An Open MAS Services Architecture for the V2G/G2V Problem

Extended Abstract

Nikolaos Spanoudakis
School of Production Engineering and Management,
Technical University of Crete, Greece
nikos@amcl.tuc.gr

Charilaos Akasiadis
School of Electrical and Computer Engineering,
Technical University of Crete, Greece
akasiadi@intelligence.tuc.gr

Georgios Kechagias
School of Electrical and Computer Engineering,
Technical University of Crete, Greece
gkechagias@isc.tuc.gr

Georgios Chalkiadakis
School of Electrical and Computer Engineering,
Technical University of Crete, Greece
gehalk@intelligence.tuc.gr

ABSTRACT

In this paper we propose an original and open multi-agent system architecture for the important and challenging to engineer vehicle-to-grid (V2G) and grid-to-vehicle (G2V) energy transfer problem domain. To address the features required, we define two novel design patterns that can be used with statecharts in many real-world situations. The first one is based on the well-known factory design pattern, and the second on the class generalization relationship. These patterns can be coupled with ASEME, an agent-oriented software engineering methodology that uses statecharts for the inter- and intra-agent control models. The latter also fits well with the FIPA standards-compliant JADE agent platform that we used for implementation.

KEYWORDS

open multi-agent systems; smart grid; design patterns

ACM Reference Format:
Nikolaos Spanoudakis, Charilaos Akasiadis, Georgios Kechagias, and Georgios Chalkiadakis. 2019. An Open MAS Services Architecture for the V2G/G2V Problem. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

In the emerging Smart Grid [1–3, 5, 13], buildings, but also vehicles, become active consumers and producers of energy, and need to be integrated into the Grid. Not only is the Smart Grid an electricity network with diverse consumers and producers, it is also a dynamic marketplace where heterogeneous devices appear and need to connect [11]. To date, several Smart Grid-related business models and information systems' architectures have been proposed, but they do not adhere to particular standards [7].

Such energy markets naturally reflect systems where not one player can force others to use her products; players or stakeholders can come along their own business models; and stakeholders can have diverse goals in negotiating their consumption and offer. Moreover, these systems allow for pro-activeness of the players who pursue their goals and sociability—as they can form dynamic partnerships or coalitions, but also react and /or adapt to a changing dynamic environment. All these characteristics point to open *multiagent systems (MAS)* and agent technology in general [9, 12, 21]. To the best of our knowledge, however, existing Smart Grid approaches fail to make proper use of existing engineering MAS research paradigms, and do not cover the aforementioned requirements, as they are mostly closed proprietary systems. This is particularly true for the specific sub-domain that motivated this work, i.e., the Vehicle-to-Grid (V2G)/ Grid-to-Vehicle (G2V) problem [10, 13]. Briefly, in G2V approaches "smart charging" might not initiate instantly upon EV connection, but can be delayed according to various factors [14, 19, 20], e.g., renewable production levels, demand from other EVs, pricing, and so on. Complementary to G2V, V2G takes advantage of the storage capabilities of EV batteries, and allows their controlled discharging for supporting the Grid during times of energy supply shortage [6, 10].

Now, to realize an open system, agents need to use predefined protocols to interact. However, when diverse stakeholders come in they need to work the protocols with their own algorithms and/or goals. In this paper we propose two new design patterns, that on the one hand allow the developers to re-use the protocol parts and logic defined in the framework, and on the other hand to customize key functionality or capabilities according to their needs/goals.

2 ARCHITECTURE

We assume that agents coexist in a microgrid infrastructure that can be interconnected with other parts of the Grid through distribution and transmission networks. When a microgrid requires power that can not be generated locally, it can import it; while a local energy surplus can be exported to the Grid and create additional profits for its electricity producers, according to energy market regulations [11].

In particular, the agent types in our system are (see Figure 1): the *(a)* Electric Vehicle agents (EV), the *(b)* Charging Station agents (CS), the *(c)* Electricity Producer agents (EP), and the *(d)* Electricity Consumer agents (EC). We also assume the existence of a regulatory service, or possibly a for-profit private service, that consists of *(i)* a Station Recommender agent (SR), *(ii)* an Electricity Imbalance agent (EI), and *(iii)* a Mechanism Design agent (MD).

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

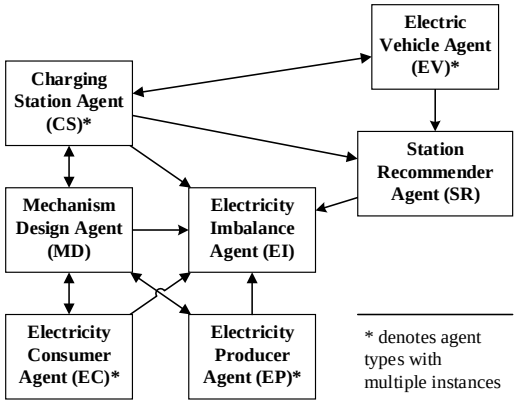

Figure 1: The agents and their interconnections in our V2G/G2V architecture.

EV agents aim to optimize a utility function set by the EV owner—e.g., always have enough energy to realize the next trip, achieve so by the minimum cost, etc. *CS agents* manage the physical gateways (i.e. connectors, parking slots) by which EVs connect to the grid and create profit by charging their batteries. They can also negotiate with *EV agents* regarding an existing charging agreement. This leads to better utilization of the station infrastructure, and maximizes its profit. The *SR agent* recommends to EVs a subset of the available CS and charging slots that match most with the EVs preferences (e.g. time of arrival/departure, distance, battery state). The *EI agent* aggregates information from agents regarding their expected energy profiles, and calculates the periods of electricity shortage and surplus. Then, it provides the imbalance levels to all interested parties, for them to plan their electricity consumption and production activities. The *MD agent* is an intermediate trusted third party entity, responsible for calculating dynamic prices and managing the payments of the various contributor types. Finally, the various *EP* and *EC* agents periodically report their predictions on expected production and consumption levels respectively, to enable accurate imbalance forecasts for the planning horizon.

3 DESIGN PATTERNS FOR OPEN PROTOCOLS

The architecture above poses several challenges for the rest of the development process. Clearly, there is a need to accommodate different methods for decision-making based on user preferences, or on the business model of a stakeholder, or on the agent that implements a protocol role. As an example, both *EV* and *CS* agents have the capability to negotiate, however, it is obvious that they will most likely employ different algorithms to do so.

Thus, we need to cater for agents following protocols to realize their goals in the system, while being able to define their own algorithms, policies or business rules. These cannot be foreseen at design time for all future stakeholders in an open system. To address these challenges we were inspired by the protocol module code generation feature of ASEME [15, 16]. ASEME defines a code generation process for the popular JADE agent platform [4] that is ideal for open systems as it implements the FIPA standards (Foundation of Intelligent Physical Agents, http://www.fipa.org).

The inter-agent control model of ASEME (i.e. the design phase model for agent interactions) is a statechart and the ASEME integrated development environment (IDE) [18] allows to generate code for it and store it in its own package [17]. However, this feature is not connected with the intra-agent control model (i.e. the design phase model for the individual agent modules coordination) code generation, and, even though code generation is automated, it only generates the control code, not the action methods of the agents: in practice, all behaviours action codes must be rewritten for each agent. This, however, poses certain risks. The protocols' intended flow may be disrupted by the code of programmers, or the same code must be rewritten in several packages.

To remedy this situation we identified two distinct cases. In the first case, we have a decision-making behaviour—e.g., in a negotiation protocol, where the objects of the negotiation are quite clear, however, there are different strategies for the agents to employ. We would like to allow the diverse agent developers to develop their own strategies. Thus, a specific functionality with clear parameters needs to be able to be supplied by the developing team: that is, the developers need to associate a specific algorithm to a behaviour's action method.

Thus, we define a design pattern, where the developers need to associate a specific algorithm to a behaviour's action method. To remedy this situation we relied on the well-known by practitioners *factory design pattern* [8]. According to this pattern, we use a factory method that returns an instance of a method respecting an API which can be dynamically selected, even at run-time.

Consider now the case of *SR agents*: it is clear that a service provider gets a request, processes it and then replies with an appropriate response. The process part, however, can be very different and complicated, and can change not only based on policy, but also based on the agent's data structures and architecture. Thus, the agent now needs to define a new capability. This time it is not just the algorithm that changes. The implementation of this capability may involve its engagement in other protocols or the undertaking of many different activities. For example, a broker might want to request a service from a third party in order to service a request.

Thus, we need another design pattern based on developing an activity that will take place within the relevant state in the protocol. This is not implemented in the protocol package. Now, although all agents use the same protocol package, they each develop their own specific state activity, or *handler behaviour*. The latter can be just a simple behaviour doing a specific action or a complex behaviour with many sub-behaviours.

4 CONCLUSION

We presented a novel architecture for the V2G/G2V energy transfer problem domain. Our approach addresses the needs for openness, and the coverage of diverse business models via the definition of a number of key agent types and the development of open protocols. These can be delivered along with the ontology to any interested parties, which can subsequently build their own agents given their expertise and business case. We identified two design patterns, allowing participating agents *(a)* to dynamically select functionalities and *(b)* to define their own implementations of abstract behaviours.

REFERENCES

[1] Charilaos Akasiadis and Georgios Chalkiadakis. 2017. Cooperative electricity consumption shifting. *Sustainable Energy, Grids and Networks* 9 (2017), 38 – 58.

[2] C. Akasiadis and G. Chalkiadakis. 2017. Mechanism Design for Demand-Side Management. *IEEE Intelligent Systems* 32, 1 (2017), 24–31.

[3] P. Asmus. 2010. Microgrids, Virtual Power Plants and Our Distributed Energy Future. *The Electricity Journal* 23, 10 (2010), 72 – 82.

[4] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. 2008. JADE: A software framework for developing multi-agent applications. Lessons learned. *Information and Software Technology* 50, 1 (2008), 10 – 21. https://doi.org/10.1016/j.infsof.2007.10.008

[5] M. J. Burke and J. C. Stephens. 2017. Energy democracy: Goals and policy instruments for sociotechnical transitions. *Energy Research & Social Science* 33 (2017), 35 – 48. Policy mixes for energy transitions.

[6] F. Christianos and G. Chalkiadakis. 2016. Efficient Multi-criteria Coalition Formation Using Hypergraphs (with Application to the V2G Problem). In *Multi-Agent Systems and Agreement Technologies - 14th European Conference, EUMAS 2016, and 4th International Conference, AT 2016, Valencia, Spain, December 15-16, 2016.* 92–108.

[7] E. Espe, V. Potdar, and E. Chang. 2018. Prosumer Communities and Relationships in Smart Grids: A Literature Review, Evolution and Future Directions. *Energies* 11, 10 (2018), 2528.

[8] E. Gamma. 1995. *Design patterns : elements of reusable object-oriented software.* Addison-Wesley, Reading, Mass.

[9] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. 2006. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 13, 2 (2006), 119–154.

[10] W. Kempton, J. Tomic, S. Letendre, A. Brooks, and T. Lipman. 2001. *Vehicle-to-Grid Power: Battery, Hybrid, and Fuel Cell Vehicles as Resources for Distributed Electric Power in California.* Institute of Transportation Studies, Working Paper Series qt0qp6s4mb. Institute of Transportation Studies, UC Davis. https://ideas.repec.org/p/cdl/itsdav/qt0qp6s4mb.html

[11] W. Ketter, J. Collins, and P. Reddy. 2013. Power TAC: A competitive economic simulation of the smart grid. *Energy Economics* 39 (2013), 262 – 270.

[12] P. Papadopoulos, N. Jenkins, L. M. Cipcigan, I. Grau, and E. Zabala. 2013. Coordination of the Charging of Electric Vehicles Using a Multi-Agent System. *IEEE Transactions on Smart Grid* 4, 4 (Dec 2013), 1802–1809.

[13] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. 2012. Putting the 'smarts' into the smart grid: a grand challenge for artificial intelligence. *Commun. ACM* 55, 4 (2012), 86–97.

[14] A. Seitaridis, E. S. Rigas, N. Bassiliades, and S. D. Ramchurn. 2015. Towards an Agent-Based Negotiation Scheme for Scheduling Electric Vehicles Charging. In *Multi-Agent Systems and Agreement Technologies - 13th European Conference, EUMAS 2015, and Third International Conference, Athens, Greece, December 17-18, 2015.* 157–171.

[15] Nikolaos Spanoudakis. 2009. *The Agent Systems Engineering Methodology (ASEME).* Ph.D. Dissertation. Paris Descartes University.

[16] Nikolaos Spanoudakis and Pavlos Moraitis. 2011. Using ASEME Methodology for Model-Driven Agent Systems Development. In *Agent-Oriented Software Engineering XI*, Danny Weyns and Marie-Pierre Gleizes (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 106–127.

[17] N. I. Spanoudakis and P. Moraitis. 2010. Modular JADE Agents Design and Implementation Using ASEME. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2010, Toronto, Canada, August 31 - September 3, 2010.* 221–228.

[18] N. I. Spanoudakis and P. Moraitis. 2015. Engineering Ambient Intelligence Systems Using Agent Technology. *IEEE Intelligent Systems* 30, 3 (2015), 60–67.

[19] K. Valogianni, W. Ketter, J. Collins, and D. Zhdanov. 2014. Effective Management of Electric Vehicle Storage Using Smart Charging. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.* 472–478.

[20] M. G. Vayá and G. Andersson. 2012. Centralized and decentralized approaches to smart charging of plug-in Vehicles. In *2012 IEEE Power and Energy Society General Meeting.* 1–8.

[21] M. Wooldridge and N. R. Jennings. 1995. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* 10 (1995), 115–152.