A Cloud-based Development Environment using HLA and Kubernetes for the Co-simulation of a Corporate Electric Vehicle Fleet

Kasim Rehman, Orthodoxos Kipouridis, Stamatis Karnouskos,
Oliver Frendo, Helge Dickel, Jonas Lipps, and Nemrude Verzano

***Abstract*— Decision makers in modern enterprises need to assess so-called "what-if" scenarios and select the best of breed among the available alternatives. Although simulation plays a pivotal role in niche cases, it is an undervalued tool in the broader enterprise context. Its proliferation has not kept up with recent technological advances such as big data, cloud computing, graphics processing units clusters, and cross-layer enterprise integration. Based on such new developments, new capabilities can be realized that go beyond single-purpose simulations, and are an excellent fit for studying heterogeneous, independently developed System-of-Systems (SoS), in a multitude of scenarios in complex and dynamic enterprise environments. Co-simulation efforts can provide new insights on the enterprise operations including compelling visualizations of possible simulated alternatives, thereby assisting decision makers in their strategy selection. To move beyond niche applications, however, simulated systems need to integrate in real-time, a continually increasing amount of data (both real and simulated), stemming from various domains and their systems. This work presents a way to achieve such distributed simulations in modern enterprise environments, based on High-Level Architecture (HLA), as well as their coordination via the Run-Time Infrastructure (RTI). In an example scenario featuring a co-simulation of a corporate electric vehicle fleet, it is shown how various software subsystems interact to enable business users to visually investigate scenarios, as well as how such a system can be deployed and operated within a modern enterprise IT landscape.**

I. Introduction

Modern companies operate in highly dynamic and competitive environments, in which they strive for high performance. However, to be able to capitalize on market opportunities and achieve superior performance, companies need to be able to have fine-grained visibility of assets and processes, timely integration of real-time data, as well as overall resource utilization, in the best possible way to achieve the firm's strategic objectives [1]. For this to be realized, C-level decision makers need to be able to anticipate and act on emerging trends quickly.

Visualization of hypothetical future situations (so-called "what-if scenarios") can contribute to this goal, by making their impact more tangible and comprehensible without the necessity of in-depth technical knowledge. Moreover, key situations can be exemplified which in turn may help C-level executives to make decisions that guide the enterprise effectively. The backbone of such capabilities lies with simulations that can enable experimentation with reality-near hypothetical situations, that can eventually help the management to take better decisions. Furthermore, coupling simulation and modern 3D or even augmented/virtual reality visualizations can more effectively bring across key points and empower decisions makers.

While the importance of simulation for the enterprise is evident, it is also becoming increasingly challenging to realize. Enterprise systems in many cases constitute of heterogeneous, large-scale systems that are independently developed and deployed, and are expected to seamlessly interact with each other, which can be perceived as a Systems-of-Systems (SoS) [2]. Modeling the behavior of such a SoS is a complex task, and requires deep expertise, which becomes even more challenging if several of them need to be simulated and capture their interactions. Hence, a more realistic approach is to bring together their simulators into a common landscape and couple them over a common communication-driven infrastructure, eventually creating a *co-simulation* [3]. Therefore, a way to integrate multiple available simulators and interact with them in a coordinated fashion over a common infrastructure is highly desired. For enabling co-simulation, the High-Level Architecture (HLA) [4], an IEEE standard for distributed simulation, can be utilized.

This work presents efforts towards bringing together several simulated entities, i.e., in a co-simulation [3], over a framework based on HLA, as well as their coordination via the Run-Time Infrastructure (RTI), the middleware implementing the HLA. Specifically, the approach focuses on enabling the execution and deployment of containerized HLA federations (based on the RTI-infrastructure), as well as simulation-supporting software components, in form of microservices, for instance for model provision and preprocessing. The goal is to shed some light on how modern technologies can be used to (i) enhance visibility on company assets via visualization, and (ii) enable co-simulation, over a common infrastructure utilizing modern cloud technologies.

To exemplify the proposed approach, the case of Electric Vehicle (EV) fleet behavior is investigated. Company car fleets are becoming increasingly electrified, and as such opportunities are sought to capitalize on these assets [5]. However, corporate EV fleet behaviors are still not well understood, and decisions pertaining to infrastructure, cost, business process impact, company policies etc. need to be better supported [5], [6]. Therefore, not only existing behaviors need to be visualized, but also future scenarios need to be studied in order to assess the impact of various parameters. A corporate EV fleet scenario is implemented with the

All authors are with SAP, Dietmar Hopp Allee 16, D-69190, Walldorf, Germany {name.surname}@sap.com

utilization of multiple simulators and on the development infrastructure proposed.

The rest of the paper is structured as follows: First, the motivation scenario and technologies are shortly described in section II. The overall integration architecture is discussed in section III, while the specific extension and realization of it is exemplified for the EV use case in section IV. Finally, an overall discussion of conceptual and technical experiences is realized in section V, while section VI presents the conclusions or this work.

II. Technologies

A. Co-Simulation: HLA and RTI

Co-simulation provides the theoretical basis as well as a set of techniques and tools that enable a global simulation of a coupled system via the composition of various simulation units [3]. A comprehensive simulation and subsequent presentation of the outcomes is key to achieve a better understanding of potential future situations. For realizing inter-operable co-simulations, the HLA has been proposed and standardized, with the latest version published in 2010 and called *HLA Evolved* [4]. HLA is a simulation systems architecture framework with an emphasis on re-usability and interoperability of simulations, driven by the need for cost-effectiveness, quality and timeliness [7]. In this context, HLA has already been utilized in several industrial efforts [8].

In distributed environments, mediators are used to enable indirect communication among the participants, and in HLA this role is carried out by the *RunTime Infrastructure (RTI)*, which provides common interface services for synchronization and data exchange. Each executed simulation that adheres to the HLA interface to RTI is called *federate application* and shares a commonly specified data communication, i.e., the federation object model (FOM) Document Data (FDD). The federate applications and the FOM that are used to achieve some specific objective, are called *federations*. It should be noted that the RTI itself does not provide any storage but only mediates the data routing (i.e., the exchange of HLA objects) among the federates, using a publish/subscribe pattern. Each federation also includes a *Federation Manager* component that assumes the coordinator role and among other tasks, it creates the federation, manages synchronization points, etc.

To enable the interfacing of the federate with the RTI, a Local RTI Component (LRC) provides the ability to the federate, to make calls and get callbacks from the LRC. In some cases, a Central RTI Component (CRC) that coordinates the runtime components, is also part of the deployment. However, not all RTI implementations include a CRC [9]. While the HLA defines an interface which federates use for their data exchange, the exact implementation, is RTI dependent. Therefore, the current common practice is for all federates use the same RTI implementation, and in our case, it was decided to use the Portico RTI [10].

B. Deployment: Container Orchestration with Kubernetes

Modern enterprise IT environments are shifting towards containerized applications (e.g., in Docker), leveraging a lightweight approach to encapsulate an application's dependencies and operating environment. The resulting container image is portable, and can easily be distributed and deployed on existing IT infrastructure and across arbitrary IaaS providers. The increasing complexity and networking of large-scale containerized applications, calls for effective management and deployment tools. Kubernetes is an open-source system that adds container orchestration capabilities and enables operators to easily automate their deployment, scaling and management within a cluster of physical or virtual machines [11]. Furthermore, Kubernetes allows developers to declare what their application needs to be successfully deployed and operated, by providing information about required resources (e.g., CPU, GPU, and memory), exposed ports, image versions and more. The engine interprets these deployment descriptions and creates *Pods* that feature one or more containers including the prepared application and provide access to other resources such as disk storage or network. This approach enables automated lifecycle management and deployment of applications, while it also adheres to modern enterprise IT needs for monitoring and scalability.

C. Visualization: AR/VR

To allow for powerful 3D visualizations of virtual worlds, game engines such as Unity 3D [12] can be a great candidate. Unity 3D is a cross-platform game engine which can be used to create 2D and 3D graphics via its scripting API. Unity 3D has already been utilized as a visualization tool in different areas [13] and is used thanks to its support of various AR and VR platforms for several industry-specific use cases [14].

III. Integration Architecture

In order to address the challenges regarding large-scale integration and scalability of heterogeneous co-simulation, Figure 1 presents a high-level architecture of a distributed simulation system, based on HLA, that can be deployed and scaled using cloud technologies. The Fundamental Modeling Concepts (FMC) notation [15] is used to visualize the main architecture parts and their interactions.

Each independent simulation unit (simulator) is implemented as a federate (shown as "Simulator Federate" in Figure 1) and consists of two main software components that are containerized. The first container includes the simulation program itself, and the second acts as a connector to the RTI infrastructure. This RTI-connector container includes the required from HLA components, such as a federate ambassador, a controller, as well as the local RTI component (LRC). The federate controller does the bulk of the work, performing common HLA actions, and also communicates with the simulation controller (depending on the interfaces that are exposed on the simulator side). As shown in Figure 1, a controller in each federate takes care of registering with the RTI. The controller is responsible for sending and processing of information so that the actual simulator

Preprint version

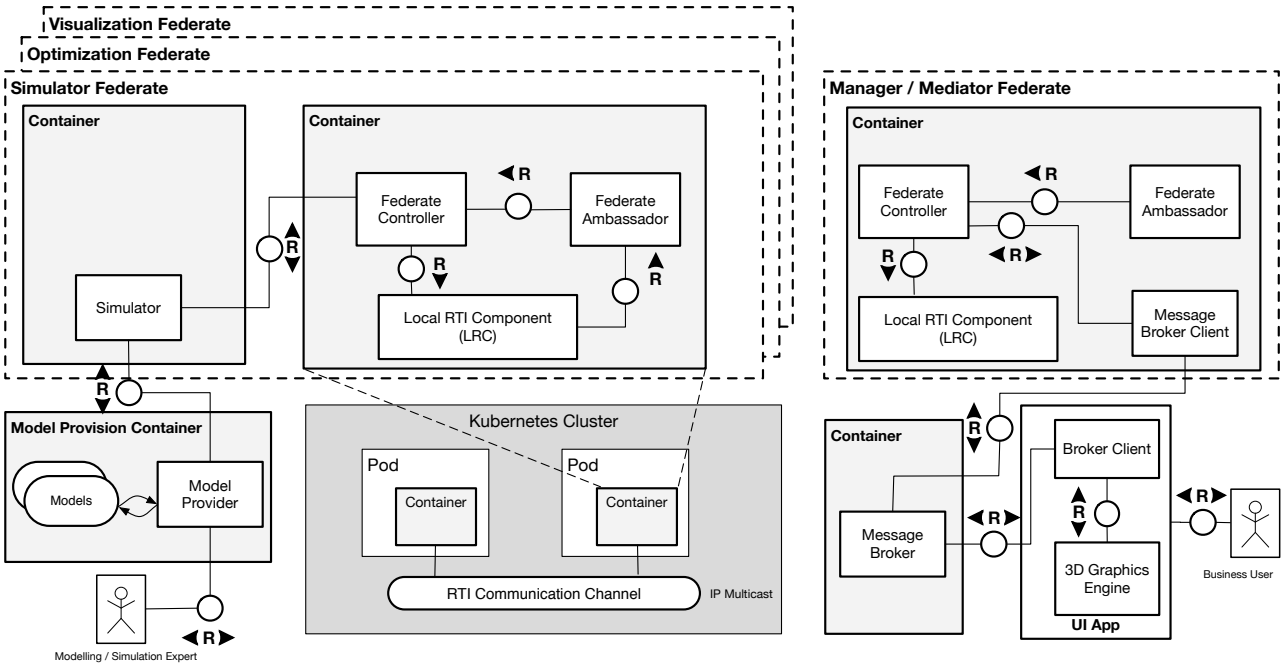

Fig. 1. Overall Architecture of the cloud-based co-simulation environment.

can remain oblivious of the RTI. This architecture ensures that all simulators whose corresponding federates participate in the same federation move in lockstep throughout the simulation. For the simulators to move in lockstep each needs to provide a `do_next_timestep` method. Furthermore, it is the controller who controls the respective simulators (or simulation client) via their API, with commands and parameters in response to the messages received from the RTI via the federate ambassador. The RTI expects each federate to provide a federate ambassador component for callbacks to the synchronized data and interactions.

The RTI is designed to be decentralized with each federate communicating via IP multicast to other federates. RTI providers such as Portico encapsulate all this functionality in libraries with a well-defined interface that each client can use to access it [16]. Behind this interface lies a local RTI component (LRC), that is responsible for the message exchange among the federates.

Next to simulators, other components can be also coupled using the RTI-connector (containing the Federate Controller and Ambassador, as well as LRC) to get access to the simulation data available on the RTI channel. Examples can be optimization and visualization tools. The proposed approach enables the deployment of optimization algorithms that work with the simulation data, and provide feedback during runtime to the simulators. An example of such an application is given in section IV.

Communicating with each federate but in a separate container, resides a repository that stores and serves the system models used for each simulator. This provides a separation of concerns to the co-simulation system. Stakeholders with the domain knowledge to design accurate models of the simulated systems can create and upload them into the repository, without the need for expert knowledge for the overall co-simulation system.

A prerequisite for enhancing the decision making in the enterprise through co-simulation, is the ability to expose simulation data and results, to external systems and users, so that useful conclusions can be extracted. Examples of such systems can be visualization, data analytics or machine learning tools. Although the RTI is designed to be decentralized, it is not unlikely that some orchestration of the various federates is required in the case of complex scenarios. In order to perform these two tasks, namely exposing data to non-RTI systems and federate orchestration, an additional federate is needed. In the context of an RTI-based architecture, such a federate is often presented as *Manager/Mediator Federate*, which channels data from the RTI to a message broker and vice versa [13].

In this approach, inside the Manager/Mediator Federate resides an additional message broker component. Using a message broker enables this federate to act as a bridge between RTI and the messaging channel receiving messages from one side (HLA federation) and sending them on to the other (external non-RTI-connected systems). The message broker offers, therefore, the required flexibility and scalability that a co-simulation requires to expose simulation data to clients that have not been adapted to integrate with the RTI. The clients connected on the other side of the messaging channel are not part of the federation (as defined by the HLA), and therefore do not share the same functionality, especially with respect to timing as compared to a federate. The protocol translation can be programmed to minimize marshaling/unmarshaling efforts. There are a number of

Preprint version

technologies that can be used to implement such a messaging service, most commonly through publish/subscribe frameworks such as MQTT.

An effective presentation of the simulation results to the stakeholders is of prime importance for the decision making process. Enabling concurrent visualization schemes and user interfaces that can adequately present information from different simulators requires a global overview of the co-simulation system. RTI can support this by implementing visualization functionalities encapsulated as a federate. In this way, visualization tools get access directly to the synchronized data objects, making sure that the presentation in sync with the simulation. In the case that there is no hard requirement for visualization, meaning that simulation can move on without having to hold for visualization that updates its state, the message broker component can be used. As shown on the architecture in Figure 1, a 3D Graphics Engine (e.g., Unity 3D) can use a broker client to get access to the shared data objects that are to be updated in a scene.

Overall the flow of actions to execute a co-simulation based on this approach would include the following steps. In a preliminary step, domain experts create a scenario and upload simulation models to the repository. The FOM file containing information about the objects that need to be shared among simulations gets updated. Subsequently, through a UI connected to a broker client, the user sends a command to the manager federate to start the simulation. The various simulators start executing their code in a synchronous way (guaranteed by the RTI). The synchronized simulation objects that are used as input for the simulation can be sourced from real enterprise information systems and IoT components (e.g., sensors and measurement devices). During execution time, the manager federate abstracts some parameters and publishes their data to the broker, making them available to be visualized by suitable tools, such as 3D graphics engines.

The modular design of the proposed architecture tries to enable the coupling of the different systems over RTI and in parallel to take advantage of modern cloud technologies, tools, and services, for development, deployment, and operation. To this end, all components are containerized using Docker. Containers provide an isolated process execution environment, that is well optimized for execution on cloud-based infrastructure. This capability is harnessed by introducing Kubernetes as container orchestrator. It allows uniting several virtual machines to form a distributed container environment, thus leveraging the advantages of a cluster. All of the parts of the architecture shown in Figure 1 can be deployed to Kubernetes clusters.

In order to ensure continuous delivery in smooth development cycles this process is automated using a Continuous Integration (CI) pipeline that, upon detected changes in the code base, automatically performs several steps i.e. (i) compilation of local and remote sources, (ii) installation of packages to the local repository, (iii) building and tagging of Docker images, (iv) uploading of images to the container registry, and (v) updating of deployments within the Kubernetes cluster to latest version of image available.

IV. Use Case: Corporate Electric Vehicle Fleet

A. Scenario

A corporate EV fleet scenario dealing with simulation and visualization pertaining to EV charging is an excellent candidate for exemplifying the approach taken in this work. Co-simulation can improve planning and operation decisions by enabling a low-cost what-if analysis, in the early stages e.g. of planning the corporate EV fleet operations. An EV fleet has significant implications for various aspects of an enterprise. There is an interplay between the EV and its usage, the power flow, the charging infrastructure and the need to visualize the system status, e.g., state of charge, entrance on campus premises and charging management, etc. These and more related aspects should be taken into account by the fleet manager for timely planning, resource allocation, and infrastructure expansion.

The incentive to apply co-simulation stems from the need to simulate scenarios that imply the interplay of several simulators and enterprise services, under a common architecture, i.e., enabling their co-simulation. Specifically, we want to investigate how a company can increase its resource utilization in the existing electrical infrastructure. To achieve this, one should be able to simulate the behavior of the corporate EV fleet, during a daily cycle, and take into account the normal energy-related consumption of the enterprise (to avoid overloading). The assumption made in this scenario is that EVs used by employees can be charged both at home, as well as at company premises. For the fleet manager, in order to accurately study the effects of the fleet, and decide on a proper course of action, various metrics are important.

Firstly, the State of Charge (SoC) of each EV at the time of arrival on company premises is important, in order to be able to optimize their assignment to charging stations and the power allocated to them. In order to calculate the SoC, the routes of an employee driving his/her car during the day (especially going from home to the company campus and vice versa) need to be taken into account. Additional parameters to be considered are the EV battery discharging rate, and the minimum required state of charge (for a car to reach its destination when leaving company premises). Once the vehicle arrives at the corporate campus, it is assigned to a charging station and recharges. Further conditions apply to the overall load that can be supported by the corporate campus charging infrastructure and need to be taken into consideration.

In this scenario, the goal is to simulate the EVs, with respect to their energy consumption, starting from various locations entering the company premises at the beginning of a workday and their charging process. The charging infrastructure is managed by a charging optimization component that allows regulating the power delivered at each charging station over time; simply said favoring the use of "cheaper" over more "expensive" electricity. For the utilized charging optimization realized [6], electricity prices can be acquired via a service that reflects the current contracts or acquires

Preprint version

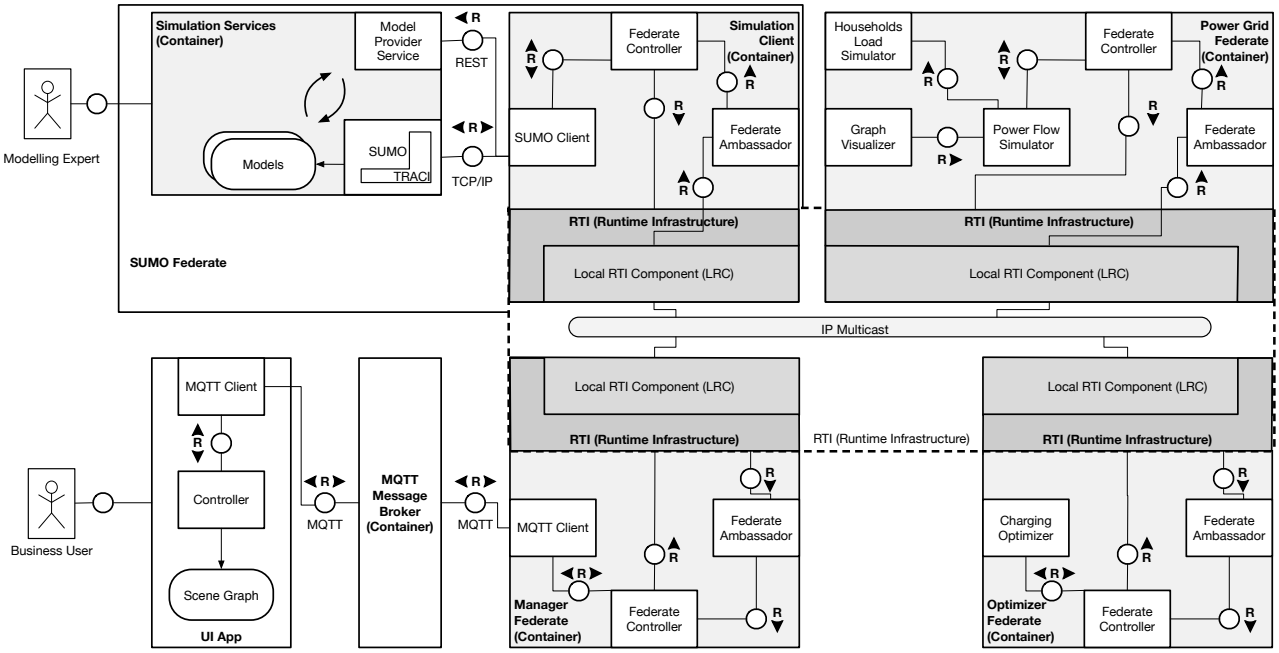

Fig. 2. Implemented scenario architecture

them in real-time from the energy provider. Each time a car connects to a charging station the load on the electrical grid may need to increase according to its optimized charging schedule. We also want to be able to analyze and compare the total charging power over time with and without intelligent charging [6].

B. Co-simulation components

To realize the use case scenario, several simulators and services need to be brought together and integrated as federates in the architecture, in order to cover the posed requirements. The resulting implementation architecture is shown Figure 2. The main components are:

1) EV Traffic Simulation with SUMO: The scenario deals with EVs, their routes, and battery utilization. To have realistic behaviors, traffic simulation tools can be utilized to create data about EV behaviors. The Simulation for Urban MObility (SUMO) [17] enables modeling and simulation of intermodal traffic systems that can include road vehicles, public transport, and pedestrians. Its extensive functionality covers creating models, setting of street network parameters, calculation of routes, and a basic visualization. In addition, it offers several plug-ins targeting e-mobility such as vehicle, driver, and battery consumption models, as well as charging stations capabilities. [18]. The traffic simulation is based on a static graph with nodes (junctions) and edges (streets & lanes). EV-types are modeled using (custom) models with specific characteristics specifying physical, dynamic and electrical characteristics. The data generated by SUMO include arrival and departure times, battery SoC and waiting times (at charging stations). Simulation data can be communicated to other components and be used as a data generator for a multitude of scenarios (for instance for testing charging optimization algorithms as discussed see below).

2) EV Fleet Charging optimization: Another component that is added as a federate, is that of an EV fleet charging optimization. This features the implementation of an intelligent charging optimization functionality in Java [6], that is responsible for creating charging schedules per car. The main provided functionality is setting the available power for each 15-minute time-slot per charging station.

3) Power Flow: A load flow study is valuable for the study of an enterprise electrical grid with multiple load centers, such as large data-centers, HVAC, and sanitary facilities. The addition of EVs charging loads can lead to overloading. Therefore, home-grown simple power flow simulator implemented in Java is used, encapsulated as a federate. The simulator can accept as input power loads from various (real and simulated) sources, including the EVs charging loads from SUMO, and provide analysis of the system's capability to adequately supply the connected load. Such a component could be expanded to provide information about total system and individual line losses, as well as the real and reactive power flowing in each power line. For now, it is limited to calculate the magnitude of the voltage at each bus. The study of household loads, when the EVs charge at home, is also enabled.

C. Implementing the Integration Architecture

The architecture in Figure 2 shows how the above-described components are integrated using HLA/RTI. It fills the abstract skeleton architecture from Figure 1 with domain-relevant components. One difference is that our model provider in this implementation is tightly-coupled with SUMO simulator at this point, as it made little sense to offer it to other components as a service in a container. As can be

Fig. 3. Visualization in Unity 3D of incoming EVs and their allocation to the parking charging stations

seen, there are four distinctive parts: (i) SUMO, (ii) the power grid, (iii) a charging optimizer, and (iv) a manager/bridge to an MQTT channel, all of which are connected over the RTI. A UI for business users is also integrated over an MQTT publisher/subscribe pattern.

From an operational perspective, two main workflows are supported. The first one involves a modeling expert preparing SUMO models and uploading them to a model provider, while the second one involves the business user executing multiple simulations with different parameters, in order to gain insights on the achieved impacts. Each user interacts the system through their bespoke UI.

The RTI allows federates to be time regulating and/or constrained, or neither. The former indicates that it can request all other federates move to next time steps. The latter indicates it moves to the next time step together with all other federates. We have chosen all federates to be both constrained as well as regulating, allowing for very tight synchronous integration. A use case for a non-regulating federate would be a viewer that is not interested in any control of the simulation.

In Figure 3 the current outcome and status of the co-simulation execution are visualized using Unity 3D. Visualization is done at run-time based on the of the generated simulation data (simulation time). The company employees come with their cars to the company premises, along the road shown on the left side of Figure 3. Subsequently, they enter the corporate parking area (shown on the right side of the figure), and if they have an EV, they park on the dedicated spots featuring a charging station. EVs can be clearly seen as they are over-imposed with information about the battery capacity as well as the respective state of charge (which is also graphically visualized as a battery on top of each EV).

The UI operated by the business user, as shown in Figure 3, provides some configuration options in order to set up and control the overall co-simulation. Note that these are limited to the EV aspects, as this UI is customized for the specific business need. On the top-left side of Figure 3, it is seen that the high-level aspects that can be configured are the overall number of cars (the corporate fleet size) as well as the percentage of these cars that are EVs (the others are considered to be conventional cars). In addition, the number of charging points can be adjusted, which has an impact on the simulation, since it is linked with the charging optimization strategies. Indicatively other info may be displayed e.g. the CO_2 emissions as shown on the top-right of Figure 3. Different areas can be selected, based on open maps, and for the specific screenshot in Figure 3 the corporate premises of SAP in Walldorf, Germany are selected (indicated as *"WDF_Area"* on the top-left of the figure. Traffic patterns and car trajectories for the specific maps are automatically done via SUMO. Finally, the start/stop of the simulation can also be controlled, via a command that is appropriately propagated to all of the components of the co-simulation.

To be able to visualize in real time the running simulation, the information about each EV such as its current speed, CO_2 emission, current battery capacity and engine type is transferred from the SUMO Federate to the Unity 3D with a message broker in a synchronous way through the RTI. Updates of all objects are handled directly in the frame when the messages are received by the Unity application. In this way, it is possible to have the best breed of two worlds, i.e., the co-simulation running on the backbone of the HLA/RTI, as well as a timely visualization of the ongoing co-simulation via non-RTI applications such as the one implemented in Unity and shown in Figure 3.

V. Discussion

The overall coupling of different independently developed software components via an HLA-empowered architecture has provided promising results. One of the main reasons

is the interoperability among the heterogeneous simulation systems that was achieved, as well as the common communication over the RTI. Although the simulations were simple, it was possible to bring everything to the cloud, utilizing the modern container-driven approach for development, testing, deployment, and operation of the simulation. The end-result enabled a timely development of the planned co-simulation and demonstration of the proposed scenario. However, during this process, several challenges also emerged, which ought to be considered for future endeavors and are discussed in this section.

Having implemented a multi-federate system for our use case we can now evaluate how well this architecture supports additional simulators increasing the utility of the system. Our experience showed that it is easy to integrate more simulators (or other producers and consumers of simulation data) as long as there is a good architecture blueprint that provides a reusable separation of concerns, applicable to all federates, which was the motivation behind the generalized architecture shown in Figure 1. The scenario presented in the previous section included only some aspects of studying an EV fleet of company cars. To enable the more detailed study of complex, heterogeneous systems, additional coupling to the grid infrastructure simulation needs to be present. As future work, one could expand it to include not only several instances of SUMO, but also a simulation of other aspects e.g. of the smart grid infrastructure via frameworks such as Mosaik [19].

The RTI provides two mechanisms to share data: data objects can be either shared between federates with updates in one being reflected in proxies held in all others, or alternatively, predefined event types called *interactions*, that can be used to send events with or without data to all other federates. Attempting to maintain a single unified data model across different contexts has been shown to often lead to wasted effort [20]. Less coupled, event-like interaction between federates can be achieved by the use of interactions. By large, however, the communication via reflected simulation objects does have great benefits as the RTI automatically takes care of assigning updates to the correct local object via unique handles. By allowing the sharing of simulation objects the RTI provides a powerful mechanism for different simulators to operate on, and even hand over any simulation object to a subsequent simulator.

For the various what-if scenarios using co-simulation approaches, the different simulation units for each of the individual system aspects, can be perceived as black boxes, and thus executed in different environments. Taking advantage of this property, modern microservice and container-based architectures prove rather suitable approaches for developing and deploying co-simulation concepts, e.g., as shown by the use case utilized in this work. In addition, as enterprise IT infrastructures expect any application to be easily deployed and monitored, the containerization and cloud-deployment approach is a good fit. The containerization of existing tools such as SUMO is proof for the rapid deployment and easy integration of the solutions. Alternatively, the user would have to spend a significant amount of time to configure and run the simulator in his/her environment, while now this complexity is hidden within the container itself. Additionally, a benefit of using scalable deployments in the cloud is the potential of executing multiple co-simulation instances, implementing different scenarios in parallel and comparing the results. As future work, a dispatcher component will be deployed to spawn new instances and distribute the simulation tasks accordingly.

Given the numerous middleware options available in the context of the enterprise IT, one question is to what extent does it make sense to use in conjunction with the RTI message brokers, REST calls, microservices, etc. for communication. In our system we try to channel as much communication as possible through the RTI, avoiding federates talking to each other through some alternative channel. This provides more flexibility for accommodating interaction patterns in the future and a simpler communication model. One improvement of this architecture, for example, could involve promoting the microservices to the status of a federate. Pursuing such an approach should result in an architecture in which the only communication channel within the federation is the RTI while other communication channels may lead out from the federation for purposes of integration into an existing landscape. We intend to investigate such issues in the next iteration where we hope to make progress on integrating with existing enterprise services to access real-time and master data.

Implementing an RTI-based architecture comes at a cost. Subscribing, publishing, and sending messages, unlike modern message brokers, involves a lot of manual work leading to an increase of accidental [21] complexity of an already essentially complex system. On the positive side, the RTI takes care of the highly challenging problem of time synchronizing distributed software components. A hypothetical custom implementation of co-simulation using an off-the-shelf modern message broker would potentially run into issues of out-of-sync messages (e.g., arriving too late or too early), leading to non-deterministic transient states.

Federates are distributed peer-to-peer components, which in itself results in complexity when trying to get them to synchronize, say at the start of a simulation. The RTI does offer synchronization mechanisms for this purpose, but some complexity remains since, for example, the actions to be performed by each federate are temporally coupled. In our system, we have chosen to implement a manager federate that takes on coordination responsibilities.

When designing an HLA-based co-simulation, a decision that needs to be made for most components is whether to have components acting within the boundary of one federate or to promote them to a federate in their own right. Although distribution always comes at the cost of more complexity, synergy effects are achieved by components producing and consuming simulation data without knowing about downstream consumers or upstream producers, and with data being manipulated more than once throughout the process by different federates. In our use case, we chose to include

several components of the power flow simulator within the Power Grid Federate to remain within the boundary of that federate but in other use cases, they could also be implemented as federates.

With regards to visualization, for the use case of EV fleets, the visualization of the system state at any given time during the simulation is of prime importance, as it provides valuable insights to the stakeholders, by revealing possible hidden interrelations and dependencies between the systems. The possibility of providing rich, easily deployed visualization tools that can combine outputs from multiple simulators and can be also be used to control additional data manipulation functions such as optimization functionalities is rather valuable.

Finally, as enterprises typically collect and store a huge amount of business data, its combination and correlation with simulations can result in more realistic what-if scenario assessments which reduce the risk. For instance, in our use case, the combination of business data pertaining to the characteristics of the EV fleet (vehicles, batteries, charging sessions etc.), enabled us to integrate in a more realistic manner the routes calculated by SUMO as well as the intelligent charging on the corporate premises. Further fine-grained integration of business data into the virtual models, as well as potentially customization of them to match the goals of the what-if scenario, is expected to significantly enhance the quality of simulation results.

VI. Conclusion

A simulation may pose a compelling tool for the investigation of what-if scenarios, especially if coupled with modern AR/VR visualization capabilities, and can help decision makers to better anticipate situations and assess the impact of potential decisions. Through modeling of the respective system characteristics and functions, the operation of a system over time can be simulated, thus allowing the study of individual systems. However, especially due to the complexity and heterogeneity involved in a system of systems scenarios, their development needs to based on modern principles of reusability, interoperability as well as rapid development and deployment.

A co-simulation scenario based on corporate EV fleet is used as motivation and a co-simulation environment where multiple simulation units interacting with each other using HLA/RTI is presented. The development and deployment are done in a modern cloud-based environment utilizing container technology managed by Kubernetes, while the simulation results can be visualized in 3D via Unity. The benefits of our architecture are as follows:

While the approach is promising and can enhance the system-wide simulation scenarios in enterprises, there are still several issues that need to be handled, in order to simplify the integration of new simulators.

It would be interesting to evaluate the performance of this architecture for large simulations, and, in particular, how computing resources are consumed by the individual sub-components. We have started optimizing performance at various points of the system and hope to continue to do so before making qualified performance measurements.

An additional challenge to be addressed as future work concerns the linking of the data and results produced from such co-simulations, with other enterprise systems for real-time analytics and management, as well as the combination of business data and services.

References

[1] S. Karnouskos, "Efficient sensor data inclusion in enterprise services," *Datenbank-Spektrum*, vol. 9, no. 28, pp. 5–10, Feb. 2009.

[2] BKCASE Editorial Board, *The Guide to the Systems Engineering Body of Knowledge (SEBoK) v1.8*, R. D. Adcock, Ed. The Trustees of the Stevens Institute of Technology, Hoboken, NJ, Mar. 2017. [Online]. Available: https://www.sebokwiki.org

[3] C. Goncalves Gomez, C. Thule, D. Broman, P. Larsen, and H. Vangheluwe, *Co-simulation: State of the art: Technical Report*. arXiv.org, Feb. 2017, 1702.00686. [Online]. Available: https://arxiv.org/abs/1702.00686

[4] "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules," pp. 1–38, Aug. 2010.

[5] S. Detzler, D. Ilić, S. Karnouskos, and C. Kindermann, "Investigating electric vehicles as a promising alternative to static storage solutions," in *2014 IEEE International Electric Vehicle Conference (IEVC)*. IEEE, 17–19 Dec. 2014.

[6] O. Frendo, S. Karnouskos, N. Gaertner, O. Kipouridis, K. Rehman, and N. Verzano, "Charging strategies and implications for corporate electric vehicle fleets," in *IEEE International Conference on Industrial Informatics (INDIN), Porto, Portugal*, Jul. 2018.

[7] O. Topçu and H. Oğuztüzün, *Guide to Distributed Simulation with HLA*. Springer International Publishing, 2017.

[8] C. A. Boer, A. de Bruin, and A. Verbraeck, "A survey on distributed simulation in industry," *Journal of Simulation*, vol. 3, no. 1, pp. 3–16, Mar. 2009.

[9] P. Ross, "Comparison of high level architecture run-time infrastructure wire protocols – part one," in *Simulation Technology and Training Conference (SimTecT), Adelaide, Australia*, 2012.

[10] "Portico." [Online]. Available: https://github.com/openlvc/portico

[11] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *ACM Queue*, vol. 14, pp. 70–93, 2016.

[12] "Unity engine." [Online]. Available: https://unity3d.com/

[13] E. Kucera, O. Haffner, and R. Leskovsky, "Interactive and virtual/mixed reality applications for mechatronics education developed in unity engine," in *2018 Cybernetics & Informatics (K&I)*. IEEE, Jan. 2018.

[14] D. Ma, X. Fan, J. Gausemeier, and M. Grafe, Eds., *Virtual Reality & Augmented Reality in Industry*. Springer, 2011.

[15] "Fundamental Modeling Concepts (FMC) notation." [Online]. Available: http://www.fmc-modeling.org

[16] C. Steinbrink, A. A. van der Meer, M. Cvetkovic, D. Babazadeh, S. Rohjans, P. Palensky, and S. Lehnhoff, "Smart grid co-simulation with MOSAIK and HLA: a comparison study," *Computer Science - Research and Development*, vol. 33, no. 1-2, pp. 135–143, Sep. 2017.

[17] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012.

[18] T. Kurczveil, P. Á. López, and E. Schnieder, "Implementation of an energy model and a charging infrastructure in SUMO," in *Simulation of Urban Mobility*. Springer Berlin Heidelberg, 2014, pp. 33–43.

[19] S. Rohjans, S. Lehnhoff, S. Schutte, S. Scherfke, and S. Hussain, "mosaik – a modular platform for the evaluation of agent-based smart grid control," in *IEEE PES ISGT Europe 2013*. IEEE, Oct. 2013.

[20] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.

[21] F. P. Brooks, Jr., "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987.