Using Modelling and Simulation as a Service (MSaaS) for Facilitating Flexibility-based Optimal Operation of Distribution Grids

Moritz Stüber, Lukas Exel and Georg Frey
Chair of Automation and Energy Systems, Saarland University, Campus A5 1, 66123, Saarbrücken, Germany

Keywords: Modelling and Simulation as a Service (MSaaS), Simulation as a Service (SIMaaS), Service-oriented Architecture (SOA), Flexibility, Power-to-Heat.

Abstract: In this contribution, initial results of a research project on modelling and simulation as a service (MSaaS) within the context of optimal operation of distribution grids by exploiting flexibility are summarized. Based on a brief description of service-oriented architecture (SOA), definitions and key aspects of MSaaS as well as open conceptual and technical challenges are outlined. Some specific aspects and challenges are then related to a possible application: a service that predicts the flexibility that a power-to-heat system can offer is implemented as an instance of MSaaS. The calculations are based on the current state of the system and a model defined in the equation-based, object-oriented language Modelica, as well as historical data when available. Thereby, an accurate and reliable representation of the flexiblity becomes available that facilitates its use for system balancing, energy market participation and grid operations. This functionality is exposed through a representational state transfer (REST)-based service interface, designed to allow for the straightforward integration with existing systems and other virtual resources. The service architecture and initial results of the implementation are described.

1 INTRODUCTION

The term "modelling and simulation as a service (MSaaS)" describes the effort to provide users with modelling and simulation (M&S) functionality over the internet by using a service-oriented architecture (SOA) to structure applications and cloud computing technology to efficiently host them (Cayirci, 2013b). Before introducing MSaaS in more detail by elaborating proposed benefits and key aspects in section 2, a brief introduction to M&S as well as an introduction to SOA will be given below; an introduction to cloud computing can be found in Cayirci (2013b, section 2) and Mckee et al. (2017a, section 3).

From a systems engineering perspective, M&S can be seen as a way to calculate physical quantities in technical devices that is particularly suited for analysing complex, multi-domain systems. In the context of modelling and simulation using equation-based, object-oriented languages (EOOLs) such as Modelica[1], a model generally consists of a system of implicit differential algebraic equations (DAEs) which can be numerically approximated by a suitable solver for a given set of parameters and boundary conditions. Based on the resulting trajectory, a user can predict the behaviour of the real or imagined system that the model represents. Compared to experiments with prototypes, M&S offers faster and safer experimentation as well as access to internal states that cannot be measured in a real experiment (Cellier, 2013). Due to the acausal formulation of the models, EOOLs allow for fast assembly of complex system models by reusing available component models.

The first step in creating a model is finding an abstraction of a system that is suitable for the purpose of the model. In-depth domain knowledge is required in order to understand and describe the *relevant* properties of the system at hand. Second, the model needs to be implemented in a formal modelling language, which requires expertise in the modelling formalisms, languages and algorithms used because attention to the numerical properties of the model and the use of structuring mechanisms that ensure the reusability of the model must be paid. M&S environments such as Dymola[2] provide support regarding the implementation of the models by checking

[1] https://modelica.org/

[2] https://www.3ds.com/products-services/catia/products/dymola/

Stüber, M., Exel, L. and Frey, G.
Using Modelling and Simulation as a Service (MSaaS) for Facilitating Flexibility-based Optimal Operation of Distribution Grids.
DOI: 10.5220/0006899106230630
In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018) - Volume 2*, pages 613-620
ISBN: 978-989-758-321-6

for syntactic mistakes, providing access to model libraries, offering the possibility to graphically create models by assembling component models, translating the models to executable form, triggering their execution and allowing users to analyse the results. However, it remains the user's responsibility to make sense of the calculated trajectories and to ensure that they are valid. In practice, obtaining accurate parameter values and measurement data for verification and validation of developed models represents a major challenge.

Applications of M&S include dimensioning of technical devices, finding control strategies and assessing the quality and feasibility of ideas as well as control and training applications (hardware-in-the-loop, human-in-the-loop). For example, M&S could be used to predict the energy generated by a photovoltaic system or to compare different possibilities for increasing the own use of energy within a household from a technical and economical perspective. In this contribution, it is used to determine the flexibility of a power-to-heat (PtH) system, which is then used as an input to a larger process.

SOA is "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" (OASIS, 2006). The ISO/IEC standard 18384 (ISO/IEC_18384-1:2016, E) defines SOA as "an architectural style that supports service orientation", where *service orientation* is defined as an "approach to designing systems in terms of services and service-based deployment" and a *service* is defined as a "logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a 'black-box' to consumers of the service". In other words, a service describes the capability, the specification and an offer to perform work for someone and can also be seen as a mechanism to match the capability of the service provider with the need of the service consumer. SOA is a way of structuring and offering a functionality that promotes reuse, growth and interoperability by focusing on tasks and business functions and acknowledging the existence of ownership boundaries. The OASIS Reference Model for Service Oriented Architecture (OASIS, 2006) identifies six major concepts of SOA and describes their properties and relations: visibility, service description, interaction, contracts & policies, real world effect and execution context.

Provided that a service is visible to the consumer, interaction with the service results in a retrieval of information and/or in a change of the shared information about affected entities (real world effect). Thereby, it contributes to the goal of the service consumer, which is generally unknown to the service provider. The interaction with a service is defined by an information model and a behaviour model, which are exposed through the service interface. The service interface is formally described in the service description, which includes applicable contracts and policies. The infrastructure elements, process entities and policy agreements of a specific service interaction are called execution context. The execution context may evolve during an interaction, for example if the service participants agree to encrypt the following communication. Visibility, interaction and real world effect can be seen as the dynamic elements of a service interaction as they result from the actions of the service participants. In contrast, service description, policies and execution context can be seen as supporting aspects that are mostly defined by the service provider.

According to OASIS (2006, section 2.3), SOA is well suited for developing complex, yet manageable software systems that scale well and allow remote users to choose the services that they need. However, implementing a SOA in a robust way based on standards and best practices, fully exploiting the underlying concepts, can be challenging(Rodríguez et al., 2016) . Furthermore, systematic approaches for creating value by transitioning to a service-oriented architecture as well as concepts and algorithms for the automated composition of services are missing.

2 M&S as a SERVICE

By combining the concepts and tools of SOA and cloud computing with the functionality of modelling and simulation, it is hoped to help a wide audience make informed decisions when confronted with complex questions: the various methods, languages and tools summarized under the term modelling and simulation allow finding quantitative information about the behaviour of a system by numerically approximating the behaviour of a virtual representation of that system; SOA and cloud computing allow users to access this capability over the internet.

Currently, factors that prevent a more widespread use of modelling and simulation amongst engineers include a lack of knowledge regarding concepts, modelling languages, algorithms and tools, the cost of tools and model libraries as well as problems with the usability of user interfaces and data formats. Also, current M&S environments are not designed to be used as part of a larger process. Factors preventing the general public from using modelling and simulation include missing awareness of its capabilities, missing

background knowledge, missing feeling for the steps required to arrive at a solution and the challenge of using complex M&S software environments.

MSaaS attempts to address the aforementioned problems by introducing a new layer of abstraction: instead of dealing with methods and data in the form of models, parameters and unprocessed simulation results, a service consumer interacts with an interface designed for specific tasks/business functions. Thus, the needed functionality is separated from the technical prowess and infrastructure required to implement it. The service interaction results in the retrieval of *information* as defined by Rowley (2007): "[...] information is defined in terms of data, and is seen to be organized or structured data. This processing lends the data relevance for a specific purpose or context, and thereby makes it meaningful, valuable, useful and relevant".

MSaaS is defined as a "means of delivering value to customers to enable or support modelling and simulation (M&S) user applications and capabilities as well as to provide associated data on demand without the ownership of specific costs and risks" by the specialist team MSG-131 at NATO (MSG-131, 2015)v. Cayirci (2013b) sees MSaaS as a "model for provisioning modelling and simulation (M&S) services on demand from a cloud service provider (CSP), which keeps the underlying infrastructure, platform and software requirements/details hidden from the users", which aligns well with the previous definition but highlights the use of cloud computing technologies. This aspect is important for distinguishing the current research activities from earlier attempts, summarized under the term web-based simulation (WBS), as retraced by Wang and Wainer (2016, section 2). Consequently, MSaaS is one form of software as a service (SaaS) (Cayirci, 2013b; Mckee et al., 2017a) that "inherits" the general benefits and challenges of cloud computing, but also offers opportunities peculiar to the field of M&S.

The proposed value of MSaaS (Cayirci, 2013b; MSG-131, 2015, section 2.6) lies in an increase in usability *and* functionality on the one hand and a possible decrease in costs on the other hand. Increased usability and functionality are expected to stem from better accessibility of M&S resources, their reuse for reproducing results and the creation of new functionality through composition of resources:

- The *accessibility* of M&S resources is increased through broad network access that allows on-demand self-service while having very light requirements on the client (hardware, operating system (OS), software).
- Accessible resources that hide their complexity and implementation from the user (encapsulation) make the *reuse* of knowledge possible, for example the provision of product models by companies.
- Also, providing an interface to an operational simulation environment can enhance the credibility of a simulation study by ensuring its *replayability* and makes sure that a user can always access the latest version of the software.
- Implementing new functionality by *composition* of services is facilitated by provisioning M&S resources as a service. Composition can reduce the time and effort needed for implementation and extend the scope of M&S resources by using them in conjunction with functionality that is not normally available in an M&S environment.

A decrease in costs can be achieved by measuring the service usage, which allows pay-per-use options, by scaling the infrastructure according to demand and by increasing the degree of capacity utilization through resource pooling. Furthermore, the service consumer does not need to invest in hardware, software and personnel, which could increase the willingness of companies to try to integrate M&S into their workflow.

Researchers have worked on combining M&S with web technologies for more than 20 years by imagineering the possibilities (Fishwick, 1996; Mckee et al., 2017a), proposing architectures for implementation (Cubert and Fishwick, 1997; Al-Zoubi and Wainer, 2011; Ribault and Wainer, 2012; Shekhar et al., 2016), pondering risk, trust and accountability (Cayirci, 2013a) and investigating the composability of M&S services (Tolk, 2013; Tolk and Mittal, 2014). Below, we will elaborate what we perceive as the key aspects of MSaaS.

Service Interface. The *service interface* defines the possibilities for interaction with a service. It combines the underlying functionality with the service's execution context. The possibilities and limitations of the latter define the degree to which the ideas and proposed benefits of SOA and cloud computing applied to M&S can be realized. This applies to both the general characteristics of cloud computing and internet technologies as well as the specific characteristics of the chosen service architecture.

Service Architecture. The *service architecture* structures an implementation and comprises the underlying design concepts, for example representational state transfer (REST), the use of standards such as OpenAPI or functional mockup interface (FMI), the software tools and

middleware, the deployment technology as well as the available computing power.

Service Composition The full potential of transitioning to using SOA for providing M&S capabilities cannot be realized when limiting the implementations to providing existing M&S functionality *in its current form* over the internet as a web application. An increase in value stems from new use cases and target audiences for M&S technology and new functionality for users which can be achieved by *service composition*. Examples are presented by Wang and Wainer (2016); Wainer and Wang (2017); Mckee et al. (2017b). However, service composition represents a conceptual and technical challenge because a consistent and viable semantics needs to be guaranteed across services in order to achieve user acceptance of the proposed solutions. Tolk and Mittal (2014), therefore, propose that a consistent representation of truth can be seen as the definition of composability and state that it can be achieved by alignment of data and the orchestration of the individual services.

Contracts and Policies Providing functionality for a *remote* user makes an explicit and comprehensive consideration of the implications on security, privacy, intellectual property and accountability necessary (Cayirci, 2013b, section 4, 5). This means that an analysis of risk and trust needs to be performed and adequate countermeasures need to be implemented. Ideally, the countermeasures are integrated into the service architecture *by design*. From the perspective of a SOA, *contracts and policies* are the mechanisms for dealing with the aforementioned. In general, *contracts* are negotiated between the service participants whereas *policies* are enforced by the service provider. They govern the willingness of a service to perform a request and can be related to infrastructure-oriented as well as business-oriented matters. Two examples that represent the need for *business-oriented* policies and highlight the importance of considering the definition of policies and contracts thoroughly are *privacy* and *intellectual property*.

Significant conceptual and technological challenges have to be addressed before the full potential of MSaaS can be understood and, ultimately, realized. Consequently, cloud-based M&S or MSaaS was identified as one of the "grand challenges for modelling and simulation" by Taylor et al. (2015, section 4).

Conceptual challenges include questions such as "what makes sense?", "which granularity of services makes sense?", "how can SOA and cloud computing be exploited to full capacity?", "how can trust regarding the reliability of results be induced?", "how can trust regarding the use of sensitive data be induced?" and "how can a consistent representation of truth be guaranteed across services?" as well as "how to derive viable business models?". On the technical side, it is necessary to combine the expertise, tools and methods from M&S, SOA and cloud computing in order to find a software architecture that facilitates the sustainable development of M&S services. Additionally, solutions for formulating and enforcing contracts and policies and ensuring the security of the service as well as its reachability need to be found.

In the next section, an envisioned application of MSaaS for facilitating the flexibility-based optimal operation of distribution grids is described.

3 DETERMINATION OF THE FLEXIBILITY OF A POWER-TO-HEAT SYSTEM

The increasing share of energy generated by solar and wind power plants imposes challenges on the electrical power grid. Because of the dependency of these power plants on the weather, utilities have to find ways to guarantee the stability of the grid despite the volatile nature of the energy generation. One possible way of stabilizing the grid is to use loads flexibly and consume power when it is generated, in other words using *flexibility* for optimal operation of the grid. Flexibility is defined as "the changes in consumption/injection of electrical power from/to the power system from their current/normal patterns in response to certain signals, either voluntarily or mandatory" (SG-CG/M490/L, 2014, section 5.1).

3.1 Scenario

As an exemplary application, the energy consumption of a building complex is investigated with respect to possiblities to adapt the consumption of electrical energy on demand. In the system under investigation, two such possibilities exist: first, the warm water can be generated by using submersion heaters instead of the normal gas heating system, which represents a load that can be switched on and off flexibly. The submersion heaters are mounted inside the storage tank. Second, there is a solar photovoltaic system which can be throttled on demand. Consequently, the overall system represents a buffered flexibility source which is controllable within bounds.

In order to reliably determine *how much* energy

can be consumed by the PtH system and/or not generated by the solar panels for a given time frame, modelling and simulation of the overall system is necessary due to its complexity: among other factors, the flexibility that can be offered by the PtH-components depend on the current state of the system, the maximum allowed temperature of the water, the temperature of the fresh water, the warm water consumption, and the heat losses to the environment; the flexibility of the photovoltaic system mainly depends on the weather conditions, but for example also on the implemented operating strategies et cetera.

As a first step, a very simple model of the building was implemented using the modeling language Modelica. It mainly consists of a model for warm water generation and -storage and the control logic that ensures that whenever the PtH-components are active, the water is *not* heated by the gas heating system. The level of the thermal energy storage, defined as the difference of the current temperature of the water inside the tank to the minimum temperature divided by the allowed temperature range, is introduced as a characteristic value. Parameters of the used models were set to plausible values, but there is no direct relation to a real building; currently, the purpose of the model is to provide plausible data as part of a proof-of-concept implementation. The electric power generated by the solar power plant as well as warm water consumption, heating demand and consumption of electrical power are defined as inputs to the system model.

The flexibility arising from throttling the solar power plant is calculated by subtracting the electric energy consumption inside the building from the generated power. The flexibility provided by the PtH-components is calculated as follows: the model of the building is used as the plant model within a closed control loop, using the level of the thermal energy storage as the measured process variable and 100 % as the value of the desired set-point. The PtH-components are activated and the power they are required to provide to the system in order to always keep the level of the thermal energy storage at the desired set-point is defined as the flexibility they can offer.

3.2 Service Architecture

Providing an answer to the question "how much energy can be consumed/not generated on demand by the system in the next 24 h?" represents the capability that shall be offered *as a service* in order to enable the service consumer to offer the calculated flexibility to potential users. The service shall also integrate necessary auxiliary steps, such as querying forecasts for weather and user behaviour, thus hiding the underlying complexity of the calculation from the service consumer.

Consequently, the service is realized as an instance of *service composition* from a technical point of view: while the endpoints of the service that provides the calculation of the flexibility for a certain system are specific to the needs of the service consumer, different services are called in the background and the results are combined such that the desired functionality follows, as shown in Figure 1.

First, the necessary input for the simulation of the building are collected, namely the forecast for the electric power generated by the solar power plant, which in turn requests an up-to-date weather forecast before triggering the simulation of the underlying model, as well as forecasts for the behaviour of the residents (warm water consumption, heating demand, consumption of electric power). Then, the actual simulation is triggered by sending a request to a simulation as a service (SIMaaS)-instance.

For each of the developed services, a formal *service description* needs to be found based on the description of the desired service functionality. The service description "represents the information needed in order to use a service" (OASIS, 2006, section 3.3.1) and can comprise information on the service's reachability, functionality, its interface and applicable policies. It exposes the set of capabilities offered by the service provider to a remote audience, but its expressivity and therefore the degree to which these capabilities can be exposed is defined by the service architecture and the underlying architectural style. Consequently, the development of the service description and the development of the service architecture go hand in hand. In the presented case, the service interface adheres to the architectural style REST due to its advantages over other approaches, following the argumentation presented by Wang and Wainer (2016, section 2).

According to Verborgh et al. (2015, section 3.1), REST is "a style for system architectures, dictating several architectural constraints, rather than a technology, or architecture in itself". Web services adhering to REST are called *RESTful web services* and their service interface is also referred to as *REST-API* (Rodríguez et al., 2016, section 2). REST applies to client-server systems, based on the idea that the resulting separation of concerns facilitates scalability and longevity. The constraints characterizing REST concern the identification of resources, the manipulation of resources through representations, the exchange of self-descriptive messages (stateless interactions) and the use of hypermedia as the engine of

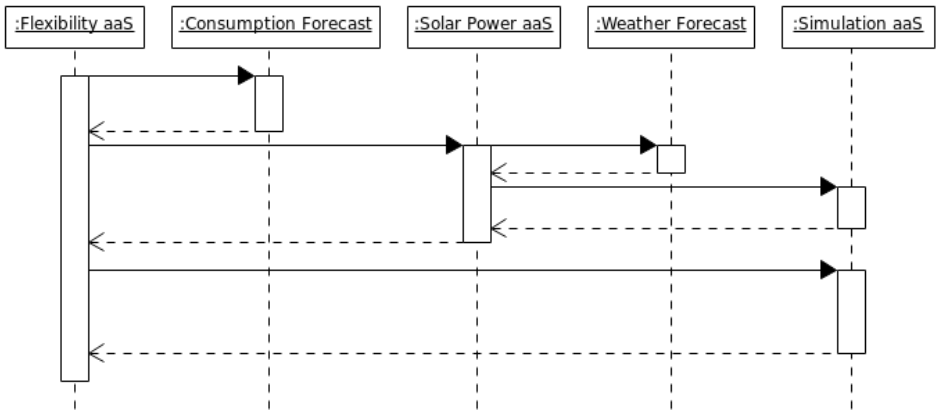

Figure 1: The functionality of the service that provides the desired information is realized by combining several services.

application state (HATEOAS); they are intended to result in a *uniform interface* (Verborgh et al., 2015, section 3.3).

In the abstract, a REST-API gives access to a set of *resources*. Resources are unique conceptual entities of the service interface; the semantics of a resource always remains the same. Conceptual entities of the application domain are mapped to resources when defining the service functionality. Resources can be considered as nouns that are uniquely identified by URIs. Interaction with resources occurs by applying a set of verbs defined by the hypertext transfer protocol (HTTP) specification to these nouns. Applying an HTTP-verb to a resource results in a transaction of a *resource representation* that is defined by the (hyper)media type agreed upon by the service participants. Therefore, the *same* service can be used to integrate with a variety of existing systems by implementing the appropriate resource representation.

Defining the resources provided by a given service is not sufficient for describing the *service interface*, which defines the specifics of how to interact with the service in order to invoke its functionality. As the exchange of messages represents the "primary mode of interaction with a service" (OASIS, 2006, section 3.2.2), the structure and semantics of the data that needs to be exchanged have to be defined as well, resulting in the service's *information model*. The corresponding *behaviour model* defines the possible interactions with the service as well as their temporal relationships and properties and also needs to be described.

In order to describe these details in a standardized way that is readable by both humans and machines, a web service description language should be used. Because the software under development is described in terms of resources, actions and representations instead of objects, methods and attributes, a web service description language promotes the realization of SOAs. The resulting document, the service description, serves as documentation for humans and can be used to automatically derive server stubs, thus significantly reducing the amount of code that needs to be implemented manually. Therefore, the interfaces of all developed services are described according to the OpenAPI-specification[3].

The performance of a service instance is defined by the characteristics of the implementation *and* the characteristics resulting from deploying it on a certain hardware using a certain technology. Thus, based on the answer to the question "which performance characteristics do I envision for the service?", an answer to the question "how do I deploy the service in order to fulfil them?" needs to be found. Aspects to be considered include the formulation, measurement and testing of key performance indicators (KPIs) as well as the scalability of the chosen solution. Since MSaaS is described as the combination of M&S, SOA *and* cloud computing, using infrastructure as a service (IaaS), platform as a service (PaaS) and SaaS as well as associated technologies like Docker in order to achieve the characteristics associated to "the cloud" also need to be considered.

Typically, M&S environments and -tools have extensive dependencies. In order to avoid problems due to those dependencies and in order to clearly separate the different modules of a service, it was decided to use Docker as a software container platform. After explicitly defining the dependencies of a software once, Docker guarantees that the software runs no matter where it is deployed and has a number of additional advantages, such as its suitability for agile software development processes.

[3] https://github.com/OAI/OpenAPI-Specification

3.3 Discussion

The predictions delivered by the developed service are currently not yet applicable in a real practical application. However, implementing the functionality *as a service* increases both the accessibility of simulation resources and the reuse of knowledge: instead of using a complex and typically expensive M&S-environment, which requires manual work and/or scripting, the service consumer sends a simple HTTP-request to a web API. This facilitates the reuse of an analysis, such as the calculation of the flexibility, within a larger process. Therefore, also the reuse of knowledge and capabilities is promoted. For example, a SIMaaS-instance is used by both the service that predicts the power generated by a solar power plant as well as by the service that predicts the flexibility.

The use of service composition for implementing a certain functionality has two major advantages: first, it allows defining analyses in the problem domain without having to take care of how the individual parts required for the analysis are implemented. For example, obtaining the latest forecast for the solar power generation reduces to calling an application programming interface (API), the complexity of handling the underlying model and auxiliary data such as the weather forecast are hidden from the consumer of this API. Second, using services makes it easy to change them; for example, all forecast services could be replaced by services that provide measurement data, thus allowing to investigate the quality of the forecast services and allowing to perform parameter fitting et cetera.

Many aspects of using MSaaS for providing information as part of a larger process have not been considered yet. Most importantly, these include considering the interoperability of services both from a conceptual and a technical point of view, and the scalability of the developed solutions. Moreover, threats to the security of the solutions as well as possible mitigations need to be considered.

With respect to the service functionality, a process for evolving model instances based on parameter estimation shall be devised and implemented in future work: in the first iteration, an initial set of parameters is chosen. In subsequent iterations, the parameters are optimized by *automatically* comparing the simulation results to measurement data, which is linked to the model instance and accessible from within the service.

4 CONCLUSION

MSaaS represents a field of study at the intersection of M&S, information science and software engineering with the goal of increasing the accessibility of and audience for the formalisms to describe knowledge about a system and the algorithms used to perform experiments based on the resulting models that were found in M&S research. Concepts developed in information sciences, such as service-orientation, provide the necessary theoretical background to transfer existing solutions to the new problem domain of M&S and to identify and reason about challenges peculiar to this domain, for example the composability of M&S-services. Software engineering methods and tools, in particular cloud computing, the concept of representational state transfer (REST), API first-development and containerization enable the realization of solutions that meet the functional *and* non-functional requirements resulting from a specific service concept.

An exemplary application of MSaaS, namely providing an accurate and reliable description of the flexibility of a PtH system that can be used to mitigate unwanted states of the distribution grid in subsequent processes, was outlined.

ACKNOWLEDGEMENTS

This work was supported by the SINTEG-project "Designetz" funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) under grant 03SIN224.

REFERENCES

Al-Zoubi, K. and Wainer, G. (2011). *Distributed Simulation Using RESTful Interoperability Simulation Environment (RISE) Middleware*, pages 129–157. Springer Berlin Heidelberg, Berlin, Heidelberg.

Cayirci, E. (2013a). A joint trust and risk model for msaas mashups. In Pasupathy, R., Kim, S.-H., Tolk, A., and Kuhl, M. E., editors, *Proceedings of the 2013 Winter Simulation Conference (WSC)*.

Cayirci, E. (2013b). Modeling and simulation as a cloud service: A survey. In Pasupathy, R., Kim, S.-H., Tolk, A., and Kuhl, M. E., editors, *Proceedings of the 2013 Winter Simulation Conference (WSC)*.

Cellier, F. E. (2013). The complexity crisis: Using modeling and simulation for system level analysis and design. In *3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, Reykjavik, Island.

Cubert, R. M. and Fishwick, P. A. (1997). A framework for distributed object-oriented multimodeling and simulation. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 1315–1322, Washington, DC, USA. IEEE Computer Society.

Fishwick, P. A. (1996). Web-based simulation: Some personal observations. In *Proceedings of the 28th Conference on Winter Simulation*, WSC '96, pages 772–779, Washington, DC, USA. IEEE Computer Society.

ISO/IEC_18384-1:2016(E) (2016). Information technology – reference architecture for service oriented architecture (soa ra). ISO/IEC 18384-1:2016(E), ISO/IEC.

Mckee, D., Clement, S., Ouyang, X., Xu, J., Romano, R., and Davies, J. (2017a). The internet of simulation, a specialisation of the internet of things with simulation and workflow as a service (sim/wfaas). In *11th IEEE International Symposium on Service-Oriented System Engineering (SOSE 2017)*. IEEE.

Mckee, D., Clement, S., Xu, J., Romano, R., Lopez, J., and Battersby, D. (2017b). The internet of simulation: Enabling agile model based systems engineering for cyber-physical systems. In *IEEE 12th System of Systems Engineering Conference (SoSE 2017)*, Waikoloa, Hawaii, USA. IEEE.

MSG-131, S. T. (2015). Modelling and simulation as a service: New concepts and service-oriented architectures. Final Report AC/323(MSG-131)TP/608, North Atlantic Treaty Organization NATO.

OASIS (2006). Reference model for service oriented architecture 1.0. Technical report.

Ribault, J. and Wainer, G. (2012). Using workflows and web services to manage simulation studies (wip). In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, TMS/DEVS '12, pages 50:1–50:6, San Diego, CA, USA. Society for Computer Simulation International.

Rodríguez, C., Baez, M., Daniel, F., Casati, F., Trabucco, J. C., Canali, L., and Percannella, G. (2016). Rest apis: A large-scale analysis of compliance with principles and best practices. In Bozzon, A., Cudre-Maroux, P., and Pautasso, C., editors, *Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings*, pages 21–39, Cham. Springer International Publishing.

Rowley, J. (2007). The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*, 33(2):163–180.

SG-CG/M490/L (2014). Overview of the main concepts of flexibility management. Technical Report Version 3.0, CEN/CENELEC/ETSI.

Shekhar, S., Abdel-Aziz, H., Walker, M., Caglar, F., Gokhale, A., and Koutsoukos, X. (2016). A simulation as a service cloud middleware. *Annals of Telecommunications*, 71(3):93–108.

Taylor, S. J. E., Khan, A., Morse, K. L., Tolk, A., Yilmaz, L., Zander, J., and Mosterman, P. J. (2015). Grand challenges for modeling and simulation: simulation everywhere–from cyberinfrastructure to clouds to citizens. *SIMULATION*, 91(7):648–665.

Tolk, A. (2013). Interoperability, composability, and their implications for distributed simulation: Towards mathematical foundations of simulation interoperability. In *17th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*.

Tolk, A. and Mittal, S. (2014). A necessary paradigm change to enable composable cloud-based m&s services. In *Proceedings of the 2014 Winter Simulation Conference*.

Verborgh, R., van Hooland, S., Cope, A. S., Chan, S., Mannens, E., and de Walle, R. V. (2015). The fallacy of the multi-api culture: Conceptual and practical benefits of representational state transfer (rest). *Journal of Documentation*, 71(2):233–252.

Wainer, G. and Wang, S. (2017). Mams: Mashup architecture with modeling and simulation as a service. *Journal of Computational Science*, 21:113–131.

Wang, S. and Wainer, G. (2016). Modeling and simulation as a service architecture for deploying resources in the cloud. *International Journal of Modeling, Simulation, and Scientific Computing*, 07(01):1641002.