase in runtime of the distributed simulation. Separating federates with larger step-sizes can enable them to execute at large step-sizes. An example using the same principle but with Functional Mock-up Interface (FMI) was previously published in [23].

Another interesting example is federates that dynamically change step-sizes during the co-simulation. For instance, if a fire extinguishing exercise uses a water spraying federate, this federate does nothing until a fire has been reported. Once the water spraying federate begins putting out an active fire, it reduces its step-size to a very small value to keep track of highly dynamic simulation of fire propagation. Splitting this federate into a separate federation can help the other federates run at larger step-sizes and thus execute in less time.

E. *Organizing Principles*

When a federation has different sets of federates simulating different aspects of the systems, it can be useful to organize the related federates into separate federations. For example, in a biological simulation of the human body, different body functional systems (e.g., nervous system, digestive system, respiratory system, muscular system) represent fundamentally different concerns in the body and as such might be better organized as separate federations. These federations will interact with each other for their inter-dependencies.

F. *Different IT Policies*

This category refers to situations where the use of multiple HLA federations is needed due to different information technology (IT) policies within an organization. For example, some federates may be located at different (potentially remote) locations or may require different access mechanisms such as passing through a network firewall, and their use in distributed simulations may require complying with IT policies. This, in turn, may require federates to be packaged into different HLA federations.

Another use case in this category may emerge due to high computational requirements of federates. In this case, depending on where and how much compute resources are available, the federates will need to be grouped into separate federations and managed accordingly.

IV. Mechanisms for Multiple HLA Federations

In this section, various mechanisms (or architectural patterns) to develop integrated multiple HLA federations are described. Various patterns have emerged in the literature as described in Section II, but a consistent and comprehensive categorization is missing. Therefore, five representative mechanisms (Fig. 2) that cover these patterns are presented below:

- ***M1: Shared Federate***: Shared federates are typically used for largely independent federations that only need to interact to utilize a limited resource (e.g., a remote laboratory). Here, these independent federations are created normally, but a new *shared* federate is created to participate in all the federations. The shared federate is responsible for passing messages between the independent federations and for providing logical time translation when the time advancement mechanisms of federations are not synchronized. The use cases that involve a limited resource, such as an elevator shared between multiple organizations on different floors of a building, or a remote server that provides integral services to multiple, independent federations, fit within the scope of this mechanism. Another example could be a Quality of Service (QoS) module that monitors resource utilization (e.g., CPU, memory, storage, database, network bandwidth) and performance across federations and performs modifications in their execution for better balancing the resource loads and their priorities. Fig. 2(M1) shows the architecture of how a shared federate is used in multiple federations. In this case, the shared federate is a member of both federations 'a' and 'b'. This mechanism can also provide performance benefits compared to using a single federation containing all the federates. This benefit is demonstrated in the case study in Section V.
- ***M2: Shared Federation***: The mechanism *M1* can be further extended when, instead of a single limited resource, different federations need multiple resources and services. A crucial aspect of this architectural pattern is that the different resources and services (provided by a shared federation) can be used independently of each other. The airport example shown in Fig. 1 is a good candidate for this mechanism. In that example, multiple airlines' federations need to access a large number of airport infrastructure and services. However, the airport services such as airport security and airport transport are largely independent. The resource contention occurs when multiple federations (in this case the different airlines) attempt to utilize the same service at the same time. Note that in this architecture the independent federations do not share a federate, and any interaction among them occurs only via the shared federation services. As shown in Fig. 2(M2), the federations 'a' and 'b' execute independently but use a federate that participates in a third federation ('c') to access c's services.
- ***M3: Parallel Connected Federations***: This mechanism effectively generalizes M1 (that used only a single shared federate) by allowing all independent federations to use a shared federate with each of the other integrated federations. These independent federations execute in parallel but interact directly through shared federates. This is in contrast to the mechanism M2, where the independent federations interacted only through the services of a separate federation. A good use case of this from the electrical grid domain is integrated systems of power transmission, distribution, and markets. Each of these systems operate independently, but also are connected to each other. For example, the transmission system must maintain the power quality and quantity desired by the city power distribution system. The transmission system also interacts with power markets (both bulk power and local power markets). The distribution system may also interact with markets through smart grid systems where many participants can use transactive controllers to buy and sell power. As shown in Fig. 2(M3), the independent federations 'a', 'b', 'c' are running concurrently but do use shared federates for interacting with each other when necessary for their coordination and control.
- ***M4: Hierarchical Federations (a.k.a. Federation of Federations)***: This mechanism reflects how things are typically organized in a large organization, where the organization can have several smaller entities or departments, which can further have smaller entities or divisions internally, and so on. Modeling real-world operations that follow this organization requires similar federation designs. Another important advantage of hierarchical federations is that it can easily scale to accommodate very large federations. Owing to the limitations of RTIs, a single federation with thousands of federates is not practically feasible. However, composing related federates in a hierarchical manner allow them to be executed in parallel, potentially split across computation clusters in a cloud, while still time synchronized and exchanging data. In Fig. 2(M4), the federations {'a', 'b'} are used in a higher-level federation 'ab', federations {'b', 'c'} are used in federation 'bc', and federations 'ab', 'bc' are used in the root level federation 'abbc'.
- ***M5: Clustered Federations***: This mechanism represents the use case where many independently operating federation clusters are analyzed in a larger application context. Consider the use case of joint military operations being conducted in different countries (or areas). Each of these military operations may use different military forces (e.g., army, air force, navy, or space force) that conduct missions in a coordinated manner to achieve common objectives. The individual military force operation may involve many organizations and commands. For example, the army might operate using many remote military bases, utilize a mix of weapons (e.g., machine guns, cannons) and military equipment (e.g., tanks), and deploy a range of communication, control, and reconnaissance tools and software. Therefore, this independent yet communicating processes and workflows might be modeled well using

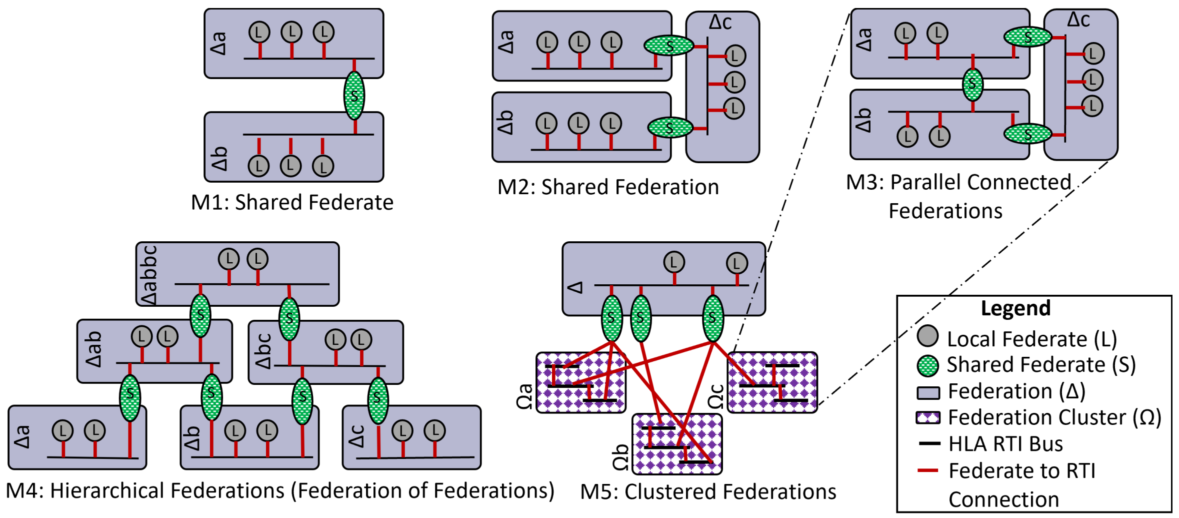
M1: Shared Federate
M2: Shared Federation
M3: Parallel Connected Federations
M4: Hierarchical Federations (Federation of Federations)
M5: Clustered Federations
Legend
Local Federate (L)
Shared Federate (S)
Federation (Δ)
Federation Cluster (Ω)
HLA RTI Bus
Federate to RTI Connection

Fig. 2. Mechanisms for Multiple HLA Federations

mechanism M3 (i.e., parallel connected federations). At the same time, the overall organization conducting these joint military operations can be modeled using a top-level federation that interacts with the concurrently running cluster of connected federations. Fig. 2(M5) shows the use of three connected federations (each using a M3 mechanism internally) ‘a’, ‘b’, and ‘c’ and a top-level federation that is connected to all three federations.

Note that multiple mechanisms can be utilized for addressing specific application requirements and it depends on the constituent systems that are modeled in the federations, how they are organized, where they are physically located, and their specific resource and performance needs. Appendix A shows how different mechanisms in Section IV can be useful in addressing the requirements listed in Section III.

V. Case Study

The following case study demonstrates the performance benefit achieved with multiple federations using an example CPS. Advanced manufacturing systems use communications networks and distributed control to operate equipment in response to changing factory conditions and customer needs. This case study models a simplified manufacturing plant that adjusts its outputs based on the performance of its products in a market.

A manufacturing company has internal organizations for purchasing of raw materials, production of products, and selling of products at market. The production, market, and buyers are implemented as separate federates. Figure 3 shows a federation diagram for this example created in UCEF. The federates in Figure 3 were implemented in Java for two scenarios: one where all the federates were integrated into a single federation, and one with separate federations for the manufacturing company (Federation 1) and the potential buyers (Federation 2).

The production federate uses the income from the market to produce a random distribution of three products. These products are sent to the market federate, which implements a Dutch auction. The market offers the products at a high asking price and waits for buyers to accept the price with a bid. If enough buyers do not accept the price, the market reduces the asking price and starts another round of the auction. The price will continue to be reduced until all products are sold.

A. Single Federation Implementation

When the market receives product, it starts the auction. Each auction round lasts one logical time step. The market collects bids for the current asking price, then determines if all the products have been sold. If a product remains unsold, the market starts another round with reduced asking price. If no product remains, the market ends the auction and sends the income to the production company. The auction duration depends on the initial asking price and the buyer behavior, and each auction will require a different number of rounds (and thus require a different amount of logical time progression).

For the production federate, an unknown amount of logical time will progress between sending a product to market and receiving the income in return. Thus, the production federate has dynamic logical time progression responsive to the market. This can be implemented in two ways: either it can use the HLA *NextEventRequest* service to wait for the income, or it must also have a logical time step of 1 s and continuously poll if the auction has concluded. This paper implemented the production federate with a logical time step of 1 s.

B. Multiple Federation Implementation

A single federation implementation requires the production federate to waste logical time waiting on the auction. If the average auction requires 99 rounds, then the production federate will perform meaningful work in only 1 of 100 logical time steps. Based on HLA implementation, the synchronization requirements of additional time steps and the network traffic from the buyers can have unnecessary performance impact. An

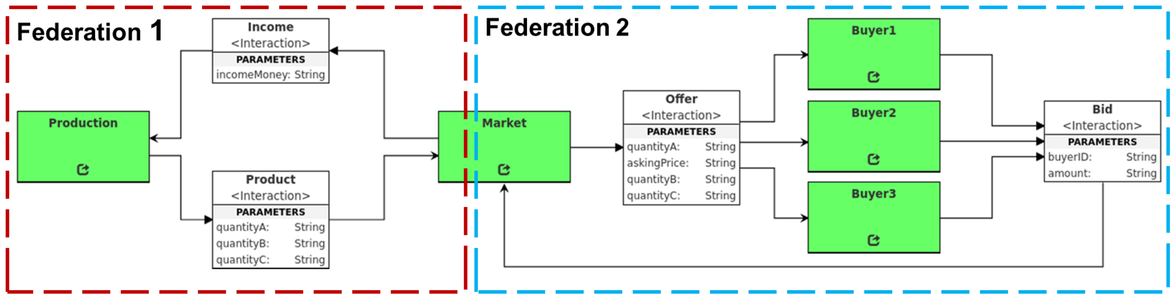

Fig. 3. Federation Model for the Manufacturing Scenario

alternative implementation is using two federations to isolate the behavior of the market from the production company.

As shown in Figure 3, Federation 1 contains the production and market federate. In every logical time step, in Federation 1, the production federate will receive its latest income from the market and produce some amount of product. Federation 2 contains the market federate and the buyers. The market implementation is the same as for the single federation case, with one logical time step (in Federation 2) corresponding to one auction round. Each federation has independent logical time progression, and there is no mapping between the logical time values due to the random nature of the auction.

The market federate co-exists in two different federations. It was implemented in Java using two threads where each thread represents one of its identities: one thread dedicated to Federation 1, and another to Federation 2. The main method controls the logical time progression of both threads, and ferries messages between the two federations using the HLA services. This corresponds to mechanism M1 in Section IV.

C. *Performance Results*

In this example, the quantity of products is determined by income amount and a random function in the production federate. The testing starts from the market federate sending 1,000 dollars to the production federate as the initial income. After the Buyer federate reaches the logical time of 10,000 s, the federations is stopped and the number of interactions in the federations is collected. Table 1 shows the testing results of single and multiple federation implementations.

TABLE I
COMPARISON OF REGISTERED NUMBER OF INTERACTIONS

Design		**Time Step**	**#Interactions**
Multiple Federations	Federation 1	23	23
	Federation 2	10,000	9,999
Single Federation		10,000	10,015

In the multiple federation design, Federation 2 (i.e., auction federation) registered 9,999 interactions, while Federation 1 (i.e., production federation) only registered 23 interactions. On the other hand, 10,015 interactions were propagated in the single federation design during the test. The result shows that the multiple federation structure effectively reduces the network traffic through the production federate by 99.8%. Moreover, the number of HLA *TimeAdvanceRequest* calls in the production federate dramatically decreased from 10,000 to 23. Such designs can help low-performance or low-power simulation devices by reducing unnecessary interactions and calls to RTI services in federations.

VI. Conclusions & Future Work

Real-world use of HLA federations for evaluating complex scenarios can become highly challenging, sub-optimal, and even practically infeasible. In these situations, to manage the complexity of scenarios, federation designers need to resort to decomposition, reorganization, and parallelization. This paper reviewed related work on various methods used to deal with these complexities and provided a classification of these real-world requirements. Further, it identified a set of key mechanisms that can be used to develop integrated multiple HLA federations. It also provided use cases where each of these mechanisms are well-suited. Furthermore, it demonstrated the use of integrated multiple HLA federations using a detailed case-study.

Our future work focuses on developing quantitative metrics to measure different mechanisms of creating integrated multiple HLA federations and designing realistic scenarios and evaluating them with these metrics. The outcome of these evaluations will be reported in future publications. These metrics will further serve as a useful guideline for federation designers, particularly when multiple mechanisms are applicable for a given scenario. In addition, UCEF will be extended with a library of tools and reusable components that will simplify creation of the various mechanisms for integrating multiple HLA federations.

Acknowledgment

This work at Vanderbilt University is supported in part by the US National Institute of Standards and Technology (NIST) (award #70NANB21H164) and the US National Security Agency (NSA) (award #H98230-18-D-0010). Official contribution of the NIST and NSA; not subject to copyright in the US. Certain commercial products are identified in order to adequately specify the procedure; this does not imply endorsement or recommendation by NIST or NSA, nor does it imply that such products are necessarily the best available for the purpose.

References

[1] E. Griffor, C. Greer, D. Wollman, and M. Burns, ”Framework for Cyber-Physical Systems: Volume 1, Overview”, 2017, DOI: 10.6028/NIST.SP.1500-201.

[2] IEEE Std 1516-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)- Framework and Rules., 2010.

[3] B. Martin, T. Roth, E. Griffor, P. Boynton, J. Sztipanovits, and H. Neema, ”Universal CPS environment for federation (UCEF),” In 2018 Winter Simulation Innovation Workshop, 2018.

[4] T. Roth, E. Song, M. Burns, H. Neema, W. Emfinger, and J. Sztipanovits, ”Cyber-Physical System Development Environment for Energy Applications,” 11th Int. Conf. on Energy Sustainability, 2017, DOI: 10.1115/ES2017-3589.

[5] H. Neema, J. Sztipanovits, C. Steinbrink, T. Raub, C. Bastian, and S. Lehnhoff, ”Simulation integration platforms for cyber-physical systems,” Proc. of the Workshop on Design Automation for CPS and IoT (DESTION ’19), ACM, pp. 10–19. DOI: 10.1145/3313151.3313169.

[6] X. Koutsoukos, G. Karsai, A. Laszka, N. Himanshu, B. Potteiger, P. Volgyesi, Y. Vorobeychik, and J. Sztipanovits, ”SURE: A Modeling and Simulation Integration Platform for Evaluation of Secure and Resilient Cyber–Physical Systems,” in Proc. of the IEEE, vol. 106, no. 1, pp. 93-112, Jan. 2018, DOI: 10.1109/JPROC.2017.2731741.

[7] J. Mirkovic, T. V. Benzel, T. Faber, R. Braden, J. T. Wroclawski and S. Schwab, ”The DETER project: Advancing the science of cyber security experimentation and test,” IEEE International Conference on Technologies for Homeland Security (HST),2010, pp. 1-7, DOI: 10.1109/THS.2010.5655108.

[8] K. Arne and P. Bernd., A Hierarchical Simulation Based Software Architecture for Back-Testing And Automated Trading. Proceedings - 25th European Conference on Modelling and Simulation, ECMS 2011. pp. 275-282. 10.7148/2011-0275-0282.

[9] D. Cetinkaya, Aa. Verbraeck, and M.Seck, Applying a model driven approach to component based modeling and simulation pp.546-553. 2010, 10.1109/WSC.2010.5679131.

[10] J. Dingel, and D. Garlan, and C. Damon, ”Bridging the HLA: Problems and solutions,” Proceedings of sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications, pp. 33-42, 2002, IEEE. DOI: 10.1109/DISRTA.2002.1166886.

[11] J. Dingel, D. Garlan, and C. Damon ”A feasibility study of the HLA bridge. Carnegie Mellon University,” 2018. DOI: 10.1184/R1/6587399.v1.

[12] C. Yoo M-W and K. TG. ”High-Level Architecture service management for the interoperation of federations,” SIMULATION. 2015;91(6):566-590. DOI: 10.1177/0037549715584403.

[13] B. Breholée and P. Siron. “Design and implementation of a HLA inter-federation bridge.” 2003.

[14] X. Wei and M. Xiaodong, ”Research on Multi-Federations Interoperability Based on Dynamic Adaptive Bridge Federate,” 2008 International Conference on Computer Science and Software Engineering, 2008, pp. 1049-1053, DOI: 10.1109/CSSE.2008.547.

[15] T. Roth, M. Burns, and T. Pokorny, ”Extending Portico HLA to Federations of Federations with Transport Layer Security,”, Fall Simulation Innovation Workshop, Orlando, FL, US, 2018.

[16] ”Portico: open-source HLA RTI,” https://github.com/openlvc/portico.

[17] W. Cai, S. Turner, and B. Gan ”Hierarchical federations: an architecture for information hiding,” Proceedings 15th Workshop on Parallel and Distributed Simulation, pp. 67–74, 2001.

[18] J. H. Ahn, M. G. Seok, C. H. Sung and T. G. Kim, ”Hierarchical Federation Composition for Information Hiding in HLA-Based Distributed Simulation,” 2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications, 2010, pp. 223-226, DOI: 10.1109/DS-RT.2010.35.

[19] D. Trihinas, G. Pallis and M. D. Dikaiakos, ”AdaM: An adaptive monitoring framework for sampling and filtering on IoT devices,” IEEE International Conference on Big Data (Big Data), 2015, pp. 717-726, DOI: 10.1109/BigData.2015.7363816.

[20] D. Fullford ”Distributed Interactive Simulation: its past, present, and future,” Proc. of the 28th conference on Winter simulation, pp. 179-185. 1996.

[21] SISO-STD-001-2015, ”Real-time Platform Reference Federation Object Model (RPR-FOM),” https://www.sisostds.org.

[22] M. Gütlein, B. Wojciech, C. Renner, and A. Djanatliev, ”Performance evaluation of HLA RTI implementations,” IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 1-8. IEEE, 2020.

[23] H. Neema, J. Gohl, Z. Lattmann, J. Sztipanovits, G. Karsai, S. Neema, T. Bapty, J. Batteh, H.Tummescheit, C. Sureshkumar, Model-Based Integration Platform for FMI Co-Simulation and Heterogeneous Simulations of Cyber-Physical Systems. 235-245. 2014, 10.3384/ecp14096235.

Appendix A
Mapping Requirements to Mechanisms for Integrating HLA Federations

The table below lists how real-world requirements can be addressed using various mechanisms discussed in the paper.

Requirement	**Examples**	**Mechanisms**
Shared limited resource	(a) A shared elevator among multiple organizations on different floors of a building; (b) A remote server that provides integral services to multiple, independent federations; (c) QoS module monitoring resource utilization and performance across federations and balancing load and priorities among them	M1
Services shared among independent entities	(a) Multiple airlines federations using the shared airport infrastructure and services; (b) Multiple shipping companies utilizing the seaport and its services for loading, unloading, and parking	M1; M2
Multiple independent and interdependent large entities	(a) Transmission system, distribution system, and power markets; (b) Power distribution grid, city transit system, and charging stations network	M3
Large hierarchically organized systems, workflows, or processes	(a) Large organizations with subdivisions and hierarchically organized processes	M4
Federations with extremely large number of federates	(a) Large SoSs and IoTs involving thousands of federated components	M4; M5
Common analysis and tuning of many independent federation clusters	(a) Large independent military operations/exercises with independent command and control, but in accordance with high-level mission guidance	M5
Need to hide/filter information among subsystems or processes	(a) Flow of filtered information from top-level management to supervisors to individuals; (b) Public office negotiating internal matters but presenting only public reports	M1; M4
Use of existing federates with fixed FOMs or different RTI implementations	(a) Federates with legacy interfaces (e.g., federates that previously conformed to Distributed Interactive Simulation (DIS) [20] supporting RPR-FOM [21]); (b) Federations built using commercial RTIs (e.g., Mak [22] or Pitch [22]) needing to work with federates/federations designed for specific research activities using open-source RTIs (e.g., Portico [16]	M1; M3
Simulating systems that communicate over the physical networks in accordance with different IT policies	(a) Large systems located at different organizations in different locations and using different networking topologies, policies, and technologies; (b) Systems using different third-party services that have different policies for authentication, authorization, and usage	M2; M3; M4