Multi-Agent Systems for Power Engineering Applications—Part 2: Technologies, Standards, and Tools for Building Multi-Agent Systems

S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, T. Funabashi

***Abstract*—This is the second part of a 2-part paper that has arisen from the work of the IEEE Power Engineering Society's Multi-Agent Systems (MAS) Working Group.**

Part 1 of the 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 2 of the 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. The 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 1 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 2) 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?

S. D. J. McArthur, E. M. Davidson, and V. M. Catterson are with the Institute for Energy and Environment, University of Strathlcyde, Glasgow, UK (email: s.mcarthur@eee.strath.ac.uk).

A. L. Dimeas and N. D. Hatziargyriou are with the Power Division of the Electrical and Computer Engineering Department of the National Technical University of Athens, Greece.

F. Ponci is with the Electrical Systems Department, University of South Carolina, USA.

T. Funabashi is with the Meidensha Corporation, Japan.

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 behaviour, 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

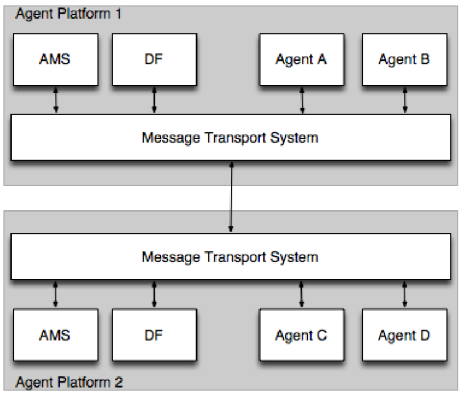

Fig. 1. The FIPA Agent Management Reference model

developed by the different companies and organizations [4]–[10]. These standards impact on not only methods for inter-agent 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 (Figure 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 own 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 inter-agent 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].

TABLE I
FIPA-ACL MESSAGE STRUCTURE

Message field	Description
performative	Type of communicative act
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Content language
encoding	Encoding of content
ontology	Ontology used
protocol	Protocol for conversation
conversation-id	ID for conversation control
reply-with	Conversation control parameter
in-reply-to	Conversation control parameter
reply-by	Conversation control parameter

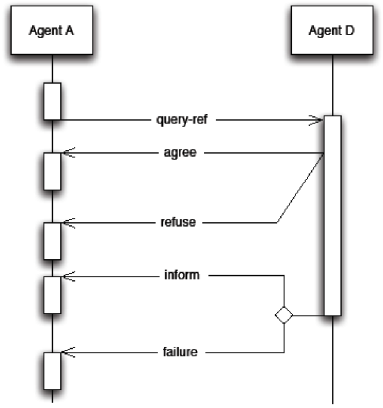

Fig. 2. Agent interaction diagram showing the protocol for the query-ref communicative act

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 US 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 (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]. Figure 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 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) [10]. In addition to the content languages above, other content languages, such as Description Logic (DL) [14] and DARPA Agent Markup Language (DAML) [15], which is closely related to the OWL standards [16] developed by the semantic web community, have found favor with some MAS developers.

The choice of content language is important, as the chosen language will shape how a given ontology is expressed. Some multi-agent systems [17]–[19] 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, RFD 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) [20] contain a class hierarchy of *concepts*, *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:

$$onCircuit(Circuit, Fault)$$

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:

$$Delete(TransformerData)$$

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 [4]–[10], 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 agent A and agent D in Figure 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;
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, 4 and 5 above.

B. An 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.

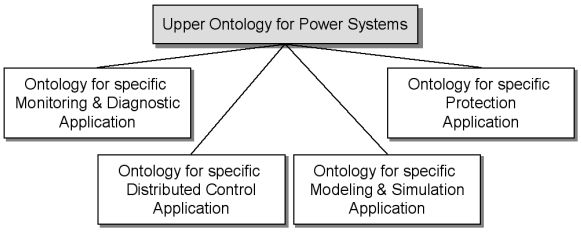

Fig. 3. Extending an upper ontology

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 Figure 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.

Figure 4 shows part of an upper ontology based on CIM. CIM provides a structured class hierarchy of many of the fundamental concepts in power engineering, especially in the description of power system plant and topology. The upper ontology in Figure 4 is not a simple port of the CIM data model. In order to preserve the meaning of the inheritance relationship some aspects of the CIM data model have been modified. For example, in CIM all classes inherit from the Naming class. While all plant items may have a name, inheritance is an "is-a" relationship; a breaker, for example, is not a name and should therefore not inherit from Naming.

Additionally, an ontology distinctly separates concepts from predicates, whereas the CIM data model has to contain all relationships within concept definitions. In CIM, the ProtectionEquipment definition contains an attribute called Operates_Breaker, which indicates which breaker can be operated by each particular protection device. However, in an agent messaging situation this is more correctly modeled by a predicate called Operates, as the operation of the breaker is not a feature of the protection device.

Despite these differences, CIM offers a considered and detailed model of power systems concepts, and is therefore a good starting point for an upper ontology. The plant descriptions are directly appropriate for all four main areas of agent

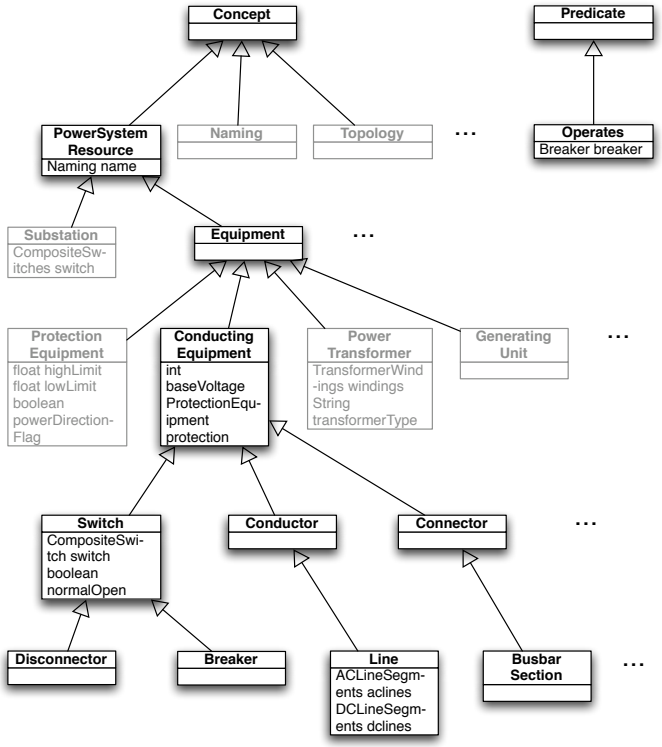

Fig. 4. Class hierarchy of part of an upper ontology based on CIM

research in the power domain: monitoring and diagnostics, protection systems, distributed control systems, and modeling and simulation, allowing the interchange of information about specific plant. Depending on what is being modeled, other CIM packages may serve are starting points for ontologies for those applications. For example, the energy consumer package may help support the ontology requirements of certain modeling and simulation applications.

The full definition of an upper ontology will take time and community support, which will be facilitated by publishing the ontology openly. It will be made available on the Multi-Agent Systems Task Force web site, for community use and comment.

Recommendation: if interoperation with other multi-agent systems is desirable, then the use of a common upper ontology will ease the integration of MAS from different developers. Developers can then extend the upper ontology to include concepts and predicates required for their applications. An upper ontology based on CIM is proposed in this paper and will be made available via the IEEE MAS Task Force website.

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.

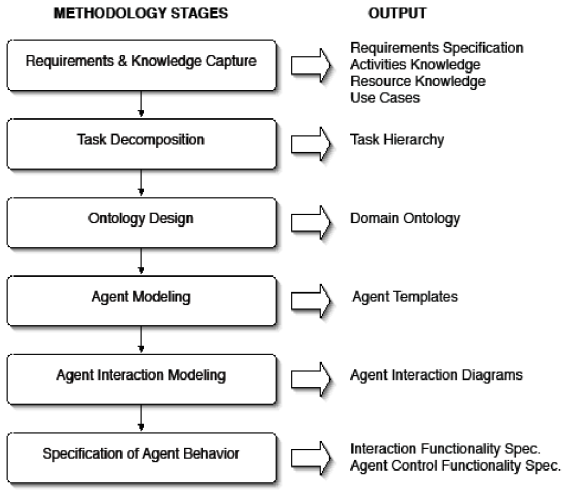

Fig. 5. Agent design methodology stages and their output, used during the design of the PEDA system [17]

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. An example methodology

Figure 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 Figure 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 that specific problem. In such an approach the behavior of the multi-agent system is design-directed rather that emergent, i.e. the MAS designer has built in all the interactions required for agents to fulfill their own goals and, in doing so, meet the design intention of the system as a whole.

What if, on the other hand, other agents are to be added in the future? How do design methodologies support the reuse of existing agents? If multi-agent systems are to be truly open, then interactions of an agent with other future agents should also be considered.

An alternative approach may be to consider agents in isolation. What can an agent know/believe? What actions can an agent take? How do these relate to the ontology the agent supports?

Another point worth considering is the fact that all the methodologies above are standards-agnostic: the choice of standards is not explicitly considered during the analysis and design process. Yet the decision to adhere to a particular set of standards constrains design choices. An agent's communicative abilities, i.e. the communicative acts and interactions it supports, could be determined by knowledge it has and the ability of the ontology to express that knowledge.

The greatest problem with all the methodologies above is the lack of experience in their use. Few of the methodologies have been applied to more than a handful of example applications. Despite this, the methodology shown in Figure 5 contains all the stages required to design an agent system for any specific task, and is therefore a suitable process to follow.

However, such a top-down approach can lead to a rigid agent structure, where agents are less socially able and flexibly autonomous than may be desired. Some consideration should be given to which types of communication are possible for each agent to engage in, rather than specifically what communication is required for the task at hand.

Recommendation: design methodologies provide a structured analytical approach to the design of multi-agent systems. Hence the use of a methodology, such as the one above, is recommended. However, MAS developers should be aware the current methodologies do not guarantee fully flexible and extensible solutions. Careful consideration should be given to the types of communication each agent can engage in, as this is key when promoting flexibility and extensibility.

V. Implementing Autonomous Intelligent Agents

The property of autonomy is often key in the decision to bring agent technology to bear on a particular power engineer-

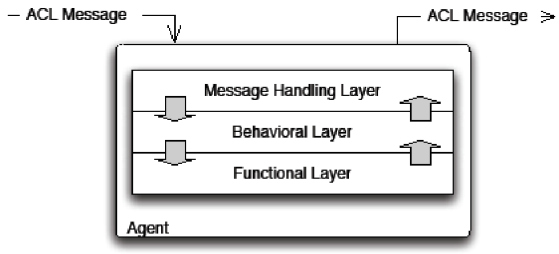

Fig. 6. Layered architecture employed by JADE agents

ing 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 (Figure 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 is 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.

The Java Agent Development Framework (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 re-use agent designs and functionality for the benefit of the whole community. Therefore, there is a role for toolkits that allow the re-use 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. Conclusions

Part 1 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 acknowledge the input, discussions and efforts of the Multi-Agent Systems Working Group members. The discussions at meetings and panel sessions helped in the creation of this paper.

REFERENCES

[1] IEC, "Energy Management System Application Program Interface (EMS-API) - Part 301: Common Information Model (CIM) base," 2005, document IEC 61970-301.

[2] A. F. Vojdani, "Tools for real-time business integration and collaboration," *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 555–562, May 2003.

[3] IEC, "Communications Networks and Systems in Substations," 2005, document IEC 61850.

[4] Foundation for Intelligent Physical Agents (FIPA), "Agent Management Specification," 2002, http://www.fipa.org/specs/fipa00023/SC00023J.html.

[5] ——, "FIPA Agent Message Transport Protocol for HTTP Specification," 2002, http://www.fipa.org/specs/fipa00084/SC00084F.html.

[6] ——, "FIPA Agent Message Transport Protocol for IIOP Specification," 2002, http://www.fipa.org/specs/fipa00075/SC00075G.html.

[7] ——, "FIPA ACL Message Structure Specification," 2002, http://www.fipa.org/specs/fipa00061/SC00061G.html.

[8] ——, "FIPA Communicative Act Library Specification," 2002, http://www.fipa.org/specs/fipa00037/SC00037J.html.

[9] ——, "FIPA Agent Communication Language Representation Specification," 2002, http://www.fipa.org/specs/aclreps.tar.gz.

[10] ——, "FIPA Content Language Specifications," 2003, http://www.fipa.org/repository/cls.php3.

[11] T. Wittig, N. R. Jennings, and E. M. Mandan, "ARCHON — A framework for intelligent co-operations," *IEE-BCS Journal of Intelligent Systems Engineering*, vol. 3, no. 3, pp. 168–179, 1994.

[12] S. Talukdar, "Asynchronous teams: Cooperation schemes for autonomous agents," *Journal of Heuristics*, vol. 4, no. 4, pp. 295–321, 1998.

[13] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language," in *Software Agents*, J. Bradshaw, Ed. The MIT Press, 1997.

[14] A. Borgida, "Description Logics in data management," *IEEE Trans. Knowl. Data Eng.*, vol. 7, pp. 671–682, 1995.

[15] "DARPA Agent Markup Language (DAML)," http://www.daml.org/.

[16] "OWL Web Ontology Language," http://www.w3.org/TR/owl-features/.

[17] E. M. Davidson, S. D. J. McArthur, J. R. McDonald, T. Cumming, and I. Watt, "Applying multi-agent system technology in practice: Automated management and analysis of SCADA and digital fault recorder data," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 559–567, May 2006.

[18] S. D. J. McArthur, S. M. Strachan, and G. Jahn, "The design of a multi-agent transformer condition monitoring system," *IEEE Trans. Power Syst.*, vol. 19, no. 4, pp. 1845–1852, Nov. 2004.

[19] V. M. Catterson, E. M. Davidson, and S. D. J. McArthur, "Issues in integrating existing multi-agent systems for power engineering applications," in *Proc. 13th International Conference on Intelligent Systems Application to Power Systems*, 2005.

[20] "Java Agent Development Framework (JADE)," http://jade.cselt.it/.

[21] Foundation for Intelligent Physical Agents (FIPA), "FIPA Ontology Service," 2001, http://www.fipa.org/specs/fipa00086/X00086D.html.

[22] A. Kalfoglou and M. Schorlemmer, "Ontology mapping: the state of the art," *The Knowledge Engineering Review*, vol. 18, no. 1, pp. 1–31, 2003.

[23] IEC, "Communications for Monitoring and Control of Wind Power Plants," 2006, document IEC 61400-25-2.

[24] C. A. Inglesia, M. Garijo, J. C. Gonzalez, and J. R. Velasco, "Analysis and design of multi-agent systems using MAS-CommonKADS," *Intelligent Agents IV: Agent Theories, Architectures and Languages*, pp. 313–326, 1998.

[25] G. Schrieber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. van der Velde, and B. Wielinga, *Knowledge Engineering and Management: The CommonKADS Methodology*. The MIT Press, 1999.

[26] F. M. T. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Treur, "DESIRE: Modelling multi-agent systems in a compositional formal framework," *Int. Journal of Cooperative Information Systems*, vol. 6, no. 1, pp. 67–94, 1997.

[27] M. F. Wood and S. A. Deloach, "An overview of the multi-agent systems engineering methodology," in *Proc. 1st International Workshop on Agent-oriented Software Engineering*, Jun. 2000, pp. 207–221.

[28] M. Wooldridge, N. R. Jennings, and D. Kinney, "The Gaia methodology for agent-oriented analysis and design," *Journal of Autonomous Agents and Multi-agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.

[29] S. D. J. McArthur, J. R. McDonald, and J. A. Hossack, "A multi-agent approach to power system disturbance diagnosis," in *Autonomous Systems and Intelligent Agents in Power System Control and Operation (Power Systems)*, C. Rehtanz, Ed. Springer-Verlag, Jul. 2003, pp. 75–99.

[30] M. Wooldridge, "Intelligent Agents," in *Multi-agent Systems*, G. Weiss, Ed. The MIT Press, Apr. 1999, pp. 3–51.

[31] B. C. Williams, M. D. Ingham, S. H. Chung, and P. H. Elliott, "Model-based programming of intelligent embedded systems and robotic space explorers," *Proc. IEEE*, vol. 91, no. 1, pp. 212–237, Jan. 2003.

[32] E. E. Mangina, "Review of software products for multi-agent systems," available from http://www.agentlink.org/.