Rational Agents for Decentralized Environments

Jonathan Dale and Apperson Johnson

Quantum Leap Innovations, 3 Innovation Way, Newark, DE 19711, USA

{jd,ahj}@quantumleap.us

Keywords: Multi-Agent Systems, Distributed and Decentralized Systems, Software Interoperability, Service-Oriented Architectures, Robustness, Communications Stack, Ontologies

Abstract

Given the emergence of new and varied energy producers, consumers, and combinations thereof, software processes and services that work on our behalf must adopt the qualities of intelligent distributed systems to address challenges including: local control of processes, local ownership of data, loose coupling and late binding, authentication and non-repudiation, balancing of competition and cooperation, and graceful degradation. Rational agents provide a basis for achieving the robustness and efficiency we seek. Agents can be owned by different organizations, can respect boundaries of authority and proprietary control, and can represent appropriate interests while working in concert with other agents and human operators to achieve common goals. This approach supports dynamic, decentralized detection of both faults and opportunities, and enables persistent online simulation and optimization. Veterans of the pioneering Agentcities project, have demonstrated the importance establishing a multi-layer agent communication stack, realized in abstract, intermediate, and instance levels of concreteness. This approach is especially applicable the emerging SmartGrid, which can promote standard business process descriptions among its membership. The paper presents the rationale of the agent communication stack, its relevance to energy grid participants, and outlines a *Rational Agent* architecture, which provides agent behaviors as services, affording integration with existing and future service-oriented architectures.

1. Challenges of Interoperability

Some of the great challenges of interoperability include: the local control of processes, local ownership of data, authentication and non-repudiation of marketplace entities, achieving a balance of competition and cooperation, determination and implementation of policies and protocols that are fair to participants, all while adhering to both regulated and pragmatic requirements for system robustness.

1.1. Future Generation/Transmission/Consumption: A More Complex Environment

Distributed generation, and the advent of new generation technologies provides potential efficiencies and reduction of the environmental costs of energy. In many cases, a single entity may be both producer and consumer, depending on semi-controllable factors such as demand, and uncontrollable, semi-predictable factors such as local weather. Because of reduced control over the spectrum of generating entities, transmission demands may well be more chaotic than they are today, affected by producer/consumer switching, temperature, overcast, and wind speed. Offsetting some of this variation, monitoring and management of consumption will become pervasive, and endpoints will enjoy additional options in production, consumption, and local energy storage.

1.2. Decentralized Control

The addition of many new parties to the energy grid will introduce new management difficulties, especially for organizations that must maintain spinning reserve to offset potential system failures. Distributed generation facilities will not have the extensive management and business infrastructure of electric utilities, making central coordination impossible. If unexpected changes do occur in the distributed generation landscape, there may not be a human operator to answer the phone, regardless of contractual commitments. This eventuality requires that coordinating entities *at all levels* must maintain models of consumption, production, and reliability, and should continually seek ways to hedge against both physical and economic calamities. Each of these entities will benefit from some sharing of information and models, but it is impossible for a central single entity to obtain complete information about the system, due to both the inherent locality of some variables, and to legal limits of visibility in competitive markets. That said, there is a sizeable opportunity to save money and reduce environmental impact by achieving better and more pervasive control of local energy use. Such control is a pure win for both consumers who avoid high costs, and for producers, who effectively satisfy greater demand with the same capital investment. Pervasive sensors, communication, and multi-level models of energy systems and energy markets can provide both efficiency and robustness.

1.3. Authority, Autonomy, and Discovery in Decentralized Energy Markets

Due to the fact that entities in the new energy environment will play both competitive and cooperative roles, and, because even innocuous information such as the generator maintenance schedules can afford competitors with a pricing advantage, appropriate control of proprietary information is required for maintaining a fair market. Market participants must enjoy a fundamental level of autonomy, but must be able to negotiate that level of autonomy where there is economic benefit. They must be able to lease the authority to control resources and to regain that authority smoothly as leases expire. Additionally, the market itself is likely to become so fluid that any historical directory of participants is at least partially incorrect. Entities must be able to discover markets, opportunities, and other entities *dynamically and opportunistically*, as they become visible and available. Such forms of discovery also permit the rapid reconfiguration of resources, as new generation, storage, and control technologies emerge.

2. Multi-Agent Systems and Service Oriented Architectures

Multi-agent systems (MASs) grew out of early efforts in distributed artificial intelligence, and have become an active area of both research and application. Quoting Katia Sycara[1]: "The characteristics of MAS are that: (1) each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint; (2) there is no system global control; (3) data are decentralized; and (4) computation is asynchronous." Agents within a multi-agent system confront the same fundamental limitations as do humans, that of *bounded rationality*. They may have access to great amounts of information and may have abilities to model, predict, and decide quickly given the current known state, but agents are always acting with only partial information about the world. A MAS plays a role analogous to that of human society; it provides a context for agents to effectively tackle problems that are too large, or too pervasive for any single individual to solve.

Some features of typical MASs include: Use of specialized Agent Communication Languages (ACLs) for inter-agent communication; use of common ontologies to ground the information communicated among agents, use of formal roles for agents to play in a given interaction, explicit interaction protocols to support cooperation, and the use of directory services and subscribe/publish models to permit communication among a changing population of agents. Additionally, recent MAS platforms are typically constructed in layers, providing basic communication at the lowest levels, up to modeling, planning, learning, and, even a degree of introspection at the highest levels.

Many parallels exist between MAS approaches and those of service-oriented architectures (SOAs). Like MASs, SOAs are typically aimed at solving problems in a decentralized environment, often one in which different entities "own" the different components and data involved in the overall process. SOAs use specialized languages for communication, and may subscribe to (formal or informal) ontologies. They often use directory services and subscriptions, have some sense of roles within transactions, and adhere to well-defined protocols. However, unlike MASs, most SOAs are typically aimed at a static problem of constructing a particular, well-defined, persistent application from components. Accordingly, the lifetime of SOA components may be very long, and some components may be relatively monolithic—rather than dynamically emerging to meet demand. Furthermore, SOA components may lack the flexibility and variety of roles that are possible with the elements of a MAS.

3. Rational, Goal-Oriented Agents

Rational, goal-oriented agents provide a basis for achieving the physical and economic robustness and efficiency that is needed in critical infrastructure such as energy systems.

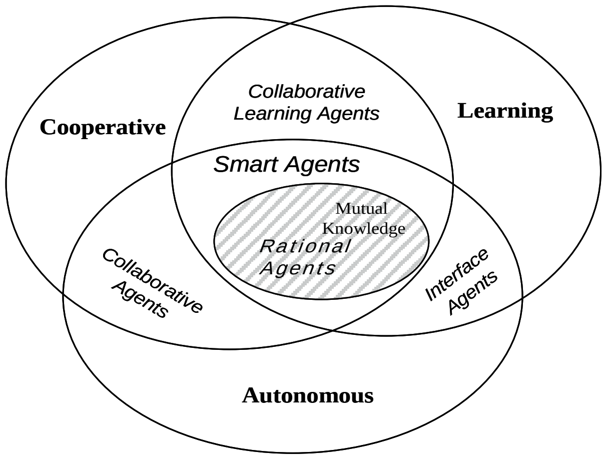


Figure 1. Software Agent Space (adapted from H. Nwana[2])

Figure 1 shows an agent typology derived from an original conceptualization by Nwana in 1996. In this view, the salient features of agent are their abilities to cooperate, to learn, and to behave autonomously. Different emphases among those features provide agents with distinct uses and strengths. The original figure puts "smart agents" at the intersection of autonomous cooperative learning agents, but the typology has recently been extended to include Rational Agents, that maintain mutual knowledge. A key aspect of Rational Agents is the ability to consider models about the domain of interest, including the models of agents, (themselves and others), agent commitments, and agent capabilities. As mentioned before, these models are incomplete and predictions from the agent's models are necessarily imperfect. However, agent systems are constructed to support effective behavior even when individual actions may be in error.

Agents may be owned by different (legal/organizational) entities and can be constructed to act consistently on the behalf of those entities even while acting in collaboration with agents owned by other entities. In some cases, such as electronic markets, legal entities are better served by lending at least temporary authority to agents, which can react quickly to opportunities, rather than hand-reviewing every suggested transaction. In other cases, the sheer information volume that agents may encounter precludes detailed human oversight, and, organizations are best served by reviewing only the salient information gleaned from agent interaction. However, in either case, the internal and private data that is owned by the entities is preserved (ownership boundary) since its divulgence can represent a competitive advantage to other, competing entities. It is the ability of agents to persistently represent the interests of their associated entities, and, to react in reasonable ways to both opportunities and faults, which makes them particularly well suited to the challenges of distributed generation, control, and energy use.

4. Relevant Work from the Agentcities Experiment

Current SmartGrid stakeholders have widely varying goals, capabilities, processes, and terminology that present both challenges and opportunities to interoperability efforts. Veterans of the pioneering Agentcities project, a multi-company interoperability test bed to exercise Foundation for Intelligent Physical Agents (FIPA) standards[3], have wrestled with these problems, and, have demonstrated the importance in establishing a multi-layer agent communication stack, which is realized in abstract, intermediate, and instance levels of concreteness. This approach is especially applicable to the SmartGrid, as standard ontologies and common business process descriptions emerge.

4.1. The Agentcities Interoperability Testbed

The aim of the Agentcities project was to demonstrate interoperability among independently developed agents that followed the FIPA ACL and FIPA models of behavior. Though it appears, on the surface, that the FIPA specifications alone may be sufficient for full interoperability, this proved not to be true. In fact, there are many design decisions and points of potential disagreement that do not arise with agents that are developed by a single monolithic organization. The Agentcities project was able to construct a number of applications from disparate agent components, over the 2001-2004 timeframe, but was only successful after extending the specifications for agent behavior, communication, and interaction beyond the FIPA standard.

4.2. The Agentcities Communication Stack

Tables 1, 2, and 3, (from Dale, et al.[4]), illustrate the three levels of concreteness that were found to be necessary to support effective interoperability among independently developed agents and agent applications. In these tables, the *communication context* defines the relation between elements and the domain in which they are interpreted; the *conversation* describes the exchange of messages that comprise a communication episode; a *message* is one atomic communication item transmitted between agents; *content* is the specific information contained in the message, and, *domain descriptors* are the references to the world model that are used to construct the messages.

Layer	Model Level
Communication Context	Agreement on one or more representations or indicators which determine the environment. This might include, for example, logical representations about world states.
Conversation	Agreement on one or more representations for sequences of messages which can be used to express the structure and semantics of message sequences.
Message	Agreement on one or more communication languages which can be used to express communication messages.
Content	Agreement on one or more content languages which can be used to express states of the world and be embedded in messages.
Domain Descriptions	Agreement on one or more representations that can be used to specify descriptions of domains relevant to a communication episode.

Table 1: Abstract Model Level Definitions

Layer	Intermediate levels
Communication Context	Agreement of the definition of elements that are part of the world and can be used by participants in the communication episode to interpret the meaning of statements.
Conversation	Agreement on a subset of message sequences and conversation patterns which are standard for the environment. These definitions represent commonly used concrete instances of the whole range of possible conversations that could be expressed in the chosen model languages.
Message	Agreement on a set of message types or message templates that are used in the interoperability environment.
Content	Agreement on a set of content types or content templates that are used in the interoperability environment.
Domain Descriptions	Agreement on a set of descriptions for domains that are available in the interoperability environment, for example, an ontology library.

Table 2: Intermediate Level Definitions

Layer	Instance Level for a Single Communication Episode
Communication Context	Use of a set of specific elements of the world that are relevant to a particular communication episode.
Conversation	Use of one or more specific conversation patterns (often only one) used for a particular communication episode.
Message	Use of one or more specific message types used in a particular communication episode. Often this choice is guided by the selected communication pattern.
Content	As for the Message level: Use of one or more specific content expressions in each of the Messages sent in the Communication Episode.
Domain Descriptions	Use of one or more specific domain models in the communication episode.

Table 3: Instance Level Definitions

4.3. Significance of the Communication Stack to Grid Interoperability Standards

Interoperable software systems, regardless of their architecture, must share a common view that permits effective communication. For the simplest static systems, the common view consists of message and data definitions, for more capable systems, there must be agreement about how to interpret metadata, while still more capable systems require commonality among models and model elements. Industry standards bodies are uniquely positioned to create standard ontologies that facilitate the semantic levels of communication among components. These ontologies provide metadata about all relevant referents in the domain of discourse and formally describe the relations between concepts in that domain. Without unified, standard ontologies, groups will invariably develop partial or ad-hoc conceptualizations of domain elements, and those domain models will very likely be incompatible, and may ultimately become a barrier to the composition of services from existing capabilities. Content and domain descriptions within the agent communication stack can be grounded in these formal ontologies, enabling decoupled systems to cooperate in both persistent and occasional applications.

5. HERMES: A Rational Agent Platform

HERMES is a recently developed multi-agent platform that provides agent behaviors as services, affording integration with existing and future SOAs. It is constructed in layers that extend from basic message transport, up to high-level planning and domain-modeling. The HERMES platform is a conceptual descendant of several pre-existing agent approaches and standards, including RETSINA[5], DECAF[6], FIPA[7], and JADE[8].

5.1. Wrapping Services with Agent Behavior

SOAs have recently emerged from a long evolution through completely proprietary IT systems that required end-to-end uniformity and single-vendor solutions, to solutions constructed from a few prime subsystems involving multi-year $MM integration efforts, to today's component-ware, in which service providers, service brokers, and service requestors may be assembled rapidly to meet emerging business needs.

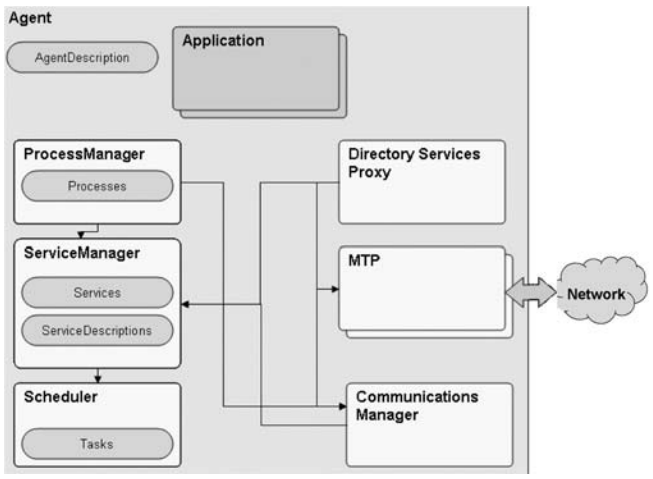


Figure 2: Principle Features of an Individual Hermes Agent

As illustrated in Figure 2, HERMES agents can offer their capabilities as applications in a standard service-oriented environment, and can, conversely, "agentify" existing SOA components.

5.2. Distributed Processes in HERMES: Explicit Models of Interaction

An agent platform provides more than individual agent interfaces. It also provides mechanisms that support agent collaboration to accomplish tasks. In the HERMES environment, a useful abstraction for such collaboration is the multi-agent process, which is typically accomplished by several agents playing specialized roles within a defined interaction protocol. To facilitate the definition of these multi-agent processes, HERMES provides a process language, which is an extension of the pre-existing Business Process Execution Language (BPEL), and offers a process editor for visually constructing processes and for monitoring their execution.

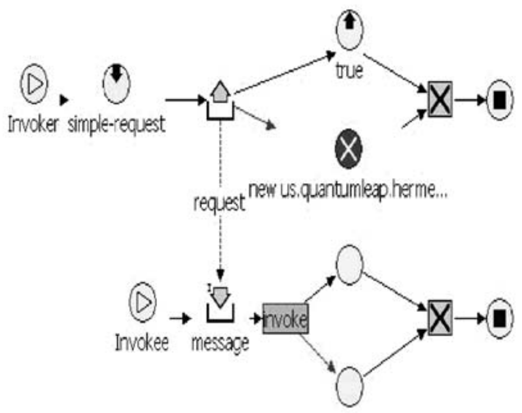
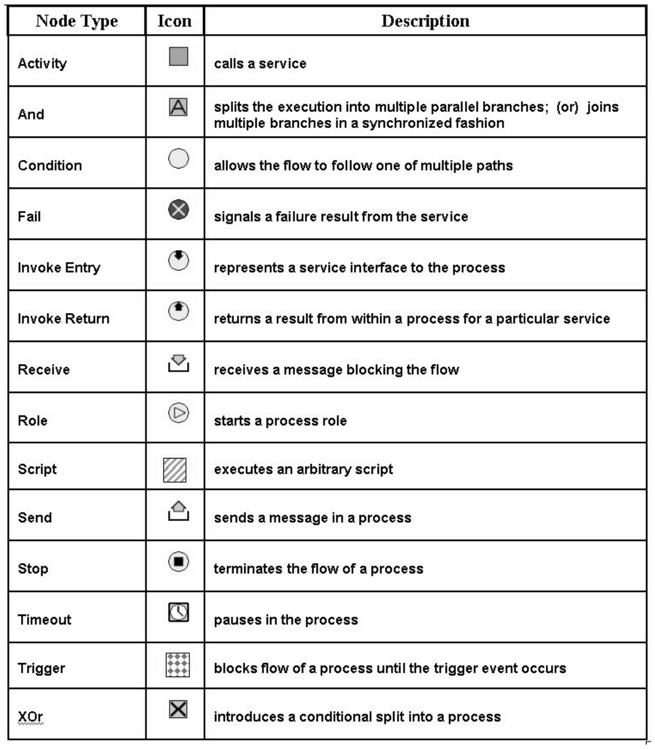


Figure 3: The Simple-Request Process Interaction Diagram

Figure 3 illustrates a simple-request process, as constructed in the HERMES process editor, which uses the Eclipse Rich Client Platform. This process can be invoked, tested and monitored within the editing framework, enabling rapid development and debugging of complex, multi-agent applications.

Figure 4 lists the process primitives that are used to construct multi-agent processes. To date, the process editing environment has been used to define several important interaction protocols, including: *fipa-request*, *centralized-execution*, *simple-request* and *contract-net*. Extensions of these protocols may be particularly relevant to the challenges of scheduling resources to satisfy physical energy demands efficiently in a competitive environment[9,10]. Additionally, beyond bidding-style protocols to perform allocations, agents with sufficient models of market costs and projections of ongoing demands can play the role of market-makers for efficiently brokering the best matches on the basis of local market optimality[11]. This approach can be cascaded to multiple levels of a hierarchy, with brokers at each level acting persistently to optimize the value of the market that it manages.



Node Type	Icon	Description
Activity	■	calls a service
And	▣	splits the execution into multiple parallel branches; (or) joins multiple branches in a synchronized fashion
Condition	○	allows the flow to follow one of multiple paths
Fail	⊗	signals a failure result from the service
Invoke Entry	⊙	represents a service interface to the process
Invoke Return	⊙	returns a result from within a process for a particular service
Receive	⊻	receives a message blocking the flow
Role	▷	starts a process role
Script	▨	executes an arbitrary script
Send	⌂	sends a message in a process
Stop	◙	terminates the flow of a process
Timeout	◷	pauses in the process
Trigger	▦	blocks flow of a process until the trigger event occurs
XOr	⊠	introduces a conditional split into a process

Figure 4. Icons and Semantics for the Hermes Process Editor

5.3. Pro-activity Amongst Rational Agents (PARA)

PARA is a high-level planning, reasoning, and decision-making capability that runs on top of HERMES to model complex and dynamic distributed systems. It provides an event-calculus-based[12] planning engine that can be deployed as a component within a HERMES agent. The logic framework extends standard SOAs by wrapping services of an agent with meaningful descriptions (meta-data) that specify the service pre-conditions and effects within a formal model of the application domain. The driving force behind an agent's activity is a set of goals, which can be triggered by events in the agent's environment (sensor data, communications), or can be given by other agents or users. Each agent is able to autonomously determine a sequence of actions to achieve their goals and each goal or action can be related to information and resource requirements. A PARA agent is able to continually evaluate its environment and adjust and prioritize its actions as a result of new events so that they are in line with its existing set of goals. High-level cooperation is enabled by sharing goals and establishing commitments to goals among agents, which allow agents to coordinate activities without sharing the details of how each agent will achieve its goals. At the same time, PARA agents retrieve information and secure resources that are required to accomplish their goals. PARA provides novel extensions to traditional event-based planning in that it is *reentrant*, supporting continual re-planning, and, that it reasons about *concrete time-points,* permitting agents to synchronize activities without recourse to polling or other costly control mechanisms.

6. Conclusion

We have presented arguments that an agent-view of services can support much of the flexibility, robustness, and configurability demanded by the emerging distributed-generation and demand response environment. To get the most out of agent platforms, there must be broad agreement on both the software interfaces that permit interaction, and on the semantic grounding that supports shared models about the domain and domain participants. Ongoing interoperability among separately developed dynamic components is particularly challenging and requires powerful tools both for construction and for monitoring of the resulting processes. We are approaching a point where every energy component, from refrigerators, to home generators, to distant nuclear reactors, will be accessible to some level of pervasive monitoring and control. That control will not be effective unless there is consistency among the models of system participants. Multi-agent communication and coordination approaches, coupled with standard ontologies, can catalyze standardization in both programmatic interfaces and in shared conceptual models that are a prerequisite for interoperability.

Acknowledgements

Portions of the system described in this paper were developed with funding by the United States Office of Naval Research (ONR), under contract No. N00014-08-C-0036. Much of the Rational Agents system and documentation was conceived and developed by Michael Kolb and Dave Cleaver, of Quantum Leap Innovations, Inc. under the ONR Contract.

Biographies

Jonathan Dale is Director of Distributed Systems at Quantum Leap Innovations. With 15 years of experience in Multi-Agent systems, Web Services and the Semantic Web, including work as a Senior Researcher with Fujitsu Laboratories of America, Jonathan's research efforts have focused on the next generation of agent–based technologies for the Internet. He has been actively involved in the work of many industrial standards organizations, such as the World Wide Web Consortium (W3C), the Foundation for Intelligent Physical Agents (FIPA) and the Agentcities Task Force (ACTF). Jonathan's work has become a significant contribution to standards bodies has been written into the official specifications in for agents, Web Services and the Semantic Web. A co-founder of the Agentcities initiative and member of the FIPA Architecture Board, Jonathan has been working within the Global GRID Forum (GGF) as part of the Semantic GRID workgroup. He has published numerous papers in journals, magazines, conferences and workshops within his field and has written a number of patents which are currently being reviewed by the US Patent Office. Jonathan earned his PhD in Computer Science at the University of Southampton, UK, and his BS with Honours in Computer Science at Staffordshire University, UK.

Apperson H Johnson is Chief Science Officer at Quantum Leap Innovations. Apperson Johnson has been working in applied Artificial Intelligence since the early 1980's, having been a principal author of a natural language interface system, several inference systems, several knowledge representation systems, an object oriented modeling system, a robust scaleable optimization system, and a system to support automatic markets for complex products. A Summa Cum Laude graduate from the Chemistry-Biology program in West Chester State University, he also holds a Master's degree in Computer Science from the University of Delaware, where he occasionally teaches. Apperson is the chief technical author of 3 issued US patents and has 19 patents pending.

References

[1] Sycara, K, "MultiAgent Systems". In: *AI Magazine 19(2)*, pp. 79-92, 1998.
[2] Nwana, H, "Software Agents: An Overview". In: Knowledge Engineering Review 11(3), pp. 205-244, 1996.
[3] Dale, J and Mamdani, E, "Open Standards for Interoperating Agent–Based Systems". In: Software Focus, 1(2), John Wiley and Sons, 2001.
[4] Dale, J., Willmott, SN., and Burg, B., "The Agentcities Initiative: Connecting Agents Across the World". In: *Innovative Concepts for Agent–Based Systems*, Truszkowski, W, Rouff, C and Hinchey, M G (Eds.), Lecture Notes in Computer Science (2564), pp. 453–457, Springer–Verlag, 2003.
[5] Sycara, K., Giampapa, J., Langley, B., and Paolucci, M., "The RETSINA MAS, a Case Study," In: *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications*, Alessandro Garcia, Carlos Lucena, Franco Zambonelli, Andrea Omici, Jaelson Castro, ed., Springer-Verlag, July 2003, pp. 232-250.
[6] Graham, J. and Decker, K., "Towards a Distributed, EnvironmentCentered Agent Framework". In: *N. Jennings and Y. LeSperance (eds.): Intelligent Agents VI Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, LNAI. Springer-Verlag, Berlin, 2000
[7] Flores-Mendez, R., "Towards the Standardization of Multi Agent Systems Architectures: An Overview", In: *ACM Crossroads - Special Issue on Intelligence Agents, Vol. 5 (4),* ACM Press, Summer, 1999
[8] Pinsdorf, U., and Roth, V., "Mobile Agent Interoperability Patterns and Practice", In: *Proc. of 9th Annual IEEE Int'l Conf. and Workshop on the Engineering of Computer-Based Systems (ECBS)*, Apr. 8-12, Lund, Sweden, 2002
[9] Sandholm, T and Lesser, V, "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework". In: *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 328-335, San Francisco, 1995.
[10] Waldspurger, C, Hogg, T, Huberman, B, Kephart, J and Stornetta, W, "Spawn: A Distributed Computational Economy". In: *IEEE Transactions on Software Engineering 18(2)*, pp. 103-107, February 1992.
[11] Yarom, I, Rosenschein, JS and Goldman, CV, "The Role of Middle-Agents in Electronic Commerce". In: *IEEE Intelligent Systems 18(6)*, pp. 15-21, November/December 2003.
[12] Shanahan, M, "The Event Calculus Explained". In: *Springer Verlag, LNAI (1600)*, pp. 409-430, 1999.