Implementation of new consumer model in RAPSim to allow home management system integration

Michael Penz
Institute of Computer Technology (ICT)
TU Wien
Vienna, Austria
e01226716@student.tuwien.ac.at

Marcus Meisel
Institute of Computer Technology (ICT)
TU Wien
Vienna, Austria
marcus.meisel@tuwien.ac.at

Thilo Sauter
Center for Integrated Sensor Systems / ICT
Danube University Krems / TU Wien
Wiener Neustadt / Vienna, Austria
thilo.sauter@donau-uni.ac.at

***Abstract*—On one hand, the worlds rising electricity demand can lead to increased renewable, clean, environmentally friendly generation. On the other hand, this increase of volatility poses many new challenges for today's power supply system to integrate this hard to predict generation. One direction many efforts are undertaken, is the introduction and increase of flexibility on the consumer side, leading to more smartness and communication need between power consumption and production. In the customer premises, home energy management systems are being developed to increase own consumption for users, but also offer an access point for other stakeholders, to receive market or grid signals, to accomplish timing and coordination. To be able to analyze, e.g. the energy flows and to gain a better understanding of scaling effects; there is a need for simulation frameworks like RAPSim. This paper describes the implemented extension of RAPSim with an additional consumer component, which allows integrating selected functions of home energy management into the simulation. The simulation allows the new consumer component to closely approximate the behavior of home energy management systems that have a photovoltaic system and a battery storage installed. Further possibilities to extend the components and features in the future are also pointed out.**

***Index Terms*—RAPSim, customer energy management system, smart grid simulation tool**

I. Introduction

The global trend of the worlds electricity consumption is increasing [1]. The International Energy Agency (IEA) reports that the worldwide electricity consumption rose by about 1,200 Mtoe in the years from 1970 to 2015 and it is still going up [2]. Especially during the past decade, this awareness leads to a subsidy triggered increase of decentralized renewable energy generation plants through wind turbines, private photovoltaic systems, and suchlike. Today, the decentralized generation and storage capacity by far is not where it needs to be to meet 2050 goals, but severe grid management challenges become apparent, leading to a potpourri of innovative solutions such as presented in [3].

A. Home Energy Management Systems as part of the solution

Some opportunities are a reduction of the power grids load which decreases the transmission losses and a reduction of the energy price for the consumer. Many of the approaches are trying to balance volatile generation by increasing the flexibility on the consumer side. One way this increase can be achieved is by the introduction of smart energy management systems, able to switch or manage loads within the customer premises to increase overall efficiency and to coordinate energy demand with times when production is available. The aim is to add communication, digital processing, automation, and computer-based remote control, to enable the consumer to participate in balancing the energy consumption to flatten appearing power peaks [4]. Home energy management systems, serve a significant scope of functions such as monitoring energy flows, setting schedules for energy use, controlling smart appliances in real time, or giving information to the consumer. There are a lot of smart HEMS solutions on the market. The functional range can be extended almost arbitrarily. Various smart appliances are available which can be connected to complex systems, managed by a control unit. To be able to analyze the effects of home energy management systems on smart grids, the goal is to integrate them into smart grid simulation frameworks.

B. Extending selected simulation tool with flexible loads

Simulation software makes it possible to analyze smart grids dependent on the smart grid structure, participants, weather conditions, and many more influencing factors. RAPSim is one of those open source smart grid simulation frameworks. It is possible to set up and configure a smart grid, and calculate various energy flow and distribution scenarios depending on selected algorithms.

This paper aims to integrate features of energy management systems into RAPSim to enable more dynamic simulations and to allow to simulate their impact on the energy flows of smart grids. The goal is to implement a new consumer model that includes selected features, which are suitable to be integrated into RAPSim. After a simulation is carried out, the changes are made available and discussed.

II. Micro-grid simulation tool RAPSim

RAPSim is an open source framework for micro-grid simulation developed by Manfred Pöchacker, Kristofer Schweiger and Sabrina Huber at the Institute of Networked and Embedded Systems of AAU Klagenfurt. The underlying programming language is Java. It aims to help better understand energy flows in smart grids with a decentralized integration of renewable energy sources [5]. RAPSim's purpose is to make it

possible to analyze smart grid constellations by various algorithms. It determines consumed energy, energy demand, power grid load, and much more. It allows simulating either on and off-grid scenarios. Weather dependent processes are based on an underlying climate model that includes information about wind speed, cloud factor, as well as some other conditions.

When setting up the basic structure of the smart grid to simulate, the user has the opportunity to choose between many different participants in the power grid. All components can be manually placed, linked, and configured in the graphical editor. After configuration of the smart grids structure and properties, an appropriate algorithm for calculation can be chosen and the simulation can be started. The algorithm is executed in individual steps in a given period, and results can be read directly, or they can be saved to a CSV file.

For the further tasks of this paper, RAPSim was chosen, as it is the most suitable simulation framework for the implementation. This is because RAPSim is an open source software and its design provides excellent extensibility. Due to the structure of RAPSim, it is possible to implement new components and to integrate them into simulations. This is the main reason why this framework was used. Other reasons are the good usability and clarity of RAPSim which allows quick creation of an executable simulation scenario. A detailed comparison of RAPSim with other simulation frameworks can be found in [6].

III. Implementation of new component and results

In this chapter, the program structure of RAPSim is stated. Subsequently, features of home energy management systems suitable for integration are determined. An additional component including this features is implemented and the new components effects are discussed based on a performed simulation.

A. Selection of properties to be implemented

Concerning a smart grid simulation, home energy management systems change the behavior of a house as a component. In the following, functions are listed that most home energy management systems have in common and those suitable for integration into RAPSim are chosen. The functions have been determined by an analysis of a selection of different systems in advance to this paper. Energy management systems are controlling room conditions, such as lightning, humidity, and temperature, as well as smart switches, photovoltaic systems, and battery storage. Light control, heating control, and smart switching are not suitable to be simulated with RAPSim, or it would only be possible with high programming effort, and the effects on the simulation would be too small to notice. RAPSim does not provide information about human behavior which would be necessary to determine energy flows dependent on human habits. Therefore those processes will be added by using an average load profile. Private photovoltaic systems and in-house battery storages are the most significant impacts on the simulation. They can completely change the energy demand required from the public power grid. All information needed to implement those features like weather conditions, or daytime is provided by RAPSim, and therefore those are well suited to integrate into the framework. The actual implementation is described in chapter III-B2.

B. Component development

The following describes the software structure of RAPSim, after which a new house component is implemented that has a home energy management system installed. Subsequently, a simulation including the new component is carried out and the results are analyzed.

1) Framework structure: The implementation of RAPSim is split into two separate projects, which are called JGridMap and RAPSim. The JGridMap project handles the user interface and is responsible for the graphical representation. It allows the user to set up the structure of the smart grid. Whereas the RAPSim project is the simulation environments actual core, it involves the implementation of the logic of all components and the algorithms for calculating the simulation. All information and features of grid components are included in the packages `sgs.model.gridObjects` and `sgs.model.objectModels`. A `gridObject` represents one element in the simulation environment. Each of them contains an `objectModel` which is responsible for internal calculations, which means every `gridObject` has its `objectModel` encapsulated. The inheritance tree of a `gridObject` is shown in Fig. 1. Abstract classes are marked with "A." "M" indicates which classes can instantiate a valid component object. New models can be added by inheriting from the given abstract classes. All implemented models need to inherit from the abstract `SmartGridObject` at least.

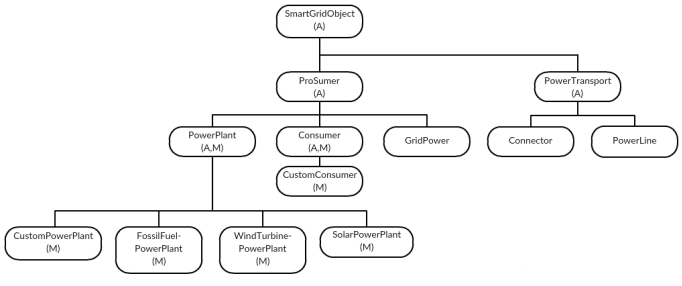

Fig. 1. Inheritance tree of a `gridObject`.

The abstract classes provide the base functions of any component like managing the property window, representing the component in GUI, maintaining the object icons, as well as managing the objects parameter set. More detailed information can be found in [7].

The RAPSim framework already contains the implementation of various models such as the `FossilFuelModel`, `SolarPeakPowerModel`, `SolarSquareMeterModel`, `RandomWindTurbineModel`, `UsualWindTurbineModel`, and the `ConstantPowerModel`. Each `gridObject` contains one of those models. The complete inheritance tree of the currently implemented models is

shown in Fig. 2. Its structure is adapted to the `gridObject` tree in Fig. 1.

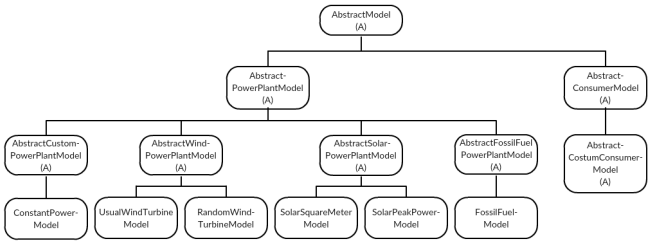

Fig. 2. Inheritance tree of currently implemented `objectModels`.

2) Implementation of additional Consumer Object: The aim is to implement a new component which simulates a household using an energy management system. RAPSim already contains two consumer objects called `ConstantDemandModel` and `ResidentialAverage-LoadCurveModel`. Those existing components provide a demand more or less independent of the rest of the simulation. This work aims to create a consumer component with a more dynamical approach by adding a photovoltaic system and a battery storage to a house itself.

To leave the user the choice to decide whether a home management system is installed, an additional consumer object is implemented as follows. To add a third consumer component, which is called `EnergyManagementModel`, it needs to extend the `AbstractCustomConsumerModel`. As a result, the new component is automatically integrated into the simulation environment and the user interface. The user now has the option to choose out of three consumer objects. To alter the behavior of the new component, the override methods `updateVariables` and `initVariableSet` are implemented in the model's class. Within the method `initVariableSet`, all properties of a component, which can be seen or changed by the user in the property window are set. The method `updateVariables` is the core of the model. All function calls and calculations are initiated there. `updateVariables` is executed each time step of the simulation and therefore, its runtime needs to be kept as low as possible, so as not to slow down the program. Its parameters provide data about the current time, and the weather conditions.

At first, all vital information to calculate the demanded power needs to be clarified. Therefore, a variable for each characteristic is declared and set in the `initVariableSet`. The needed parameters for the photovoltaic system are its squaremeters, longitude, and latitude. The necessary parameters for the battery storage are battery capacity and whether a battery is installed or not. Those attributes can all be set by the user in the property window. Also, the number of people living in the household is also queried in order to be able to calculate their energy requirements later.

The households base demand is determined by using an average load profile indicating the demand depending on the daytime. In particular, an H0 load-curve, created by TU München, is used. The values are based on statistics of the energy agency NRW. H0 profiles are normalized to 1,000 kWh per year [8]. The used load curve contains values that indicate power consumption in intervals of 15 minutes. The load profile is provided in CSV format, and was added to the projects data folder to enable the use in source code. To load the data of the CSV file into the component, the method `loadLoadProfile` was implemented. All values are written into a global string array variable one by one. As the file contains more than 35,000 values it needs to be loaded in the object's constructor to ensure, it is only loaded once.

Because the load profile was formatted so that it contains only the following values without any markings, there needs to be a logic that assigns the current simulation time to the corresponding value in the CSV file. Therefore, the method `getTimeID` was implemented. Its source code is shown in Listing 1.

```
private int getTimeID(GregorianCalendar
    currentTime) {
     int timeID = 0;
     timeID = (currentTime.get(Calendar.
         DAY_OF_YEAR) - 1) * 24 * 60 +
         currentTime.get(Calendar.
         HOUR_OF_DAY) * 60 + currentTime.
         get(Calendar.MINUTE);
     return (int)((((double)timeID)/15)
         +0.5);}
```

Listing 1. Code of `getTimeID` method

To receive the location of the value of interest in the CSV file, the simulation's current date and time must be converted to past minutes since the beginning of the year. Then it is calculated how many fifteen minute steps are needed to get there because the values are given in this scaling in the load profile. Since the resolution of time steps can be set to any value, which might not be fifteen minutes or a multiple of it, the result needs to be rounded. This can be done via a temporary conversion of the quotient to the int data type. Because Java rounds to the smaller number by default, 0.5 needs to be added before conversion to ensure a correct result. Summarizing the described `getTimeID` method returns the load profiles cell number corresponding to simulation time.

The calculation of real demand involving the households person count is carried out in a new method `getAverageLoadFromProfile`. Therefore the yearly energy consumption is determined based on the households person count. The values used, are based on a statistic of EnergieAgentur.NRW. More information can be found here [9]. Therefore, the yearly average energy consumption is set to 1,600 kWh for one person, 2,400 kWh for two persons, 3,200 kWh for three persons, 4,000 kWh for four persons and an additional 500 kWh for each person more in the household. This is determined by a switch condition to keep the running time low. In a try-catch construct the actual value from the

H0 string array is read and parsed into a double for further calculations. In case of an exception, the variables remain at their standard values. The program code for parsing is shown in Listing 2.

```
try{load = Double.parseDouble(loadProfile
    .get(timeID));
    persons = Integer.parseInt(
        personCount.getValueDouble()
         .toString().split("\\.")[0]);}
```

Listing 2. Parsing the households person count to integer

After converting the values to the correct data type, the calculations can be carried out. The households actual consumption is calculated through equation (1).

$$ActualConsumption = \frac{H0value \cdot AnnualConsumption}{1,000\,kWh} \tag{1}$$

The resulting value represents the baseline consumption and serves as a basis for further adoption. At this point, the newly implemented component is functional and can be integrated into the simulation.

3) Implementation of additional Photovoltaic System and Battery Storage: The photovoltaic system and the battery storage were integrated to add the chosen features of home energy management systems and introduce some dynamic into the component. a new method `calculatePhotovoltaicPP` was implemented to calculate the energy currently being produced by the photovoltaic system. It has access to the actual date, the weather, as well as to Gregorian Calendar data through its parameters. It is necessary to include the sun position and cloudiness into the calculation to ensure an exact result. RAPSim provides methods for determining the sun position and therefore the solar intensity in the package `solarCalculations`. Importing this allows executing the calculations directly within the component's source code. It should be noted that the height at which the photovoltaic system is installed, is set to zero meters above the sea level in the whole scenario. The actual energy production of the photovoltaic system is determined by multiplying the photovoltaic systems area, with the solar intensity, the free sight factor, and its efficiency. Concerning the information used, the photovoltaic systems area is set by the user, the solar intensity is calculated through RAPSim's `solarCalculations` package, and the free sight is identified by using the cloudiness factor which is included in the weather parameter and the efficiency is set to a constant 0.15. Listing 3 shows the exact implementation. Also, to enhance the accuracy of the calculation, the user can enter the alignment of the photovoltaic cells based on its longitude and latitude. These attributes are queried via the property window and they have a direct influence on the solar intensity. To calculate the power output of the photovoltaic system the method `calculatePhotovoltaicPP` is implemented.

```
AzimuthZenithAngle sun = PSA.
    calculateSolarPosition(currentTime,
    latitude.getValueDouble(), longitude
    .getValueDouble());
freeSight = 1-weather.getCloudFactor();
solarIntensity = SolarCalculations.
    getSolarIntensity(sun.getZenithAngle
    (), 0);
currentProduction = squareMeters.
    getValueDouble() * solarIntensity *
    freeSight * 0.15;
return currentProduction;
```

Listing 3. Calculation of energy produced by the photovoltaic system

The power production resulting from method `calculatePhotovoltaicPP` is influencing the households demand from the power grid. Therefore, to cover a specific proportion of the previously determined consumption, the produced power is used. In the next step, a battery storage was implemented, so that if there is surplus energy from the photovoltaic system, it can be stored into a battery and used for a future demand later.

Method `checkBattery` was implemented to accomplish this. It saves surplus energy to the battery and provides available energy in case of demand. The method's flow chart is represented in Fig. 3.

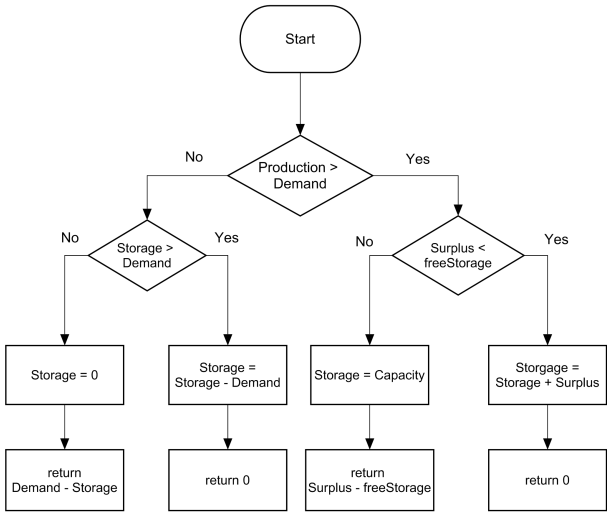

Fig. 3. Flow chart of `checkBattery` method.

The information needed for battery storage handling are the values of currently produced power, the amount of energy in the battery storage and the time resolution of the simulation steps. It is critical to note that the current power values given in W need to be converted to an amount of energy over time in kWh, to enable the interaction with the battery. To determine the amount of energy being produced or consumed during two simulation time steps, (2) is used.

$$Energy\ (kWh) = \frac{Power\ (W) \cdot Resolution\ (min)}{60 \cdot 1000} \tag{2}$$

Since the data is only available in intervals of 15 minutes, it is assumed as an approximation, that the performance is constant over this period. This allows cross-unit calculations. Depending on the current production and the amount of energy in the battery, the method returns the remaining demand to the calling function.

So far, all functions required for the process have been implemented. The administration of the calls is done in the `updateVariables` method. It is automatically called by the RAPSim source code in intervals of the respective resolution set in the simulation settings. The order of calls is shown in Fig. 4.

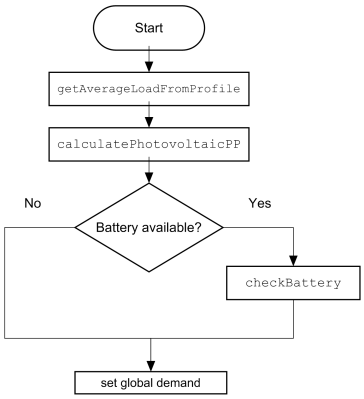

Fig. 4. Call order of `updateVariables` method.

At this point, the component is fully functional and ready to use in simulation. It reacts to the current weather situation and manages optional battery storages.

C. Simulation results with new energy management model

To validate the behavior of the new consumer component, a simulation scenario was created in RAPSim, which contains the two existing consumer components `ConstantDemandModel` and `ResidentialAverageLoadCurveModel`, as well as the new `EnergyManagementModel` component. The simulation setup is shown in Fig. 5.

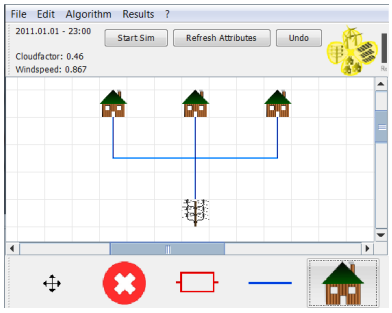

Fig. 5. Simulation setup of three building models.

All three models are connected to each other, as well as to the public power grid, to supply the demanded power. This setup already allows running a simulation. Before starting it, the properties of the components need to be adjusted to get reasonable results. Therefore the demand for `ConstantDemandModel` is set to 500 W and the annual consumption of `ResidentialAverageLoadCurveModel` is set to 4,000 kWh. This value scales the values of the H0 load profile. To obtain similar orders of magnitude for the comparison, the number of persons of the `EnergyManagementModel` is set to four, which also corresponds to an annual consumption of 4,000 kWh. Furthermore, it is set that a battery storage with a capacity of 10 kWh and a photovoltaic system with an area of fifteen square meters are installed in the house (universal values in Austria at the time of writing).

For the simulation, the simple-distribution algorithm is chosen. It determines the consumption of all components and carries out an energy flow analysis. The simulation is carried out for one day from 01:00 to 22:00 in time steps of 60 minutes. The results are displayed in Fig. 6. Besides the values calculated by the algorithm, the battery contents and the current production of the `EnergyManagementModel` are also indicated to be able to understand better what happens in the background. Those can be seen in Fig. 7.

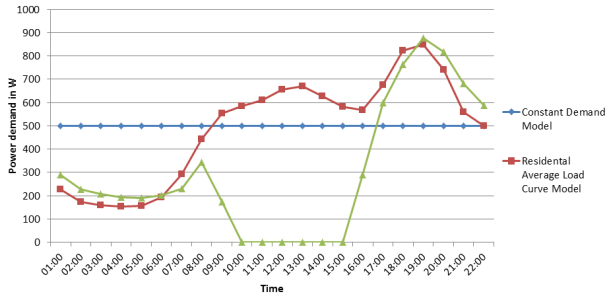

Fig. 6. Simulation results, power demand

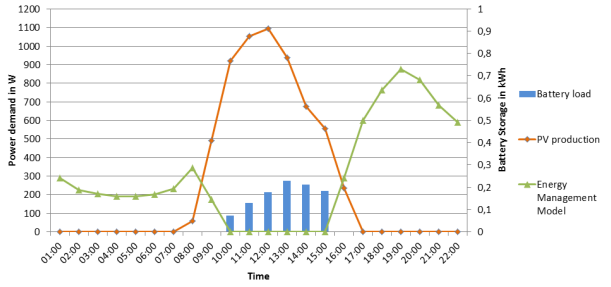

Fig. 7. Simulation results, battery storage and PV production

Fig. 6 and Fig. 7 illustrate the function of the new component. It is recognizable that the consumption of the `ConstantDemandModel` does not change over time and is constant at the pre-set value of 500 W. The results of the `ResidentialAverag-LoadCurveModel` are directly proportional to the given load profile. This is independent of any internal simulation influences, except the season and whether its a workday or a day off, such as weekends and holidays. In comparison, the newly implemented `EnergyManagementModel` is a dynamic solution approach that

takes into account the different conditions of the simulation. It can be seen that at the beginning, the PV system does not produce any power because no sun is shining and therefore the solar intensity is zero. When the battery storage is empty, the consumption values are determined by the given load profile. As the simulation continues and the solar intensity increases, the line-chart shows that the photovoltaic system begins to produce power at 8:00. This value reduces the consumption delivered by the load profile. The higher the solar intensity gets, the higher the power production and the lower the demand from the power grid. At 10:00, the production exceeds the consumption and the battery is charged with the surplus energy. The demand value is set to zero from this point on. When the consumption exceeds the production in the evening, the energy is taken from the battery until it is empty. The remaining demand is covered by the power grid. As the test simulations show the component works.

IV. Conclusion

After a summary, this chapter discusses, which improvement effects the new component has on the simulation. Finally, there is an outlook on what was not done for this work, and what can be changed or added to RAPSim in the future.

A. Summary

This paper describes the role of energy management systems and their simulation. At first, the simulation framework RAPSim is introduced. Subsequently, it was determined which features of home energy management systems are best suitable for integration into a simulation framework. Concluding, a new consumer component with an integrated photovoltaic system and a battery storage was implemented. Afterward, the new component has been tested in simulations and the results have been analyzed.

B. Discussion

RAPSim had only two consumer models by default: the `ConstantDemandModel` and the `Residential-AverageLoadCurveModel`. The fist provides a constant demand which does not change over time whereas the latter provides a demand according to a given load profile. Different day profiles depend only on the day of the week and the season. As a result, the values often repeat. Both models are very static and there are only a few options. To be able to simulate better growing numbers of households equipped with a home energy management system as well as producing energy themselves or having a battery storage installed, a further component was added in this work, which makes it possible to simulate an in-house photovoltaic system and a battery storage. This extends the possibilities of RAPSim and allows to simulate more individual adjustments in order to be able to better approximate reality. Some suggestions for further extension are given in the section IV-C.

C. Future aspects and improvements

Even if the component is fully functional, there are some things that can be improved or added. The component implemented in this paper offers many possibilities for future expansion. For example, the use of the load profile could be replaced by a simulation of human behaviors. Furthermore, other ways of private energy production and consumption could be added. There are many features of home energy management systems that could be included in the component, which, while not having a huge impact on the simulation, will make it more versatile.

Another improvement is about the use of surplus energy. At the moment, the surplus energy produced, which is left after the battery is filled, is neglected. The method `checkBattery` could, therefore, be enhanced so that the energy is supplied back to the power grid. However, this would require a substantial change in the RAPSim consumer model, since the component in the simulation could then no longer be implemented as a pure consumer, but as a generator as well.

To improve the usability, there is the suggestion for RAPSim itself to add the ability to enter boolean values in the property window. This would be very helpful for activation or deactivation of features by the user, or to determine the results of yes or no queries. Currently, the internal datatype `SingleVariable` only provides the input of real or complex numerical values. Therefore, in the `checkBattery` method, to query whether a battery is installed, a double value must be read in and then converted in order to use it in a switch condition.

In advance, it is planned to enhance the software with an interface to the real world to feed external information into the program. This will create many new possibilities to improve the quality and accuracy of the simulations.

References

[1] V. Crastan: “Weltweiter Energiebedarf und 2-Grad-Klimaziel”; Springer Verlag Vieweg, 2016. -ISBN 978-3-662-53421-2

[2] International Energy Agency: “Key world energy statistics”; 2017.

[3] M. Meisel, T. Leber, F. Kupzog, K. Pollhammer, M. Ornetzeder, J. Haslinger, J. Sterbik-Lamina, P. Wächter, A. Schiffleitner, M. Stachura: “Lastmanagement für intelligente Stromnetze in Österreich (Smart Response)”; techn. Report for FFG ; Berichts-Nr. 825527, pp. 41, 2012.

[4] S. Kollmann, S. Wilker, M. Meisel, A. Wendt, L. Fotiadis, T. Sauter: “Local intelligence for a customer energy management system equipped with smart breakers”; In: 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), 2017, pp. 1-4

[5] M. Pchacker, T. Khatib, W. Elmenreich: “The microgrid simulation tool RAPSim: Description and case study”; IEEE Innovative Smart Grid Technologies-Asia (ISGT ASIA), 2014. -ISSN 2378-8534, p. 278-283

[6] Aron Kondoro, Imed Ben Dhaou, Diana Rwegasira, Amleset Kelati, Naiman Shililiandumi, Nerey Mvungi, Hannu Tenhunen, “Simulation Tools for a Smart Micro- Grid: Comparison and Outlook”, 21st FRUCT (Finnish-Russian University Cooperation in Telecommunications) Conference, 6-10 November 2017, Helsinki, Finland.

[7] M. Pchacker, W. Elmenreich: “Model implementation for the extendable open source power system simulator RAPSim”; In: 2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES), 2015, pp. 103-108

[8] W. Schellong: “Lastprole und Lastmanagement”; Springer Berlin Heidelberg, 2016, pp. 375-417. -ISBN 978-3-662-49463-9

[9] L. M. Holm: “Stromverbrauch fr den 1, 2, 3, und 4 personen-haushalt pro jahr”; 2016