Experimental Validation of BDI Agents for Distributed Control of Electric Power Grids

Issicaba, Diego; Rosa, Mauro A. ; Prostejovsky, Alexander Maria; Bindner, Henrik W.

Published in:
Proceedings of 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe

Publication date:
2017

Document Version
Peer reviewed version

Experimental Validation of BDI Agents for Distributed Control of Electric Power Grids

Diego Issicaba and Mauro A. Rosa
Labplan/INESC P&D Brasil
Department of Electrical and Electronic Engineering
Federal Unversity of Santa Catarina
Florianopolis-SC, Brazil 88040–900
Email: diego.issicaba@inescbrasil.org.br

Alexander M. Prostejovsky and Henrik W. Bindner
Center for Electric Power and Energy Systems
Department of Electrical Engineering
Technical University of Denmark
Roskilde, Denmark 4000
Email: alepros@elektro.dtu.dk

***Abstract*—This paper presents initial laboratory experiments designed to test belief-desire-intention agent reasoning in a web-of-cell context. The work introduces the application of the bridge between JASON and the Common Artifact Infrastructure for Agents Open Environments to agent and environment modeling, respectively. Belief-desire-intention reasoning is achieved through JASON's engine while artifacts monitor and control grid devices using dedicated JAVA objects. Experiments have been conducted in SYSLAB, a testbed for distributed power system control and distributed solutions, located at the Risø campus of the Technical University of Denmark. Experimental results show the feasibility of applying belief-desire-intention reasoning to WoC control using a test case where tie-line power flow setpoints must be followed.**

***Index Terms*—Power distribution, Multi-agent systems, Smart grid.**

NOMENCLATURE

A&A	Agents and Artifacts
BDI	Belief-Desire-Intention
CArtAgO	Common ArTifact Infrastructure for Agents Open Environments
DMS	Distribution Management System
DTU	Technical University of Denmark
ELECTRA	Electricity Committed Towards long-term Research Activity
IRP	Integrated Research Programme
JADE	Java Agent Development Framework
PV	Photovoltaics
WoC	Web-of-Cell

I. INTRODUCTION

INNOVATIVE concepts are the main catalysts to rapid advancements in science and technology. The power industry has been evolving by absorbing innovative concepts in order to build more effective processes, technologies, products, services and business models. Innovations have been motivated by initiatives, which foment generation through renewable energy, interconnection of distributed energy resources, adoption of decentralised management solutions, customer participation and power system modernisation as a whole.

978-1-5386-1953-7/17/$31.00

The main driver behind these initiatives is the changing landscape of electric power systems. The integration of small-scale renewable energy sources and the active involvement of power consumers challenge the established (de)centralized monitoring and control systems. The Web of Cell (WoC) approach proposes a distributed solution, where cells, as opposed to traditional control areas, possess a high degree of automation and can be located throughout all voltage levels [1]. This concept is promoted by the European Liaison on Electricity Committed Towards long-term Research Activity Integrated Research Programme (ELECTRA IRP) [2], which targets aligning national research efforts towards developing new approaches for distributed control.

The WoC concept can be facilitated using the agent paradigm, which provides a suitable approach to design and test distributed solutions. Multi-agent systems have been already applied in areas of power engineering including monitoring [3], [4], [5], diagnoses [6], fault location, isolation and power restoration [7], [8], [9]. However, most of multi-agent applications to power engineering relies on the midware Java Agent Development Framework (JADE) [10], ignoring aspects related to computational modeling of the environment. Also, most applications lack actual experimentation in laboratory environments and in the field.

This paper presents initial laboratory experiments designed to test belief-desire-intention (BDI) agent reasoning to WoC control. Goal-directed behaviors interrelated with agent planning are then emulated using JASON [11], a Java-based interpreter for an extended version of AgentSpeak, which in turn is based on the BDI architecture. The application of Common ArTifact Infrastructure for Agents Open Environments (CArTAgO) [12] is proposed, allowing both a bridge to JASON and environment artifacts. BDI reasoning is achieved through JASON's engine while CArtAgO's artifacts are designed to act over JAVA objects, which in turn can read and act over monitoring and controlling devices through a laboratory communication infrastructure. The experiments have been conducted in SYSLAB, a smart grid laboratory facility provided with smart components and flexible configuration. Experimental results show the feasibility of applying BDI reasoning to cell control, more specifically in a test case

where tie-line power flow setpoints must be followed.

The paper is organised as follows. Section II introduces the agent-artifact framework [13] designed to the experiments. Section III presents results for initial laboratory experiments on BDI reasoning to WoC control. Section IV outlines conclusions and final remarks.

II. Agent-Artifact Framework to BDI Cell Control

The key principle behind the WoC is to *solve local problems locally* by splitting the grid within cells, as illustrated in Fig. 1. Cells must communicate with each other to enable distributed collaboration and information exchange, in similarity with the block-oriented approach proposed in [13]. A cell operator is then responsible for establishing and maintaining autonomous control mechanisms contributing to an adequate and secure operation. Among the modeling possibilities for the cell operator, the BDI agent architecture stands out as a successful mechanism to model human-like reasoning, planning and decision-making, with the additional benefit of allowing analysis via model checking.

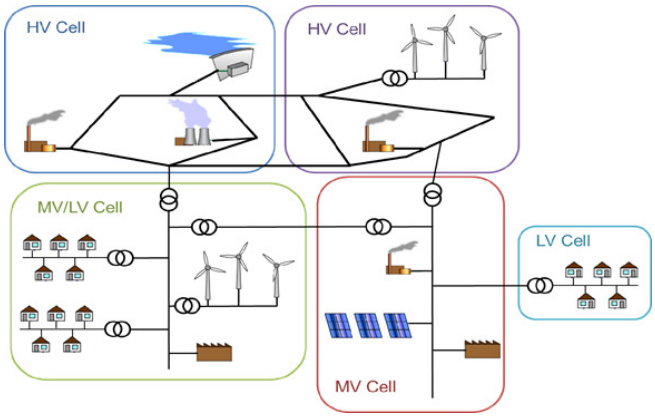


Fig. 1. Overview of the WoC concept that proposes a flat hierarchy accross all voltage levels. [14].

The BDI model is utilised in AgentSpeak, a logic-based agent-oriented language whose recent versions can be interpreted using JASON. If well-utilised, JASON allows high-level representations of agent reasoning through AgentSpeak and, at the same time, a sophisticated use of legacy code and object-oriented models implemented in Java. Differently from JADEX, JASON belongs to the group of theoretically-rooted agent-oriented programming languages, outlining a strong emphasis on rigorous formal semantics [15]. Furthermore, similarly to JADEX, JASON can use the JADE infrastructure. JASON is the standard instance utilized in CArTAgO documentation, code examples and literature. All these features made JASON the natural choice towards modelling and testing BDI-based cell control processes.

A. Agent Modeling and Reasoning

Following the WoC paradigm, a `cellagent` has been abstracted as cell operator. Each `cellagent` is provided with a *belief base* defined by a set of literals, where beliefs are modeled as *predicates* in symbolic forms [16], [17] such as

```
on(cmp-X09-ABB)[source(percept)]
```

which denotes that "the component named *cmp-X09-ABB* is on". The instruction `[source(percept)]` denotes an *annotation* meaning that a sensor placed in the environment is the source of the belief. If `on(cmp-X09-ABB)[source(percept)]` is placed in an agent's belief base, it means that currently the agent believes that the predicate is true. Using *theoretical reasoning*, inconsistencies can be found through rules, such as

```
alarm(state,Comp) :- off(Comp)[source(dms)] &
                     on(Comp)[source(percept)]
```

which assigns an alarm if contradictory information has been delivered by distinct sources regarding component status.

States of affair of interest are modeled as *achievement goals* using the mark "!". In order to verify whether the agent believes a set of literals, *test goals* are marked with "?". For instance, the literal

```
?voltage_value(cmp-Y09-ABB,Vmag)
```

unifies `Vmag` with the voltage magnitude the agent believes is associated to component `cmp-Y09-ABB`, while the literal

```
!open(neigh_switches)
```

denotes that the agent has the goal of achieving a state of affair in which it believes `open(neigh_switches)` is a true statement. The marks "+"/"-" symbolize addition/deletion, either of a belief or a goal.

For instance, let us consider the agent meta plan abstracted to isolate a cell shown in Fig. 2, as usually defined in agent design methodologies. Simplified versions of plans derived from this meta plan are shown below.

Description: Actions are taken to isolate the cell aiming at supporting the service restoration.
Context: The cell is assigned de-energized.
Functionality: `outage management`.
Trigger: `+sensor(·)`.
Incoming messages: [←Cell,DMS] Inform message.
Outgoing messages: [→Cell,DMS] Inform message.
Percepts: `sensor(·)`, `energized(·)`, `isolated(·)`.
Actions: `open_switch(·)`, `hmi_update(·)`.
Used data: Outage management data.
Produced data: Outage management updated data.
begin
  - Open neighboring switches.
  if *successful* **then**
    - Mark topology status as `isolated(·)`.
    - Inform neighbors.
  end
  else
    - Proceed according to other meta plans.
  end
end

Fig. 2: Example of meta plan: cell isolation.

```
@mp18_02 +!isolate_itself : energized(false,·)
<-...;!open_list_switches(ListOfIDs).

@mp18_03 +!open_list_switches([]) : true <- true.

@mp18_04 +!open_list_switches([ToName|T]) : true
<- ...;!open_switchID(ToName);
!open_list_switches(T).
```

In JASON, these plans model the following reasoning: *to handle the addition of the achievement goal corresponding to isolate the cell, if the cell is believed to be de-energized, then pursue the goal of opening neighboring switches*.

Similarly to this example, several abstracted meta plans can have a large set of plans covering from simple inferences to complex reasonings. Each agent has then a threaded reasoning cycle with the steps: (i) perceive the environment, (ii) update belief base with perceptions; (iii) receive communications; (iv) select socially acceptable messages; (v) select an event to be handled; (vi) retrieve and determine applicable plans; (vii) select one applicable plan; (viii) select an intention (course of action) and execute one step of the intention.

JASON interpreter includes several inner complex functionalities such as libraries for internal actions, customization options, plan patterns, and so forth. These functionalities have been exploited to model the agent plans of actions.

B. Environment Modeling and Implementation

JASON is provided with an environment infrastructure centralized in a unique object, a strategy which somehow seems not perfectly linked (in modeling terms) with the idea of decentralizing services and resources to support collective and individual activities. Furthermore, in computational modeling of power systems, several objects are not directly affected by agent actions or have attributes to be perceived by agents. This revels the need for an abstraction beyond the concept of an object in order to model environments that allow testing and validating agent-based solutions in power engineering.

The infrastructure mentioned above has been led by the classical notion of environment used to identify *the external world that is perceived and acted upon* by agents. There exists, however, a modern view of environment as a *first-class abstraction* of agent system engineering where to encapsulate services and resources to aid agent activities. This modern view is employed in the agents and artifacts (A&A) meta model of CArtAgO, where *artifacts* are conceived as general resources and tools to be shared and exploited by agents. From the point of view of designing, artifacts are basic modules to structure and organize environments, providing a general-purpose model to shape functionalities available to agents. On the other hand, from the agent point of view, artifacts are first-class entities structuring a world that the agents can affect in runtime. Artifacts are utilized according to the modeling layers illustrated in Fig. 3.

In the figure, one might observe the explicit separation of agent, artifact and object-oriented modeling layers. Summarily, the agent modeling layer aggregates agents, whilst the object-oriented modeling layer is comprised of computational objects

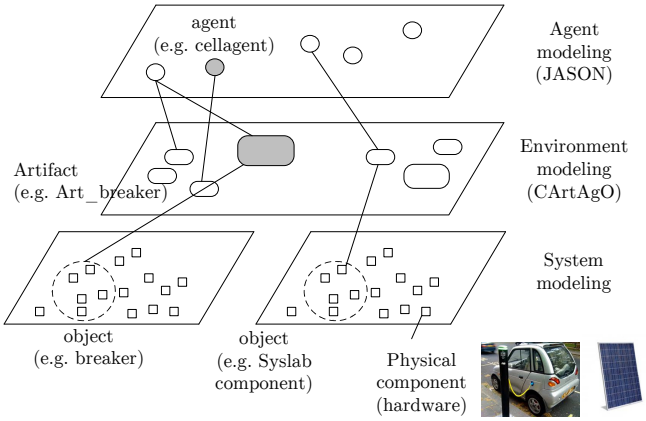


Fig. 3. Environment modeling layers.

for components (e.g. switch, node, line). In the artifact modeling layer, there are the artifacts which interact directly with the agents and may encapsulate several data in the form of computational objects. Moreover, they provide sets of operations and observable properties to the agents. These *operations* are computational processes which may be triggered by agents or other artifacts, whilst the *observable properties* are the variables whose value can be perceived by agents which are observing the artifact.

Artifacts have been defined for the cell components directly involved on agent capabilities utilised in the experiments. Physical components can be acted upon and monitored through objects atributed to the artifacts. The bridge between the modeled artifacts and agents is shown in Fig. 4, where concept mappings are marked through filled squared end connections. In this bridge, one can verify the roles of agent and environment simulation of JASON and CArTAgO, respectively. Using this bridge, agents can be allowed to consult a manual with the description of artifact properties, create/dispose/link artifacts, use operations, perceive observable events, observe/perceive properties and join/quit workspaces. Observable properties/events are provided as literals (e.g. `on(cmp-X09-ABB)[source(percept)]`) to belief bases and ex-

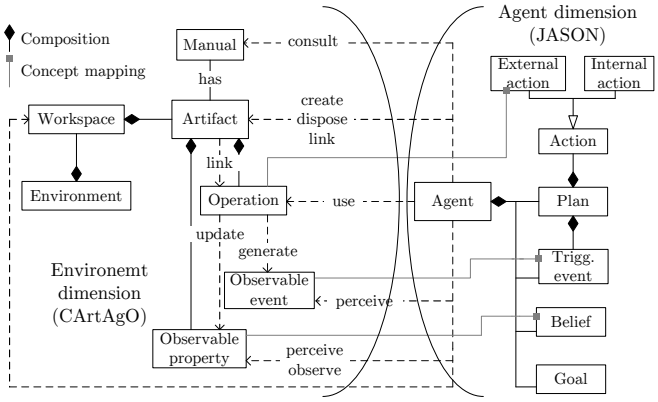


Fig. 4. Bridge between JASON and CArtAgO (adapted from [18]).

ternal actions are performed through artifact operations.

III. Laboratory Experiments

In order to test the application of BDI agents in SYSLAB, a smart grid laboratory designed for testing advanced control and distributed solutions to power systems. SYSLAB is placed at DTU campus and involves 4 interconnected electric sites/buildings. Among other components, SYSLAB facilities includes a 400 V grid with flexible configuration, renewable energy units such as wind turbines and photovoltaic (PV) modules, embedded computing power and flexible communication platform, real-time monitoring, back-to-back converter, controllable loads, vanadium battery and electric vehicles.

Fig. 5 shows the SYSLAB layout with the used configuration marked in red, and the corresponding single-line diagram highlights the components utilised in the experiment. The properties of the components are listed in Table I, where P_{min} and P_{max} are the minimum and maximum powers, and P_0 stands as the operating points where applicable. These experiments are related to evaluating the feasibility of managing, with simple reasonings, cell power imbalances using load and battery control, aiming at following tie-line power flow setpoints. In the figure, closed breakers form a connected grid comprising DTU interconnection, 1 PV module (PV319), 1 bank of controllable loads (building 319), 1 PV module (building 117), 1 controllable vanadium battery and 1 additional PV module (PV117). The grid connection between buildings has been achieved by closing breakers C1 and A2. The remaining adjacent breakers have been opened to guarantee isolation from other experiments.

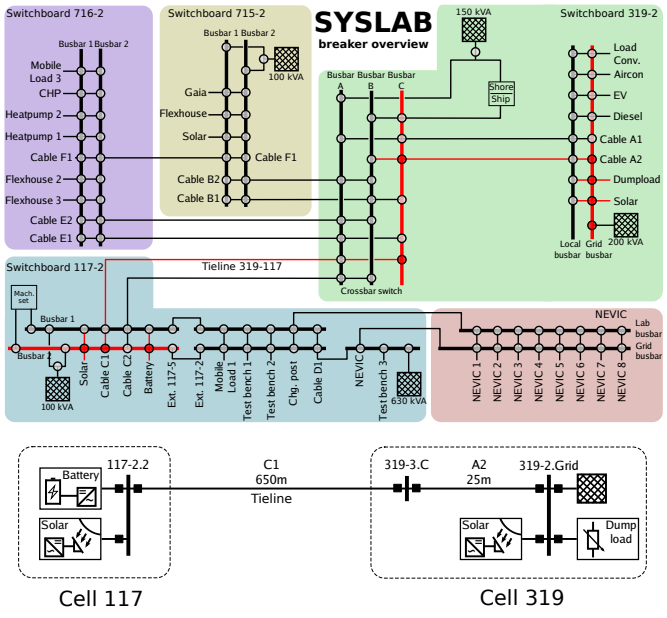


Fig. 5. Layout of the experimental facility SYSLAB (used configuration marked in red) and the corresponding single line diagram.

Within the frame of this work, buildings 319 and 117 are considered cells and `cellagents`, named `cellagent319` and `cellagent117`, are assigned to each building. Each `cellagent` must establish and maintain the operation of its assignee. Among sets of intentions, in order to achieve such target, each `cellagent` must monitor and control power imbalances of its assignee. If tie-line power flow setpoints are changed, total cell inner imbalance must be changed as well. Hence, actions must be taken aiming to return to a "normal operation" where setpoints are followed adequately.

TABLE I
Properties of devices used in the experiment.

Device	Cell	Busbar	P_{min} (kW)	P_{max} (kW)	P_0 (kW)	Description
Battery	117	117-2.2	-15.0	15.0		Vanadium redox type, 190 kWh
Cable A2	319	319-2.G 319-3.C				$G = -117.7$ S, $B = -28.6$ S
Cable C1	319 117	319-3.C 117-2.2			± 2.0	Tieline 319-117, $G = -7.74$ S, $B = -4.46$ S
Dumpload	319	319-2.G	-78.09	0.0		Resistor load bank, 256 steps
Grid	319	319-2.G			0.0	200 kVA
Solar	319	319-2.G	0.0	10.1		Orientation az. 180°, el. 40°
Solar	117	117-2.2	0.0	1.0		Orientation az. 100°, el. 20°. Due to a fault in the controller, the inverter clamped P_{max} to 1 kW instead of the nominal 10.1 kW.

In BDI, this sort of reasoning can be placed according to simple plans as follows:

```
!normaloperation.                                        (i)
```

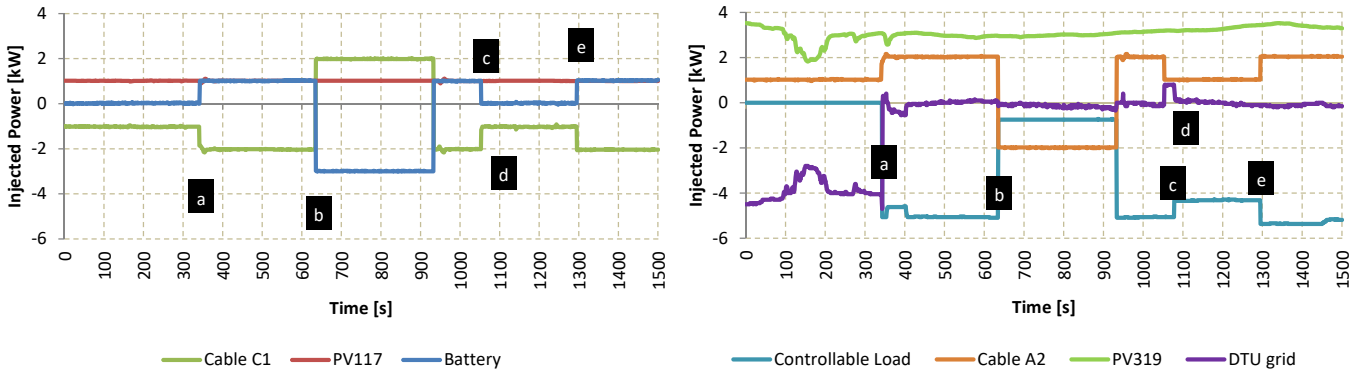
```
+!normaloperation: normaloperation <- .wait(3000);
!normaloperation.                                        (ii)
```

```
-!normaloperation : not normaloperation
<- ?imbalanceError(E);
   ?imbalance(I);
   setResource(I,E);.wait(7000);
   !normaloperation.                                     (iii)
```

in which (i-ii) the `cellagent` cyclically targets achieving a state of affair related to normal operation. If there is not a literal (or set of literals) in its belief base indicating such state of affair, (iii) the `cellagent` starts the execution of a plan which begins with the unification of auxiliary literals `I` and `E` with cell inbalance and inbalance error, respectively.

The referred unification is performed by reasoning over literals corresponding to the observable properties of artifacts, namely those related to tie-line power flows, through the process called theoretical reasoning. The imbalance data is then used to alter properties of artifacts in real time, such as load consumption (cell 319) and battery charging (cell 117). In case of identifying any problem during these processes (e.g. communication failure, device failure), the belief base is automatically updated though artifact observable properties and a discussion between neighboring `cellagents` initiate in order to try solving the problem at least partially.

In Fig. 6, the injected active power (generator convention) in the main components of the experiment is shown. Data covering 25 minutes of system operation have been obtained



(a) Cell 117 (b) Cell 319

Fig. 6. Injected powers in each cell.

on 29-Apr-2016 starting at 12:50. Before initialising the agent system, SYSLAB power grid has provided roughly 4 kW to the external grid and a power flow of around 1 kW has been measured in the interconnection from cells 117 to 319. Although solar radiance was nonnegligible during the day of the experiment, PV117 production has been clamped to 1 kW due to a controller malfunction. On the other hand, PV319 production varies from 1.8 kW to 3.7 kW during the day since its inverter maximum power generation has been specified at 5 kW.

The agent system initialisation is marked at [a], where an auxiliary agent corresponding to the Distribution Management System (DMS) sends updated instructions to the cell agents, demanding null power flow at the DTU grid interconnection and 2 kW flowing from cell 319 to cell 117. Using the real-time monitoring of power flows and BDI plans, `cellagent117` identifies the necessity of increasing inner production in 1 kW. Following a priority order embedded in external action `setResource(I,E)`, this increase has been ordered to the battery control artifact. In cell 319, a demand increase is required to the controllable load to values of around 5 kW, aiming at absorbing all inner generation from both cells. In [b], the DMS imposed a sudden requirement indicating the inversion of power flow between cells. Each `cellagent` controlled its cell resources accordingly, where `cellagent117` required the battery artifact to charge at 3 kW and `cellagent319` ordered load reduction to 1 kW. BDI reasoning operated adequately and DMS requirements have been met, as shown in Fig. 6.

In [c], the DMS required the power flow from cell 117 to cell 319 to return to the value of 2 kW. Setpoint orders have been sent to the resources accordingly. However, after some minutes, the battery flagged failed operation and shut down. This process has been included in the experiment through the battery artifact, where the shut down has been scheduled to occur after a maximum number of inner operations. Additional setpoint orders have been sent by `cellagent117` to the battery artifact to confirm the updated belief of failed operation. Concomitantly, `cellagent319` has identified an abnormal operation state by analysing tie-line power flows. Since its inner inbalances have been reasoned to be adequate, a belief has been generated indicating that a problem occurred out of its cell. After confirming the problem, `cellagent117` conveys a message to `cellagent319` flagging "abnormal operation, inner solution not found". Hence, `cellagent319` confirms message reception indicating that it will take actions to at least nullifying DTU grid power flow. The controllable load is utilised to nullify DTU grid power flow and, some minutes after, the battery returns to normal operation and all setpoints are followed adequately.

IV. Final Remarks

Multi-agent systems have been applied considerably in power engineering aiming at modelling and simulating distributed solutions. Nevertheless, most of applications have been designed using the midware JADE, which lacks an explicit environment modelling framework. To overcome this problem, this work presents the first application of JASON and CArTAgO to power system operation and control. Using an A&A meta model, BDI agents have been modelled to interact with artifacts which encapsulate JAVA objects, which in turn are able to monitor and control grid devices through a laboratory communication infrastructure. The framework has been utilised in laboratory experiments where BDI agents are responsible to WoC control. Experimental results show the feasibility of applying BDI reasoning to WoC control using a test case where tie-line power flow setpoints must be followed. Future work will extend the application envisioning A&A general interactions to test and validate distributed solutions to WoC operation and control.

V. Acknowledgements

The work in this paper has been supported by the European Commission under the FP7 project ELECTRA (grant no: 609687) and INESC P&D Brasil. The authors would like to thank all involved people at the Technical University of Denmark and SYSLAB.

REFERENCES

[1] A. Prostejovsky, O. Gehrke, A. M. Kosek, F. Coffele, and A. S. A. E. Zaher, "Distributed framework for prototyping of observability concepts in smart grids," in *2015 International Symposium on Smart Electric Distribution Systems and Technologies (EDST)*, pp. 593–600, Sept 2015.

[2] ELECTRA IRP, "European liaison on electricity committed towards long-term research activity integrated research programme." http://www.electrairp.eu (Online; accessed 17-June-2015).

[3] E. E. Mangina, S. D. J. McArthur, J. R. McDonald, and A. Moyes, "A multi agent system for monitoring industrial gas turbine start-up sequences," *IEEE Transactions on Power Systems*, vol. 16, pp. 396–401, Aug. 2001.

[4] S. D. J. McArthur, S. M. Strachan, and G. Jahn, "The design of a multi-agent transformer condition monitoring system," *IEEE Transactions on Power Systems*, vol. 19, pp. 1845–1852, Nov. 2004.

[5] V. M. Catterson, S. E. Rudd, S. D. J. McArthur, and G. Moss, "On-line transformer condition monitoring through diagnostics and anomaly detection," in *Proceedings of the International Conference on Intelligent System Applications to Power Systems (ISAP)*, (Curitiba, Brazil), Nov. 2009.

[6] J. A. Hossack, J. Menal, S. D. J. McArthur, and J. R. McDonald, "A multiagent architecture for protection engineering diagnostic assistance," *IEEE Transactions on Power Systems*, vol. 18, pp. 639–647, May 2003.

[7] M. M. Nordman and M. Lehtonen, "Distributed agent-based state estimation for electrical distribution networks," *IEEE Transactions on Power Systems*, vol. 20, pp. 652–658, May 2005.

[8] I. S. Baxevanos and D. P. Labridis, "Implementing multiagent systems technology for power distribution network control and protection management," *IEEE Transactions on Power Delivery*, vol. 22, pp. 433–443, Jan. 2007.

[9] G. Zhabelova and V. Vyatkin, "Multiagent smart grid automation architecture based on IEC 61850/61499 intelligent logical nodes," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 2351–2362, May 2012.

[10] JADE, "Java Agent DEvelopment framework." http://jade.tilab.com/.

[11] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology, Chichester, England: John Wiley & Sons, Nov. 2007.

[12] A. Ricci, M. Piunti, and M. Viroli, "Environment programming in multi-agent systems: An artifact-based perspective," *Autonomous Agents and Multi-Agent Systems*, vol. 23, pp. 158–192, Sep. 2011.

[13] D. Issicaba, *Block-oriented Agent-based Architecture to Support the Power Distribution System Operation*. PhD thesis, Faculty of Engineering of Porto University, May 2013.

[14] R. D'hulst, J. M. Fernández, E. Rikos, D. Kolodziej, K. Heussen, D. Geibelk, A. Temiz, and C. Caerts, "Voltage and frequency control for future power systems: the electra irp proposal," in *2015 International Symposium on Smart Electric Distribution Systems and Technologies (EDST)*, pp. 245–250, Sept 2015.

[15] R. Píbil, P. Novak, C. Brom, and J. Gemrot, "Notes on pragmatic agent-programming with jason," in *Programming Multi-Agent Systems* (L. Dennis, O. Boissier, and R. H. Bordini, eds.), vol. 7217 of *Lecture Notes in Computer Science*, pp. 58–73, Springer Berlin Heidelberg, 2012.

[16] A. S. Rao, "Agentspeak(L): BDI agents speak out in a logical computable language," in *Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW)*, vol. 1038, (Eindhoven, Netherlands), pp. 42–55, Jan. 1996.

[17] D. Issicaba, J. A. P. Lopes, and M. A. Rosa, "Agent-based simulation of active distribution grids following a jason-based approach," in *Proceedings of the International Conference on Intelligent System Applications to Power Systems (ISAP)*, (Tokyo, Japan), Jul. 2013.

[18] O. Boissier, R. H. Bordini, J. F. Hubner, A. Ricci, and A. Santi, "Multi-agent oriented programming with JaCaMo," *Science of Computer Programming*, vol. (in Press), 2011.