Scalable Integration of 4GL-Models and Algorithms for massive Smart Grid Simulations and Applications

Thomas Preisler[1], Gregor Balthasar[2], Tim Dethlefs[3], Wolfgang Renz[4]

Abstract

This paper presents a scalable integration approach for algorithms and models written in fourth generation (programming) languages for massive Smart Grid simulations as well as applications. While fourth generation languages (4GL) focus on rapid application development and the reduction of lines of code, they lack of integration and scalability features. Nevertheless, they are widely spread and often used by engineers. The scalable integration of such elements is achieved in this paper by wrapping the 4GL-models and algorithms with established web technologies like RESTful web services and load balancing. The provision of a seamless integration concept allows engineers to focus on rapid application development and liberates them from integration efforts.

1. Introduction

Today's energy grid undergoes a structural change towards the so-called Smart Grid. The power grid will no longer be dominated by a relatively small number of large coal and nuclear power plants, but rather by a large number of distributed, renewable energy resources (DER). Hereby the control and coordination of this large number of DERs in order to balance the generation and demand is the main challenge. Due to the quantity and restrictions of the involved components it is a challenging task. With respect to the grid stability control strategies need to be developed as well as evaluated and tested particularly. To achieve this a wide range of simulation scenarios have to be established facilitating an economical transition towards a Smart Grid and ensuring its reliability [13]. The approach presented in this paper introduces a concept for the scalable integration of 4GL-models and algorithms for such Smart Grid simulations. It is also used in an ongoing industrial project dealing with the load-management of electrical heaters.

Fourth generation languages (4GL) are programming languages focussing on rapid application development while reducing the lines of code. The expression 4GL was coined by [9]. Recently this interpretation has gained new attention through the introduction of model based software development [1]. Examples for such 4GLs are MATLAB, Simulink, Modelica, GAMS (General Algebraic Modelling System) and many more. These languages respectively the languages offered by the according software products are used in many research projects for rapid prototyping as well as the realization of models or algorithms, e.g. for demand side management and the integration of regenerative energy resources into smart grids [4]. They reduce the overall development effort through the usage of comprehendible application-oriented paradigms. Furthermore, both the good

[1] Hamburg University of Applied Sciences, 20099 Hamburg, Germany, Thomas.Preisler@haw-hamburg.de, Faculty of Engineering and Computer Science, Department of Information and Electrical Engineering

[2] Hamburg University of Applied Sciences, 20099 Hamburg, Germany, Gregor.Balthasar@haw-hamburg.de, Faculty of Engineering and Computer Science, Department of Information and Electrical Engineering

[3] Hamburg University of Applied Sciences, 20099 Hamburg, Germany, Tim.Dethlefs@haw-hamburg.de, Faculty of Engineering and Computer Sciences, Department of Information and Electrical Engineering

[4] Hamburg University of Applied Sciences, 20099 Hamburg, Germany, Wolfgang.Renz@haw-hamburg.de, Faculty of Engineering and Computer Science, Department of Information and Electrical Engineering

readability and the user-oriented representation of 4GL programs facilitate the maintainability as well as the extensibility. This results in a significant reduction of development time and costs.

The integration of such 4GL models or algorithms into operative systems or large-scale simulations is a considerable challenge, as most of them have to be executed on their specific runtime environment. Even if they can be exported as independent external models and therefore be integrated into the infrastructure of operative systems (resp. simulations), a scalability problem remains. One possible solution is to rewrite the algorithms and models so they use the same programming language as the operative system or simulation. This may solve the integration and scalability problems, but it raises new problems and accordingly efforts: these rewritten parts have to undergo the whole process of validation and verification again in terms of ensuring their correct behaviour. In addition to these efforts also comes the task of rewriting the code in a different programming language with possibly lesser application-specific language elements.

The remainder of this paper is structured as follows. The next Section introduces related work on simulation interoperability and presents previous work, which forms the fundament for this approach. Section 3 introduces the scalable integration concept for 4GL elements that avoid the afore-mentioned integration problems and efforts by wrapping them with established web-technology and using load-balancing techniques to ensure scalability. The Section also describes how the integration approach is embedded into a service based simulation framework and how it can be facilitated for combined hard- and software Smart Grid simulation models. Section 4 concludes the paper and gives an overview about future work.

2. Simulation Interoperability

There exist many different tools and approaches for the simulation of Smart Grid scenarios or simulation interoperability in general. The military domain has been a major driver in this field. The development of simulation interoperability standards started in the early 1990s and resulted in the High Level Architecture (HLA), an IEEE Standard for modelling and simulation [8]. The HLA is a technical architecture developed to facilitate the reuse and interoperability of different simulation systems and assets. It provides a general framework, which can be used by developers to structure and describe their simulation systems and to interoperate them with other simulation systems. But due to its complexity it is hardly used outside the military domain [3].

A more specific approach is introduced in [2]. The IPSYS framework is a versatile framework for the simulation of integrated power systems with multiple forms of energy- and control-structures. It focuses on the system performance of large amount of renewable energy resources. However standardized interfaces to the simulated power systems are currently not offered, which makes it difficult to integrate external resp. heterogeneous simulation models.

The GridLAB-D simulation tool [5] is a more recent approach. It is developed by the U.S. Department of Energy (DOE) at the Pacific Northwest National Laboratory (PNNL) in cooperation with industrial partners. It allows the specification of a wide-range of simulation scenarios and the simulation of millions of independent devices. External links to 4GL-simulation models, e.g. MATLAB or MySQL are also supported.

Another interesting approach is the Mosaik framework [12,13]. Compared to the GridLAB-D framework it is more explicitly designed for the composition of heterogeneous simulation models. It enables the reuse and combination of existing simulation models and simulators to create large-scale Smart Grid scenarios. This is achieved by the interaction of four main components:

- A programming language independent API that allows the integration of existing simulators, simulation models or control strategies into the Mosaik framework. This enables e.g. the integration of MATLAB models as described in this paper.
- A scenario definition API for the description of large-scale simulation scenarios. Here, different simulation processes as well as models or the connections of the different entities can be described.
- A simulation manager, that is able to start simulators resp. to connect to an already running instance.
- An event-discrete simulation execution to coordinate the execution of all simulators. Therefore, each simulator or sub-simulation can have a different step size, which may even vary during the execution of the simulation.

The approach presented in this paper originates from the work undone in [4] where the load schedules and load shifting potentials of a high number of electrical consumers were determined using massive simulation. The simulation was realized by stand-alone models of heat pumps, night storage heaters and the supplied building, which were built and validated using Matlab/Simulink. These models were exported as dynamic link libraries (DLL) and integrated into a Java-based multi-agent platform using Java Native Access (JNA)[5]. The resulting multi-agent based simulation system was able to simulate up to 10.000 households, where each household was represented by an agent encapsulating an exported Matlab/Simulink model. This approach was consequently extended to the more versatile and scalable approach presented in this paper by using established technology like REST web services and load-balancers.

Contrasting to IPSYS [2], GridLAB-D [5] or Mosaik [12,13] the work presented in this paper focuses on the seamless integration of 4GL-models and algorithms for both Smart Grid simulations as well as applications. By using standardized REST web services as a wrapping mechanism ,the wrapped 4GL-models and algorithms can be integrated in both operative systems and simulations without any further efforts. Therefore, the approach presented in this paper does not concern the composition and coordination of simulations or the formal description of simulation scenarios. A further differentiation follows up in Figure 1. It shows how the approach presented in this paper complements the related approaches, which focus on the functional requirements of simulations like the provision of semantic models of the according application domain. The approach presented in this paper complements these approaches by focusing on the fulfilment of non-functional requirements like the application independent technical integration of 4GL models for simulations and also operative systems.

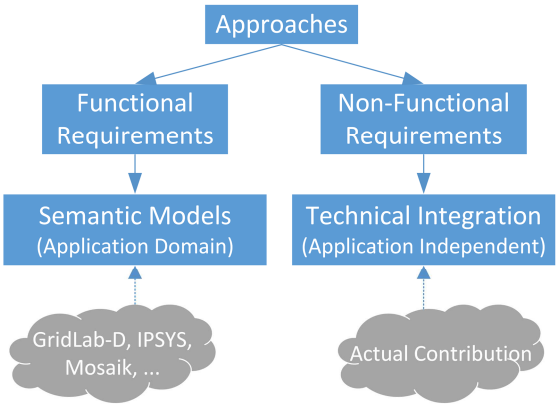

Figure 1: Comparison of related approaches and the presented one.

[5] https://github.com/twall/jna

3. Scalable Integration of 4GL-Models and Algorithms

This Section describes how the scalable integration of 4GL-models and algorithms for both Smart Grid simulations and applications can be achieved. First, the integration approach is described on a conceptual level before it is architectural embedded into a service-based simulation framework. An example on how the proposed approach can be facilitated into a combined hard- and software Smart Grid simulation model concludes the Section.

3.1. Integration Concept

The scalable integration of 4GL models and algorithms is based on standard web technology. The functions offered by the 4GL elements are wrapped by REST web services following the service-oriented architecture (SOA) paradigm [6]. Thus, each 4GL model or algorithm that shall be called from a productive or simulation system is accessible as a REST service method. Figure 2 depicts an integration example for 4GL code exported from MATLAB. With the *Builder JA*[6] MATLAB code can be exported into Java archives (JAR) allowing the code to be called directly from any Java application. However, in case of MATLAB exports, the machine executing these JARs still requires the MATLAB Compiler Runtime (MCR) to be installed. The MCR is a self-contained set of dynamic libraries called by MATLAB applications outside of a MATLAB installation. As the MCR is thread-safe, it is not possible to execute (even independently) exported code parallel within one enclosing process. Therefore, in order to allow a scalable execution of exported MATLAB code, multiple processes are required. Figure 2 shows how the proposed scalable integration approach could solve this issue. The functions offered by the JAR files are wrapped by REST web services [7] and deployed to a Tomcat[7] application container. Each Tomcat instance is a single operating system process allowing the parallel execution of (independent) MATLAB functions. In order to achieve a scalable solution, multiple Tomcat instances are required. The propagated approach makes use of a load balancer that distributes the REST web service calls to multiple Tomcat instances (on single or multiple machines). A first proof of concept uses the open source Apache HTTPD[8] web server combined with the *mod_jk*[9] module as a load balancer. This configuration utilizes round robin scheduling to distribute the service calls to the according Tomcat instances. The load-balanced distribution of the service calls is carried out transparently for the calling application. In case of bottlenecks new Tomcat instances can be added effortless without the need of changing any code on the calling side thus allowing a scalable integration of MATLAB code (or any other exported 4GL code). Additionally it is possible to use the integration concept with a different load balancer utilizing a more complex, application-dependent scheduling algorithm.

The here-described approach is used in a current industrial Smart Grid project dealing with the load-management of electrical heaters. Control algorithms written in MATLAB are exported into JARs, which are wrapped by REST service interfaces and deployed to a Tomcat application container. These services are used for the integration of 4GL-models into a large-scale multi-agent simulation for validation, as well as in the actual operative systems. An optional load-balancer ensures the scalability and fast response times for both the large-scale simulation and the operative application.

[6] http://www.mathworks.de/products/javabuilder/index.html

[7] http://tomcat.apache.org/

[8] https://httpd.apache.org/

[9] http://tomcat.apache.org/connectors-doc/

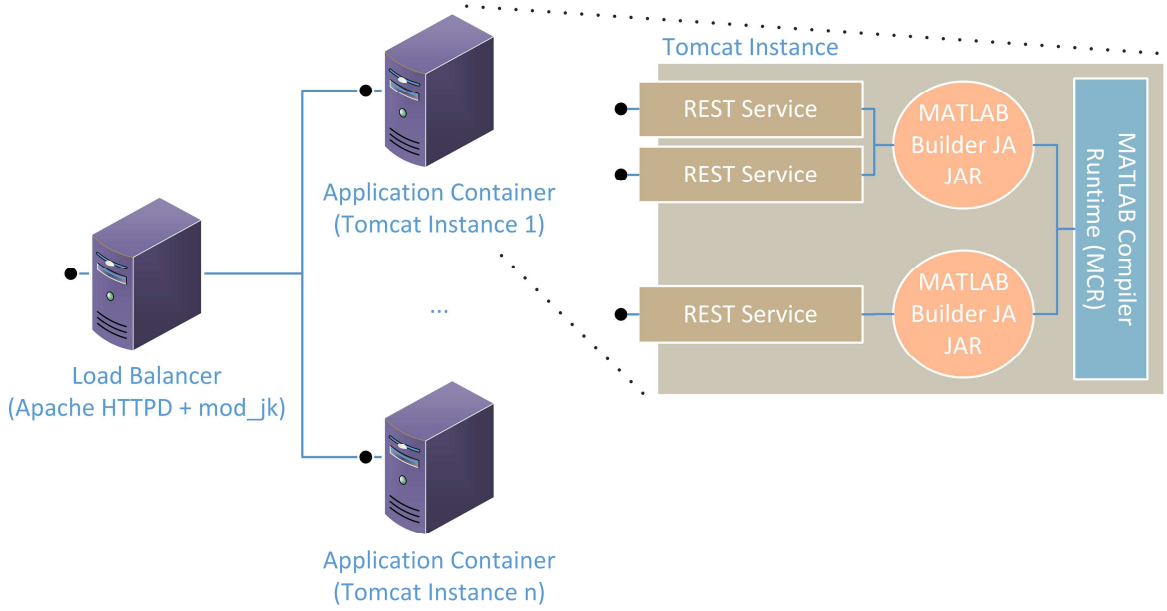

Figure 2: Scalable Integration of MATLAB Code

3.2. Simulation Integration

In [10] a component-based architecture for a service-based simulation framework is presented in order to integrate microscopic and macroscopic (sub-)simulations. All components communicate through a Service Bus and are able to offer simulation services via well-defined interfaces. The framework supports a domain-specific declarative description of simulation scenarios. These descriptions are processed by a Scenario Manager and contain information about all simulation components required for a specific scenario. The according requests from the Scenario Manager are routed to the Simulation Component Manager utilizing the Service Bus. This scalable component handles the lifecycle of all simulation components. According to requirements it starts additional components of the required type and sends corresponding requests via the Service Bus. Heterogeneous simulation components based on different technologies are connected to the system through an attached Adaptive Manager component. This manager masks the specific properties of a simulation component with generic interfaces (black-box) and extends it with adaptive behaviour. Therefore, they can be integrated into existing simulation systems. The Adaptive Manager allows to monitor the execution state of a simulation component and if necessary also to control it. In case of a simulation request the Adaptive Manager can access existing simulation data (Database) or execute a corresponding (sub-)simulation. Thus, the system is able to dynamically execute macroscopic simulations with regard to mesoscopic effects [11].

Figure 3 illustrates the architecture and shows how the scalable integration of 4GL code fits into it. Following the concepts proposed in [10] the Apache HTTPD used as a Load Balancer adopts the role of an Adaptive Manager. It handles the service requests, monitors the execution state of the Tomcat instances and calls the corresponding web services offered by the Tomcat container. In this context a Tomcat instance wrapping the 4GL code is a simulation component that offers specific simulation services. According to e.g. a Multi-Agent System consisting of multiple agents such a scalable 4GL model simulation component may consist of multiple duplicated Tomcat instances. Of course these adaptive simulation components containing a Load Balancer and multiple application containers may occur several times within a complex simulation systems.

In conclusion, the proposed concept enables the scalable integration of 4GL models and algorithms for massive (Smart Grid) simulations and allows a fast transition to operative systems. This is

achieved by combining the advantages of fourth generation languages widely spread in engineering disciplines with the advantages of advanced high-level programming languages.

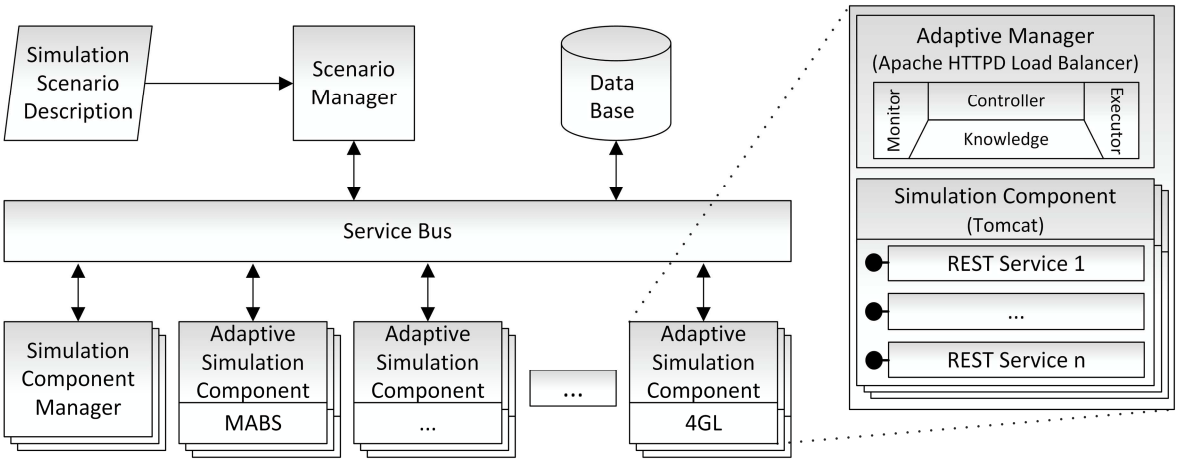

Figure 3: Scalable Integration of 4GL Models in a Component- and Service-based Simulation Framework (extending [10])

3.3. Example: Combined Hard- and Software Smart Grid Simulation Model

Because the standardized REST APIs can be used via different wide area networks (WAN), e.g. the Internet, the proposed solution for the integration of 4GL-models and algorithms can also be used for interconnected hard- and software simulation models in different organizational structures (resp. networks). Therefore, it is required that the physical devices also offer REST APIs, so that they are able to communicate with software models resp. either simulated or actual users. Figure 4 illustrates this concept. The depicted Smart Grid simulation model consists of three different domains. The User Domain maps the user interactions with the system. This might be a technical user who starts and stops the simulation or a functional user who utilizes the simulation system in order to validate the systems behaviour to the given inputs. The Simulation and Control Domain maps the simulated entities, e.g. household or devices models, which simulate their actual counterparts and the control and regulation structures of the system. In this domain the proposed integration approach can be used to integrate 4GL-models (simulation) or algorithms (control) in a scalable fashion. The last domain is the Physical Domain where the actual hardware and devices are situated. Similar to a hardware-in-the-loop (HIL) simulation the simulated entities can either be used to test the physical devices. Vice versa the physical devices can be used to test control structures and algorithms in combination with the simulated ones.

4. Conclusion and Future Work

In this paper we proposed a solution for the scalable integration of 4GL-models and algorithms into Smart Grid simulations and applications. Fourth generation languages (4GL) focus on rapid application development and the reduction of lines of code. They are widely spread across engineers also in the Smart Grid domain but they lack of integration and scalability features. By wrapping the models and algorithms with established REST web services and applying load-balancing techniques to the services the scalable integration is achieved. The proposed solution allows engineers to focus on the application development using their well-known tools and liberates them from integration efforts by providing a seamless integration concept. This concept allows it to integrate the models and algorithms into simulations as well as operative systems without requiring further efforts. We presented the integration concept and its interaction with an envisioned service-based simulation framework. A short example showed how the approach supports the combination of hard- and software Smart Grid simulation models.

The presented approach does not concern with the composition and coordination of simulations or the formal description of simulation scenarios. As described in Section 2 there exist many other approaches, like e.g. IPSYS, GridLAB-D or Mosaik for the interoperability of simulations and the integration of heterogeneous simulation models, which exceed the presented approach in these terms.

Future work will include the integration of the proposed approach into the envisioned service-based simulation framework as well as the realization of a massive multi-agent based simulation system utilizing the approach to integrate MATLAB-models and algorithms. The purpose of this simulation is to test algorithms for a smart load-management of electrical heaters.

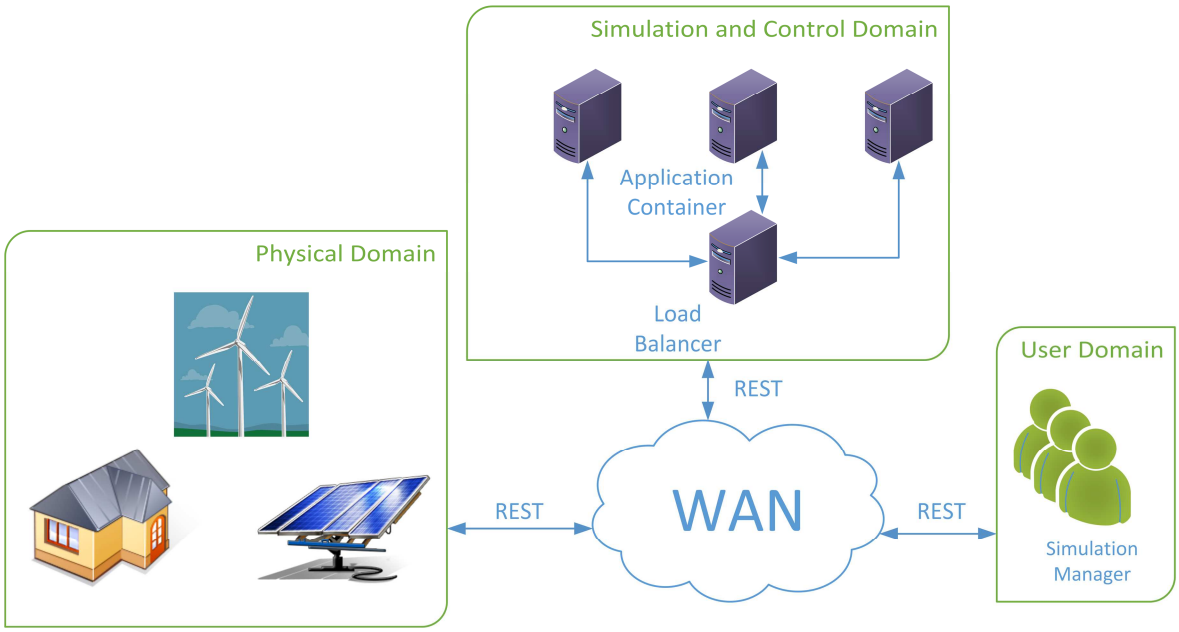

Figure 4: Combined Hard- and Software Smart Grid Simulation Model

References

[1] Beydeda, S., Bock, M., and Gruhn, V., "Model-Driven Software Development," 2006. Springer Berlin Heidelberg, ISBN: 978-3-540-25613-7.

[2] Bindner, H. W., Gehrke, O., Lundsanger P., Hansen, J. C., and Cronin, T., "IPSYS – A simulation tool for performance assessment and controller development of integrated power system distributed renewable energy generated and storage," *Risø National Laboratory*, 2004.

[3] Boer, C. A., de Bruin, A., and Verbraeck, A., "Distributed simulation in industry – a survey part 3 – the hla standard in industry," In *Proceedings of the 2008 Winter Simulation Conference*, 2008.

[4] Braunagel, J., Vuthi, P., Renz, W., Schäfers, H., Zarif, H., and Wiechmann, H., "Determination of load schedules and load shifting potentials of a high number of electrical consumers using mass simulation," In *2013 22nd International Conference on Electricity Distribution*, Stockholm.

[5] Chassin, D. P., Schneider, K., and Gerkensmeyer, C., "GridLAB-D: An open-source power system modelling and simulation environment," In *Transmission and Distribution Conference and Exposition*, 2008.

[6] Erl, T., "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services," Prentice Hall PTR, Upper Sadie River, NJ, USA, ISBN: 0131428985.

[7] Fielding, R., "Architectural styles and the design of network-based software architectures," 2000. Doctoral Dissertation, University of California, Irvine, ISBN: 0-599-87118-0.

[8] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLS) – Framework and Rules, IEEE Std 1516-2000, 2010.

[9] Martin, J., "Application Development without Programmers," 2000. Prentice Hall PTR, Upper Sadie River, NJ, USA, ISBN: 0130389439.

[10]Preisler, T., Balthasar, G., Dethlefs, T., and Renz, W., "Servicekomponenten-basierte Architektur für mikroskopische und makroskopische Simulation der städtischen Energieversorgung," In *VDE-Kongress 2014 Smart Cities* (to be published).

[11]Renz, W., Preisler, T., and Sudeikat J. "Mesoscopic stochastic models for validating self-organizing multi-agent systems," In *IEEE 6th International Conference on Self-Adaptive and Self-Organizing Workshops (SASOW)*, 2012

[12]Schütte, S., Scherfke, S., and Tröschel, T., "Mosaik: A Framework for Modular Simulation of Active Components in Smart Grids," In *1st International Workshop on Smart Grid Modeling and Simulation (SGMS)*, 2011.

[13]Schütte, S., Scherfke, S., and Sonnenschein, M., "mosaic – Smart Grid Simulation API," In *Proceedings of SMARTGREENS 2012 - International Conference on Smart Grids and Green IT Systems*, edited by B. Donnellan, J. P. Lopes, J. Martins, and J. Filipe: SciTePress, 2012.