# A DOMAIN-SPECIFIC LANGUAGE FOR SIMULATION COMPOSITION

Steffen Schütte
OFFIS – Institute for Information Technology
Escherweg 2
26121 Oldenburg, Germany
Email: steffen.schuette@offis.de

**KEYWORDS**

DSL SmartGrid Simulation Composition Scenario Xtext

**ABSTRACT**

The SmartGrid, the power grid of the future, is comprised of a large number of distributed, partially controllable energy consumers and producers that need to be matched to ensure that generation and consumption is balanced. Modelling & Simulation is a powerful approach for analysing and testing such complex systems. In this paper a scenario specification based on a domain-specific language (DSL) is presented which can be used to formally describe simulations and scenarios using these simulations. This formal description, in combination with a simulation framework that is able to interpret the description, allows the automatic composition of the simulations.The DSL has been tested with a lightweight simulation framework developed for a current project that analyses the impact of electric vehicles on the distribution grid. Although this work focuses on the domain of SmartGrids, the DSL-based approach can of course also be applied to other domains.

## INTRODUCTION

Modelling & Simulation is a powerful approach for analysing and testing complex systems such as the SmartGrid – the power grid of the future. In the SmartGrid a large amount of distributed energy resources (DER), such as highly efficient heat and power plants, wind turbines or photovoltaic modules as well as controllable consumers need to be coordinated to ensure that generation and consumption are balanced at any time. This is a challenging task due to the number and restrictions of the involved components. Control strategies for this complex and new task still need to be developed and in particular evaluated and tested.

In literature different approaches for simulating such SmartGrid scenarios comprised of markets, distributed energy resources and controllable consumers are presented by (Karnouskos and De Holanda, 2009), (Widergren et al., 2004) and others. Whichever approach one chooses, the scenarios that are to be simulated need to be defined in some formal way if the overall simulation shall be composed and parametrised automatically from available components/models. In this paper a formal scenario and simulation specification based on a domain-specific language (DSL) is presented. This approach has been developed in combination with a small, lightweight simulation framework that has been developed for the GridSurfer project (BMWi, 2010). In this project, OFFIS develops and evaluates control strategies for electric vehicles (EVs) in combination with other DER, such as photovoltaic modules. Although this work focuses on the domain of SmartGrids, the DSL-based approach can of course also be applied to other domains.

## RELATED WORK

Different approaches for simulations in the field of SmartGrids already exist, e.g. (Widergren et al., 2004), (Karnouskos and De Holanda, 2009). None of these, however, describe in detail how to specify the scenarios which are a major input for the simulation. Those who do, give a brief description of some kind of XML configuration. This paper wants to show a straight-forward, alternative approach for a formal description of simulation scenarios using a DSL that is easy to understand, even to non IT experts.

The presented approach has been developed to meet the simulation needs of the GridSurfer project and is not directly based on a solution presented in literature. The simulation of markets as described by (Widergren et al., 2004) is not part of the GridSurfer project and the agent-based approach described by (Karnouskos and De Holanda, 2009) was considered unnecessarily complex for the requirements of the project.

Besides the domain-specific approaches described above, there are other approaches that can be found in literature which are domain independent and could be used as well, such as (Moradi, 2008) or (Benali and Ben Saoud, 2010). Again, these were considered too complex as the focus of these works is placed on how to match the simulation requirements with a large number of components from a component repository. In the GridSurfer project just a hand full of simulation models needs to be used in different configurations and combinations so that referring to concrete simulation models from within the scenario specification is sufficient. Thus, the matching problem does not occur. Nevertheless, an easy to understand and consistent specification of the different scenarios should be possible.

The approach presented in this paper has been inspired

Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

by the works of (Prasanna et al., 2005) that in turn are based on the approach of Model-Integrated Computing (MIC) by (Sztipanovits and Karsai, 1997) as described later. As this approach is based on the creation of a domain-specific modelling environment, the project domain is introduced first.

**The Project Domain**

The simulation environment that has been developed for the GridSUrfer project had to provide enough flexibility to simulate the different scenarios required for the project evaluation. To meet these requirements the scope of the environment and the scenario modelling approach presented in this paper have been limited to the domain of SmartGrids and in particular to the domain elements that were required for the GridSurfer simulation. There are three major types of domain elements (in our case simulation models) that were identified:

- Control algorithms (monitoring the state of the grid and/or the resources and controlling the resources if possible)
- Distributed energy resources (DER)
- The power grid (connecting the DER)

Figure 1 illustrates these elements and their relations as they can occur in a real SmartGrid.

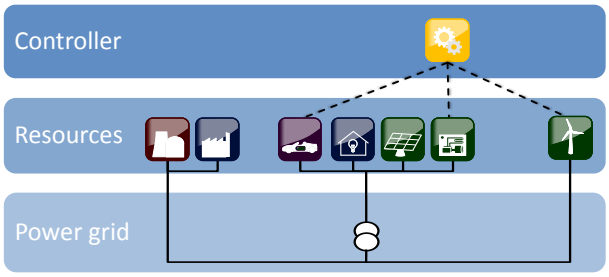


Figure 1: Types of SmartGrid elements and considered relations

The DER are represented by simulation models developed for the GridSurfer project or reused from earlier projects. The power grid is represented by a simulation model which has been implemented using the open-source power-flow analysis Pylon (Lincoln, 2009). The control algorithms and the integration to the simulation framework are currently being developed and not within the scope of this paper. While other approaches for simulation composition try to achieve arbitrary model couplings, a specific characteristic of the project domain presented here is that the DER models are only coupled via the power grid, as can be seen in figure 1. However, this still allows the specification of a large number of scenarios while keeping the complexity low. In the future the DSL shall be extended to support the specification of submodel relations (e.g. to use a battery model as submodel for an electric vehicle model or alone as stationary storage).

In the remainder of this paper, a simple scenario shall serve as an example to illustrate the approach. This scenario is shown in figure 2 and contains private homes that are equipped with photovoltaic (PV) modules. The homes and their PV modules are connected to different points (A..D) in a small grid topology and the power flow analysis shall be used to calculate the voltage and load conditions at the different nodes of the network as well as the overall load flow at the transformer.

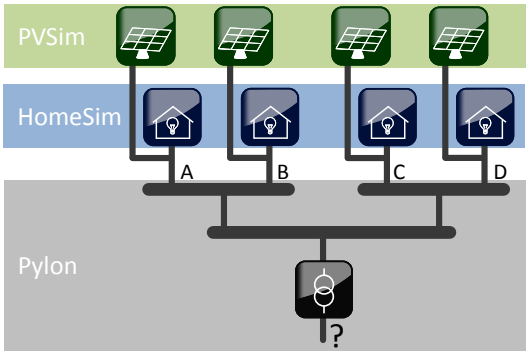


Figure 2: Example of a simulation scenario

## THE APPROACH

### Model-Integrated Computing

The presented approach follows the three step process of Model-Integrated Computing (MIC) (Sztipanovits and Karsai, 1997). First, in the metalevel process, software engineers formally specify and configure a domain-specific environment which is automatically generated. In a second step, this environment is used by domain engineers to specify domain models. Finally, model interpreters synthesize executable programs from the domain models or generate data structures for tools. Figure 3 shows these steps. In our case the interpreter generates input for the simulation framework. Compared to other model based approaches, such as Model-Driven-Design (MDD), MIC models do not only capture the architecture of the software but also it's environment. Information that is often more comprehensive and likely to change than the models of the software itself (Sztipanovits and Karsai, 1997).

This approach has been adopted because the creation of a domain-specific modelling layer also allows people without in-depth knowledge of the simulation framework to use it for modelling their scenarios of interest. This is in particular beneficial for students and other researchers who want to use it to evaluate their works, for example SmartGrid management strategies.

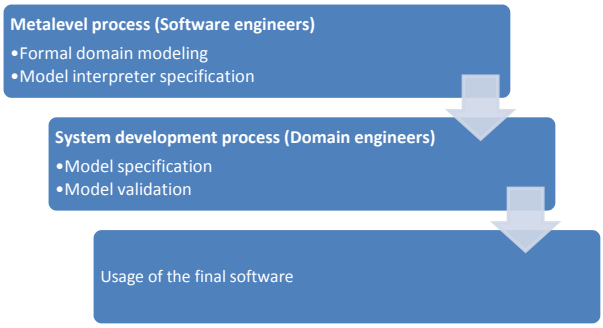


Figure 3: Model-Integrated Computing method according to (Sztipanovits and Karsai, 1997)

### Xtext

For our project Xtext (Eclipse Foundation, 2010b) was chosen as implementation for both, metalevel and system development process. Xtext has an active community and is quite easy to learn. It has various advantages over XML based solutions as (Efftinge and Zarnekow, 2010) demonstrate. Compared to proprietary MIC Environments such as GME, Xtext is much more lightweight and common as it is a component of the Eclipse Modelling Framework and integrates with the Eclipse platform. It allows easy and straight forward creation of own domain-specific languages based on a simple EBNF-style grammar. The definition of a DSL corresponds to the MIC metamodelling level.
For the system development process Xtext automatically generates a DSL specific editor as Eclipse plugin, which can be used to define models using the DSL. This editor includes "the parser, the type-safe abstract syntax tree (AST), the serializer and code formatter, the scoping framework and the linking, compiler checks and static analysis aka validation and last but not least a code generator or interpreter." (Behrend et al., 2010) Figure 4 depicts our approach based on the three steps of the MIC approach. The puzzle parts represent the different technological choices that were made. These parts could also be implemented using other technologies.

Each metamodel has one or more corresponding instances (models) on the system development level. In other words, the metamodels define a domain specific class structure which is instantiated by the simulationist on the system development level, using the domain specific language editor that Xtext has generated. For being able to automatically simulate different scenarios such as the example shown in figure 2, three different metamodels had to be implemented.
The simulation metamodel allows to create formal descriptions of the available simulations. These include the definition of the simulation's step size, its inputs and outputs and the available configuration parameters. To sum up, simulation implementations can be described on the system development level using a domain specific language defined on the metalevel. Next, a metamodel for

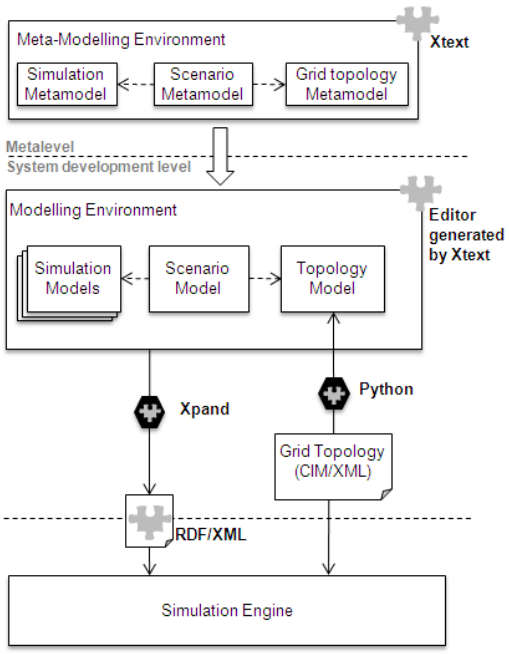


Figure 4: MIC-based approach for SmartGrid simulation

scenario description has been defined. It allows the simulationist to specify the number and properties of the simulation models to use within the simulation and in case of DER models, how these are connected to the power grid. Therefore the scenario metamodel contains references to the simulation metamodel. This way, it can be made sure that in the scenario description only those simulations and parameters are used that are available. Finally, for being able to specify to which nodes of the grid the simulated DERs shall be connected, the scenario metamodel needs to have references to the metamodel of the power grid topology. Currently, the resulting topology model is just a plain list of nodes contained in the topology and not further elaborated in this paper. It can be generated from a CIM/XML (IEC, 2008) compatible topology description using a small Python script. In the next chapter all metamodels are presented in detail and it will be shown how the definition of the DSL using Xtext looks like.
For generating the input to the simulation framework Xpand (Eclipse Foundation, 2010a) is used, a template-based language for generating arbitrary files from EMF-models, such as the Xtext models. In our case RDF/XML files are generated, containing all relevant information of the models and the scenario. This information is interpreted by the simulation framework and the simulation implementations are instantiated and executed accordingly.

## METALEVEL PROCESS

During the metalevel process, the domain-specific modelling environment that is to be generated needs to be modelled in a formal way. The different metamodels that have been briefly introduced above are now described in

more detail.

**Simulation Metamodel**

The simulation metamodel describes the simulations that are to be composed and is shown in figure 5 as UML class diagram. The metamodel used for the GridSurfer project contains some more details that have been omitted here as they are not required for understanding and presenting the basic idea of the approach.

Listing 1: Definition of the simulation metamodel using the Xtext-Grammar

```
enum Role:
   DER| grid;
enum DataType:
   int | string | float | boolean |
       datetime;
enum Meaning:
   PowerFlow;

Simulation:
   'Simulation' name=ID 'of 'role=Role
    (isContinuous?='is continous'|
    ('has step size from 'minStepSize=INT
        'to 'maxStepSize=INT 'minutes'))
      ('has Parameters'
      (simParameters+=SimParameter)*)?
      'containing' (models+=Model)*;

SimParameter:
   name=ID'as 'type=DataType;

Model:
   'Model' name=ID 'with '
      multiplicity=('one'|'1..*') '
          instances'
   'having'
      ('Parameters'
      (modelParameters+=Parameter)*)?
      ('Inputs'
         (inputs+=ModelData)*)?
      ('Outputs'
         (outputs+=ModelData)*)?;
ModelParameter:
   name=ID'as 'type=DataType;
ModelData:
   name=ID 'as 'type=DataType(':'meaning=
       Meaning)?;
```

The metamodel supports specification of continuous simulations (such as the power flow calculation with Pylon) and event-discrete simulations as needed for centralised DER management approaches. In the latter case the minimal and maximal step size can be specified. A simulation can have a certain role according to the classification in figure 1. This is in particular useful for automatically determining the execution order of the simulations as described below. Each simulation contains one or more different models. Each model can have different input and output data, e.g. the power drawn from the grid or the battery state of an EV. Also, simulations and models can have configuration parameters. The datatypes are to be specified for each of these elements. The in- and output data can have an optional meaning, e.g. to let the framework know which output of a DER model represents the generated power. For each model the possible multiplicity can be specified, e.g. how many models (which represent real entities, such as an EV) can be simulated. In listing 1 the corresponding Xtext specification for this metamodel is shown. The classes from the UML diagram became Xtext entity definitions (e.g. 'Simulation:'). Aggregations are created by using the entity definitions and assigning them to a variable name. Enumerations can explicitly be defined using the 'enum' keyword. For mor information about the syntax see (Eclipse Foundation, 2010b). Listing 2 shows how the PV simulation of the example scenario is described using the domain-specific language that Xtext has generated.

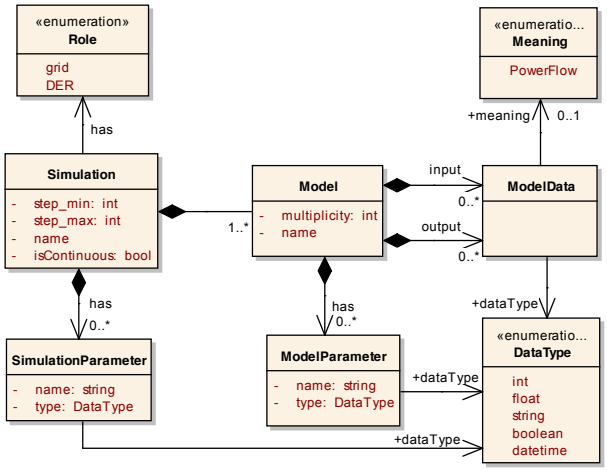


Figure 5: Metamodel for simulation description

Listing 2: Excerpt of the PV-Simulation description in the generated editor

```
Simulation PVSim of DER
  has step size from 1 to 15 minutes
  has Parameters
   simulationStart as datetime
   simulationEnd   as datetime

  containing Model Photovoltaic with 1..*
      instances
   having Parameters
     PV_maxActivePower  as float
     PV_Angle           as float
     PV_IMPP            as float
     PV_UMPP            as float
   Outputs
     generation as float:PowerFlow
```

Based on the meaning of the output data and the role of the simulation model, the simulation framework can create the dataflow. In our scenario (see figure 2), for example, the DER simulations need to be stepped first to

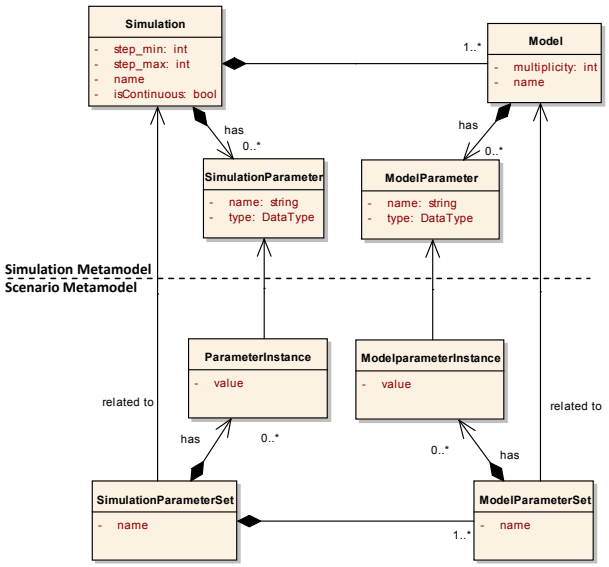


Figure 6: Parameter assignment

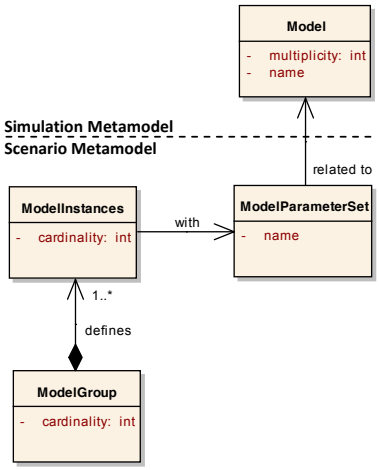


Figure 7: Model instantiation and grouping

determine the power values at the four nodes. In a second step, the data is sent to the power flow simulation that can be stepped afterwards.

Listing 3: Simulation and model parameter specification

```
Scenario:
'Scenario' name=ID
 (simulationParameterSets+=
     SimulationParameterSet)+
 (modelInstances+=ModelInstances)*
 (modelGroups+=ModelGroup)*;

SimulationParameterSet:
 'SimulationParameterSet' name=ID 'for
     simulation' sim=[Simulation] 'with'
 (parameterInstances+=ParameterInst)*
 (modelParameterSets+=ModelParameterSet)+;

ParameterInst:
 param=[SimParameter] '= ' value=STRING;

ModelParameterSet:
 'ModelParameterSet' name=ID 'for model'
     model=[Model]
 'with parameters'
 (parameterValues+=ModelParameterInst)*;

ModelParameterInst:
 param=[ModelParameter] '= ' value=STRING;
```

**Scenario Specification Metamodel**

Next, the scenario metamodel, which needs to reference the simulation metamodel, can be defined. It shall be a formal description of the scenario that is to be simulated to allow an automatic composition of the required simulations. In our example (figure 2) the scenario needs to contain information about the type of PV modules to simulate (e.g. how the PV simulation is parametrised), the number of houses and PV modules to simulate and their connection to the grid.

A separation of the parameter specification from the rest of the scenario definition is beneficial, as there can be quite a number of parameters and in this way, a redundant parameter specification can be avoided. Figure 6 shows this modelling approach as UML class diagram. A *SimulationParameterSet* references a simulation and contains parameter instances, e.g. concrete values for the parameters of the simulation, and one or more *ModelParameterSets* which contain the parameter instances for a specific model that the simulation referenced by the *SimulationParameterSet* contains. The Listing 3 shows how this is implemented using Xtext. References between the scenario and simulation part of the metamodel are implemented using square brackets.

In our scenario, the PV modules shall only be connected to the same connection point as the simulated home, as it is assumed that the PV modules are mounted on top of the houses roof. Therefore model groups are introduced to allow specification of models that physically belong together. Figure 7 show the according extension of the scenario metamodel. A *ModelGroup* allows the specification of 1 or more model instances that are to be grouped and a cardinality for the group. The class *ModelInstances* specifies how many instances of a specific model (represented by the desired model parameter set) are to be created. In our example we have 4 groups with one home and PV simulation, each.

What is still missing is a possibility for specifying the connection points of the simulated DERs, e.g. a node in the electrical power grid to which the DER is electrically connected (ResourceConnectionPoint, RCP). This connection point is an Xtext reference to a node in the grid topology metamodel already shown in figure 4. Most of the DER have fixed connection points (e.g. wind turbines, PV modules, etc...). EVs, however, require a way to specify a variable connection point as they move be-

tween different charging stations. This can be done by binding the connection point to a certain property value of the simulation model (e.g. when the output 'location' of the vehicle model has the value 'home', the EV is connected to grid node 'A' and else to 'B'). For a better understandability these extensions are not presented here and only the parts of the metamodel required for modelling the example scenario are shown. Listing 4 shows the Xtext definition for the model group and connection point extensions. In the example scenario only 4 connection points are to be specified. In larger scenarios, however, the specification of each connection point becomes impractical. Therefore, a different connection strategy called *RandomRCP* has been introduced. This strategy connects all specified instances to randomly but reproducibly chosen nodes of the power grid. This will be used for the example scenario specification shown in the next section. Currently, only nodes that have not been connected are returned by the algorithm (no two house-/PV groups are connected to the same node). However, additional parameters for specifying the connection behaviour could be required. The Xtext metamodel can be extended easily to incorporate such parameters and the model editor is automatically updated by the Xtext framework to support the new elements.

Listing 4: Grouping and grid connection metamodel

```
ModelGroup:
   'Create' cardinality=INT 'instances of
       ModelGroup' name=ID
   'connected to grid 'connectionStrategy=
       ConnectionStrategy
   (modelInstances+=ModelInstances)+;

ModelInstances:
   'Create' cardinality=INT 'instances of
       ' modelParameterSet=[
       ModelParameterSet|QualifiedName]
   ('connected to grid 'connectionStrategy
       =ConnectionStrategy)?;

ConnectionStrategy:
   FixedRCP|RandomRCP;
FixedRCP:
   //Reference to grid-topology model
   'at rcp:'rcp=[RCP];
RandomRCP:
   'at random RCP';
```

### System Development Process

Based on the metamodel specified above, Xtext can generate an editor plugin for Eclipse which is capable of syntax highlighting and auto completion. Although advanced XML editors also offer auto-completion, Xtext allows to specify additional consistency checks and custom auto completion enhancements, thus offering support for the domain-modeller in specifying the scenario so that it is valid for the simulation framework. Also, references are type-safe, while using XML the references just need to be IDs in a valid syntax, e.g. a check has to be implemented manually as XML pre-processing step. The scenario definition for our example is shown in listing 5.

Listing 5: Model of the example scenario

```
Scenario Example1
  SimulationParameterSet default for
      simulation GridSim with
      ModelParameterSet default for model
          Pylon with parameters
      csv_file = "<file_path>"

  SimulationParameterSet pv_params for
      simulation PVSim with
   ModelParameterSet private_pv for model
       Photovoltaic with parameters
     PV_maxActivePower = "900"
     PV_Angle = "30"
     PV_IMPP  = "7.55"
     PV_UMPP  = "23.8"

  SimulationParameterSet homesim_params
      for simulation HomeSim with
   ModelParameterSet private_home for
       model Home with parameters
     load_profile = "vdew_h0"

  Create 4 instances of ModelGroup Home_PV
   connected to grid at random RCP
   Create 1 instances of homesim_params.
       private_home
   Create 1 instances of pv_params.
       private_pv
```

It defines one parameter set for each of the different simulations (HomeSim, PVSim, GridSim/Pylon) and defines a group *Home_PV* that represents homes with PV installation. 4 instances of this group are to be instantiated during the simulation.

The scenario description and the description of the referenced simulations is interpreted by our simulation engine during the composition phase and the required simulations are instantiated and parametrised. The definition of how to interpret the scenario description based on the metamodel is obviously as important as the metamodel itself. However, the interpreter component is not within the scope of this paper.

## CONCLUSIONS & FUTURE WORK

The approach presented in this paper is currently being used in the GridSurfer project. First scenarios could already be successfully defined and simulated. It could be shown that the scenario definition based on the DSL is easy to understand, even to non IT experts, and is clearly arranged. Another advantage is the loose coupling between the scenario specification and the simulation framework. When a change in the interface or the functionality of the simulation framework is made, the

possibly large number of scenario specifications does not need to be touched but rather only the Xpand generator needs to be adapted. With all these advantages, the scenario specification acts as a real specification artefact.
In the future more domain-specific extensions are planned, such as the integration of more roles (besides 'DER' and 'grid') will be implemented to enable a wider range of scenarios. Also the DSL shall be extended to support the specification of submodel relations (e.g. to use a battery model as submodel for an electric vehicle model or alone as stationary storage). Currently the simulations are only coupled via the grid simulation. However, this is sufficient for the current project and still allows the specification of a broad range of scenarios while keeping the complexity at a minimum. Finally an electricity standard conform integration (CIM/IEC61850) of control strategies is planned to allow reuse of the evaluated strategies for a broad range of simulation components.

## ACKNOWLEDGEMENTS

The GridSurfer project is funded by the German Federal Ministry of Economics and Technology (01 ME 09 017).



## REFERENCES

Behrend, H., Clay, M., Efftinge, S., Eyshold, M., Friese, P., Köhnlein, J., Wahnenden, K., and Zarnekow, S. (2010). Xtext user guide. http://www.eclipse.org/Xtext/documentation/1_0_0/xtext.pdf.

Benali, H. and Ben Saoud, N. B. (2010). Towards a component-based framework for interoperability and composability in modeling and simulation. *Simulation*, 87(1-2):133–148.

BMWi (2010). Gridsurfer - inter-urbane integration von elektrofahrzeugen in energiesysteme inklusive batteriewechselkonzept. 2. February 2011, http://www.ikt-em.de/de/GridSurfer.php.

Eclipse Foundation (2010a). Xpand website. 2. February 2011, http://wiki.eclipse.org/Xpand/.

Eclipse Foundation (2010b). Xtext website. 2. February 2011, http://www.eclipse.org/Xtext/.

Efftinge, S. and Zarnekow, S. (2010). Goodbye xml - befreiungsakt mit xtext. 10. January 2011, http://it-republik.de/jaxenter/artikel/Goodbye-XML-Befreiungsakt-mit-Xtext-3459.html.

IEC (2008). Iec 61968-13. 7. February 2011, http://webstore.iec.ch/preview/info_iec61968-13\%7Bed1.0\%7Den.pdf.

Karnouskos, S. and De Holanda, T. N. (2009). Simulation of a smart grid city with software agents. *2009 Third UKSim European Symposium on Computer Modeling and Simulation*, pages 424–429.

Lincoln, R. (2009). Pylon website. 8. February 2011, http://pylon.eee.strath.ac.uk/.

Moradi, F. (2008). *A Framework for Component Based Modelling and Simulation using BOMs and Semantic Web Technology*. Ph.d. thesis, KTH Stockholm.

Prasanna, V., Orangi, A., Da Sie, W., and Kwatra, A. (2005). Modeling methodology for application development in petroleum industry. *IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005.*, pages 445–451.

Sztipanovits, J. and Karsai, G. (1997). Model-integrated computing. *Computer*, 30(4):110–111.

Widergren, S., Roop, J., Guttromson, R., and Huang, Z. (2004). Simulating the dynamic coupling of market and physical system operations. *Power Engineering Society General Meeting, 2004.*, (6-10 June 2004):748–753.

## AUTHOR BIOGRAPHIES

**STEFFEN SCHÜTTE** studied information technology in Hannover, Lüneburg and Wolverhampton. He received his Master degree in 2007 at the Leuphana University of Lüneburg. After working as a software developer in the automotive industry for two and a half years he decided to join the eMobility project GridSurfer at OFFIS in Oldenburg, Germany as scientific research assistant. His email is steffen.schuette@offis.de.