Multi-Agent Systems for Power Engineering Applications—Part II: Technologies, Standards, and Tools for Building Multi-agent Systems

Stephen D. J. McArthur, Senior Member, IEEE, Euan M. Davidson, Member, IEEE, Victoria M. Catterson, Member, IEEE, Aris L. Dimeas, Student Member, IEEE, Nikos D. Hatziargyriou, Senior Member, IEEE, Ferdinanda Ponci, Member, IEEE, and Toshihisa Funabashi, Senior Member, IEEE

Abstract—This is the second part of a two-part paper that has arisen from the work of the IEEE Power Engineering Society’s Multi-Agent Systems (MAS) Working Group.

Part I of this paper examined the potential value of MAS technology to the power industry, described fundamental concepts and approaches within the field of multi-agent systems that are appropriate to power engineering applications, and presented a comprehensive review of the power engineering applications for which MAS are being investigated. It also defined the technical issues which must be addressed in order to accelerate and facilitate the uptake of the technology within the power and energy sector.

Part II of this paper explores the decisions inherent in engineering multi-agent systems for applications in the power and energy sector and offers guidance and recommendations on how MAS can be designed and implemented.

Given the significant and growing interest in this field, it is imperative that the power engineering community considers the standards, tools, supporting technologies, and design methodologies available to those wishing to implement a MAS solution for a power engineering problem. This paper describes the various options available and makes recommendations on best practice.

It also describes the problem of interoperability between different multi-agent systems and proposes how this may be tackled.

Index Terms—Multi-agent systems.

I. INTRODUCTION

Part I of this paper examined the properties of multi-agent systems (MAS) and discussed how MAS technology offers the means to create flexible, extensible, and fault tolerant systems; and also a modeling approach for creating complex systems or market models.

This part of the paper (Part II) is concerned with the design and implementation of such systems. There are two fundamental questions to be considered, namely:

• How should an autonomous intelligent agent be built for power engineering applications?

• How should a society of agents be built for power engineering applications?

Agents are currently being investigated for a wide range of applications within the community, from monitoring and diagnostics to network control. The justification for their use often lies in the allegedly inherent properties of flexible autonomy, reactivity, pro-activeness, social ability, the distributable nature of agents, the possibility of emergent behavior, and the fault tolerance of agent systems. In reality, the design decisions and specific implementation techniques used for an agent can constrain it to the point that these properties are not displayed. For this reason, it is essential that current best practices are followed when developing a multi-agent system.

This paper discusses the various options and identifies the current state-of-the-art. Consideration is given to MAS standards and their relation to existing data standards such as the Common Information Model (CIM) [1], and how to best allow interoperability between agents from different designers. Design methodologies are examined, with a brief overview of one example approach to MAS design. Finally, agent anatomy is identified as an area requiring further research, through a description of several anatomies and the technologies they employ, but a lack of comparative information.

Importantly the recommendations presented in this paper are not recommendations for designing and implementing MAS per se but recommendations based on consideration of the application of MAS specifically to power engineering.

II. STANDARDS AND INTEROPERABILITY

The use of standards is important when developing multi-agent systems for power engineering applications. Utilities are striving for increased integration between previously separate systems [2]. Recent standards, such as the power systems CIM [1], which promotes open interfaces between energy management systems from different vendors, and IEC 61850 [3], which promotes interoperability between devices within substations, highlight this point. If the application of MAS technology is to be widespread within power engineering, then the adoption of standards that promote interoperability between systems in the future would be advantageous, if not a necessity.

In recent years, the Foundation for Intelligent Physical Agents’ (FIPA) standards have become the de facto standards used by MAS developers in the computer science community and beyond. In 2005, FIPA was formally accepted as a standards committee of the IEEE Computer Society.
FIPA aims to define specifications and standards that can be used to support interoperability between agent-based systems developed by the different companies and organizations [4]–[10]. These standards impact on not only methods for interagent communication, but also on the basic architecture a multi-agent system should implement.

A. Multi-Agent System Architectures

The FIPA Agent Management Reference model defines “the normative framework within which FIPA agents exist and operate. It establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents” [4]. Under the FIPA model (see Fig. 1), an agent resides on a particular agent platform which provides some sort of message transport system to allow the agents to communicate. FIPA offers standards for the use of certain message transport protocols such as HTTP [5] and IIOP [6].

Each agent platform includes two utility agents: the agent management service (AMS) agent, which is compulsory, and the directory facilitator (DF) agent, which is optional. The AMS acts as white pages, maintaining a directory of agents registered with the MAS platform. The DF acts as yellow pages, maintaining a directory of agents and the services they can offer other agents. An agent can use the DF to search for other agents that can provide services to aid it in fulfilling its particular goals.

Many early multi-agent systems had closed architectures where the specific interactions were effectively “hard wired” at design time. The FIPA Agent Management Reference model, on the other hand, provides an open architecture, i.e., an architecture to which agents can easily be added and removed. In many power engineering applications, this extensibility is one of the key benefits of the use of agents.

B. Agent Communication Languages

Mechanisms for the communication between agents underpin their social abilities. As agent technology has matured, a number of different methods for interagent communication have been developed. Early multi-agent systems, such as ARCHON [11], used proprietary communication languages. Other systems have also used blackboard system-type approaches to enable communication between agents [12].

One of the first agent communication languages (ACL) to be used by different researchers across different fields was the Knowledge Query and Manipulation Language (KQML) [13], which emerged in the early 1990s through the U.S. government’s DARPA knowledge-sharing program. In recent years, KQML has been superseded by FIPA-ACL [7].

FIPA-ACL has its roots in speech act theory and incorporates many aspects of KQML. A FIPA-ACL message contains 13 fields (see Table I). The first and only mandatory field in the message is the performative field that defines the type of communicative act or speech act. By classifying the message using a performative, FIPA-ACL ensures that recipients will understand the meaning of a message in the same way as the sender, removing any ambiguity about the message’s content.

FIPA specifies 22 performatives or communicative acts that define the type of message content and the flow of messages expected by each agent during specific classes of communicative act [8]. Fig. 2 illustrates the flow of messages specified by FIPA for a query-ref interaction. For example, agent A may be interested in the details of distributed generators currently connected to a local MV network. If agent D is responsible for the management of that network, agent A could ask agent D for details of all the cases it knows of where local generators are currently connected, by using the query-ref communicative act.

There are a number of different ways the characters of FIPA-ACL messages can be encoded and sent between FIPA-compliant agents [9].

C. Content Languages and Ontologies

The content of a message comprises two parts: content language and ontology. The content language defines syntax, or grammar, of the content. The semantics or lexicon is drawn from the ontology. The content language and ontology employed are

<table>
<thead>
<tr>
<th>Message field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>performative</td>
<td>Type of communicative act</td>
</tr>
<tr>
<td>sender</td>
<td>Participant in communication</td>
</tr>
<tr>
<td>reply-to</td>
<td>Participant in communication</td>
</tr>
<tr>
<td>content</td>
<td>Content of message</td>
</tr>
<tr>
<td>language</td>
<td>Content language</td>
</tr>
<tr>
<td>encoding</td>
<td>Encoding of content</td>
</tr>
<tr>
<td>ontology</td>
<td>Ontology used</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol for conversation</td>
</tr>
<tr>
<td>conversation-id</td>
<td>ID for conversation control</td>
</tr>
<tr>
<td>in-reply-to</td>
<td>Conversation control parameter</td>
</tr>
<tr>
<td>reply-by</td>
<td>Conversation control parameter</td>
</tr>
</tbody>
</table>
declared in the “content” and “ontology” fields of a FIPA-ACL message, respectively.

FIPA has proposed standards for four different content languages: FIPA-Semantic Language (FIPA-SL); Knowledge Interchange Format (KIF); Resource Definition Framework (RDF); and Constraint Choice Language (CCL). In addition to the content languages above, other content languages, such as Description Logic (DL) and DARPA Agent Markup Language (DAML), are also used. Each of these languages is designed to express different types of information, such as physical concepts such as substations and transformers, and less tangible concepts such as feature vectors.

The choice of content language is important, as the chosen language will affect how a given ontology is expressed. Some multi-agent systems are reported to use the FIPA-SL content language, simply because it is the only one of the four FIPA content language specifications to reach a stable standard: KIF, RDF, and CCL are still experimental and liable to change. Although the FIPA-SL standard has been in existence since 1997, it only became a stable standard in 2002.

The ontology describes the concepts of a domain and the relationship between those concepts in a structured manner. For example, ontologies for use with the Java Agent Development Framework (JADE) contain a class hierarchy of concepts, including predicates, and agent actions. Concepts, as the name suggests, model domain concepts: physical concepts such as substations and transformers, and less tangible concepts such as feature vectors. Predicates specify concept relationships, and can always be evaluated as true or false. An example predicate in a power engineering ontology would be

\[
\text{onCircuit(Circuit, Fault)}.
\]

This could be used to discuss whether a fault occurred on a particular circuit. An action is a special type of concept specifically for communicative acts such as request and call-for-proposal, where agents discuss an event happening. An example action is

\[
\text{Delete(TransformerData)}.
\]

This action could be used to allow agents to discuss the deletion of particular facts from their local data stores. The requirement for particular subclasses of these three will change depending on the communication models employed in a system.

Agents use the ontology for the passing of information, formulating questions and requesting the execution of actions related to their specific domain.

**Recommendation:** When implementing a multi-agent system, if interoperability with other systems is desirable, then standards for basic MAS architecture, agent communication language, and content languages should be adopted. At the time of writing, FIPA standards, described briefly above, are recommended.

### III. Interoperability for Power Engineering Applications

The FIPA standards go a long way in promoting interoperability between multi-agent systems. If different developers adhere to the same set of FIPA standards, then the agents they have developed should be able, at a basic level, to interoperate. Consider agents A and D in Fig. 2. By supporting FIPA standards [4]–[10], agents should be able to discover each other’s existence and then interact. However, while the agents may be able to send each other messages using FIPA-ACL, unless they employ a common ontology, they will not be able to parse and understand the content of the messages they receive.

Currently different developers of multi-agent systems tend to develop their own application-specific ontologies. This leads to different systems using different ontologies. Although the ontologies are different, power engineering systems tend to capture common concepts, such as “substation,” “transformer,” and “circuit-breaker.” The problem is that the way these concepts are represented in the ontologies is different. In other words, the agents speak the same language but do not share a common vocabulary.

#### A. Using Multiple Ontologies

FIPA’s solution to the problem of using multiple ontologies comes in the form of an ontology agent that provides a number of ontology-related services [21]. The list of possible services is given as:

1. locating and accessing public ontologies;
2. maintaining a list of public ontologies;
3. translating expressions between ontologies;
4. providing information about the relationship between two terms or ontologies; and
5. identifying an ontology common to two agents.

There are a number of issues with implementing this solution, not least of which is that the relevant FIPA standard is still experimental. Part of the problem may be that the state-of-the-art in ontology mapping [22] falls short of what is required to automate services 3–5 above.

#### B. Upper Ontology for Power Engineering

In [19], Catterson et al. examined the possibility of integrating two multi-agent systems called PEDa [17] and COMMAS [18]. In order to allow PEDa agents to communicate effectively with COMMAS agents, they had to define the mappings between the two ontologies. As [19] highlighted, this was not a straightforward task.

If the deployment of MAS technology becomes widespread within the power arena and utilities demand inter-operation between systems from different vendors/developers, the cost of the creation of multiple mappings between different ontologies may be prohibitive.

One alternative, which Catterson et al. mooted in [19], would be to create an upper ontology for power engineering applications. The upper ontology would contain the basic concepts of the domain. While not detailed enough for specific applications, the ontology would ensure that different multi-agent systems would employ the same basic representation for common concepts, such as “substation,” “transformer,” “conducting equipment,” and how they are related. Agent developers could use the upper ontology as a starting point for developing ontologies for specific applications. The property of inheritance would ensure that different multi-agent systems would share the same representation for common concepts, reducing the complexity of ontology mapping should it be required. This is shown in Fig. 3.

Currently, there is no standard upper ontology for power systems. However, existing power engineering standards, such as IEC 61850, CIM, and IEC 61400-25 [23], may provide data models that can be used as a foundation for an upper ontology.
Fig. 3. Extending an upper ontology.

Fig. 4. Class hierarchy of part of an upper ontology based on CIM.

Fig. 5. Agent design methodology stages and their output, used during the design of the PEDA system [17].

IV. DESIGNING MULTI-AGENT SYSTEMS

Since the mid 1990s, a number of different methodologies have emerged for the specification and design of multi-agent systems, developing or extending traditional software engineering approaches and knowledge engineering approaches. MAS-CommonKADS [24], for example, extends the CommonKADS knowledge engineering methodology [25]. DESIRE [26], MaSE [27], and Gaia [28], on the other hand, owe more to object-oriented software development methodologies.

MAS design methodologies tend to share some common features: a conceptualization phase where the problem to be solved is specified; an analysis phase; and a design phase that uses the results of the analysis phase to produce agent designs of varying detail.

A. Example Methodology

Fig. 5 illustrates the different stages of the design methodology that McArthur et al. used to specify and then design the PEDA system. Details of the methodology can be found in full in [29]. Each stage of the methodology produces material that is used in the subsequent stages of the design process.
The methodology begins with a structured knowledge engineering stage, specifying the system requirements and capturing the knowledge needed to fulfill those requirements. During the task decomposition stage, the requirements specification and knowledge captured in the previous stage are transformed into a hierarchy of tasks and subtasks. These tasks may include the functions performed by legacy systems. In the case of PEDA, legacy intelligent systems were used to provide data analysis functions. After task decomposition, the ontology can be designed.

The agent modeling stage uses the task hierarchy and ontology design to identify a group of autonomous agents with the abilities to perform the tasks in the task hierarchy. An agent can encapsulate one or more tasks and each of the tasks in the hierarchy must be attributed to at least one agent. The outcome is a set of agent models that specify the tasks the agents should be able to perform. The methodology also identifies the tasks which can be attributed to legacy systems and for which new code needs to be generated.

Once the agents have been identified, the interactions the agents must support have to be defined. These interactions are specified in interaction diagrams similar to Fig. 2.

The final stage of the process is the specification of the interaction functionality of the agent and the control functionality of the agent. This amounts to the specification of the behavior an agent should display.

B. Alternative Design Approaches

The MAS design methodologies referenced above all share one common feature: they begin with a particular problem to solve and specify, to varying degrees, a MAS that will solve and specify, to varying degrees, a MAS that will solve the problem. None of the design methodologies referenced in the previous section offer criteria for the selection of a specific style of agent implementation so that they display the correct levels of reactivity, pro-activeness, and social ability. In some respects, one of the benefits of taking an agent approach is that the way the agents achieve their characteristics is immaterial: an agent can be conceptualized as a black box which sends and receives messages and interacts with its environment in an autonomous manner. However, the practicalities of engineering multi-agent systems means that developers need a working knowledge of the different agent design options or agent anatomies, and the characteristics of the agents with those anatomies.

A. Agent Anatomies

Numerous approaches to building individual autonomous intelligent agents can be found in the literature: Belief Desire and Intention (BDI) agents; reactive agents; agents with layered architectures [30]; and agents implemented using model-based programming [31], to name but a few.

The BDI approach to building agents is based on mental models of an agent’s beliefs, desires, and intentions. There are many different implementations of the BDI approach.

Reactive agents are normally associated with the subsumption model of intelligence. The core property of reactive agents is that they do not perform reasoning through symbolic manipulation; instead they react to inputs from their environment and messages from other agents. Ease of implementation is an advantage of this approach, but the pro-activeness of the agents it produces is arguable.

Several layered agent anatomies are discussed in [30]. As an example, agents developed for the JADE platform tend to consist of three basic layers: a message handling layer; a behavioral layer; and functional layer (see Fig. 6).

The functional layer embodies the core functional attributes of the agent, i.e., the actions the agent can perform. The behavioral layer provides control of when an agent will carry out specific tasks. Should the functional layer produce new data, for example, the behavioral layer will instruct the message handling layer to inform interested agents of the new data. Similarly, the action taken by an agent in response to the receipt of a new message is decided in the behavioral layer.

The message handling layer is responsible for the sending and receiving of messages from other agents, implementing the
relevant ACL and ontology parsers, as well as the functionality for the control of conversations with other agents.

Research by NASA into autonomous spacecraft has also produced some interesting techniques for implementing remote agents, with the aim of displaying a greater degree of autonomy [31]. These agents couple a reactive planner with a model-based reasoning (MBR) engine. The agent has an explicit set of goals and a model of itself. Based on the state of the model, the agent uses its planner to decide the actions it needs to carry out in order to achieve its goal. Should the agent lose its ability to carry out some action, due to a hardware failure or its environment changing in some way, the MBR engine detects this and updates the agent’s model. The planner can then use the updated model to create a new plan of action of how to fulfill its goals.

It may be that as more intelligent, flexibly autonomous agent anatomies are explored, the limitations of the AI techniques which give an agent its underlying intelligence become the barrier in building agents which display the required levels of reactivity, pro-activeness, and social ability. An empirical evaluation of different agent anatomies would certainly help inform the design choice of those developing MAS applications in power engineering.

Recommendation: Currently, there is insufficient information to support a recommendation of any specific agent anatomy. Further research and comparative data are required.

B. Tools for the Implementation of Agents and Multi-Agent Systems

In recent years, both commercial and open source agent development tools have become available [32]. When implementing a multi-agent system, judicious selection of MAS development tools is required. Firstly, the toolset has to comply with the standards to which the developers wish to adhere. Secondly, agents implemented using the chosen toolset must display a level of robustness required for the application at hand.

JADE [20] has become a firm favorite with researchers in power engineering in recent years. While JADE’s support of FIPA standards and the robustness of the agents that can be implemented make it attractive, JADE also promotes a certain style of agent implementation which may not be optimal for exploiting autonomy.

Regardless of the underlying agent anatomies, there is the opportunity to reuse agent designs and functionality for the benefit of the whole community. Therefore, there is a role for toolkits that allow the reuse of existing agent functions, behaviors, and capabilities tuned for applications to power engineering problems. The publication of ontologies may also help reduce the development costs of multi-agent systems and promote interoperability between them.

VI. CONCLUSION

Part I of this paper considered fundamental terms and definitions relating to multi-agent systems technology, and discussed why it is being investigated for a number of power engineering applications. This second part followed on to examine how such a system should be designed and implemented.

Engineering multi-agent systems is complicated by a number of factors: competing standards; difficulties associated with interoperability and ontologies; the choice of a range of design methodologies; and the choice of a number of different agent anatomies and implementation strategies. However, the lack of experience in producing industrial strength multi-agent systems is probably the stumbling block for the technology. With that experience should come better understanding of the effectiveness of different standards, design methodologies, and agent anatomies. This paper provides guidance and information on the state-of-the-art in these technical areas, to aid the uptake of this technology within the power industry.

This paper is based on the successful research and implementation of multi-agent systems by the authors. In addition, it draws upon the experience of the members of the Multi-Agent Systems Working Group, within the Intelligent Systems Subcommittee of the IEEE PES PSACE committee.

ACKNOWLEDGMENT

The authors would like to thank the Multi-Agent Systems Working Group members for their input, discussions, and efforts. The discussions at meetings and panel sessions helped in the creation of this paper.

REFERENCES


Stephen D. J. McArthur (M’93–SM’07) received the B.Eng. (Hons.) and Ph.D. degrees from the University of Strathclyde, Glasgow, U.K., in 1992 and 1996, respectively.

He is a Professor in the Institute for Energy and Environment, within the Department of Electronic and Electrical Engineering. He is Director of the British Energy Advanced Diagnostics Centre and Co-director of the Strathclyde Centre for Applied Intelligent Systems. He currently chairs the IEEE PES Multi-Agent System Working Group. His research interests include intelligent system applications in power, condition monitoring, fault diagnosis, and multi-agent systems.

Euan M. Davidson (M’06) received the M.Eng. degree from the University of Strathclyde, Glasgow, U.K., in 2000.

He is a Research Fellow at the Institute for Energy and Environment, within the Department of Electronic and Electrical Engineering. His research interests include: multi-agent systems; model-based reasoning; and the application of intelligent system techniques to diagnostic, power system control, operation, and management problems.

Victoria M. Catterson (M’03) received the B.Eng. (Hons) and Ph.D. degrees from the University of Strathclyde, Glasgow, U.K., in 2003 and 2007, respectively.

She is a Research Fellow within the Institute for Energy and Environment at the University of Strathclyde. Her research interests include multi-agent systems, agent architectures, and condition monitoring and diagnostics.

Eris L. Dimeas (S’04) was born in Athens, Greece, in 1977. He received the diploma in electrical and computer engineering and the Ph.D. degree from the National Technical University of Athens (NTUA), Athens, Greece, in 2001 and 2006, respectively.

He is currently a Senior Researcher at the Electric Energy Systems Laboratory of NTUA. His research interests include dispersed generation, artificial intelligence techniques in power systems, and computer applications in liberalized energy markets.

Dr. Dimeas is a member of the Technical Chamber of Greece.

Nikos D. Hatziargyriou (S’80–M’82–SM’90) was born in Athens, Greece, in 1960. He received the diploma in electrical and mechanical engineering from the National Technical University of Athens (NTUA), Athens, Greece, and the M.Sc. and Ph.D. degrees from the University of Manchester Institute of Science and Technology, Manchester, U.K.

He is currently Executive Vice-Chairman and Deputy CEO of the Public Power Corporation of Greece and Professor at the Power Division of the Electrical and Computer Engineering Department of NTUA. His current interests include transmission and distribution systems, dispersed and renewable generation, and artificial intelligence techniques applied in power systems.

Dr. Hatziargyriou is member of the EU SmartGrids Technology Platform, of CIGRE SCC6, and of the Technical Chamber of Greece.

Ferdinanda Ponci (M’01) received the M.S. and Ph.D. degrees in electrical engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively.

In 2003, she joined the Department of Electrical Engineering, University of South Carolina (USC), Columbia, as an Assistant Professor. She works with the Power and Energy Research Group, USC, on the research and development of the electric ship: a project that is funded by the U.S. Navy. Her current research interests include agent-based control, integrated environments for distributed measurement, and monitoring and diagnostics of electrical systems.

Toshihisa Funabashi (M’90–SM’96) was born in Aichi, Japan, on March 25, 1951. He graduated in March 1975 from the Department of Electrical Engineering, Nagoya University, Aichi, Japan. He received, in March 2000, a Doctor degree from Doshisha University, Kyoto, Japan.

He joined Meidensha Corporation in April 1975 and has engaged in research and development on power system transient simulation and integration of distributed generation. Currently, he is a Senior Engineer of the Power Systems Sales and Engineering Division.

Dr. Funabashi is a chartered engineer in the U.K., a member of IET, and a member of the IEEE of Japan.