Grid Mind: Prolog-Based Simulation Environment for Future Energy Grids

JAN ROSECKY[1,2], FILIP PROCHAZKA[2] and BARBORA BUHNOVA[1]

[1] *Faculty of Informatics, Masaryk University, Brno, Czech Republic*
Email: {j.rosecky,buhnova}@mail.muni.cz
[2] *Mycroft Mind, a. s., Brno, Czech Republic*
Email: {jan.rosecky,filip.prochazka}@mycroftmind.com

submitted 29 April 2015; accepted 5 June 2015

Abstract

Fundamental changes in the current energy grids, towards the so called smart grids, initiated a range of projects involving extensive deployment of metering and control devices into the grid infrastructure. Since in many countries, the choice of supportive information and communication technologies (ICT) for the grid devices still remains an open question, benchmarking tools aimed at predicting their behavior in the deployed solution play an essential role in the decision-making process.

This paper presents a Prolog-based simulation environment, named Grid Mind, primarily intended for the very purpose. The tool was successfully used to generate simulation scenarios in several smart-grid related projects and became a self-standing simulation tool for the evaluation of information and communication technologies used to deliver low-voltage metering and monitoring data. The tool is continuously evolving, aimed to become an integral part of the future energy grid design in the Czech Republic and beyond.

KEYWORDS: Simulation Environment; Smart Grid; Communication and Networking; ICT; Prolog

1 Introduction

With the growing emphasis on energy security policies, the electricity distribution grids are currently facing a profound transformation challenge. The new approach, referred to as the smart grid, implies intelligent distribution throughout the grid, aiming at automated problem detection and handling as well as power balancing in order to minimize distribution-related losses and fully use the potential of unregulated renewable power sources (Amin and Wollenberg 2005).

Having up-to-date, accurate and complete data is crucial for achieving the aforementioned goals, hence serious attention must be paid to measuring and detecting mechanisms as well as distribution of relevant data to other grid agents. The latter implies the need for a solid communication and information-technology infrastructure developed along with the energy infrastructure. Its requirements, side-by-side with the economical and legal constraints, form an investment dilemma of vital

importance for the future development of the energy grids (Patel et al. 2011; Trefke et al. 2013).

Apart from the pilot projects, simulation has been playing a major role in the design of smart-grid solutions. A number of smart grid simulation tools and environments has been introduced (Mets et al. 2014), focusing on simulating the communication and power infrastructure of the grid. However, power distribution companies still require more detailed analysis of the behavior of regular data reading, signaling and control processes planned in particular deployment scenario, which shall better support the design of their smart grid solutions. Together with the particularities of the Czech distribution grids, this need has formed an opportunity for close academia and industry cooperation in 2010, when the development of the Grid Mind simulation environment began. Soon after that, the deployment of 3.5 million intelligent end-point meters (smart meters) was simulated in the CERIT Scientific Cloud with the help of the devised concepts (Křenek et al. 2013).

In this paper, we introduce a SWI Prolog-based smart-grid simulation tool, named Grid Mind, used as a rule-based simulation scenario generator in (Křenek et al. 2013) and continuously developed since. The tool has become a self-standing smart grid simulator, whose second generation was successfully used in several industrial projects with practical implications.

The paper is structured as follows: An overview of related work, mainly focusing on other smart-grid simulation tools and environments, is presented in Section 2. Section 3 introduces the overall architecture of the domain-independent core of the tool and its main components. A brief presentation of the modeling approach used in a real industrial project is provided in Section 4. Finally, Section 5 sums up the lessons learned, reflecting on the major issues and drawbacks, most of which are being solved in the on-going work, also mentioned in the section.

2 Related Work

Recently, an extensive review on existing smart grid modeling and simulation tools (Mets et al. 2014) concluded that the growing importance of simulating smart grid dynamics is mostly driven by two needs: to evaluate either (1) demand-response strategies in local communities, so-called smart grid cells, or (2) overall wide-area monitoring, protection and control architecture.

As for the former, various agent-based energy and communication infrastructure co-simulation tools have been introduced to analyze the influence of different types of control strategies, e.g. dynamic pricing or centralized control algorithms, on the grid cell sustainability. Mosaik (Schütte et al. 2012) is a powerful modular Python-based simulation environment, enabling an integration of existing simulators in order to simulate the power dynamics of areas comprising thousands of nodes. (Kosek et al. 2014) present the comparison of two power and communication grid co-simulation frameworks (MasSim resp. JadeSim and IPSYS-DE), orchestrated in Mosaik. (Mets et al. 2011) present a comprehensive simulation environment using the OMNET++ simulator as a development platform. The Coupled Simulator (Bergmann et al. 2010) is built on Network Simulator NS-2 and the commercial

Siemens' Power System Simulation (PSSTM) as a benchmark to newly developed communication and control techniques.

The wide-area monitoring, protection and control simulators test the influence of real-time sensor data availability on the overall grid stability guaranteed by the supervisory control and data acquisition (SCADA) systems. EPOCHS (Hopkinson et al. 2006) combines the PSCAD/EMTDC power simulator and NS2 network simulator in a co-simulation, glued together with the Runtime Infrastructure (RTI). Another approach, based on the High-Level Architecture (HLA), is presented in (Georg et al. 2012), where DIgSILENT PowerFactory and OPNET network simulator are coupled together as HLA federates. (Lévesque et al. 2014) have only recently introduced a preliminary implementation of a HLA-based simulator with the ambition to cover all the three smart grid perspectives, as presented in (IEEE 2011): the power systems, the grid communications and the related information technology. The time-stepped co-simulator implementation integrates OMNET++ with Java processes generating IEC 61850 communication protocol messages.

Our work touches both the measuring-data harvesting (e.g. for demand-response strategies) and grid monitoring and control, complementing the state of the art with a more in-depth analysis of the dynamics of metering-data delivery delay and incompleteness, caused by simultaneous data transfers and communication channel overloading, as well as the influence on the data volume and transaction load on the metering data management system.

3 Grid Mind Platform

Grid Mind (Rosecky 2015) is a modeling and discrete-event simulation tool built in SWI Prolog, primarily aimed at analyzing the behavior of a particular smart-grid information and communication infrastructure, hardly predictable due to the emergent nature of the modeled environment. The platform supports automated simulation execution and evaluation, including the export of the results into the GMIG visualization framework (Pompe 2015). Primarily, the tool was used to estimate the abilities of a given infrastructure to collect and provide particular types of data, analyzing the data freshness, completeness and reliability, rapidly changing with the number of concurrently executed data transfer jobs. The environment is controlled via the Prolog console, the models are formed using a versatile Prolog-based domain-specific language. The choice of a logical programming language facilitates the knowledge representation and orientation in the muddle of models, information from the simulation runs and their results. Moreover, the choice was made with regards to several constrained optimization problems expected to be solved in the future, e.g. synthesis of an optimal set of model parameters according to a specific price function or solving complex constraint-based queries, indicating the future integration of constraint logic programming (CLP).

The overall architecture of the domain-independent environment core (of Grid Mind v2), presented in this section, is depicted in Figure 1. Element storage and transformation tools form the structural core of the environment, described in Subsections 3.1 and 3.2 – a knowledge base to build the models in. The Modeler parses

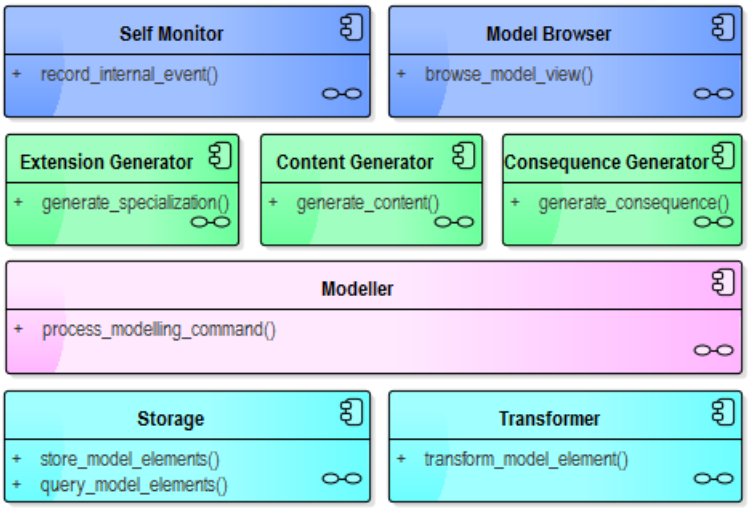

Figure 1. Component diagram of the Grid Mind v2 architecture.

```
object( node_substation, [ secondary_substation ], [
    how_many_consumption_points( integer:30 )
] ).
```

Figure 2. Example element – node secondary substation template – in Grid Mind v2 notion.

the model language into a canonical representation using a definite clause grammar (DCG). The three solver components, handling the model dynamics, are presented in Section 3.3. Finally, Self Monitor provides a simple logging interface, and Model Browser is responsible for *json* export to the GMIG visualization framework using the HTTP JSON library.

3.1 Knowledge Representation

Model elements and relations between them form a (hyper)graph, stored in the Storage component, extended and modified during the simulation using a set of functions provided by the Transformer component. The elements and relations are represented as Prolog terms and predicates, respectively, stored in a Prolog database. An element in Grid Mind v2 notion is a compound term of arity 3, where the first argument is the name of the element, the second argument is a list of its abstractions and the third argument is so-called element construction, a list of roles the element plays in domain relationships. An exemplary element is in Figure 2. The relations used for orientation in the spectrum of elements, whose semantics is described further in this section, are: (i) abstraction, (ii) role, (iii) context member, and (iv) causality.

Abstraction

Elements represent concepts on various levels of abstraction and detail. Thus, in

order to enable the construction of more general models or meta levels, elements are not classified into classes, instead, they are placed into an abstraction hierarchy, inspired by the notion of concept lattices (Wille 1992). *"To be an abstraction of"*, or *"to be a specialization of"* for the inverse direction, forms a many-to-many relation, used to grow the element ontology, with the following semantics:

Definition 1. *An element S is a specialization of an element A iff S possesses all attributes of A except for those explicitly redefined.*

In the second generation of the tool, the notion of attributes referred to roles. The third generation extends it to all relations (with the exception of abstraction).

The environment is responsible for attribute inheritance, fostering the code re-usability: When an element attribute is searched, than if not found at the queried element, it is sough on the higher levels of the abstraction hierarchy.

An example of an abstraction hierarchy chain is:

`power_grid_element` ← `secondary_substation` ← `node_substation` ← `substation_1234`

According to the hierarchy and the definition of node_substation in Figure 2, `substation_1234` will also power 30 consumption points without having to state it explicitly.

Role

Domain relations, be it between elements or an element and a general term, are modeled using so-called roles. *"To play a role for"* is a ternary relation, used to model domain relations, with the following semantics:

Definition 2. *Value V plays role R for element E iff E has a property named R with value V.*

In Grid Mind v3, a role is also considered to be an element. That allows the definition of role ontology as well as structuring the roles into contexts, facilitating the queries (e.g. get all the location characteristics of a given element) by bringing additional mechanisms of role polymorphism. Definition of `node_substation` in Figure 2 assigns value 30 to `node_substation` role `how_many_consumption_points`

If V is also an element, an inverse role can be defined:

Definition 3. *IR is an inverse role to R iff for every pair of elements E1, E2: E1 plays role R for element E2 iff E2 plays role IR for element E1.*

The support of the use of the inverse roles in queries is now being implemented in the third generation of the tool.

Context

Contexts form logical clusters of elements, often serving as sub-models of particular problems. E.g.

- E.ON distribution grid;
- Procedures related to a data reading process;
- Simulation run.

Definition 4. *Element E is a member of context C iff it fits the context definition, i.e. is a logical part of the context.*

In the third generation of the tool, the relation will become ternary, taking into account the role a particular element plays in a given context.

Causality

The simulation nature of the environment implies the need to record the causalities between events. As further mentioned, event reactions cause additional events in the future, making the causal relationship ternary:

Definition 5. *Action A is a causality between events* $E1$ *and* $E2$ *iff A caused* $E2$ *in reaction to* $E1$.

A series of predicates used to store, update and query the aforementioned relations, as well as the transitive and reflexive closure of the binary ones, forms the conceptual core of the simulation environment. Blurring the barrier between types and instances naturally supports the construction of domain-specific modeling tools without getting lost in "meta-muddle", as described in (Favre 2005).

3.2 Lambda Elements

Potential relation dependencies and stochastic relations motivated the need to work with implicitly constructed relations, with actors derived as a result of an operation. Procedural knowledge-base attachments called lambda-elements or lambdas are used for this very purpose. The name has been chosen to emphasize the conceptual connection to transparent intentional logic – TIL (Duží et al. 2010), built upon the lambda calculus.

Definition 6. *Lambda element is a compound term of arity 4:*
`lambda_elem( Id, Variables, Guards, Predicates )`,

- *evaluable for given* `Variables` *iff* `Guards` *hold, and*
- *true for given* `Variables` *iff evaluable and* `Predicates` *hold.*

3.3 Solvers

The dynamics of the environment is handled by a set of so-called solvers—engines used to generate elements or generally modify the element graph using model lambda elements. Lambdas are often parts of rules passed to the solvers.

Extension Generator

Depending on the nature of relations that an element participates in, it can be classified either as an *extension* or an *intension*. The TIL-inspired classification indicates to which extent the element is modeled explicitly or, complementarily, the amount of lambda elements used in the element construction. An (ultimate) extension is an element whose relations contain no lambda. The more the element relations are constructed using lambdas, the more intensionally-modeled the element is.

Intensions form a natural way of defining patterns for generating, especially when particular role values for a given set of elements are picked randomly or depend on one another. The generating process is realized by so-called extension making—constructing explicit form of element by finding values of its intensionally-constructed relations, evaluating their lambda elements. When evaluated lambdas contain non-deterministic predicates, multiple extensions with all admissible relation combinations can be generated.

Pseudo-code in Algorithm 1 in Appendix A represents a naive extension making process used in Grid Mind v2, where an evaluable lambda is sought within all element's unevaluated lambdas. When the lambda is evaluated, the process repeats, since the evaluation may have influenced evaluability of other lambdas by instantiating a free variable shared by the lambdas. The cycle loops until a fixed point is reached, meaning no further lambdas can be evaluated. Grid Mind v3 will implement more sophisticated extension-making strategy, based on variable dependency graph. In that case, when seeking an evaluable lambda, only so far untested lambdas or those, whose guards variables may have been instantiated by one of the evaluated lambdas, are to be tested on evaluability.

Context Generator

Also referred-to as Production Rule Solver, the Context Generator represents an approach to element or general term generation, mainly used for the data exports. Initial rules, containing lambdas returning initial collection content, form the bootstrap of the resulting collection. Iterative rules contain lambdas determining the set of records to be added to the resulting collection based on its current content. The solver iteratively processes the collection until a fixed point is reached, meaning no further new records are added. The pseudo code of the process is presented in Algorithm 2 in Appendix A. The solver proves particularly useful for obtaining constrained breath-first search results initiating from a given set of elements, e.g. for finding transitive consequences of a given event.

Consequence Generator

Finally, the Consequence Generator handles the actual simulation process. Simulation timeline is a queue of events to be processed, sorted by their timestamps. Intensionally-defined registered actions (subscriptions) may react to the events, producing a set of output events to be put back into the timeline. After adding the set of initial events (e.g. "start a measuring process") to the timeline, the Consequence Generator can be executed to run a simulation, ending either when the queue is empty, or with an explicit termination (while handling an event). Causalities are recorded and stored during the simulation. Algorithm 3 in Appendix A represents the pseudo-code of the simulation run process.

4 Models

Our approach to simulation analysis builds upon the TIL notion of possible worlds (Duží et al. 2010), following the pattern:

1. **Model world(s)**. The model data can be either (i) set manually in the model files; (ii) imported, e.g. from `csv` files or (iii) modeled intensionally, using fuzzy properties shared by classes of agents.
2. **Extensionize world(s)**, use the Extension Solver to generate a set of possible worlds containing individual agents, respecting the intensional model definitions.
3. **Simulate world(s)**, use the Consequence Generator to run the simulation of each world.
4. **Evaluate world(s)**, export and analyze the simulation results.

This implies that apart from the simulation itself, the responsibilities of the environment also include the execution of individual simulation runs, orchestrating the entire analytical process. In its full extent, the environment and related visualization framework should be able to provide extensive assessment of a given set of possible technical solutions under different conditions. Practical reasons why so far this has not been made possible are discussed in Section 5.

4.1 Case Study: Communication Technologies Assessment in Low-Voltage Monitoring

This section presents the crucial models and preview of result visualization from a recent simulation project, assessing the communication technologies for transfer of low-voltage metering and monitoring data in a prepared pilot project of a Czech energy distribution company. Since the planed deployment of the monitoring devices within the project scope only covered tens of secondary substations in different locations, the simulation was used to assess the influence of sharing a communication medium, particularly a base transceiver station (BTS), by multiple substations in a fully-deployed setting, meaning every secondary substation equipped with a monitoring device. Assuming the independence of an individual cluster of substations around a BTS, the problem was reduced to simulating the behavior of 8 clusters, each containing approximately 10 substations obtaining smart meter data from individual consumption points. The simulated scenarios covered one week of life with various cellular network communication technologies implemented, under changing conditions including dynamic external load in the used communication network infrastructure or dropouts of individual communication devices at end-point meters. The dynamics of the following characteristics was observed: (i) data delivery latency; (ii) reliability (percentage of successfully transferred data) of measurement data reading, signaling and command transmitting; (iii) communication network data load; (iv) end-system data and transaction load.

The communication infrastructure was modeled as a graph of network devices connected via communication lines. A *channel* connecting two devices refers to a graph path (lines and devices) between them. The model of a network line has been abstracted to the following characteristics with values added based on pilot project observations:

- **Throughput,** meaning the amount of data transmitted through a line or de-

vice at a time unit. Channel throughput is equal to the minimum throughput of a channel component.

- **Latency,** constant delay added to the throughput-caused delay for every data transfer. Channel latency is equal to the sum of the component latencies.
- **Reliability,** i.e. the probability of losing or damaging a transferred message. Reliability of a channel is equal to the product of its component reliabilities.

Expected delivery time τ of a data batch of size S, transferred through a channel with latency L_{ch}, comprising component C with throughput T_c, external load Ex_c and amount of concurrent transfers A_c, was computed as:

$$\tau = \frac{S}{\min\limits_{c \in ch} \frac{T_c - Ex_c}{A_c}} + L_{ch}$$

When starting a new transfer or finishing an old one, the volume of transferred data and new delivery-time estimates were computed for every transfer sharing a channel component with the affected transfer. Transported data batches corresponded to those of IEC 60870-5-104 protocol (IEC 2006), used in the pilot project.

The used network communication models are a subject to further discussion and verification, planned to be conducted in cooperation with other Czech universities in the upcoming months. Our experience suggests that a similar abstraction can be applied to the communication models without a significant loss of precision of the behavior characteristics of the overall simulation setting. Interface to an existing network simulator would inevitably cause a drop of performance, closing the door to an efficient state-space analysis of a particular technical solution.

Case Study Results

The conducted project have shown that the introduced environment can be successfully employed for realistic simulations demanded by industry. The SWI Prolog database used as a knowledge base core is able to handle hundreds of thousands of element records, each having from tens to thousands of connections. When loaded, the overall general smart grid model comprised over 1,000 elements. Over 6,000 elements were stored in the knowledge base when the project models have been extensionized. Apart from other elements, simulation runs ended up generating 150,000 up to 400,000 events during 5 to 20 minute run times, depending on the complexity of the simulated scenario.

Figure 3 represents an exemplary simulation output. The setting simulated daily data reading procedures via the public GPRS infrastructure. The upper timeserie presents normalized portions of hourly simulated dropouts of communication devices at end-point meters and external network load, potentially caused by extensive data transfers in the area when using a public telecommunication operator's network. The lower timeserie represents the resulting hourly line load caused by (compressed) metering data transfers. As can be seen, lowering the capacity of the communication media raises the overall volume of transferred data, due to repetitive transfers. On the other hand, dropouts of individual communication devices at

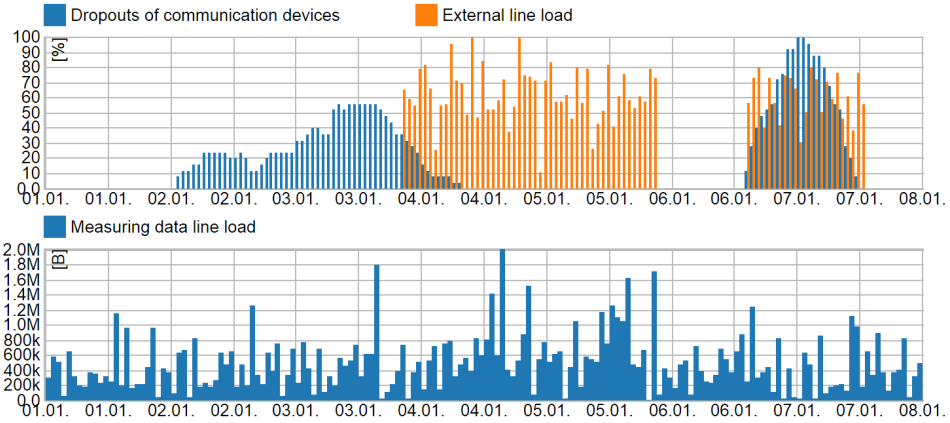

Figure 3. A week of life of the simulated setting, using the GPRS technology for the metering-data transfers.

end-point meters prevent them from communicating at all, shifting the load to the following days.

The crucial contribution of the project covered not only the results of the conducted analysis but also the structured representation of the modeled problem, revealing the potential of the used knowledge base architecture. Both helped establishing a solid ground for the planned deployment of the metering and monitoring devices onto the secondary substations in the Czech Republic.

5 Discussion

The second generation of the tool proved its usefulness in a number of projects, which at the same time helped us to identify the areas of possible future development. This chapter sums up the lessons learned from the usage of Grid Mind v2.

Due to complex model structure, dependencies and a set of specific predicates designed to use the environment, there is room for improvement from the usability perspective. In particular, the tool usability could be improved by encapsulating the *model – extensionize – simulate – evaluate* approach (see Section 4) entirely into the notion of the models. In order to do that, models should state their dependencies explicitly in a model header, to be processed by the model reader component.

Stochastic nature of the models requires multiple simulation runs to be executed in order to get statistically-relevant results. Extensive simulation analysis in a single-threaded Grid Mind v2 could be sped-up by the parallelization of individual simulation runs. When implemented, not only will Grid Mind be able to determine the probability characteristics of a particular solution, also extensive state-space analysis, determining the system behavior under various conditions, could be conducted on a computation grid in a matter of minutes.

The uprising interest of the Czech energy distribution companies, outlining several upcoming projects, has recently initiated the development of the third generation of the tool. The aforementioned points triggered a range of design changes, improving the tool performance and usability. The design of the conceptual core is

still a subject to further research. Particularly a comparison with existing Prolog-based approaches like (Chaudhri et al. 2013) or (Chisham et al. 2011) could be interesting.

6 Conclusion

The paper introduces the Grid Mind modeling and simulation environment, currently used for the analysis of smart grid information and communication technologies. The environment core is domain independent, believed to be usable for any simulation-analytical task. Domain models are structurally represented in a knowledge base, forming a graph of elements and relationships between them. Various abstraction levels, connected by the abstraction-specialization relations, represent a way of creating reusable model patterns, adjustable for specific purposes. Trinity of solvers handle the model dynamics, including the simulation engine and the generator of model elements. The models cover individual technical processes and routines, including reading and error-handling strategies.

The environment has been successfully used in several real industrial projects. The presented case study has validated its usefulness and revealed its capabilities, as well as its limitations. The architecture of the knowledge storage combined with the power of the three solvers has shown its potential, not only for the simulation analysis itself but also for forming a structured knowledge base of the local power grids and their future-development scenarios. The uprising smart-grid simulation projects demand in the Czech Republic illustrates how the fruitful cooperation between academia and industry can effectively solve complex problems in real world.

Appendix A Solver Algorithms

Algorithm 1 Extension making process

```
procedure GET_EXTENSION_OF(+Element, -Extension)
    repeat
        Evaluable_lambda ← null
        for all Lambda in Element.lambdas do
            if can_evaluate(Lambda.guards) then
                Evaluable_lambda ← lambda
                break
            end if
        end for
        evaluate(Evaluable_lambda)
        Element.lambdas ← Element.lambdas \ Evaluable_lambda
    until Evaluable_lambda = null
end procedure
```

Algorithm 2 Production rule solving process

```
procedure SOLVE_PROD_RULES(+Init_rules, +Iter_rules, -Result_set)
    Result_set.add(solve(Init_records))
    Changed ← true
    while Changed do
        Changed ← false
        for all Record in Result_set do
            for all Rule in Iter_rules do
                New_records ← Rule.apply_on(Record)
                Set_just_changed ← Result_set.add(New_records)
                Changed ← Changed ∨ Set_just_changed
            end for
        end for
    end while
end procedure
```

Algorithm 3 Simulation run process

```
procedure RUN_CONSEQ_GENERATOR(+Tmln_in, +Subscrs, -Tmln_out)
    Terminate ← false
    Tmln ← Tmln_in
    while (Tmln ≠ []) ∧ ¬Terminate do
        Event_in ← Tmln.pop()
        for all Subscr ∈ Subscrs do
            if Subscr.reacts_on(Event_in) then
                Terminate ← Terminate ∨ Subscr.execute(Event_in, Events_out)
                Tmln.put_sorted(Events_out)
                for all Event_out ∈ Events_out do
                    record_causality(Event_in, Subscr, Event_out)
                end for
            end if
        end for
    end while
    Tmln_out ← Tmln
end procedure
```

References

Amin, S. M. and Wollenberg, B. F. 2005. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE 3,* 5, 34–41.

Bergmann, J., Glomb, C., Gotz, J., Heuer, J., Kuntschke, R., and Winter, M. 2010. Scalability of smart grid protocols: Protocols and their simulative evaluation for massively distributed DERs. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. 131–136.

Chaudhri, V. K., Heymans, S., Wessel, M., and Tran, S. C. 2013. Object-oriented knowledge bases in logic programming. In *Technical Communication of International Conference in Logic Programming*.

Chisham, B., Pontelli, E., Son, T. C., and Wright, B. 2011. CDAOStore: A phylogenetic repository using logic programming and web services. In *ICLP (Technical Communications)*. 209–219.

Duží, M., Jespersen, B., and Materna, P. 2010. *Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic*. Vol. 17. Springer Science & Business Media.

Favre, J.-M. 2005. Megamodelling and etymology – a story of words: From MED to MDE via MODEL in five milleniums. In *APPEARED IN DROPS 04101, IBFI*.

Georg, H., Wietfeld, C., Muller, S., and Rehtanz, C. 2012. A HLA based simulator architecture for co-simulating ict based power system control and protection systems. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*. 264–269.

Hopkinson, K., Wang, X., Giovanini, R., Thorp, J., Birman, K., and Coury, D. 2006. Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components. *Power Systems, IEEE Transactions on 21,* 2 (May), 548–558.

IEC. 2006. *IEC 60870-5-104: Telecontrol equipment and systems — Transmission protocols*. International Electrotechnical Commission, Geneva, Switzerland. 2006.

IEEE. 2011. Ieee guide for smart grid interoperability of energy technology and information technology operation with the electric power system (EPS), end-use applications, and loads. *IEEE Std 2030-2011*, 1–126.

Kosek, A., Lunsdorf, O., Scherfke, S., Gehrke, O., and Rohjans, S. 2014. Evaluation of smart grid control strategies in co-simulation 2014; integration of IPSYS and mosaik. In *Power Systems Computation Conference (PSCC), 2014*. 1–7.

Křenek, A., Holub, P., Holer, V., Kouřil, D., Procházka, F., Hejna, Z., Guričan, M., and Muller, F. 2013. 3.5 million smartmeters in the cloud. In *Proceedings ISGC 2013*, S. Lin, Ed. PoS - Proceedings of Science, Trieste, 180–189.

Lévesque, M., Béchet, C., Suignard, E., Maier, M., Picault, A., and Joós, G. 2014. From co-toward multi-simulation of smart grids based on HLA and FMI standards. *arXiv preprint arXiv:1412.5571*.

Mets, K., Ojea, J. A., and Develder, C. 2014. Combining power and communication network simulation for cost-effective smart grid analysis. *Communications Surveys Tutorials, IEEE 16,* 3 (Third), 1771–1796.

Mets, K., Verschueren, T., Develder, C., Vandoorn, T., and Vandevelde, L. 2011. Integrated simulation of power and communication networks for smart grid applications. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2011 IEEE 16th International Workshop on*. 61–65.

Patel, A., Aparicio, J., Tas, N., Loiacono, M., and Rosca, J. 2011. Assessing communications technology options for smart grid applications. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*. 126–131.

Pompe, P. 2015. Masters thesis, Masaryk University, Faculty of Informatics. Available at http://is.muni.cz/th/325292/fi_m/.

Rosecky, J. 2015. Grid Mind website. URL http://www.mycroftmind.cz/en/grid-mind/, [Online; accessed 19-July-2015].

Schütte, S., Scherfke, S., and Sonnenschein, M. 2012. mosaik-smart grid simulation API. *Proceedings of SMARTGREENS*, 14–24.

Trefke, J., Rohjans, S., Uslar, M., Lehnhoff, S., Nordstrom, L., and Saleem, A. 2013. Smart grid architecture model use case management in a large european smart grid project. In *Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES*. 1–5.

Wille, R. 1992. Concept lattices and conceptual knowledge systems. *Computers & Mathematics with Applications 23,* 6–9, 493–515.