NIST Special Publication 1900-101

UCEF 1.0.0-ALPHA Kickoff Workshop

Workshop Report

Martin Burns
Thomas Roth

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.1900-101

C Y B E R - P H Y S I C A L   S Y S T E M S

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Special Publication 1900-101

UCEF 1.0.0-ALPHA Kickoff Workshop

Workshop Report

Martin Burns
Thomas Roth
Smart Grid and Cyber-Physical Systems Program Office
Engineering Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.1900-101

April 2019

U.S. Department of Commerce
Wilbur L. Ross Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

National Institute of Standards and Technology Special Publication 1900-101

22 pages (April 2019)

Publications in the SP 1900 subseries present information of interest to the cyber-physical systems (CPS) community, where CPS are defined as smart systems that include engineered interacting networks of physical and computational components. The series was established in 2016 by the Smart Grid and Cyber-Physical Systems Program Office of the NIST Engineering Laboratory to provide a separate identity for CPS and Internet of Things publications, including those concerned with the foundations of CPS, CPS testbed science, and CPS applications, e.g., smart grid, smart cities and intelligent transportation. The series reports on research, guidelines, and outreach efforts in CPS, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Special Publication 1900-101
Natl. Inst. Stand. Technol. Spec. Publ. 1900-101, 22 pages (April 2019)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.1900-101

Revision Tracking

Version	Date	Editor	Changes
1.0		Marty Burns	Original Version

Table of Contents

Table of Figures

Acknowledgement

The NIST Smart Grid and Cyber-Physical Systems Program Office would like to acknowledge the valuable leadership of the following individuals who helped guide the development of the Universal CPS Environment for Federation (UCEF) and the workshop activity:

NIST:

Eugene Song
Paul Boynton
Ed Griffor
David Wollman
Chris Greer
Cuong Nguyen

Vanderbilt University:

Himanshu Neema
Janos Sztipanovits
Gergely Varga
Yogesh Barve

Santa Clara University:

Hohyun Lee
Joseph Singer

Prometheus Consulting:

Geoffrey Roberts

Executive Summary

This report summarizes the results of a workshop held at NIST on July 27, 2017. The goal of this workshop was to introduce participants to the newly developed Universal CPS Environment for Federation (UCEF 1.0.0-ALPHA).

The ALPHA version of software is generally regarded as a first release that has limited operational experience with few users. Typically, as experience is gained the software evolves to BETA release that is more broadly used.

Attendees were provided with the UCEF software package, which is contained in a virtual machine. They were presented with the background on UCEF and the technologies necessary to operate the tool suite. In addition, they were led through an exercise using the tool suite to develop a joint federated experiment.

Universal CPS Environment for Federation

1. Overview

NIST has assembled a testbed for conducting research on smart grid and cyber-physical systems (SG/CPS) applications. A key component of this testbed is a software platform that enables federated experiments.

The NIST SG/CPS Program Office is working with an open source tool suite that can graphically model a federate (input/output) and generate the source code and build files for the federate based on its graphical representation. NIST calls this the Universal CPS Environment for Federation (UCEF). UCEF was developed through a collaboration with Vanderbilt University's Institute for Software Integrated Systems (ISIS).

UCEF provides a software development tool suite to build federated experiments. A federated experiment is one in which a set of independent *federates*, which are independent components of an experiment with their own simulation solver or source code, which can collaborate in a co-simulation where timing is managed, and data is exchanged between the federates as needed. A federate could be a piece of equipment or a simulation model. It can be in a local machine, in the cloud, or at geographically dispersed test facilities. Federates themselves can be co-located or distributed anywhere suitable Internet routing is available. For each solver a wrapper is developed that binds the simulation into the federated infrastructure. Example solvers include GridLAB-D™ [3] and LabVIEW™ [4].

The generated source code may be standalone, or it may have additional hardware or software dependencies that must be resolved prior to deployment. In addition to the source code, the tool suite can also generate experimental configuration files which tailor the runtime behavior of federates and include deployment information such as where the built executable for each federate should be run. Figure 1 shows a schematic representation of the tool suite and its generated products:

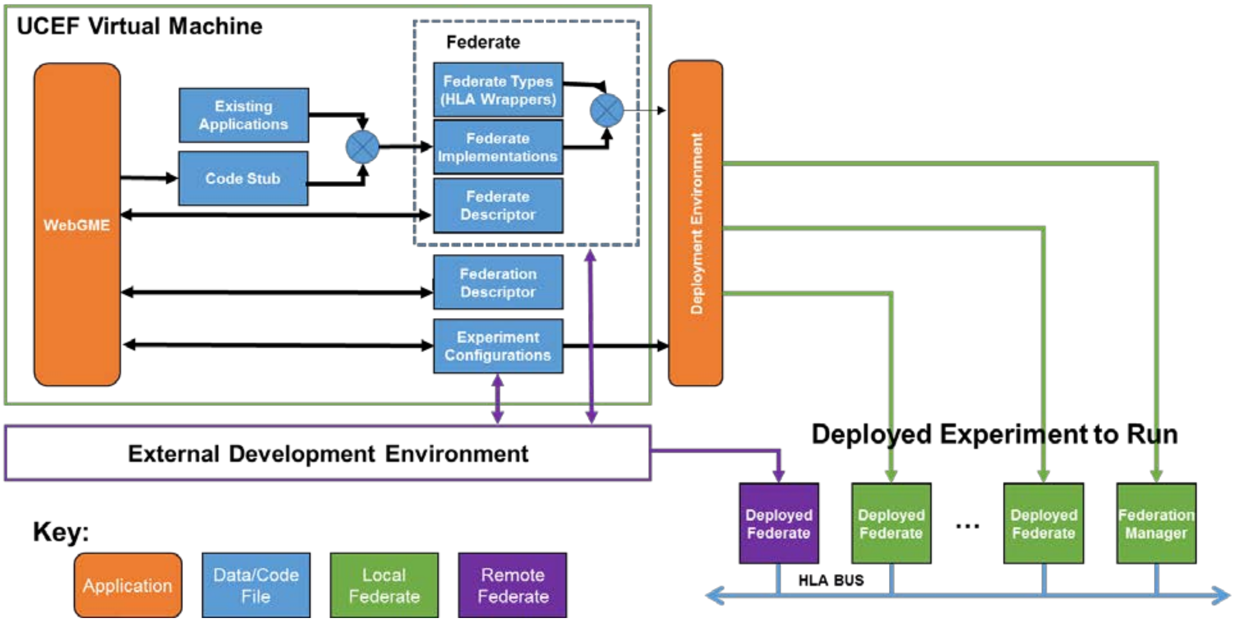

Figure 1: UCEF Composition

Shown in the figure, are the basic components of UCEF and the relationship between the tools and federated experiments:

- **UCEF Virtual Machine:** is a Linux virtual machine consisting of the operating system and a collection of software development tools to use to build federates, federations, and experiments. Core is the Web-based Graphical Modeling Environment (WebGME) graphical modeling environment. WebGME allows federates, federations, and experiments to be designed, and code generated.
- **An External Development Environment:** is used for actual federate implementation behavior. The scope of WebGME is exclusively software artifacts that allow actual federate implementations to be built on coding templates it generates in co-simulation platform-specific design tools.
- **A Deployment Environment:** tool is used to manage the deployment and running of experiments.

UCEF, at its core, utilizes an open-source implementation of the Institute of Electrical and Electronic Engineers (IEEE) Standard 1516-2010 High Level Architecture (HLA) for communications. This implementation is called Portico [1].

2. Discussion

2.1. The Capabilities of the Platform

The UCEF Platform was designed to support:

- The ability to rapidly adapt a diverse set of models and hardware systems for use as components (federates);
- The means for integrating these components, whether situated locally or connected via remote network or the Internet, into reusable and reconfigurable test systems;
- Tools for designing experiments for a given test system (WebGME and accessories); and
- The ability to run an experiment with real-time visualization and data storage for subsequent analyses.

Wrappers are now available for Java, C++, and GridLAB-D, with additional adapters underway to support LabVIEW, MATLAB, Omnet++, Courses of Action (COA), and federate unit testing.

The alpha version of the UCEF open source testbed software system (comprising more than 5 million lines of code) is freely available on Github under an open-use license, similar to an Apache 2.0 license as approved by NIST and Vanderbilt:

> Certain portions of this software are Copyright (c) 2006-2017 Vanderbilt University, Institute for Software Integrated Systems.
>
> Certain portions of this software are contributed as a public service by The National Institute of Standards and Technology (NIST) and are not subject to U.S. Copyright.
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above Vanderbilt University copyright notice, NIST contribution notice and this permission and disclaimer notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. THE AUTHORS OR COPYRIGHT HOLDERS SHALL NOT HAVE ANY OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

UCEF software is maintained on NIST’s GitHub site [2].

2.2. The Workshop Agenda

On July 27th, 2017 NIST held an introductory 1-day workshop on UCEF. Workshop participants included representatives from industry and academia, including eight groups with extensive experience in testbed architectures. The SG/CPS program's intention was to introduce the audience to the UCEF concepts and technologies and to focus on an exercise that provided hands-on experience.

The workshop agenda was composed to have the following segments:

- **Introduction:** this segment provided an overview of the NIST CPS Testbed effort in Gaithersburg and NIST invited our Vanderbilt collaborators to present their perspective and describe their role in developing the UCEF package;
- **Technical Context:** this segment provided an overview of the UCEF architecture and a brief introduction to IEEE 1516 HLA that is the underlying communications protocol used in experiment federation.
- **Exercise:** a participatory workshop exercise was conducted in which each participant was guided through the installation of UCEF on their own systems along with an initial shared experience of running a collaborative experiment involving about two dozen laptops connected over WiFi in the workshop venue.
- **Review:** this portion provided a review of the set of initial features of the UCEF platform that described the number of prominent simulation environments that UCEF would initially integrate. A key goal of this segment was to encourage the voluntary collaboration of the participants in the expansion of UCEF through their own efforts.
- **Collaboration:** finally, a section was held for collaboration opportunities and how interested parties could collaborate with NIST.

2.3. The Workshop Exercise

It was desired to engage the participants in an exercise that was both understandable and compelling. To do this, we constructed the following use case:

> Participants make house federates that collaborate with common weather and utility federates. These house federates implement a thermostat algorithm and a simplified dynamic model of the Net-Zero Energy Residential Test Facility (NZERTF) (provided by Farhad Omar, Steve Bushby and David Holmberg) [5][6]. The utility federate has two sources of power: bulk and solar each at different price. Based on the availability of solar energy and the demand from the houses, the composite current price of electricity is computed and distributed to the houses. Weather data such as solar insolation and outside air temperature is sourced for Gaithersburg, Maryland during the period of summer on July 15 of a typical meteorological year.

Then, with this use case, we sought to demonstrate the following features of UCEF:

- Install and run UCEF Virtual Machine;

- Design a federate using WebGME;
- Insert custom code in the designed federate;
- Build and run the federate; and
- Collaborate in a federated experiment (federation).

Figure 2: Workshop Exercise Design illustrates the working collaborative experiment.

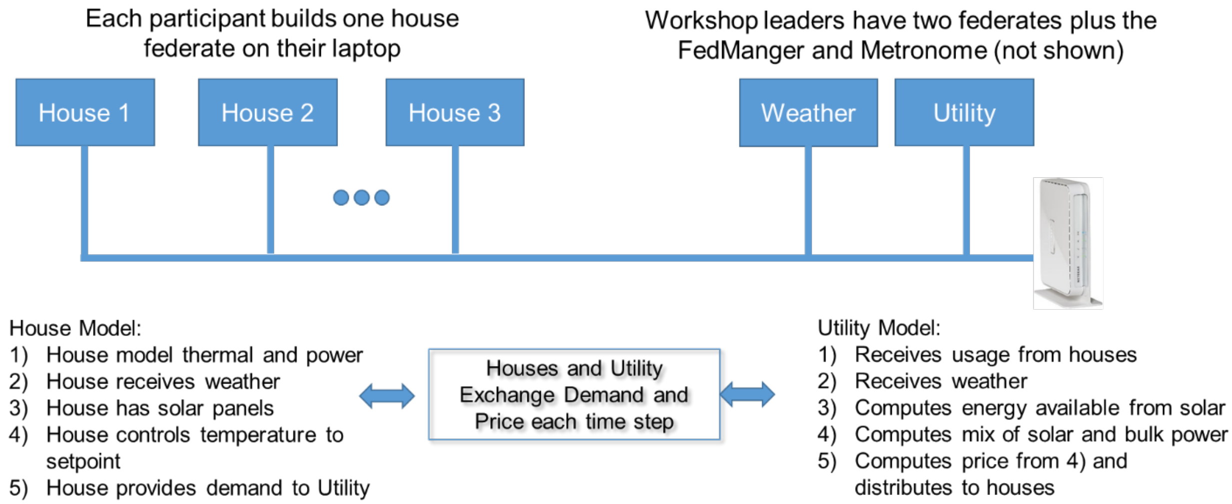

Figure 2: Workshop Exercise Design

The simulation in the workshop exercise had the following federates:

- **FedManager:** Leader of the experiment which is a required component for UCEF federations and is responsible for keeping track of federates in the experiment and metering time during the experiment.
- **Metronome:** Responsible for publishing the calendar time during the experiment. Since houses may join asynchronously and come and go, they need to acquire the current calendar time associated with the logical time progression administered by the FedManager.
- **Weather:** The weather federate publishes weather information as the experiment progresses at each logical time step.
- **Utility:** Represents the utility modeling, the supply of bulk power, and real-time price computation.
- **House:** Represents a model of a house with optional solar power inverter.

2.3.1. Utility Federate

The utility federate represents an electric utility that receives loads published by the individual houses and determines price. It has bulk electric power and power from a solar array. It computes price by combining its own solar resource with its bulk generation resource and corresponding pricing using a weighted average based on total demand and energy available from sunlight.

The price of electricity in \$/kWh is computed by:

$$Price = C_{fp} + \frac{(W_{bpused} * C_{bp} + W_{sol} * C_{sol})}{\sum W_i},$$

where:

C_{fp} = is the fixed cost of utility bulk power [0.04 \$/kWh]
C_{bp} is the incremental cost of bulk power [0.07 \$/kWh]
C_{sol} is the cost of solar power [0.00 \$/kWh]
W_i is the current house demand [W]
W_{sol} = f(Weather) [W]
$W_{bpused} = \sum W_i - W_{sol}$

2.3.2. House Federate

Each participant in the exercise runs one instance of the house federate. This house federate runs a simplified model where thermal resistance and heat capacity of the home are governed by bulk regression constants. The parameters governing the performance of the house simulation were contained in a JSON configuration file used by the house federate.

The house federate is a simplified model of the Net-Zero Energy Residential Test Facility (NZERTF) [5] located on the campus of NIST in Gaithersburg, MD. The simplified model of the house was obtained by analyzing the performance of the NZERTF over time reported in *"A Self-Learning Algorithm for Temperature Prediction in a Single Family Residence"* by Omar and Bushby[6]. This paper provides for a simplified first-order building model with properties of thermal mass, thermal resistance, heatpump capacity and performance.

The key regression constants of UA_e and τ_e came from that work. The change in indoor temperature T_i at time step k is computed as:

$$T_{i,k+1} = T_{\infty,k} + \frac{Q_{h,k}}{UA_e} + \left(T_{i,k} - T_{\infty,k} - \frac{Q_{h,k}}{UA_e}\right) \exp\left(-\frac{\Delta t}{\tau_e}\right),$$

where:

T_i is the indoor temperature [°C]
T_∞ = Outdoor Temperature retrieved from weather federate as dryBulbTemperature [°C]
Q_h = Total heat load on house – Heat pump, plug loads, occupants, … [W]

Parameters from fit to NZERTF:

$1/UA_e$ = Lumped Thermal Resistance
τ_e = Time constant
UA_e = 172.0 [W/°K]
τ_e = 104.0 [h]
hpbasepower = 1571.0 [W]
houseeffectivearea = 6.84 [m^3]

sghc = 0.89 the solar energy conversion efficiency [dimensionless]

Heat pump demand computation can be performed:

P_{hp} is the power consumed by heatpump when it is on [W] (see thermostat algorithm below)
= 7.1204*(T_∞*9.0/5.0+32.0) + hpbasepower
Q_{hp} is the thermal energy pumped by heatpump [W]
= -(323.24 * (T_∞*9.0/5.0+32.0) + 10576.0) / 3.412

A solar panel contribution of power to the house:

SolarInsolation is retrieved from weather federate and is the direct normal irradiance [W/m^2]
Q_{solar} is the heat gain from the sun [W]
= SolarInsolation * houseeffectivearea * sghc
P_{solar} is the power output from the solar panels for the house [W]
= − SolarInsolation * solarconversionconstant * solarpaneleffectivearea

A fixed load is defined:

$P_{fixedload}$ representing constant power consumption [W]

Total energy consumed by house:

$P_h = P_{hp} + P_{solar} + P_{fixedload}$

Total thermal energy flux for the house:

$Q_h = Q_{hp} + Q_{solar} + Q_{fixedload}$

Compute the indoor temperature due to heat pump and solar gain:

$T_i = T_\infty + Q_h/UA_e + (T_i - T_\infty - Q_h/UA_e) * \exp(-\text{logicalTimeSec}/\tau_e)$

Then, a thermostat algorithm can be performed (cooling only for summer):

if($T_i > T_{sp}$ + del)
heatpump on
else if ($T_i < T_{sp}$ – del)
heatpump off

where:
del is the hysteresis for thermostat algorithm [°C]
T_{sp} = comfort setpoint [°C]

2.3.3. Weather Federate

The weather federate makes use of the National Renewable Energy Laboratory (NREL)'s Typical Meteorological Year (TMY3) weather data format available for 1000 US cities[1] [7]. This federate, based on the current logical time step computes the current weather statistics (about 70 measurements including temperature, pressure, humidity, solar insolation) and interpolates the hourly NREL data to the current calendar time and temporal resolution of the simulation.

2.3.4. Metronome Federate

UCEF federates execute in logical time steps. For this experimental federation, there is a need to have a notion of clock or calendar time.

The Metronome provides the correspondence between logical time zero and calendar time. For the workshop exercise, this was July 15 at 00:00:00 local time. Additionally, it provides the number of seconds represented by each logical time step. In the case of the workshop exercise, this was 15 min (900 s). By this means, the simulation can operate as if it is July 15 in Gaithersburg, MD.

2.4. Executing the Workshop Exercise

Thomas Roth from NIST led the workshop participants through a 4 h hands-on activity using the UCEF 1.0.0- ALPHA release. The workshop exercise was intended to give a positive first impression of the UCEF software platform, to serve as a tutorial on federated experiment design, and to result in each participant having UCEF 1.0.0-ALPHA installed on their personal machine after the workshop had concluded.

2.4.1. Exercise Format

The exercise was conducted in a lecture style room with a computer running a UCEF 1.0.0-ALPHA virtual machine projected to the front of the room. The participants were led through each step of the exercise through live demonstrations on the projected laptop. A team of 7 people familiar with UCEF was available to guide audience members and provide support for technical problems encountered during the exercise. The members of this support team were Martin Burns, Geoffry Roberts, Hohyun Lee, Joseph Singer, Himanshu Neema, Gergely Varga, and Yogesh Barve. The exercise consisted of 4 major sections described below.

2.4.1.1. Installing the UCEF 1.0.0-ALPHA Virtual Machine

Each participant was given a USB flash drive that contained an image of the UCEF 1.0.0-ALPHA virtual machine and installers for software to load the image called Oracle VM VirtualBox for Windows, Ubuntu, and OS X operating systems. The participants were instructed

[1] Note: most recent update to database was 2015.

to install the version of VirtualBox appropriate for their machine and then launch the UCEF virtual machine.

2.4.1.2.Running an Example Federation

The UCEF 1.0.0-ALPHA virtual machine was packaged with a pre-built example federation that could be executed from a Bash script. The script ran a collection of federates that interacted with a GridLAB-D distribution grid simulation to drive a transaction energy scenario. Martin Burns from NIST gave a 20-minute presentation that introduced the scenario to the exercise participants, and Thomas Roth demonstrated how to run the scenario in the UCEF virtual machine and inspect its results.

2.4.1.3.Modeling a Federate from Scratch

A significant portion of the workshop exercise involved using WebGMD in the UCEF virtual machine to model a federate from scratch. All participants received a dense 2-page handout that described all the steps required to design and code-generate a federate using UCEF. Thomas Roth demonstrated the steps described in the handout at the front of the room.

This portion of the exercise was broken into 7 short presentations, each lasting between 5 min to 10 min, which demonstrated one step in the federate design process. After each presentation, the participants were given time to repeat the steps on their local machines and ask questions of the support team assembled for the exercise. Each step built upon the results of the previous one such that after all the presentations were completed each participant had modeled and produced software code (Java) for a house federate on their local machines.

The 7 steps demonstrated how to:

1. Launch WebGME and create a new project;
2. Add a federate model to the blank project;
3. Define a new message (in this case an HLA Interaction);
4. Import an existing interaction from another project;
5. Connect the interactions to the federate model;
6. Generate code for the modeled federate; and
7. Import and inspect the generated code in Eclipse.

The exercise did not require implementing the behavior of the federate created. How to implement, build, or run a custom designed federate was left to the participant off line both due to the high chance of mistakes and in the interest of time. However, the 2-page handout distributed for this section contained an appendix which provided information on how to compile and execute the generated code. A prebuilt federate was provided for the participant to run in the joint experiment described in the next section.

2.4.1.4.Joining a Group Federation

The final section was a group exercise where participants were encouraged to join a federation started by the computer linked to the projector display in the front of the room. A wireless router had been set up and participants were given login credentials to connect their machines to that wireless network. Another Bash script [9] was packaged in the UCEF virtual machine which launched a single federate on that wireless network, using a configurable name. Each participant was instructed to connect to the wireless network and run the script using a name of their choosing. The participants would then see their selected name join the federation in the textual output projected in the front of the room.

2.4.2. How the Exercise Progressed

The exercise design involved several first-time activities for the UCEF team:

- Running federates over a wireless network;
- Joining two dozen laptops in a single experiment; and
- Scaling to a 30-federate experiment.

Overall, the exercise was a substantial success with almost every participant successful at participating in the joint experiment.

During the exercise, however, a few problems arose with the installed software that prevented all participants from joining the network. A few other technical problems also occurred, but in most cases they were resolved during the workshop exercise. The following subsections describe the successes and challenges we had during the exercise.

2.4.2.1.Successes

All participants successfully installed UCEF 1.0.0-ALPHA on their personal laptops with operating systems that included Windows 7, Windows 10, OS X, and Ubuntu. All participants successfully completed the exercise to design and generate code for a Java federate. Around one-third of the participants could join the group federation over the wireless network. All steps of the exercise were completed on time.

The format of the workshop exercise engaged the audience. Almost all the participants closely followed and imitated the demonstration project. Several questions were asked after each of the 7 exercise presentations, and over 20 questions were asked by multiple audience members over the course of the workshop exercise.

The 2-page handout was extremely effective and well received. The participants received the handout an hour before the hands-on portion of the exercise, and several audience members had already gone through and completed the design of their federate before receiving the exercise and with minimal assistance from the support team.

2.4.2.2.Technical Problems

One participant received a corrupted USB flash drive during the UCEF 1.0.0-ALPHA installation that prevented successful launching of the virtual machine. It took some time to discover the cause of the problem, but it was easily resolved using a spare flash drive. Over 50 flash drives were produced in preparation for the workshop exercise, and this was the only instance where one of the drives was reported as corrupted.

Three participants were unable to launch the UCEF 1.0.0-ALPHA virtual machine after installation due to insufficient random-access memory (RAM). In one case the computer crashed when trying to launch the virtual machine. The image distributed at the workshop uses 4 GB of RAM and expects a machine with at least 8 GB of RAM available. However, this number was selected empirically and the actual minimum memory requirements for UCEF for the workshop experiment was less. These cases were all resolved by lowering the RAM usage of the affected virtual machine to 2 GB.

One participant with a Windows 10 laptop was unable to load the UCEF 1.0.0-ALPHA virtual machine due to a known issue with 64-bit virtual machines in Windows 10. Some releases of Windows 10 include the Hyper-V virtualization technology that prevents the creation of non-Windows 64-bit virtual machines. UCEF is based on 64-bit Oracle Virtual Box [10], and cannot be used together with Hyper-V. This issue was not resolved, but the affected participant used another laptop with an older version of Windows.

Two technical problems were encountered during the group wireless federated exercise. One participant was unable to connect to the wireless router, and at least one participant was unable to join the group federation despite being on the wireless network. Neither of these issues could be resolved and their root causes were unknown.

One participant attempted to work through the appendix that described how to build and run a federate and was unsuccessful despite assistance from the support team. This portion of the handout was incomplete and missing several critical sections necessary for compilation of a Java federate. These sections were later revised after the workshop had concluded, but there was no resolution during the workshop exercise itself.

2.4.2.3.Lessons-Learned

A significant amount of effort went into assembling a competent technical support team that was ultimately underutilized during the workshop exercise – primarily because the exercise went more smoothly than anticipated. Five members of the support team received a 3 h training course the day prior to the workshop. A takeaway from this workshop is that more effort could be expended to make the audience aware that a support team is available to answer questions and instruct the audience members on how to flag down support. Additionally, perhaps a more challenging degree of development on the part of the participants would have required more support and perhaps resulted in more knowledge uptake.

The topics covered in the workshop exercise did not include how to run a federation. One audience member was disappointed that no discussion on how to run the federation manager and create their own federation was included in the exercise. Several participants were directed to online documentation on the federation manager, but it’s unknown how many individuals accessed the documentation during the workshop.

The largest number of technical problems occurred during the wireless group part of the exercise. This section of the workshop exercise was also the shortest, lasting no more than 15 minutes. Given the amount of time spent on technical problems related to the wireless network, the smaller number of participants who were successful in joining the federation, and the comment on the federation manager, it would make sense to expand this section and add a tutorial on the federation manager in future events. Two additional segments could be added to the 2-page handout to describe how (1) launch the federation manager, and (2) configure a federate to join the federation.

2.4.3. Summary of Attendee Interactions

Most attendees, as previously described, met our desired profile of potential collaborators. We had attendance from representatives of 2 commercial firms, 3 national labs, 2 participants from other parts of NIST, and 2 universities. Generally, the attendees saw that UCEF was a concept that they could benefit from with the caveat that they needed to see evidence that it would be persistently advanced and maintained. There was also interest in being able to integrate domain-specific tools from their own work. There was a significant variation in the technical depth of the participants with respect to testbed platforms. Some participants were potential users of UCEF while some participants were architects themselves and saw how they could use the tools to bridge their own developments.

3. Conclusion and Next Steps

We indicated to those present that the SG/CCPS Program seeks their engagement in a community of common practice around UCEF and its ancillary technologies. For this purpose, we devised a set of activities and technical and collaborative components to use.

Our plan is to:

1. Manage the source code base on GitHub and its public repositories;
2. Use the UCEF main GitHub project to engage the community in reporting issues and suggesting improvements;
3. Use github fork and pull-request model to allow the community to contribute to the project and address defects or shortcomings as observed;
4. Use NIST Drupal presence for static information about UCEF and its goals and progress going forward;
5. Use a mail-list set up as ucef@list.nist.gov to communicate directly with a growing community of interest;
6. Complete the 1.0.0 features we had promised:
    a. COA
    b. Omnet++
    c. LabView
    d. Matlab/Simulink
    e. Gridlab-D
    f. Unit Test

   g. ucef-gateway;
7. Upon completing each of these features, conduct a planned webinar to introduce the feature and use it as an opportunity for community feedback on UCEF;
8. Additionally, the NIST team will begin several sample experiments using the CPS TestBed and the UCEF software to explore the uses of UCEF for CPS and to further refine and debug the various interfaces provided. Demonstration experiments currently underway include:
   a. Interaction of residential energy management systems with local energy markets and a general-purpose, transactive energy test system. This project will realize the set of re-usable federates based on the NIST TE Challenge Abstract Component Model [11]. By creating these components, various degrees of complexity in dynamic retail energy markets can be studied and evolved;
   b. Planning for an intelligent vehicle simulation is underway and will bring together elements of an advanced automotive system to evaluate performance of networked vehicle components;
   c. Planning is underway for a secure and scalable sensor fusion experiment where sensors are added over time, resulting in an increase in measurement accuracy. . This project will study how aggregate behaviors can allow for more precise measurements, and detect tamper and defective values based on group behaviors; and
9. A completed revision will produce a more mature version of UCEF for broader distribution in autumn of fiscal year 2018 via a second workshop or webinar.

4. References

[1] The Portico Project, GitHub, https://github.com/openlvc/portico

[2] UCEF GitHub presence, GitHub, https://github.com/usnistgov/ucef

[3] GridLAB-D, PNNL, retrieved from https://www.gridlabd.org

[4] LabVIEW, National Instruments, retrieved from http://www.ni.com/en-us/shop/labview.html

[5] Net-Zero Energy Residential Test Facility (NZERTF), retrieved on 9/27/2017 from https://www.nist.gov/el/net-zero-energy-residential-test-facility

[6] Omar, F., Bushby, S., NIST, (September 2015), *A Self-Learning Algorithm for Temperature Prediction in a Single Family Residence*, http://dx.doi.org/10.6028/NIST.TN.1891

[7] National Solar Radiation Data Base, *1991- 2005 Update: Typical Meteorological Year 3*, retrieved on 9/27/2017 from http://rredc.nrel.gov/solar/old_data/nsrdb/1991-2005/tmy3/

[8] TE Challenge Component Model, GitHub, https://github.com/usnistgov/TEChallengeComponentModel

[9] GNU Open Software Foundation (2016). The GNU *Bash Reference Manual* (Version 4.4) retrieved on 9/27/2017 from https://www.gnu.org/software/bash/manual/bash.html

[10] Oracle Virtual Box, *Virtual Box*, retrieved from https://www.virtualbox.org/

[11] NIST (2018) TE Challenge Component Model: retrieved from https://github.com/usnistgov/TEChallengeComponentModel