Universal CPS Environment for Federation (UCEF)

Martin Burns, Thomas Roth, Edward Griffor, Paul Boynton
National Institute of Standards and Technology (NIST)
100 Bureau Drive
Gaithersburg, MD 20899
martin.burns@nist.gov,
thomas.roth@nist.gov,
edward.griffor@nist.gov,
paul.boynton@nist.gov

Janos Sztipanovits,
Himanshu Neema,
Vanderbilt University
1025 16th Avenue South
Nashville, TN 37212
janos.sztipanovits@vanderbilt.edu,
himanshu.neema@vanderbilt.edu

Keywords:
cyber-physical systems, Internet of Things, high level architecture (HLA), testbed science, federated testbeds

ABSTRACT: *NIST, in collaboration with Vanderbilt University, has assembled an open-source tool set for designing and implementing federated, collaborative and interactive experiments with cyber-physical systems (CPS). These capabilities are used in our research on CPS at scale for Smart Grid, Smart Transportation, IoT and Smart Cities. This tool set, "Universal CPS Environment for Federation (UCEF)," includes a virtual machine (VM) to house the development environment, a graphical experiment designer, a model repository, and an initial set of integrated tools including the ability to compose Java, C++, MATLAB™, OMNeT++, GridLAB-D, and LabVIEW™ based federates into consolidated experiments. The experiments themselves are orchestrated using a 'federation manager federate,' and progressed using courses of action (COA) experiment descriptions. UCEF utilizes a method of uniformly wrapping federates into a federation. The UCEF VM is an integrated toolset for creating and running these experiments and uses High Level Architecture (HLA) Evolved to facilitate the underlying messaging and experiment orchestration. Our paper introduces the requirements and implementation of the UCEF technology and indicates how we intend to use it in CPS Measurement Science.*[1]

[1] NIST does not recommend or advocate any particular technology for the purposes of federation. Specific technologies mentioned are for illustrative purposes only.

1 Introduction

Cyber-Physical Systems (CPS) comprise interacting digital, analog, physical, and human components engineered for function through integrated logic and physics[1]. CPS integrate computation, communication, sensing and actuation with physical systems to fulfill time-sensitive functions with varying degrees of interaction with the environment, including human interaction. These highly interconnected systems provide new functionalities to improve quality of life and enable technological advances in critical areas, such as personalized health care, emergency response, traffic flow management, smart manufacturing, defense and homeland security, and energy supply and use. CPS and related systems (including the Internet of Things (IoT) and the Industrial Internet) are widely recognized as having potential to enable innovative applications and impact multiple economic sectors in the worldwide economy [2].

The impacts of CPS will be revolutionary and pervasive – this is evident today in emerging smart cars, intelligent buildings, robots, unmanned vehicles and medical devices [3]. The development of these systems cuts across all industrial sectors and demands high-risk, collaborative research between research and development teams from multiple institutions. Realizing the future promise of CPS will require interoperability between heterogeneous systems and development processes supported by robust platforms for experimentation and testing across domains. Meanwhile, current design and management approaches for these systems are domain-specific and would benefit from a more universally applicable approach.

The U.S. National Institute of Standards and Technology (NIST) has partnered with the Institute for Software Integrated Systems at Vanderbilt University to produce an open-source tool suite for the design and implementation of federated CPS experiments [4]. This tool suite integrates the leading simulation engines and hardware-in-the-loop with a distributed modeling and simulation architecture defined by the IEEE 1516 High Level Architecture (HLA) standard [5]. The following sections enumerate the benefits of a federated testbed architecture, describe the challenges of federated design for CPS, and introduce the high-level details of the NIST tool suite.

2 Federated Testbed Architecture

A federated testbed architecture is integrative, reconfigurable and reproducible, scalable and useable and, as such, is agnostic to the implementation details and geographic location of the individual entities that participate in an experiment. During one experimental run, the experiment entities might be simulations running on a private cloud hosted in one data center. The next run of the same experiment might replace these simulations with a collection of hardware entities distributed across testbed facilities locally, regionally, or around the world. This federated testbed architecture, when applied to CPS research, must be realized across components and their facilities in order to be able to share resources, reduce costs, and enable experimentation at both heterogeneous and larger scales.

In addition to the sharing of data and models, federation also includes the capability of a laboratory from one facility being able to provide both control and response signals to a laboratory located at another facility. The interactions in this type of remote federation are not simple data exchanges, and must contain an abstraction of the policies and disciplines employed at each location. This abstraction requires a governance model that communicates research and development priorities, and a consensus agreement that describes how the interaction between facilities will be developed and maintained. Furthermore, the laboratories involved must share a common architecture to facilitate data exchange, which can be thought of as an open-source platform that provides for the interaction of data, simulation models, and, in some cases, control signals.

Note that much work on the design and development of federated testbeds has been performed over the years in the forum where this paper is presented. Included in this work is a key reference on the recommended practice for the design of federated experiments, “Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)” [6].

This section discusses the benefits of a federated testbed architecture for CPS to motivate the need for an open-source platform for remote federation between laboratories.

Integrative

A CPS contains co-engineered, interacting networks of physical and computational components from multiple domains of technology. These domains – such as smart cities, smart manufacturing, and transportation – each have technologies and simulation engines tailored to their individual needs and experiences. CPS experimentation requires the integration of these heterogeneous, domain-specific tools into a common co-simulation platform. Figure 1 illustrates a subset of the different entities that must be integrated together to perform a CPS experiment.

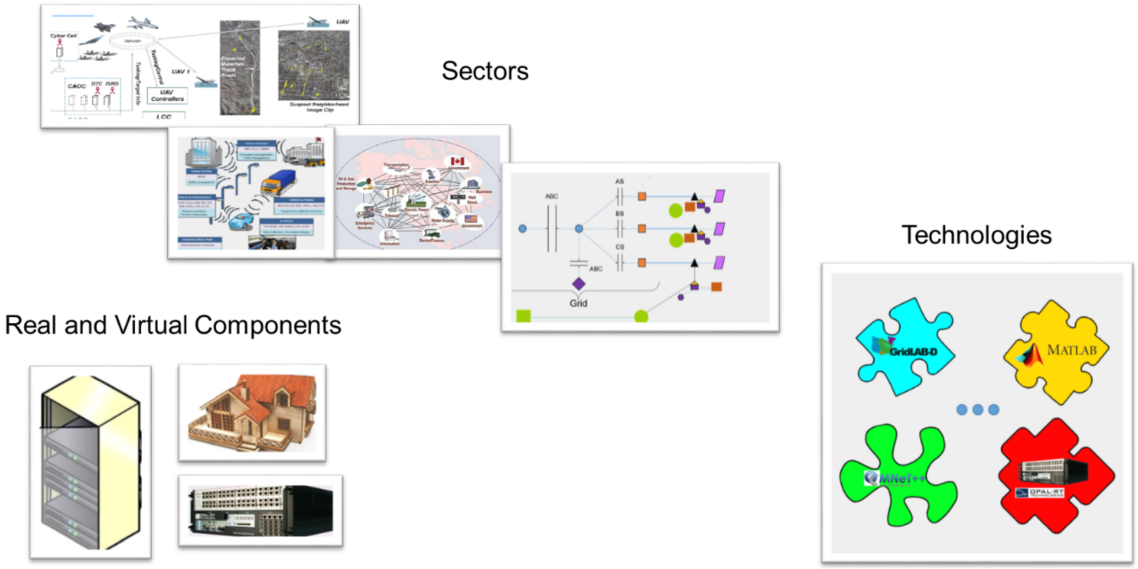

Figure 1 Ability to federate across sectors and technologies, as well as virtual or real instances of CPS.

One benefit of a federated testbed architecture is that interactions among heterogeneous federates are abstracted into a common, high level language interface where the abstract representation can be used without regard to domain-specific implementation details. This allows domain-specific simulation languages, experimental prototypes, and proprietary hardware to be integrated into composite experiments without the need to constantly create custom adapters. This adaptation should be achieved in a way that is uniform in the *native language of a component* to ensure the repeatability of an experiment. Note that it is not necessary to always expose every detail of a component interface for a federated testbed – only so much as is required for collaboration between components.

Reconfigurable and Reproducible

Once a domain-specific hardware or simulation entity has been wrapped as a federate, it can be easily composed into any number of different federations. This capability to compose experiments from individual federates is inherent to a federated testbed architecture, and facilitates the rapid reconfiguration and reproduction of CPS experiments. Repeatability of experiments is crucial to scientific research, hence there is a need for a uniform approach to federation.

Scalable

Experimentation with CPS at scale potentially requires evaluating the behaviors of, and interactions among, large numbers of individual components. Such components can be modeled using multiple modeling tools. Where required, the behavior of the model can be optimized to match the behavior of its physical equivalent. Once built this way, the models of physical systems can be scaled up to enable experimentation with system behaviors that otherwise would be difficult to achieve using physical components due to cost and complexity challenges.

Usable

A federate produced from domain-specific simulation technologies can be stored in a library with other federates abstracted from vastly different domains. A benefit of federated testbed architectures is a library of reconfigurable federates that combine:

1. unique or prototype hardware that cannot be relocated,
2. simulations created using domain-specific tools, and
3. proprietary components.

A federated testbed architecture provides the community of CPS researchers with a means of creating a rich array of different federates that can be incorporated into experiments without the need for significant development or domain-specific knowledge. Although an HLA-like integration architecture may constrain the capabilities of a federate, the libraries and common interfaces in a federated architecture add reusability to the development process.

3 Challenges for Federated Testbeds

The previous section discussed the benefits of a federated testbed architecture for the design and implementation of CPS experiments. However, a federated architecture – especially one that includes the potential to connect remote federations – leads to new challenges that would be less critical when using domain-specific and simple self-contained benchtop approaches. This section describes several of the unique challenges of federated testbeds.

3.1 Testing and Curation

It is not sufficient to address only the technical challenges required to enable federated experiments across testbeds. As experiments become more complex through integration of new federates, and as CPS applications become more reliant on experimental results to provide assurance in design and implementation, it becomes critical to characterize, validate and curate the individual federate models albeit within the scope of its intended use. It is likely that models developed for one project will be repurposed in another. The utilization of models created by others requires a means of model characterization that goes beyond the documentation produced solely for single test use. Methods of testing individual federates are necessary to gain confidence in their expected operation like unit testing [7][8] in software systems engineering.

The topics of testing and curation require that a given federate be supported by life cycle management, specification of its features and capabilities, and control of version and function.

The life cycle for federate development consists of several major steps:

1) design of the federate and its interfaces;
2) implementation of the federate in hardware or software;
3) documentation of the federate to describe its scope and usage;
4) development of an experimental test hardness to validate the federate behavior;
5) archiving of the federate, its documentation, and its testing results; and
6) the evolution of the federate with respects to defect management, enhancement and extension and versioning.

We introduce here the concept of a test harness for a re-useable federate. A test harness places the federate under test (FUT) in a test environment where its behavior can be monitored and stimulated. The development of a federate must be kept synchronized with the development of its corresponding test harness. The test harness should be able to generate a pass-fail result to support the validation of federations on a continuous basis. This continuous integration is desirable so that regression tests over federates can be periodically invoked to ensure the integrity of the designs. The archived federate must utilize versioning so that as the federate is evolved, applications can rely on stable versions for their integration. Semantic versioning should be used [9].

A federate requires description so that a potential integrator of a federate can understand its capabilities and limitations. For

example, a federate that acts as a source of weather that provides hourly data, cannot be used in a federation that demands minute by minute variations. The "federate descriptor" should provide this level of detail along with the interfaces that describe information received and emitted from the federate. The NIST CPS Framework [1] describes a collection of "concerns" that a given CPS can satisfy. A federate descriptor based on the specification of these concerns addressed can also define the test boundaries of federate testing. The HLA standard supports a version of a federate descriptor and this descriptor should be extended to cover more behavioral aspects of the federate.

Together, these dimensions can provide for a robust organization of federate designs and testing that, in turn, can encourage re-use.

3.2 Communication Protocols

A CPS can employ one or more communication protocols that its component devices and systems use to interact. There is a myriad of communications protocols used in practice for this exchange. An accurate model of the communications environment within a CPS is essential for measuring both individual and aggregate behavior. One approach to model this communication dimension would be to implement one or more native protocols within the federated experimental environment. This has the advantage of using the genuine communications stacks and hardware that the actual devices being tested or emulated utilize. An alternative approach would be to integrate one or more network simulators to emulate the set of required communication protocols.

The first approach to implement native protocols is straightforward. However, it has the following severe shortcomings:

1) difficult to create large network configurations,
2) difficult to reproduce network traffic between experiment runs,
3) difficult to inject protocol fault conditions into the network traffic,
4) lack of isolation from out of band traffic present in real world deployments.

These challenges inspired the creation of network simulation platforms such as OMNeT++ [10], NS2 [11], and NS3 [12]. These network simulators enable precise manipulation and measurement of the communication environment and increase the ease with which a given experiment can be reproduced. *In situ* communications tests are vulnerable to interference from coincident communications from uncontrolled sources, and varying latencies due to network design and routing. Such variances are important to consider in studying communications. However, they are best studied as control variables rather than asynchronous and non-reproducible environments.

With a network simulator as a co-simulator in a federated testbed environment, more accurate and reproducible test scenarios can be designed. For example, precise message injection is possible with a network simulator but not necessarily with an actual implementation of a communications network. Also, the injection of specific faults and scenarios becomes possible, and reproducible, using a network simulator.

3.3 Security and Privacy

The federates involved in remote federations can be at different geographical locations, and are typically resident on different network segments under different network security administrative domains. This is a characteristic of many Operational Technology (OT) architectures. Often, federates behind a firewall will be prohibited from accepting message traffic from federates located outside of the enterprise. To enable federation in this scenario typically requires the existence of an outgoing connection to establish a channel for communications with external federates.

Measurements in federated experiments require the same security and privacy protections as other applications in the areas of confidentiality, integrity, availability, and identity protection. Therefore, the federations that span multiple network segments must communicate through secured channels, especially when communications among federates cross enterprise boundaries.

In exposing interfaces to federates, there can be different federates with different levels of trust. For example, the developer of an experimental apparatus may utilize the federate interfaces for its own proprietary purposes. Yet, there may be some public uses of the same federate component. It should be possible to design such a federate once, where differing rights to access its interfaces can be determined at runtime. Therefore, it is efficient to build a federate interface that suits a variety of needs so that many special-purpose or experiment-unique versions do not need to be created.

For example, a manufacturer may create a federate that represents a proprietary device. In experiments on the internal network of that manufacturer, this federate may expose a great deal of information via its interfaces and these are useful in testing within the enterprise. Yet, there may be features of this interface that are beneficial to access during a collaboration with others outside the enterprise. Rather than creating a second interface for this federate that exposes a limited set of the capabilities to external users, Role-Based Access Control (RBAC) [13] can be used to expose a limited subset of capability based on the role of each collaborator. The owner role, with access to all the features of the federate, could continue to be used within the enterprise. This mixed-rights uses may be desirable with different roles being asserted within the same experiment.

Since interfaces between federates must exchange information within and outside a single enterprise environment, methods of connection must facilitate the traversing of such boundaries. RBAC permits a design it once and use with varying authorities on a per-role basis. This way, a federate can provide extensive access to authorized parties and less access for lower authorization classes even within a single federation.

Modern enterprises are protected by firewalls and other tools to defend against cyber-intrusion. Communications methods that allow outbound secure connections are more permissive than those allowing inbound communications. Therefore, communications models that permit federation exclusively through outbound secured communications, and without virtual private network tunnels, are desirable. Such communication channels should be constrained to only carry experiment-specific network traffic and not be open pipes through which unintended messaging can be injected.

3.4 Community of Collaborators

A requirement to facilitate widespread collaboration and evolution of a federated testbed architecture is that a community of developers and users forms around a common approach to identify problems, advance the technology, and increase the availability (and number) of integrated federate models.

In the absence of such a community, independent developments continually reproduce the same innovations to integrate another capability into an existing base. This produces the classic N x (N-1) permutation problem of interoperability translations. For example, if there are 5 solvers, each community has to create adaptors to integrate the other four into their scheme – thus 20 adaptations instead of 5. A robust community, should it be successfully formed, allows each new capability to be integrated once extending the scope and power of experimental capabilities that preceded it.

4 Universal CPS Environment for Federation (UCEF)

A major challenge to the federated testbed architecture discussed in this paper is the requirement for a common language and a set of development tools that facilitate development in that common language. The concept of a testbed-in-a-box is one option that could mitigate this challenge and provide a critically needed resource to the CPS research. Ideally, this would involve a software platform that handles most of the essential functionality of CPS testbeds such as monitoring sensors, controlling actuators, logging data, device interoperability, and co-simulation, as well as an extensible framework that allows different CPS applications to be developed on the platform. A facility could make this software available for download along with guidance on modeling/simulation tools, sensors, and actuators that have been tested to work with the platform. This could help research laboratories create testbeds much faster than the current approach of building testbeds from scratch with very little shared knowledge and experience. In addition, this could spur an open source community that builds applications with this platform, further increasing the adoption of CPS technology nationwide by domain experts and citizen scientists. This section describes the testbed-in-a-box developed at NIST called the Universal CPS Environment for Federation (UCEF).

4.1 Lightweight Wrappers – A Uniform Approach

One benefit of federated architectures is the ability to integrate simulators, emulators, hardware-in-the-loop and other domain-specific tools to take advantage of leading technical capabilities. To avoid the onerous requirement to re-implement each of these different entities in a common language, a means of lightweight integration is desirable that integrates a federate using a 'template' that depends only on the native language of the federate. The optimal concept is to provide only as much adaptation of an existing component as required for the exchange of information needed for a collaborative experiment. That includes being able to recognize its joining and departure, its ability to be synchronized with other federates, and to react to pauses and resume as the experiment progresses. Figure 2 illustrates how models built in different tool chains can be integrated using this concept of light adaptation.

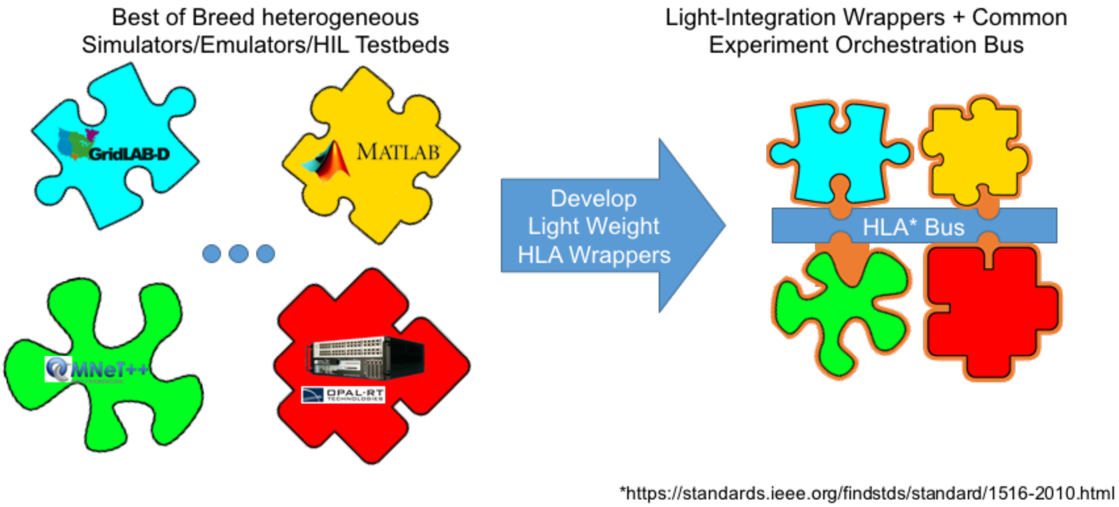

Figure 2 Lightweight Adaptation for Federation[2]

This figure illustrates four different types of domain-specific hardware and simulation entities, and how light wrappers can be used to integrate these entities to a common communication bus. By adding a thin layer of adaptation and a common communication means, models constructed on these otherwise separate platforms can be integrated for federated experimentation.

The initial release version of UCEF applies this concept of lightweight wrapping to incorporate several simulation engines with an implementation of HLA called Portico [14]. Support has been added for the simulators LabVIEW™ and Simulink™, as well as a grid simulator used in the energy industry called GridLAB-D. Network simulation is also possible in this initial release using the network simulator OMNeT++. In addition to these simulation engines, both the Java and C++ programming languages are incorporated into UCEF to support the development of software interfaces for hardware-in-the-loop and other software with socket and web application user interfaces.

4.2 Graphical Development Environment

Figure 3 shows the Web-based Graphical Modeling Environment (WebGME) developed by Vanderbilt University, which has been extended to support the development of HLA federations. Federates (green boxes) and messaging interactions (publications and subscriptions) between federates (white boxes) can be dragged in from a component palette and linked

[2] NIST does not recommend or advocate any particular technology for the purposes of federation. Specific technologies mentioned are for illustrative purposes only.

together to quickly create federations. All of the simulation engines wrapped into UCEF as discussed in the previous subsection are available for use in the component palette. A code generator integrated with the graphical environment can turn these models into executable code.

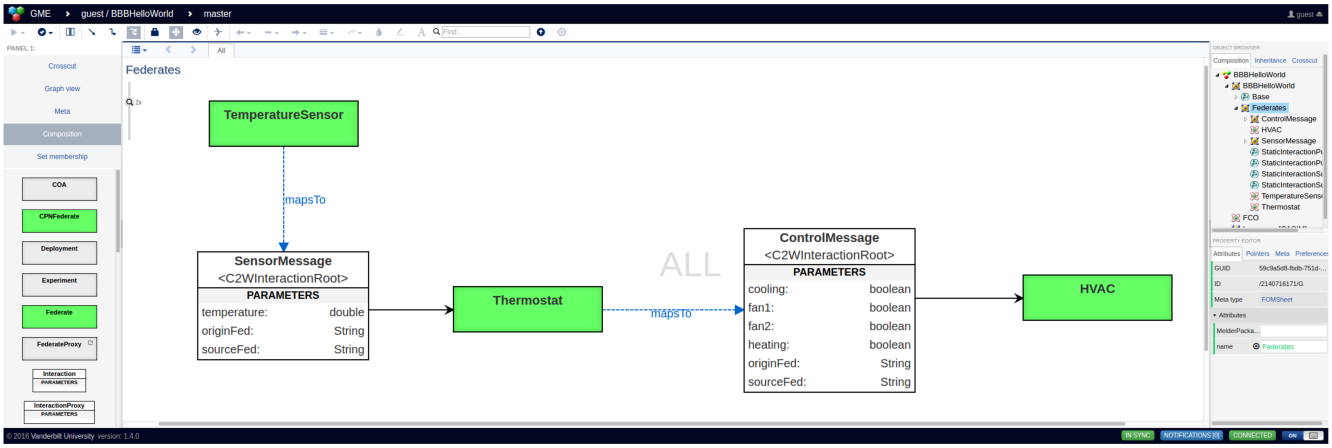

Figure 3 Web-based Modeling Environment in UCEF

4.3 Experimental Dashboards

Situational awareness of the progress of an experiment and its individual components is crucial when running federated experiments. Figure 4 illustrates how this situational awareness can have dimensions of geographic dispersion and federate status (center), dashboards of experiment status (bottom right), and graphical status of experimental measurements in progress (top left and right). This capacity for situational awareness resembles control center consoles deployed across all CPS sectors, and its addition to a federated testbed architecture allows research into the human factors of large scale CPS deployments.

Figure 4 Immersive Situational Awareness

The FIWARE platform [15] developed by an open-source community provides the capability to build experimental dashboards for CPS. This capability has been integrated into UCEF through development of a wrapper that allows interactions exchanged in an HLA federation to be pushed into the FIWARE environment.

5 Conclusion

UCEF presents a valuable assembly of technologies that enables the design, implementation, and execution of experiments through a federated architecture that is reconfigurable, reproducible, scalable, and usable.

By assembling this tool set in an easy to acquire and maintain form, better experiment design and more complex scenarios can be achieved.

NIST and Vanderbilt seek to inspire an Open Source Community around UCEF so that, as more adaptors are created and libraries of modeling components are shared, researchers and other stakeholders of CPS can benefit from these cumulative capabilities.

6 References

[1] NIST (June 2017) Griffor, E., Greer, C., Wollman, D., Burns, M., *Framework for Cyber-Physical Systems: Volume 1, Overview* (SP 1500-201), https://dx.doi.org/10.6028/NIST.SP.1500-201, NIST

[2] Columbus, L. (2017, November 27), Roundup Of Internet Of Things Forecasts And Market Estimates, 2016. *Forbes*, Retrieved on 7/10/2017 from https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#5664b318292d

[3] Rajkumar, R., Lee, I., Sha, L., Stankovic, J., (2010, June 13018), Cyber-physical systems: the next computing revolution,; *DAC '10* *Proceedings of the 47th Design Automation Conference*, Pages 731-736

[4] Roth,T., Song, E., Burns, M., Neema, H., Emfinger, W., Sztipanovits, J. (2017), Cyber-physical system development environment for energy applications, 2017 *Proceedings of the ASME 2017 11th International Conference on Energy Sustainability (ES2017)*

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

[6] IEEE. (2010). Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP) (1730-2010), IEEE Standards

[7] ANSI/IEEE. (1986). IEEE Standard for Software Unit Testing (Std 1008-1987), Page: 0_1, DOI: 10.1109/IEEESTD.1986.81001, IEEE Standards

[8] ISO/IEC/IEEE. (2013). Software and systems engineering Software testing Part 1:Concepts and definitions, (ISO/IEC/IEEE 29119-1:2013(E)), Pages: 1 - 64, DOI: 10.1109/IEEESTD.2013.6588537, IEEE Standards

[9] Semver.org. *Semantic Versioning*. Retrieved from Semver.org website http://semver.org/spec/v2.0.0.html

[10] OMNet++ *Discrete Event Simulator*. Retrieved on 7/10/2017 from https://omnetpp.org/.

[11] Sourceforge.net. *The Network Simulator – ns-2*. Retrieved on 7/10/2017 from SourceForge web site http://nsnam.sourceforge.net/wiki/index.php/User_Information

[12] Nsnam.org, *NS-3*, Retrieved on 7/10/2017 from nsnam.org https://www.nsnam.org/

[13] Ferraiolo, D.F. & Kuhn, D.R. (October 1992). "Role-Based Access Control" [PDF]. *15th National Computer Security Conference,* Pages 554–563.

[14] The Portico project, https://github.com/openlvc/portico

[15] FIWARE Foundation, https://www.fiware.org/

7 Author Biographies

MARTIN BURNS is a researcher at the National Institute of Standards and Technology (NIST). With his background in IEC and ANSI standards development for semantic models and data exchange, he has facilitated the development of the underlying Green Button technologies which define energy usage information and APIs in the Smart Grid in the US and internationally. He co-chairs the data interoperability working group for the NIST led Framework for Cyber-Physical Systems (CPS) and is a key contributor to NISTs architecture for federated testbeds for investigating the behaviors of CPS. Marty is also currently leading the IES-City Framework project for NIST. The IES-City Framework is an international collaboration with a goal of providing analysis that will help encourage convergence for interoperability of the Internet of Things (IoT), Cyber-Physical Systems (CPS), and Smart Cities.

THOMAS ROTH is working on development of the technology behind the cyber-physical testbed at the National Institute of Standards and Technology as a member of its Smart Grid and Cyber-Physical Systems program office. His research interests are in formal methods for the composition of cyber-physical systems, and the detection of compromised cyber-physical devices through comparison of their reported behavior against the constraints of the physical system.

EDWARD GRIFFOR is the Associate Director for Cyber Physical Systems in the Smart Grid and Cyber-Physical Systems Program of the National Institute of Standards and Technology (NIST). Prior to joining NIST in 2015, he was Walter P. Chrysler Technical Fellow for Electrical Engineering at DaimlerChrysler and later Fiat-Chrysler Automobiles where he founded and served as Chairman of the Chrysler Technology Council until 2015. Prior to joining the automobile industry, he was professor of mathematics and electrical engineering at multiple universities in Europe for two decades and was a key contributor to the mathematics that preceded 'formal methods' for assurance.

PAUL BOYNTON is a long-time researcher at NIST, where his work initially focused on DC-low frequency standards. More recently, he performed research with the Display Metrology Project at NIST and served as project leader. Beginning in 2009, he helped lead the effort in establishing and managing the Smart Grid Interoperability Panel (SGIP), created to accelerate the development of smart grid standards. Currently, he works for the Smart Grid and Cyber-Physical Systems Program Office, where he is manager of the NIST Smart Grid and CPS testbeds. He also manages the NIST Health IT Imaging Project.

JANOS SZTIPANOVITS is E. Bronson Ingram Distinguished Professor of Engineering at Vanderbilt University and founding director of the Institute for Software Integrated Systems. His primary research areas are model-based design of and design automation for Cyber-Physical Systems. He coauthored two books and over 350 papers in model-based design, model-integrated computing, design automation for cyber-physical systems, model-based system integration and adaptive systems. He is Fellow of the IEEE and external member of the Hungarian Academy of Sciences.

HIMANSHU NEEMA is a System Architect at Institute for Software Integrated Systems at Vanderbilt University. He holds a M.S. in Computer Science from Vanderbilt University. His primary research interests include Modeling & Simulation, Model-Integrated Computing, Distributed Simulations, Artificial Intelligence, Constraint Programming, and Planning & Scheduling. He has 19+ years of experience in research and development of software applications covering these areas and has coauthored 20+ research publications.