Simulation Model as a Service (SMaaS): A concept for integrated deployment, execution and tracking of system simulation models

Philipp Emanuel Stelzig[1] Benjamin Rodenberg[1]

[1]simercator GmbH, Germany, {philipp.stelzig,benjamin.rodenberg}@simercator.com

Abstract

System simulation is dealing with increasingly multiphysical and cyber-physical systems that involve multiple engineering domains. In development and production, system manufacturers often rely on supplier parts and their digital representations. To deal with this inherently collaborative setting in a more efficient way we propose a concept of *Simulation Model as a Service* (SMaaS) developed at simercator. In this article, we apply established workflows from software engineering to system simulation to create more efficient workflows, discuss the compliance with technical, economic, and regulatory requirements, and present a software for digital supply chain management that implements SMaaS.

Keywords: System simulation, FMI, Modelica, traceability, as a service, continuous integration, microservice

1 Introduction

System simulation is an integral part in the development of mechatronic and cyber-physical systems. These are increasingly multiphysical systems that involve multiple engineering domains and – to an increasing extent – also algorithms from fields like machine learning. This makes system simulation inherently collaborative: System manufacturers rely on specialized suppliers to deliver subsystems, *e.g.* battery packs for electric vehicles or HVAC systems for cruise ships. In addition to the physical hardware, system manufacturers also require corresponding digital assets from the suppliers like geometry data, documentation, and simulation models.

Today, system simulation experts have access to a rich set of tools that allow to model and solve system simulation models. However, little attention has been given to the actual *process* of exchanging simulation models between the various parties that are involved. Consequently, there are few technical means supporting this *supply chain of simulation models*. In this article, we introduce a concept developed by simercator that we call *Simulation Model as a Service* (SMaaS). Our proposed workflow transfers established best-practices from software engineering to system simulation to obtain and manage 3rd party simulation models and is designed to comply with technical, economic, and regulartory requirements. Finally, we present with *simercator hub* an actual implementation of a digital supply chain for simulation models implementing SMaaS.

1.1 State of the art: Modeling and file-based model exchange

System simulation knows excellent modeling languages and software tools for modeling and simulation of complex systems. In a multidisciplinary or multiphysics context, Modelica (Modelica Association 2023) is very popular and numerous softwares implement the language. The Functional Mock-up Interface (FMI, see Modelica Association (2022)) has become a de-facto standard for exchanging models between tools of different vendors: See the long list of tools[1] that now support FMI. The list of tools reaches well beyond the system simulation domain and includes also tools for models based on partial differential equations (PDEs) from solid mechanics using the finite element method (FEM) or computational fluid-dynamics problems using finite volume solvers. A simulation model according to the FMI is a ZIP-file called *Functional Mock-up Unit* (FMU) that contains – as defined in the standard – a computational model implementing the FMI and model information as structured text specifying the model inputs, outputs, and additional meta-information. Thus, the FMI standard allows to share and distribute simulation models by exchanging FMU files.

However, the file-based exchange of simulation models is not standardized and the result of a bilateral, often personal contact between model owner and model user. If model owner and model user are representatives of different companies a legal framework such as non-disclosure agreements (NDA) is often required. This bilateral approach works for a small number of models and model users. However, as a manual process it is slow, error-prone, cannot be opened to a broader audience, and is costly. If these complications prove unmanageable, simulation experts have to fall back to making their own simulation models for 3rd party components, often merely relying on datasheets.

Circuit simulation is a special case: Virtually all commercial circuit simulation tools derive from the same open sourced SPICE simulator (Vladirmirescu 2011). Other than in mechanics-centered system simulation, this made it possible to exchange at least elementary components or simple SPICE netlists (plain text). However, complex circuit models developed in commercial tools can usually not

[1]List of supporting tools https://fmi-standard.org/tools/

be imported in tools from different vendors, especially if the models are encrypted. There are model exchange platforms like Ultra Librarian.[2] Most semiconductor companies offer SPICE models for elementary components for download, often without access restrictions. Netlists for complex products are typically not openly available.

For the exchange of CAD models for supplied parts, geometry models are mostly reduced to the necessary outer features. The resulting CAD files are often made available as downloadable objects, either on company websites or via specialized CAD portals like PARTcommunity[3] or GrabCAD.[4] While sometimes no special access restrictions are imposed, legal restrictions usually apply.

Technical means to deliver, track, and maintain digital assets of real hardware products are currently in their infancies. Therefore, major efforts are being made in research projects and industrial associations. The Industrial Digital Twin Association (IDTA)[5] has formulated the *Asset Administration Shell* (AAS, see Federal Ministry for Economic Affairs and Climate Action (2022)) as a superstandard that can capture a wide variety of digital assets and even integrate with existing standards by means of *sub-models*, like with the simulation sub-model and the FMI (Industrial Digital Twin Organization 2022). The industry association Catena-X aims at establishing “a trustworthy, collaborative, open and secure data ecosystem”[6] and is developing an open source implementation.[7]

1.2 SaaS trend in simulation

In recent years, the simulation software industry is steadily moving from local software installations on users' computers to cloud services using *Software as a Service* (SaaS). This impacts the collaboration and exchange of simulation models, because either the entire modeling and simulation process is executed through the web browser in the cloud, or modeling is done locally and only the execution of computationally heavy simulations is deferred to a cloud infrastructure. One major advantage here is the good availability and on-demand allocation of computational resources. Most major simulation software vendors offer SaaS solutions. New developments tend to be cloud-native anyway, like *e.g.* SimScale[8] for PDE-based simulation or Modelon Impact[9] with its cloud-native modeling environment for Modelica.

Some simulation software vendors now allow sharing of easy-to-use and ready-to-use, *i.e.* executable simulation models as interactive web applications. This trend can also be viewed in the broader context of the so-called *Democratization of CAE* (Taylor et al. 2015). Interactive web applications can be generated *e.g.* with Modelon Impact's App Mode[10], or Siemens Simcenter Webapp Server.[11]

1.3 Related work

In the automotive industry, the research project SET Level proposed a “Credible Simulation Process” (SET Level 2021) to ensure simulation quality when integrating simulation models from different stakeholders. The maritime industry developed the *Open Simulation Platform* (OSP)[12] as an open source project[13] providing a co-simulation solution tailored to the needs of the maritime industry to “create a maritime industry ecosystem for co-simulation of 'black-box' simulation models”.[14] In the context of the OSP, the open source project *FMU-Proxy*[15] has been developed to enable single FMUs to be co-simulated via network, see Hatledal, Styve, et al. (2019) and Hatledal, Zhang, Styve, et al. (2019). FMU-Proxy enables model owners to share an original FMU by means of a proxy of the FMU, meaning the proxy FMU looks identical to the original FMU from the perspective of the user, but internally it features remote procedure calls (RPC) to a server, which evaluates the original FMU. In Hatledal, Zhang, Styve, et al. (2019, chapters 3, 4.1) one possible use-case of FMU-proxy is illustrated: Model owners can list original FMUs delivered as proxy FMUs to authorized model users through a “discovery service”. Similar to FMU-Proxy, UniFMU by Legaard et al. (2021) also uses a FMU as a communication interface and network communication, but as a means to enable running computational models in languages or tools that do not support the FMI. In Schranz et al. (2021), UniFMU is extended in that users can encapsulate the computational model hidden behind UniFMU in a Docker[16] container with all dependencies included, in order to improve portability; remote execution is not supported with UniFMU as of Schranz et al. (2021). In the defense sector, exchange of simulation models and physically distributed cooperative simulations (co-simulation) are natural requirements. For example in combat simulations, when simulated actions have to be synchronized amongst involved ships, aircraft, and vehicles. NATO has developed the concept of Modeling and Simulation as a Service (MSaaS) and created reference architectures regarding possible realizations (Siegfried, Lloyd, and TVD Berg 2018; Hannay and Tom van den Berg 2017). Note that MSaaS is different from the SMaaS that we propose, in that MSaaS describes a distributed execution framework for simulation, while SMaaS is a concept for providing building blocks to assemble simulations from.

[2] https://www.ultralibrarian.com/
[3] https://b2b.partcommunity.com
[4] https://grabcad.com/
[5] https://industrialdigitaltwin.org/
[6] https://catena-x.net/en/vision-goals
[7] https://github.com/eclipse-tractusx
[8] https://www.simscale.com/
[9] https://modelon.com/modelon-impact/
[10] https://help.modelon.com/latest/release_notes/impact_2023_2/
[11] https://plm.sw.siemens.com/en-US/simcenter/systems-simulation/webapp-server/
[12] https://opensimulationplatform.com/
[13] https://github.com/open-simulation-platform
[14] https://open-simulation-platform.github.io/
[15] https://github.com/NTNU-IHB/FMU-proxy
[16] https://www.docker.com/

Trauer et al. (2022) outline strategies to facilitate the exchange of digital twins, including simulation models, by means of trust and quality indicators.

1.4 Outline

In section 2, we discuss the differences of any formalized process to exchange simulation models compared to exchange processes for other digital assets. In section 3 we review various obstacles (technical, economic, legal, regulatory) that occur in such a process and present the resulting requirements in section 4. In section 5 we then present well-established best-practices from software engineering that are applicable to an exchange process for simulation models. In section 6 the SMaaS concept is described. Finally, simercator hub as an actual implementation of this concept is outlined as a showase in section 7, followed by conclusions and suggestions for future work in section 8.

2 Simulation models need dedicated supply chains

In the industries, system manufacturers require simulation models for supplier parts. Other than in academia, where research and simulation is mostly done as a collaborative effort with openly accessible knowledge, in the industries the involved organizations prefer to protect the internal workings of simulation models and at most provide simulation models as ready-to-use black boxes. If model user and owner work in different companies, the motivation for this is to secure a real or perceived value. But also if the model user and owner are part of the same company and merely work in different departments, distributing ready-to-use executables, in particular as FMUs, is common practice.

We will consider any computational model that may take part in a dynamic system simulation and that produces numerical outputs from given inputs as a simulation model. Examples are traditional system simulation models (event-based DAE systems), FEA or CFD simulations (PDEs), control algorithms, and data driven or machine learning models. In the world of system simulation the system under consideration is mostly built from individual submodels. Some typical examples are: 1) active power and aging models for batteries are analyzed in a renewable energy system simulation; 2) a finite element strength simulation for a damper component contributes to an overall simulation of the dynamics of a vehicle suspension; 3) a neural network inference model acts as a control algorithm for object detection in an autonomous vehicle simulation.

All the aforementioned simulation models are mostly files or a collection of files. Hence, bringing simulation models from a model owner to a model user means physically copying files to a location where the model user can access and import them into a system simulation tool. Like with other file-based digital data, this immediately leads to a number of issues: Who owns the digital data? What is the receiving party allowed to do with it? In section 3 we will address these and other questions in detail.

Finally, all involved parties need to agree on an exchange format. If all parties are already using exactly the same simulation software tool, then simulation models can be imported and further processed without major issues. If different tools are in use, the Functional Mock-up Interface (FMI) provides a tool-agnostic simulation model format. If this is not feasible, the parties first need to agree on a common software with a specific version and platform before collaboration can happen.

Here, we want to focus on the following two questions:

1. How are simulation models different compared to other digital assets?
2. Why does the FMI as a data format not satisfy all the resulting requirements?

Executable code Simulation models are executable. In order to run, various software dependencies on the user's system have to be satisfied. If dependencies, *e.g.* runtime dynamic libraries, are not met, the simulation model might not run at all or, even worse, produce wrong results.

Runtime behavior Simulation models have a runtime behavior. They produce simulation output as the result of time-varying input provided to the model and numerically solving a computational model (see the mathematical problem classes above). Inputs are generally only known at runtime and not at the time of model creation or model distribution. Here, CAD models clearly differ, because they are static and do not have a runtime behavior; the information that they convey to a model user is entirely known to the model owner at the time of creation.

Validity range Simulation models can only accurately represent reality for a limited range of inputs. However, one cannot expect a user to generally be able to judge the modeling error while the model owner has expert knowledge that helps to quantify the modeling error. Therefore, a model owner needs to know the inputs a model user is going to provide to the model, especially if the model owner has a liability for the correctness of the results. Naturally, also a model user needs to know for which inputs the model produces valid outputs. This knowledge is required both at modeling time, *i.e.* to ensure that an externally supplied model is used in a meaningful context only, as well as after simulation time, *i.e.* to ensure that a model produced meaningful results.

Maintainability Simulation models may require updates and bugfixes. If an error is discovered, in particular by the actual user, the model owner needs the ability to update the simulation model, or, to shut it down. If the number of users is small and it is known where the model has been deployed in the past, then doing this manually in a bilateral fashion is possible. However, if the number of users is large, or the model distribution is unclear, or it is used in highly critical applications, a manual update process is not feasible or satisfactory.

As a consequence, simulation models need a dedicated solution for the distribution from model owner to model user. It must ensure dependency management, usage control, traceability, and update mechanisms. These features are not part of the FMI and, from our perspective, are outside its scope. As we will show in sections 6 and 7, with a particular client/server architecture that we call *Simulation Model as a Service*, it is possible to build a supply chain for simulation models that considers the aspects above.

3 Obstacles

Today, a model user has to overcome many obstacles before a simulation model can actually be run. These obstacles are of technical, economic, legal, or regulatory nature.

3.1 Technical: Dependency management

The FMI is a widely used data format for the exchange of simulation models. A major issue of the FMI is the lack of dependency management. Both source code FMUs and compiled FMUs require dependencies to be present in the execution environment. If they are not met or wrong versions are provided, FMUs cannot run or might produce wrong results. Model users typically cannot resolve dependencies without help from the model owner. The FMI standard states that "FMUs must reduce their dependency on operating system services" (Modelica Association 2022, section 1.3) and that "tool dependencies must be documented" (Modelica Association 2022, section 2.5.3). However, beyond the requirement for written documentation in the FMU's `documentation` folder (Modelica Association 2022, section 2.5.1.1), the FMI does not provide machine-readable dependency management to support an automatic and user-friendly process.

3.2 Economic: Real and perceived value

Economic value is attributed to simulation models by both model owners and users. For users, the economic value lies mainly in working time savings, if the alternative for them means creating their own model. This allows us to roughly approximate the value for users. For owners, this is more difficult: Simulation models, at least if they represent a real product sold by the model owner, have little market value on their own, because they are merely descriptive and can only generate revenue in combination with the real product they represent. In this case, the economic value of a simulation model is mainly the added value for the buyer. However, the costs to create a simulation model are usually very well known or can be estimated as the personnel costs spent on modeling, costs for the software tools, and, if applicable, the cost for model validation. Finally, there is a negative economic value, *i.e.* the risk of economic damage that might result from the loss of intellectual property, or from damage claims for a flawed simulation model.

There is no good literature available on this subject. At simercator we have experienced that managers, decision makers, and non-experts in simulation tend to make the following mistakes:

Mistake 1	Cost for model creation equals the market value of the models.
Mistake 2	The information *contained* in a simulation model is equal to the knowledge that *went into creating* the model.

As a consequence, managers and decision-makers tend to be over-reluctant when sharing simulation models, because they overestimate the risk of distributing models. Hence, there appears to be a significant discrepancy between actual and perceived value.

3.3 Legal: Intellectual property and liability

Simulation models contain intellectual property and the models themselves are subject to copyright and property law. Model owners need to ensure that distributing simulation models does not affect any rights. Export control laws also apply.

Simulation softwares, including Modelica tools, offer the possibility to encrypt simulation models. However, the processing of encrypted models is usually vendor-specific, and, in some cases, even version-specific. The FMI's intellectual property protection relies on the binary compilation. To lesser extent, source code FMUs can be obfuscated, but by nature the internals of the model remain exposed. With simulation models, where phenomenological or reduced order models are used, it is quite often already hard to reverse-engineer product features from a descriptive model. Restricting the distribution, *i.e.* the *copying* of files that contain simulation models is not possible through technical means, but one can restrict the execution, *e.g.* by requiring runtime licenses or decryption keys.

Another big legal concern is liability for correctness. Mathematically, it is not possible to quantify *a priori* the modeling error of a simulation model, *i.e.* the accurate representation of physical reality through the model for *any input* that a user provides. The same applies to control algorithms and becomes even more pressing when the model itself evolves when the user provides new input during model usage, *e.g.* with machine learning models.

As of today, we do not know of effective mechanisms for owners to track usage after delivering the model to the user. Usage control is limited to restricting ranges of input variables and other mechanisms that are compiled into a simulation model. However, any control mechanism built into a model at compile time is based on previously known or anticipated model usage and model behavior. Consequently, without knowledge about actual usage, the model owner cannot improve on control mechanisms or discover unexpected runtime behavior. Likewise, a user cannot be warned and potential damage can only be analyzed *a posteriori* and only if the user keeps record of specific model versions together with the input and ideally output values.

3.4 Regulatory: Traceability, explainability

The recent years have witnessed the widespread adoption of computational methods from the fields of machine learning (ML), statistical methods or artificial intelligence (AI). Applications range from the prediction of human behavior for marketing purposes, artificial generation of text or graphics, chatbots, or even crime prediction, to industrial applications like object detection in quality control or control algorithms. Due to the "learning" nature and the dimensional complexity, their operational behavior is hard to predict with mathematical means and might evolve over time. Lawmakers all over the world will most likely enforce regulation on their usage soon. The European Union (EU) has taken a leading role. In 2018, it has already issued the *Ethics Guidelines for Trustworthy AI* (European Commission 2018). In 2021, the European Commission has published a proposal for the *EU Artificial Intelligence Act* (EU AI Act) (European Commission 2021b).

AI and ML are already used as computational models within system simulation, mostly as control algorithms. But also the other way round: System simulation models themselves can be used as part of AI-based or ML-based control algorithms to simulate the physical reality that is being controlled, *i.e.* as *simulation digital twins* (Boschert, Heinrich, and Rosen 2018).

The European Commission (2018, Chapter II.1.4) already demanded *traceability* and *explainability* from AI systems. The EU AI Act proposal is more specific: "[H]igh-risk AI systems" (European Commission 2021a, Annex I, Annex III) are required to implement "automatic recording of events ('logs')" (European Commission 2021b, Articles 12, 20) as well as "[c]orrective actions" (European Commission 2021b, Article 21) in order "to bring [the high-risk AI system] into conformity, to withdraw it or to recall it, as appropriate" (European Commission 2021b, Article 21). High-risk applications for AI systems according to European Commission (2021b, Section 5.2.3) and European Commission (2021a, Annex III) comprise "[m]anagement and operation of critical infrastructure [...] operation of road traffic and the supply of water, gas, heating and electricity".

Hence, it might very well be possible that model owners who provide certain computational models will legally have to ensure automated mechanisms to control distribution, maintainability (corrective action), and usage control (record-keeping and explainability).

4 Requirements

From section 3 we can now derive requirements for a realization of a supply chain for simulation models (section 2). The most important are:

Model owner A model owner must be able to

(RO1) control model distribution,
(RO2) enforce control, logging, and monitoring of model inputs and outputs,
(RO3) enforce a location for storage and execution,
(RO4) update or shut down distributed models,
(RO5) comply with data privacy laws.

Model user A model user must be able to

(RU1) obtain models ready-to-use and fitting the technical and organizational (data traffic) needs,
(RU2) know if his model usage produced valid outputs, both at modeling time and after simulation time,
(RU3) know which data is processed and communicated to the model owner, both *a priori* and *a posteriori*,
(RU4) retain full control over his own simulation models (no forced updates),
(RU5) rely on the fact that data logging does not reveal any of his own models or data.

Simulation model supply chain A supply chain for simulation models must allow for

(RC1) parallel model delivery and execution,
(RC2) integration with other digital supply chains,
(RC3) integration with simulation tools and standards.

5 Learning from software engineering

Deployment, monitoring, and maintenance of ready-to-use software have been a core challenge in software engineering for decades. Therefore, we seek best practices and inspiration in software engineering for a supply chain for simulation models.

5.1 Dependency management, software deployment, and maintenance

Modern operating systems use package managers and specific package formats to install software from given repositories, like *e.g.* Ubuntu's official package repository providing `.deb` Debian packages.[17] Package managers that take care of dependency resolution exist for all major operating systems: `apt`[18] or `rpm`[19] (Linux), `brew`[20] (MacOS), and `winget`[21] (Windows). There are also platforms like Google Play[22] or Apple's App Store[23] for mobile applications as well as dockerhub[24] for microservices in mostly cloud applications.

While FMI's role could be intepreted as a package format, there is – to the best of our knowledge – neither a *package manager with dependency management for simulation models* nor a *package repository for system simulation models*. (This statement does not apply to circuit simulation models, see section 1.1.)

[17] `https://packages.ubuntu.com/`
[18] `https://wiki.debian.org/AptCLI`
[19] `https://rpm.org/`
[20] `https://brew.sh/`
[21] `https://learn.microsoft.com/en-us/windows/package-manager/`
[22] `https://play.google.com`
[23] `https://apps.apple.com`
[24] `https://hub.docker.com/`

5.2 Services for source code development and exchange

In the software world, almost all developers rely on web-based services for exchanging software. Services like GitHub,[25] GitLab,[26] or Bitbucket[27] are used for development, collaboration with others, building, testing, and other tasks. These platforms support their users with features like version management, release management, issue tracking, continuous integration, automated testing, vulnerability scanning, user management, usage analytics, rights management, and collaborative development. The code repositories are available as open to the public (open source), or access is restricted to specific users in private repositories or on private instances of these platforms.

For model *development*, simulation experts often use the same services as software developers. In particular, version control systems like Git[28] and collaborative environments like GitHub or GitLab. Note that simulation modeling environments like *e.g.* Modelica tools take the role that integrated development environments (IDE) have in software development. Some simulation modeling environments like *e.g.* Dymola[29] already integrate version control.

5.3 Quality measures

In software engineering, developers expect several quality indicators from a software library: Update cycles, the existence of automated test and build toolchains, reliable tracking and handling of issues, but also social indicators like the number of contributors, regular commits, or response times for reported issues. Also, a large portion of software projects use the semantic versioning scheme `<major>.<minor>.<bugfix>`.[30]

In recent years, such measures are being adopted in model development, too. For example, the Modelica libraries IBPSA (Wetter, Treeck, et al. 2019), Buildings (Wetter, Zuo, et al. 2014) and AixLib (Müller et al. 2016) use issue tracking, automated test and build toolchains, as well as semantic versioning based releases.

6 SMaaS concept and architecture

The Simulation Model as a Service (SMaaS) concept developed at simercator proposes a simulation model repository software combined with a scalable, microservice-based computing backend. The core idea is to offer a simulation model not merely as a downloadable and executable file to the model user. Instead, model owners can offer their simulation model as an executable service, where the owner retains full control over model execution, distrubtion and usage. The SMaaS architecture is depicteded in the following Figure 1 and a concrete implementation is presented in the following section 7.

SMaaS incorporates different realizations of execution services. An integral part of SMaaS is a special execution service which we refer to as *simulation model streaming*.

Streaming *Simulation model streaming* describes original models being executed in the computing backend, while the user only uses a model *doppelgaenger* that is distributed as a downloadable file. The *doppelgaenger* model can be offered in different formats (*e.g.* FMI) to allow the import into various tools. Every time the model *doppelgaenger* is executed, the requested action (*e.g.* FMI function calls like `fmi2DoStep`) is deferred to the computing backend via a web-based RPC call, effectively "streaming" input data and results back and forth between *doppelgaenger* and original model.

This idea is similar to FMU-proxy by Hatledal, Zhang, Styve, et al. (2019). However, FMU-proxy uses a TCP/IP-based socket-to-socket communication where one proxy FMU (the equivalent to our *doppelgaenger*) communicates with one FMU-proxy server instance. To achieve a scalable execution service and to be able to run multiple instances of the same model in parallel when using SMaaS, we use microservices in the computing backend and a dedicated communication backbone for routing incoming requests to dedicated containers. For every request from a *doppelgaenger*, we instantiate a dedicated container from a container image repository. We then create a copy of the original simulation model in the container to perform the computations. In this fashion, the dependencies only need to be resolved once with a suitable container image, whereas the *doppelgaenger* delivered to the user does not require any special dependencies.

As a consequence, one can deliver simulation models as *doppelgaengers* in a format that the original simulation model does not even support. This comes at the price of matching model execution requests to corresponding evaluations of the original model during communication. This principle has also been used in FMU-proxy to "import FMI 1.0 models in software that otherwise only supports FMI 2.0",[31] and furthermore in UniFMU by Legaard et al. (2021). The advantage of resolving dependencies of an original simulation model in a microservice has also been exploited by UniFMU (Schranz et al. 2021).

Also, streaming inputs back and forth allows us to implement observer and maintenance mechanisms in the SMaaS concept to satisfy requirements such as control of usage, distribution, as well as ensuring traceability and reproduceability as the basis for explainability.

Browser-based Being able to execute original simulation models via web-based RPC calls on a computing backend makes it possible to offer access over the web

[25] `https://github.com`
[26] `https://gitlab.com`
[27] `https://bitbucket.org/`
[28] `https://git-scm.com/`
[29] `https://www.3ds.com/products-services/catia/products/dymola/model-design-tools/`
[30] `https://semver.org/`
[31] `https://github.com/NTNU-IHB/FMU-proxy/blob/master/README.md`

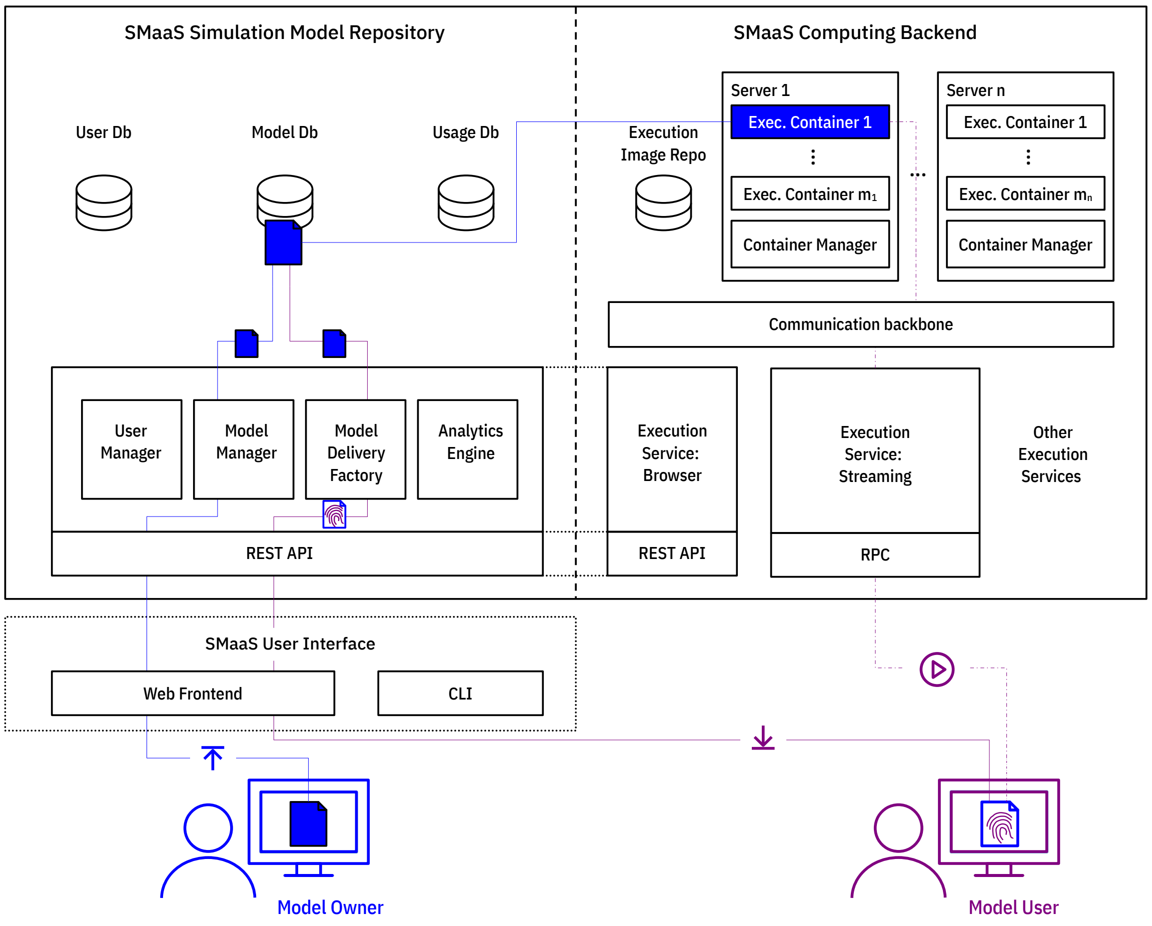

Figure 1. SMaaS concept architecture with illustration of model delivery and simulation using streaming execution

browser. Hence, within the SMaaS concept it is possible to implement a web-based frontend as an execution service, where model users can trigger model evaluations from a browser. This is particularly useful to perform simple runs in an explorative fashion, and also for users who do not have access to a simulation tool or lack the required knowledge. Hence, model owners can allow browser-based execution for a broader user audience without having to develop a dedicated web application.

Individualized model delivery Not in all applications simulation model streaming is an option. For instance, IT guidelines of a company could forbid communication to external services, or the target platform could have no (sufficiently reliable) network connection, *e.g.* a production machine or an autonomous vehicle. In these cases the only option is to deliver a copy of the original model to the model user – including all the drawbacks mentioned before. However, with the simulation model repository infrastructure it is possible to individualize copies of the simulation model and to augment them with additional control functionality. We call this *Simulation Model on a Leash* (SMoaL). It is ongoing research at simercator and might be subject to a future publication.

Finally, the SMaaS concept is not specific to the FMI, but can be adopted to future (*e.g.* microservice-based) exchange standards for computational models. For instance the Open Neural Network Exchange[32] format (ONNX).

Evaluation With the exception of RO5 and RU1, the fulfillment of the requirements from section 4 is inherently possible if SMaaS is implemented with the streaming execution service. Depending on the actual implementation, also RO5 and RU1 can be satisfied. In order to fulfill RU2 one can expose a subset of the model owner's monitoring mechanisms implemented for RO2 to the model user. We summarize the fulfillment of requirements in Table 1.

Table 1. Requirement analysis: SMaaS with streaming execution service. (* depends on implementation and deployment)

RO1	RO2	RO3	RO4	RO5	RU1	RU2	RU3	RU4	RU5	RC1	RC2	RC3
✓	✓	✓	✓	✓*	✓*	✓	✓	✓	✓	✓	✓	✓

7 Realization: simercator hub

At simercator we develop simercator hub[33] as a product that implements the SMaaS concept introduced above.[34]

[32]`https://onnx.ai/`
[33]`https://simercator.com/product/`
[34]Since the implementation is proprietary and part of commercial activity, we cannot not provide implementation details

7.1 Implementation and features

simercator hub offers a web frontend for user management, uploading or connecting to a model, and the possibility to share models with others. Simulation model streaming is implemented and FMU models in FMI 2.0 (both model exchange and co-simulation) are supported as original models that are then distributed via *doppelgaengers*. Note that the FMU *doppelgaenger* can be compiled and executed on platforms different from the ones the original FMU is supporting.

Our recent product iteration implements browser-based model evaluation, where a web-interface is auto-generated from a model's inputs and outputs and other meta-information. Python-native simulation models are supported as well, *e.g.* data-driven models, and the proprietary Python module `simercator` helps model owners to wrap their simulation models and define inputs and outputs in a single `main.py` file, so that simercator does not intrude the actual model implementation. Built-in visualization features allow the model owner to provide interactive 2D and 3D plots to the user through the web frontend.

We also feature a basic semantic versioning system that allows to enable and disable specific model versions. Furthermore, we provide integrated model usage data acquisition and basic analytics in an integrated dashboard.

simercator hub is designed as a licensable sofware with different commercial license options: 1) host a private instance of simercator hub, *e.g.* on company owned server infrastructure, or 2) work on an instance hosted and managed by simercator with the option to have a client-specific dedicated instance.

7.2 Showcase: FEM model as a service into vehicle system simulation

We now provide a fictitious showcase how simercator hub can help to achieve a significant speed-up in setting up and executing a system simulation, as illustrated in Figure 2. Here, a system engineer wants to perform a vehicle dynamics simulation for a vehicle climbing a curb at a comparatively high speed ($\approx 10\frac{\text{km}}{\text{h}}$). The simulation is used to predict potential damage to a hard rubber absorption buffer and effects on passenger comfort. To assess damage, the stresses in the absorption buffer need to be computed using FEM analysis (FEA), while the analysis of passenger comfort requires us to include a suitable buffer model into the overall vehicle system simulation. The system modeling is done in a Modelica tool (OpenModelica[35] (Fritzson et al. 2020)). In the following example, we will refer to the system engineer owning the vehicle system dynamics model as "Alice" and to the simulation engineer owning the buffer FEM model as "Bob".

Typical workflow today Today, Alice would typically use a surrogate model for the absorption buffer in her system simulation (nonlinear spring). The system simulation result for the buffer compression is then handed over to Bob to carry out the damage assessment using the FEM model. This is slow and error-prone, because the construction of a meaningful surrogate model requires a known force-compression-dataset. Alternatively, Bob can provide a FMU that contains a FEM model of the absorption buffer to Alice. However, the solver of choice (Calculix[36] (Dhondt 2004)), does not provide an FMI interface that could be used here.

Simulation model exchange using SMaaS As a preparation, Bob needs to prepare a Python script that wraps the Calculix solver of the (for simplicity) elastostatic FEA. The script calls the actual FEA as an external process. Then PythonFMU[37] (Hatledal, Zhang, and Collonval 2020) is used to turn the script into a co-simulation FMU (FMI 2.0). The FMU accepts the enforced displacement on the buffer surface in x, y, and z direction as inputs. It computes and outputs the reaction force in corresponding directions x, y, z and also returns the maximum von Mises stress from the buffer's bulge.

Figure 3 illustrates the showcase workflow with simercator hub. After logging in, Bob uploads the FMU from above to the simercator hub instance, together with text-based meta-information (or as a JSON file). The simercator hub instance provides pre-built docker images with a collection of pre-installed open source solvers, including one with Calculix that Bob uses. After that, Bob creates an account for Alice on simercator hub and authorizes Alice to view and download the FEA model as a *doppelgaenger*.

Alice now logs in to the simercator hub instance to browse models that she has access to. She can also assess whether the model fits her needs from the metadata. Alice uses the download option to retrieve the model *doppelgaenger* of the model provided by Bob. It only contains a communication library, the same `modelDescription.xml` of the original FMU, and

[35] `https://openmodelica.org/`

[36] `http://www.calculix.de/`

[37] `https://github.com/NTNU-IHB/PythonFMU`

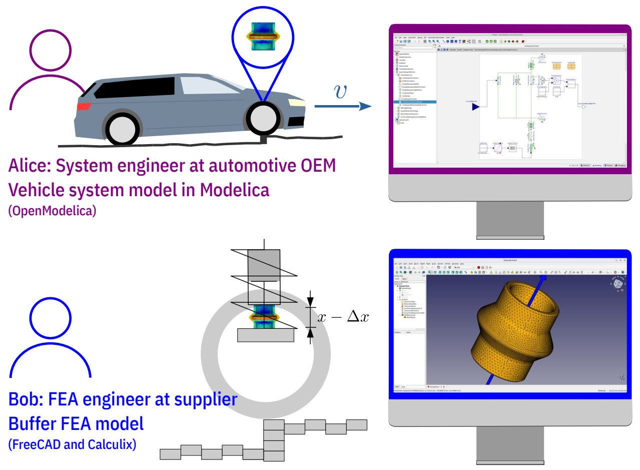
v
Alice: System engineer at automotive OEM
Vehicle system model in Modelica
(OpenModelica)
$x-\Delta x$
Bob: FEA engineer at supplier
Buffer FEA model
(FreeCAD and Calculix)

Figure 2. Showcase for a SMaaS scenario

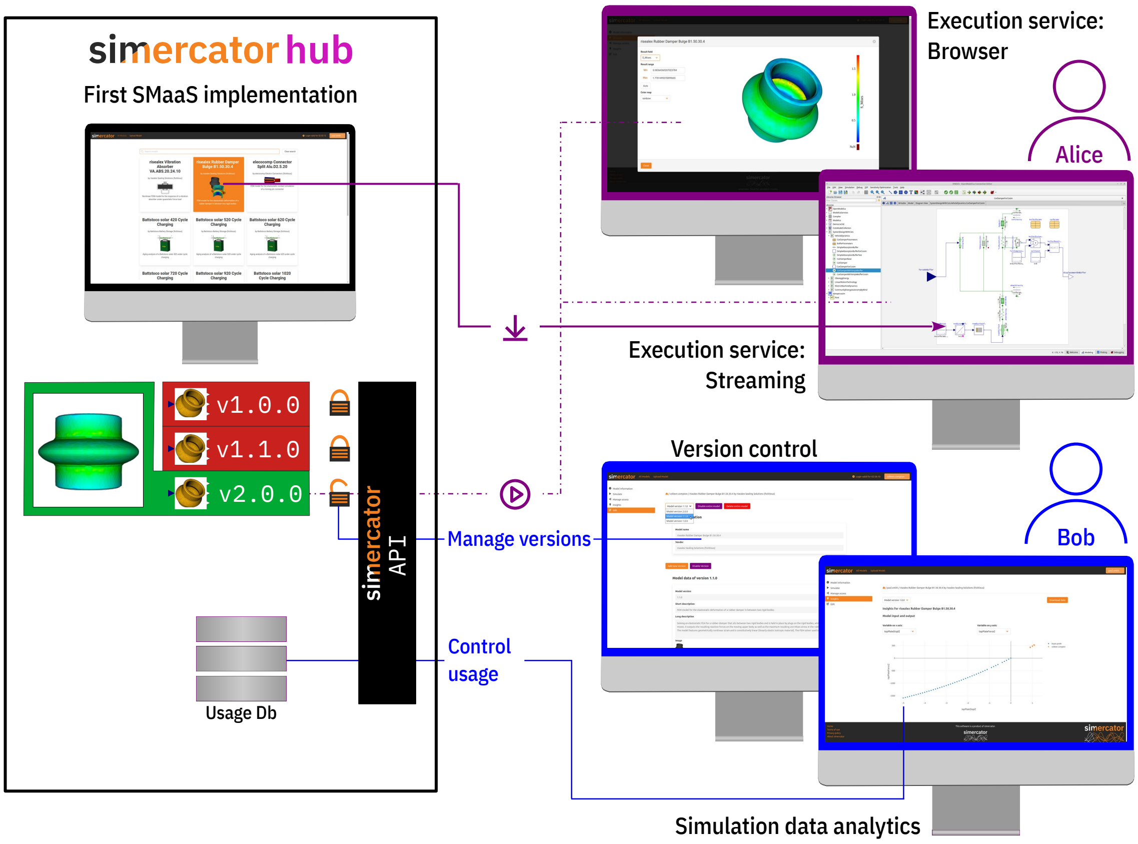
simercator hub
First SMaaS implementation
v1.0.0
v1.1.0
v2.0.0
simercator API
Usage Db
Manage versions
Control usage
Execution service: Browser
Alice
Execution service: Streaming
Version control
Bob
Simulation data analytics

Figure 3. simercator hub implementation of SMaaS. Screenshots of use in showcase.

an individually generated token that ensures that only authorized users can execute this *doppelgaenger*. Because it contains the same `modelDescription.xml`, to Alice it looks just like the original model.

Alice wants to use the vertical displacement that originates from her suspension model as an input for the buffer *doppelgaenger*. Then, she wants to use the vertical reaction force computed by the buffer *doppelgaenger* as an input for her suspension model. Therefore, Alice exports the suspension model with corresponding input and output connections into a FMU for co-simulation. Then she combines both the suspension FMU and the FMU with the *doppelgaenger* into one weakly coupled co-simulation. When the co-simulation is run, the *doppelgaenger* model for the buffer establishes communication with the simercator hub instance via the streaming process explained above. Bob does not know about the surrounding system simulation. Alice's simulation model is entire invisible to Bob. Likewise, Alice can only access the outputs of Bob's model and no internal details are exposed, such as the FE mesh or material properties.

The workflow described above takes Alice a few minutes (log-in, download, co-simulation setup). Also, Bob can now make this simulation model available to multiple experts via simercator hub's user management.

With simercator hub, also an alternative approach is possible: Using the browser-based evaluation, Alice can query the response forces for various displacements from the browser. This allows her to create a surrogate model with a nonlinear spring from data of the FEM model.

8 Conclusion and future work

We described why the exchange of simulation and computational models between system manufacturers and suppliers of mechatronic and cyber-physical systems needs a dedicated solution ("digital supply chain"). Then we outlined that today's file-transfer based exchange cannot satisfy natural technical and possible future regulatory requirements for the operation and maintenance of rolled out models. With SMaaS we have formulated a concept and an architecture that can provide a suitable solution for the exchange of simulation models and integrates with existing simulation tools and standards like FMI. Finally, we have presented our product simercator hub that implements SMaaS, illustrated how it can satisfy the identified requirements, and demonstrated SMaaS and simercator hub within a collaborative system simulation use case.

A key point for SMaaS will be whether and how organizations will accept communication to the outside when it comes to simulation, as some form of communication and collaboration is required by all implementation variants of SMaaS. Recent adoptions of package managers and software as a service (SaaS) solutions have shown that companies are willing to implement this if the gains are sufficient. This has been the case for business applications and can now be observed with the rise of SaaS solutions for simulation software. Additional efforts have to be undertaken to improve dependency management for FMI-based models or other standards that may arise in the future. Also, it remains to be seen how simulation models will be affected by future regulation. From a technical perspective, additional research is needed on performance speedup in network-based co-simulation. Finally, adoption of SMaaS or other *package repository like* solutions also depends on economic considerations with potential adopters. SMaaS implementations like simercator hub can provide infrastructure for digital supply chains. Whether companies will continue to share simulation models purely request-driven like today or use it as a digital service to differentiate from competitors will affect the rate of adoption.

References

Boschert, Stefan, Christoph Heinrich, and Roland Rosen (2018). “Next generation digital twin”. In: *Conference: TMCE 2018*. Vol. 2018. Las Palmas de Gran Canaria, Spain, pp. 7–11.

Dhondt, Guido (2004). *The finite element method for three-dimensional thermomechanical applications*. John Wiley & Sons.

European Commission (2018). *Ethics Guidlines for Trustworthy AI*. Tech. rep. European Union. URL: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai.

European Commission (2021a). *Annexes to the Proposal for a regulation of the European Parliament and of the Council laying down harmonized rules on artificial intelligence (Artificial Intelligence Act) and ammending certain union legislative acts.* Tech. rep. European Union. URL: https://ec.europa.eu/newsroom/dae/redirection/document/75789.

European Commission (2021b). *Proposal for a regulation of the European Parliament and of the Council laying down harmonized rules on artificial intelligence (Artificial Intelligence Act) and ammending certain union legislative acts.* Tech. rep. European Union. URL: https://ec.europa.eu/newsroom/dae/redirection/document/75788.

Federal Ministry for Economic Affairs and Climate Action (2022). *Part 1 – The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02)*. Tech. rep. Federal Ministry for Economic Affairs and Climate Action. URL: https://industrialdigitaltwin.org/wp-content/uploads/2022/06/DetailsOfTheAssetAdministrationShell_Part1_V3.0RC02_Final1.pdf.

Fritzson, Peter et al. (2020). “The OpenModelica integrated environment for modeling, simulation, and model-based development”. In: *Modeling, Identification and Control* 41.4, pp. 241–295.

Hannay, Jo Erskine and Tom van den Berg (2017). “The NATO MSG-136 reference architecture for M&S as a service”. In: *Proc. NATO Modelling and Simulation Group Symp. on M&S Technologies and Standards for Enabling Alliance Interoperability and Pervasive M&S Applications (STO-MP-MSG-149)*.

Hatledal, Lars Ivar, Arne Styve, et al. (2019). “A Language and Platform Independent Co-Simulation Framework Based on the Functional Mock-Up Interface”. In: *IEEE Access* 7, pp. 109328–109339. DOI: 10.1109/ACCESS.2019.2933275.

Hatledal, Lars Ivar, Houxiang Zhang, and Frederic Collonval (2020). “Enabling Python Driven Co-Simulation Models With PythonFMU.” In: *Proceedings of the 34th International ECMS-Conference on Modelling and Simulation-ECMS 2020*. ECMS European Council for Modelling and Simulation, pp. 235–239.

Hatledal, Lars Ivar, Houxiang Zhang, Arne Styve, et al. (2019). “FMU-proxy: A Framework for Distributed Access to Functional Mock-up Units.” In: Proceedings of the 13th International Modelica Conference (March 4–6, 2019). Ed. by Anton Haumer. Regensburg: Linköping University Electronic Press, pp. 157–008.

Industrial Digital Twin Organization (2022). *IDTA 02005-1-0. Provision of Simulation Models*. Tech. rep. Industrial Digital Twin Organization. URL: https://industrialdigitaltwin.org/wp-content/uploads/2023/01/IDTA-02005-1-0_Submodel_ProvisionOfSimulationModels.pdf.

Legaard, Christian M. et al. (2021). “A Universal Mechanism for Implementing Functional Mock-up Units”. In: *11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SIMULTECH 2021. Virtual Event, pp. 121–129.

Modelica Association (2022). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 3.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org/docs/3.0/.

Modelica Association (2023). *Modelica Specification, Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://modelica.org/documents/MLS.pdf.

Müller, Dirk et al. (2016). “AixLib-An open-source modelica library within the IEA-EBC annex 60 framework”. In: *BauSIM 2016*, pp. 3–9. URL: https://github.com/RWTH-EBC/AixLib.

Schranz, Thomas et al. (2021). “Portable runtime environments for Python-based FMUs: Adding Docker support to UniFMU”. In: *Proceedings of the 14th International Modelica Conference*. Linköping University Electronic Press, pp. 419–424.

SET Level (2021). *Credible Simulation Process*. Tech. rep. SET Level Research Project. URL: https://setlevel.de/assets/forschungsergebnisse/Credible-Simulation-Process.pdf.

Siegfried, R, J Lloyd, and TVD Berg (2018). “A new reality: Modelling & Simulation as a Service”. In: *Journal of Cyber Security and Information Systems* 6.3, pp. 18–29.

Taylor, Simon et al. (2015-07). “Grand challenges for modeling and simulation: Simulation everywhere - From cyberinfrastructure to clouds to citizens”. In: *SIMULATION* 91, pp. 648–665. DOI: 10.1177/0037549715590594.

Trauer, J. et al. (2022). “A Digital Twin Trust Framework for Industrial Application”. In: *Proceedings of the Design Society* 2, pp. 293–302. DOI: 10.1017/pds.2022.31.

Vladirmirescu, Andrei (2011). “Shaping the History of SPICE”. In: *IEEE Solid-State Circuits Magazine* 3.2, pp. 36–39. DOI: 10.1109/MSSC.2011.942105.

Wetter, Michael, Christoph van Treeck, et al. (2019). “IBPSA Project 1: BIM/GIS and Modelica framework for building and community energy system design and operation–ongoing developments, lessons learned and challenges”. In: *Iop conference series: Earth and environmental science*. Vol. 323. 1. IOP Publishing, p. 012114. URL: https://github.com/ibpsa/modelica-ibpsa.

Wetter, Michael, Wangda Zuo, et al. (2014). “Modelica Buildings library”. In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506. URL: https://github.com/lbl-srg/modelica-buildings.