An Architecture Analysis of Agent-based Systems

Nenad Ivezic[1,3], Mario Barbacci[2], Don Libes[3], Tom Potok[1], and John Robert[2]

[1] Oak Ridge National Laboratory, PO Box 2008, Oak Ridge, TN 37831, USA
{ivezicn, potokte}@ornl.gov
[2] Software Engineering Institute, 4500 Fifth Avenue, Pittsburgh, PA 15213, USA
{mrb, jer}@sei.cmu.edu
[3] National Institute of Standards and Technology, 100 Bureau Dr., Gaithersburg, MD 20899, USA
{nivezic, libes}@nist.gov

Abstract. In this paper, we describe a principled methodology for analysis of agent-based systems architecture. We illustrate the use of this methodology on an example of a supply chain management system. Using the methodology, it is possible to assess architectural decisions on software qualities such as performance, modifiability, and security. The focus for the methodology is risk identification, evaluation, and mitigation in the early stages of system design. We describe a test bed to analyze and evaluate agent-based systems in manufacturing enterprises that is based in part on this methodology.

1 Introduction

In this paper, we describe an analysis and evaluation methodology for agent-based systems that addresses issues of using agent technologies in complex software systems. We illustrate the use of this methodology on a supply chain management system. The methodology is applicable to the early agent systems architecture analysis and evaluation phase. The primary focus for the methodology is risk identification, evaluation, and mitigation in the early stages of system design. Architectural decisions on software qualities such as performance, modifiability, and security can be assessed. Also, we describe a test bed to analyze and evaluate agent-based systems in manufacturing enterprises that is in part based on this methodology.

2 An Agent-based Supply Chain Design Example

We have been investigating agent technologies in support of supply chain management (SCM) systems [5,6,8]. Figure 1 presents an early, conceptual agent architecture for a SCM system in support of large-scale manufacturing enterprises. (We implicitly specify an agent-based architecture by talking about agent roles within the architecture.) The major goals for such an architecture are the following:

– to understand the demand of the customers on the suppliers,
– to understand capabilities of suppliers to meet demands, and
– to make the suppliers' production processes more cost effective.

The following text discusses the roles and interactions in the supply chain design.

2.1 Supply Chain Roles

In Figure 1, the customers (blocks labeled 'C') form an alliance among themselves and suppliers (blocks labeled 'S') to achieve an effective, extended enterprise through interactions at the Global Procurement Center (GPC). As part of the customer's role, a procurement center (block labeled 'PC') is responsible for interactions with the Global Procurement Center.

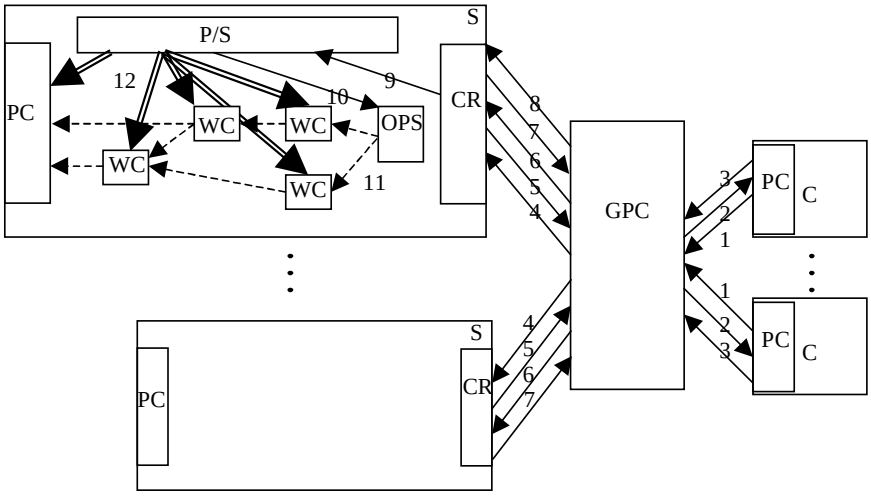

Fig. 1. The proposed supply chain management system architecture shows the roles and interactions among these roles: *Customer Representative (CR) responsible for the customer's interests; Planning and Scheduling (P/S) detailing customer orders fulfillment; Manufacturing Operations (OPS), responsible for operations; Manufacturing Work Center (WC), representing the work areas; and Procurement Center (PC), where raw materials are purchase*

Note that a supplier may assume the role of a customer with respect to it's own suppliers. Therefore, this architecture could be further extended to second- and third-level suppliers, representing the entire supply chain.

2.2 Supply Chain Interactions

The following are the interactions centered at the Global Procurement Center (GPC):

- The customers (i.e., their procurement centers labeled ‘PC’) send information about their needs for different types of products to the GPC. (Arrow labeled ‘1’.),
- The GPC issues request for demand specifications to every customer for a specified product. (Arrow labeled ‘2’.)
- Each customer procurement center (PC) provides demand specifications to the GPC (e.g., about preferred suppliers). (Arrow labeled ‘3’.)
- The GPC issues request for capacity information (related to specific products) to the suppliers (i.e. their customer representatives ‘CR’). (Arrow labeled ‘4’.)
- Each supplier provides capacity for each product delivery to the GPC (Arrow labeled ‘5’.)
- The GPC sends a request for bids for a ‘bundled’ collection of products according to the extended enterprise grouping methodology. (Arrow labeled ‘6’.)
- Each supplier bids on the requested task. (Arrow labeled ‘7’.)
- The GPC awards the task to one of the bidders. (Arrow labeled ‘8’.)

Following the task award to the winning supplier, the focus of the system interactions shifts to that supplier. The following are the interactions centered at the supplier:

- The supplier’s customer representative (CR) ‘unbundles’ the awarded task and sends the sub-task specifications to the planner/scheduler. (Arrow labeled ‘9’.)
- The planner/scheduler (P/S) determines the strategy for the task assignments and provides guidance to the manufacturing operations. (Arrow labeled ‘10’.)
- The planner/scheduler may issue work assignments to work centers (WC) (Double-line arrow types labeled 12) or the operations can send a signal to the work centers that propagate upstream (Dotted arrow labeled 11.)
- The supplier’s procurement center (PC) is contacted with the demand specification. (Arrow types 11 and 12 pointing to supplier PC.)

3 Architecture Tradeoff Analysis Method for Software Systems

In large software systems, software qualities such as performance, security, and modifiability depend not only upon code-level practices (e.g., language choice, detailed design, algorithms), but also upon the overall software architecture. A variety of qualitative and quantitative techniques are used for analyzing specific quality attributes [1]. These techniques have evolved in separate communities, each with its own vernacular and point of view, and typically have been performed in isolation. However, the attribute-specific analyses are interdependent; for example, performance

affects modifiability, availability affects safety, security affects performance, and everything affects cost.

The Architecture Tradeoff Analysis Method (ATAM) is an evolving architecture evaluation technique [7,10]. The input to the ATAM consists of system architecture and the perspectives of stakeholders involved with that system. The ATAM relies on generating scenarios to assess the architecture, where the scenarios are stimuli used to represent stakeholders' interests and to understand quality-attribute requirements. The scenarios give specific instances of anticipated use, performance requirements or growth, various types of failures, various possible threats, various likely modifications, etc.

ATAM, by itself, does not provide an absolute measure of architecture quality; rather it serves to identify scenarios from the point of view of diverse stakeholders (e.g., the architect, developers, users) and to identify risks (e.g., inadequate performance, successful denial-of-service attack) and possible mitigation strategies. The results of an ATAM evaluation reflect the concerns of stakeholders and the scenarios that they consider important. In section 4, we identify typical quality attributes, scenarios, and agent-system architectural patterns as a way to expedite evaluations of real agent systems.

4 Architecture Tradeoff Analysis Method for Agent Systems

Very broadly, agents may be thought of as software entities with the ability to undertake action autonomously in a specific embedded environment. Typically, the autonomous behavior is performed according to a general set of requests or desired goals. Additionally, agents are considered capable of communicating with other agents as determined by their own initiative. In this section, we illustrate the nature of the architecture decisions and their impact on various quality attributes of agent systems.

4.1 Quality Attributes of Agent Systems

A number of publications on the subject of agent-based systems contain typical design patterns for collaborating agents [2,3]. However, critical aspects of evaluating agent-based systems in terms of software quality attributes are not presented. We see the following set of attributes as particularly relevant to agent architecture evaluation:

- *Performance predictability*. Since agents have a high degree of autonomy, it is difficult to predict, from a high level of abstraction, individual agent characteristics (e.g., timeliness and latency) when determining the behavior of a system at large.
- *Security against data corruption and spoofing*. Agents are free to identify their own data sources (e.g., Web agents). In addition to possibly misleading information acquired by agents, there is the distinct possibility of hostile agents attempting to

spoof the system agents. The protocols of verifying authenticity for data sources by agents are a concern in the evaluation of system quality.
- *Adaptability to changes in the environment*. Agents may be required to adapt to modifications in their environment, such as changes to the agent's communication language.
- *Availability and fault tolerance*. Agents that offer services to other agents may make promises about the availability of the offered services. A related attribute is the ability of the agent system to detect, contain, and deal appropriately with external agents such that erroneous inputs are not propagated into the agent system itself.

4.2 Space of Architecture Choices

Typically, agent-system taxonomies focus on functional properties of the agents. As functionality increases, proliferation of agent "types" mushrooms; confusion is inevitable because of unavoidable overlaps and subtle distinctions.

Instead of attempting to conduct attribute tradeoffs based on ever-changing taxonomies of styles, we take a different approach [10]. First, we do not look at the functionality of the agents in the system, rather we concentrate on external properties such as communication, cooperation, and coordination actions performed by the agent-system components and connections. This defines our "architecture" domain. Second, we look at the instantaneous architecture (i.e., components and connections and the patterns of behavior among the components). Third, we identify a "central" component and conduct the analysis with this component as the focus of the interaction with the other components. This simple approach seems to describe a large variety of agent systems [10].

Specifically, we look at the number of inputs/outputs to/from the focus agent, the message processing actions (bundling, unbundling, translation) of the focus agent, and other actions specific to the focus agent. Hayden makes a good first step in this direction, but unfortunately still focuses on a "catalog" of coordination models with the inevitable overlaps and subtle distinctions that we have observed in other taxonomies [3].

In the following, we illustrate part of the architecture space with example agent patterns such as wrapper, matchmaker, broker, mediator, and contract-net.

4.2.1 Wrapper Agent

A wrapper agent allows a legacy application to be incorporated into a multi-agent system (MAS). The legacy code is extended with agent capabilities by "agentifying" it [3]. The wrapper allows agents to communicate with the legacy system using an agent communication language (ACL) by acting as a translator between the agents and the legacy system. This ensures that agent communication protocols are respected and the legacy system remains decoupled from the agents. The legacy system should be capable of supporting any of the requests, commands, or activities possible in the agent communication language or provide a graceful denial. The wrapper can examine the

traffic and filter the messages if necessary. A "top-of-the-line" wrapper would have a rather rich set of capabilities:

- it can bundle multiple agent messages into one message to the legacy system (e.g., the wrapper understands the purpose of the individual requests and "packages" them to match the functionality of the legacy),
- it can unbundle agent messages into multiple messages to the legacy system (e.g., the agent request might be too complicated for the legacy system to handle and the wrapper breaks it into pieces matching the real functionality of the legacy), and
- it can translate a message between the ACL formats and the format of the commands understood by the legacy system.

Not surprisingly, one could imagine wrappers with more modest capabilities. But aside from cost and development time, why would anyone want a simpler wrapper? The answer is directly related to the quality attributes of the architecture. For example, a wrapper that does not bundle agent messages might generate extra requests of the legacy system, but the latency of the individual responses to the agent might be shorter. In addition, a wrapper that does not unbundle agent messages would not leave the legacy system in some inconsistent state should the wrapper crash in the middle of the sequence. Finally, a wrapper with limited translation capabilities might have to reject complicated agent messages but will run faster for the acceptable messages.

4.2.2 Matchmaker, Broker, and Mediator Agents

The matchmaker, broker, and mediator agents are closely related. These agents act as intermediaries between a number of agents providing services (the providers) and a number of agents requesting services (the consumers) [2,3].

The matchmaker locates a provider corresponding to a consumer request for service and hands the consumer a handle to the chosen provider. Thus, the negotiation for service and actual service provision are separated into two distinct phases. The broker, on the other hand, directly handles all interactions between the consumer and the provider. The mediator acts like the broker with the following difference: a broker simply matches providers with consumers; a mediator encapsulates agent interactions and maintains models of behaviors over time (these are the bundling/unbundling actions).

The broker presents a tradeoff with the matchmaker in that the broker insulates the consumer and provider. Thus the broker protects the provider from hostile or unruly consumers at the expense of a doubly tasked broker component whose failure is crucial to any brokered interactions that are ongoing.

The matchmaker limits its failure impact to the negotiation phase since previously negotiated interactions are undertaken between the consumer and the provider. This flexibility, however, is provided at the expense of revealing provider identities to consumers. The matchmaker interaction in its simplest form does not guarantee secure interactions. Nevertheless, additional negotiation steps can be imagined (key exchanges, for instance) that would secure interactions at the expense of additional communication and computation overhead.

The advantage of a mediator is that it further decouples the agents: individual agents do not have to maintain models of behavior for all other agents around, and the

mediator allows each agent to vary its behavior independently. Agents need to be aware of only the mediator. The mediator pattern works if the interactions between agents are well defined, even if coordination of distributed behavior is desired (i.e., the mediator must know the valid sequences). Cooperating agents send and receive requests to/from the mediator, which routes the request in accordance with a specific conversation model. The disadvantage of this pattern is that the mediator becomes a performance bottleneck. This can be remedied as in the broker pattern, by having multiple mediators that coordinate their activities.

4.2.3 Contract Net

Finally, lets consider a contract-net agent system [9]. In contract-net, a manager agent issues a request for proposals (RFP) or bids for a particular service to all participating contractor agents. The manager then accepts "proposals" to meet the service request at a particular "cost" (or messages indicating the contractors' unwillingness to bid). The manager selects among these proposals and indicates acceptance to exactly one bidder. Optionally, the manager indicates to non-successful bidders that their bids have been rejected. The selected contractor performs the contracted work and informs the manager upon completion. Many of the same tradeoffs identified above reappear in the contract net. The manager agent works on behalf of a client and isolates the client from the contractors and provides additional protection by conducting the negotiations and the delivery of the contracted work (mediating). The down side is that the manager could be a performance bottleneck and a tempting target for an attacker.

4.3 Scenarios for Agent-Based Systems

We devise scenarios for various agent types to identify and resolve potential risks. For example, agent systems are susceptible to attacks in which an outside party intervenes in an inter-agent exchange. Attacks that require explicit knowledge of the server by the client or knowledge of the client by the server are prevented in the broker agent. On the other hand, the existence of a specific intervening agent in the communication between the client and the server might have an impact on performance (increase in latency), availability (single point of failure), and even security of information since there are now two separate communication channels to be protected. While this is unavoidable in some cases, for agent-to-agent communication it is quite possible that public key encryption can help in the positive identification of an agent prior to acceptance of a particular message. Such choices are the nature of architectural analysis. At any rate, the scenarios listed below are indicative of the considerations used in evaluating architectural properties for collaborative agents.

Scenario 1: Attacker (Consumer) Issues Massive Requests for Services. This scenario may affect performance and availability, with the possible denial of service (a broker for all services implies that a successful attack destroys all service allocations.) Possible solutions include:

- A multiple-broker solution would address this issue; however, it requires consumers to have an elaborate understanding of the brokers and their specialties, if any — a robustness/availability versus complexity/overhead tradeoff.
- Registration of the consumer with the broker and provider could allow the broker to limit the frequency of consumer requests. (For certain services and/or agent systems, this may be appropriate and could alleviate this kind of attack risk.)
- Provider is restricted to a single request type to the same consumer in a given time frame.
- All consumer/provider interactions are monitored to limit the consumer actions to expected values, including volume of messages.
- The matchmaker issues only single-provider addresses to the consumer in a given time frame.
- The matchmaker notifies the recommended provider of requests by the consumer in order to make service available for a limited time only.

Scenario 2: Provider Fails After Advertising Service. This scenario might affect availability of service, performance of the consumer, and reliability of the consumer. If the matchmaker database contains information about non-available services, a consumer can obtain a handle to an unavailable service. If the consumer blocks while waiting for the service, the system might hang. Solutions include:

- The matchmaker regularly pings service providers as assurance of presence.
- The matchmaker requires regular re-advertisement of service.
- The consumer times out and notifies the matchmaker.

Scenario 3: Spoofer (Customer) Acquires Service and then Seeks to Replace It. This scenario might affect the security of the service if the attacker spoofs the provider, reliability of the service, and performance of the provider. As a consequence, the matchmaker database may contain erroneous provider/service information, and a customer could obtain an apparently valid provider and receive an erroneous result. Alternatively, a spoofed provider could deliver an apparently valid response on contract and the consumer could expect contracted service and not receive it. Alternatively, the spoofed provider could fail to reply to a contract request and the consumer hangs. Solutions include:

- Do not provide explicit provider process or port information to the consumer.
- Use public key encryption to verify identity of the provider.

Scenario4: Attacker (Provider) Unadvertises Legitimate Provider/Service. This scenario may affect performance and availability. It is a denial-of-service attack by turning off provider access to all consumers. A solution is for the matchmaker to use public key encryption to enforce same-key advertise and unadvertise performatives.

Scenario5: Spoofer (Provider) Advertises False Service This scenario might affect availability. The provider may respond to service requests with false results or a provider may simply not answer requests with contracted response. Possible solutions include:

- The matchmaker periodically verifies service by issuing a request and observing response and server behavior.
- The consumer reports the provider performance to the matchmaker, which may deregister the offending provider.

5 An ATAM Analysis of the Agent-based Supply Chain Design

A complex agent system would rarely fit neatly into one of the patterns or styles illustrated previously. In practice, agent systems are more likely to consist of combinations of these patterns: sometimes nested and sometimes including non-agent systems. To apply the appropriate Architecture Tradeoff analyses, it is necessary to decompose the system into recognizable building blocks.

The Supply Chain Management Systems example from Section 2 seems a rather complex system but its behavior can be explained in simpler terms by recognizing several of the primitive agent patterns illustrated before. For example, the procurement centers (PC) act as mediators between the customer (C) and the global procurement center (GPC). Furthermore, the PC could also include wrapper features if the customer is a legacy system.

The general procurement center (GPC) seems to play a standard contract-net manager role between a "client" and the suppliers (S), except that it is not clear who is the client. The GPC as described seems to incorporate two different agent systems and it might be necessary to separate them to simplify the analysis. First, there is a subagent within GPC that plays the role of a broker for the collection of clients, GPC-Broker. The second subagent is a contract-net manager proper, GPC-Manager. GPC-Broker gets requests from clients (actually, the PC's mediating/wrapping the real clients, C) looking for a contract-net manager. Thus, message 1 has two parts -- (1) a request for a manager and (2) the information about the cleint's needs, assuming a manager is available. GPC-Broker contacts GPC-Manager who then interacts with the clients to build the demand specifications (messages 2 and 3). After the GPC-Manager has obtained the demand specifications, it issues a request for capabilities to the known suppliers (message 4).

The supplier, as suggested in the figure, is a complex agent system, incorporating various building blocks. The customer representative (CR) component of a supplier plays two roles. First it negotiates with GPC-Manager on behalf of the supplier (messages 5 through 8) and later, if it wins a contract, it becomes a 'client" of the Planner/Scheduler (P/S).

The Planner/Scheduler acts as a mediator between the customer representative (CR) and the operations and work centers (OPS and WC boxes), the mediation activity in this case consists of scheduling the various tasks requested by the customer representative (message 9). Depending on the schedule adopted, the behavior of the operations and work centers can range from strictly sequential to a mix of concurrent operations synchronized at appropriate time, as suggested by the supply chain design.

The nature of the contract-net established by the GPC, CR, and PC agents would suggest that availability and security might be more important than performance. To mitigate this risk, it may be necessary to use, fault-tolerant services and to prevent faults (accidental or intentional) from crashing the GPC and to prevent denial-of-service attacks. At the same time, security services might be required to prevent disclosure or tampering of the information exchanged between the customers and the GPC. These services are likely to degrade performance but this might be an acceptable price to pay. Inside the supplier we have a different situation. If we assume that

the internal components are under control of a single organization, one could expect that certain types of security issues would be less critical (e.g., using a firewall around the complex and allowing faster, unencoded internal messages) while availability and performance are more critical (e.g., the negotiated contract might include penalty clauses if the supplier does not deliver on time).

In identifying the agent-based architectural "styles," we are able to readily apply the earlier experiences, reasoning, and models developed in other analyses to this new situation. In particular, any generic Architecture Tradeoffs analyses for agents types can be made more concrete for a specific instantiation of the agent type and we can then determine if the potential risks noted in the generic case are either handled or require further investigation. We illustrate this with a generic security scenario such as "Attacker (consumer) issues massive requests for service." [10]. This scenario could be realized when one or more customers attempt to flood the global procurement center (GPC) with frivolous request for products (message 1). This type of action can seriously affect the performance and availability of the GPC in the supply chain management system. Obvious solutions or approaches to this problem include instantiating the generic "multiple-broker solution" by allocating GPC-Brokers to sets of "customers" (by location, by nature of their products, etc.). Other solutions include service limitations on individual request originators by the GPC-Broker. The key concept here is that the characteristics of the specific situation (broker in this case) implies the relevance of a set of predefined generic scenarios. In the context of an architectural tradeoff analysis of a specific system, these generic scenarios provide example of where one should start. The applicability of one or more architectural styles can suggest specific cases, problems, solutions, and analyses appropriate to the overall analysis.

6 A Test-Bed for Agents System Analysis

A test-bed is needed to ensure that the techniques used to measure the effectiveness of different agent technologies are repeatable, disciplined, and readily available. We have begun implementation of such a test-bed at the National Institute of Standards and Technologies (NIST). It is our goal to make the test-bed publicly available via the World Wide Web.

The test-bed will include metrics ranging from simple facts (e.g., contact info, literature, and number of users) to sophisticated analyses techniques (e.g., ATAM). Wide range of software qualities such as usability and scalability will be considered. The test-bed will collect the outcomes (e.g., lessons learned, performance measurements) of applying particular agent technologies to various scenarios. Our plan is to provide a number of agent-based scenarios like those illustrated in this paper within the test-bed. We expect that many scenarios will be provided from the agent research and the user communities alike. We will create example problems of varying detail such as the supply chain management example in this paper. We expect that such problems will provide for a communication medium between researchers, vendors, and users of agent technologies. As novel interesting problems are identified, we will further identify new agent-based scenarios.

Scenario application will include a variety of qualitative and quantitative techniques, ranging from discussions that follow from a scenario exploration, to a model problem and a discussion on how that model might be analyzed when instantiated. Specific metrics will be defined and analytic formulae will be adapted to calculate values of particular quality attributes. As a result of this type of analysis, we will identify a collection of sensitivity or tradeoff points. A sensitivity point is a property of the architecture that is critical for the achievement of a quality attribute (e.g., using encryption to achieve confidentiality). A tradeoff point is a sensitivity point for multiple quality attributes (e.g., encryption improves security but increases latency).

A number of these sensitivity and tradeoff points may have real consequences for the system; however, we may not always have the right information to come to a certain conclusion. For example, some decisions may not have been made at the time of the evaluation (e.g., choice of middleware or encryption algorithm).

In this context of possible application, it is also useful to study how such scenarios are applied to agent and non-agent technologies. It is unlikely that we will be able to provide objective means of comparisons among such widely disparate architectures. But it should be possible to capture practical insights in the form of comments, and measures such as estimates of cost, robustness, effectiveness, etc. We believe that the proposed test-bed is an ideal place to capture such subjective measures. In a larger sense, the test-bed stands as a focal point for both agent practitioners and the users that are interested in agent technologies.

7 Conclusion

Software architects are looking with increasing interest at agent technologies as a key ingredient in building mission-critical systems. However, lack of software engineering knowledge in the area of agent-based systems prevents ready adoption of the new, promising technology. In this paper, we present an evolving methodology to achieve, organize, and continuously update software engineering knowledge required for analysis and evaluation of agent-based systems. As in all pioneering areas, engineering knowledge needs to be carefully managed so that influx of new findings is appropriately balanced with the established best practices. The testbed that is being implemented will make this and other methodologies readily accessible to software agent community and should allow for interchange of information required in this new software engineering field. Such a test bed will help capture valuable lessons in the rapidly growing field, accelerate adoption and experimentation with the new agent technologies, and allow lower risk in adopting the technology by the end user.

Acknowledgments

Oak Ridge National Laboratory is managed by the Lockheed Martin Energy Research Corporation for the U.S. Department of Energy. Parts of the article were written un-

der contract no. DE-AC05-96OR22464 for U.S. Department of Energy. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense. Part of the work presented here is sponsored by the Defense Advanced Research Projects Agency. Any mention of products is for information only and does not imply recommendation or endorsement by NIST.

References

1. Barbacci, M.; Klein, M.; Longstaff, T.; & Weinstock, C. Quality Attributes (CMU/SEI-95-TR-21, ADA307888). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, Dec. 1995.
2. Brenner, W.; Zarnekow, R.; & Wittig, H. Intelligent Software Agents, Foundations and Applications. Berlin, Germany: Springer-Verlag, 1998.
3. Hayden, S. C.; Carrick, C.; & Yang, Q. "Architectural Design Patterns for Multi-Agent Coordination." Proceedings of the International Conference on Agent Systems '99. (Agents'99). Seattle, WA, May 1999.
4. Goodwin, R. (ed.). Proceedings of Workshop *Agent-Based Decision-Support for Managing the Internet Supply-Chains*, Autonomous Agents '99 Conference, Seattle, WA, 1999.
5. Ivezic, N., Potok, T. E., and Pouchard, L. C., Multiagent Framework for Lean Manufacturing. IEEE Internet Computing, Vol 3., No. 5, 58-59, 1999.
6. Ivezic, N. and Potok, T.E., "An Agent-Based Approach For Supply Chain Management," Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis, Vol. 3, 570-573, 1999.
7. Klein, M. and Kasman, R. Attribute-Based Architectural Styles (CMU/SEI-99-TR-022). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, Sept 1999.
8. Potok, T.E. and Ivezic, N. , "Multi-agent spare part grouping system for logistics optimization," Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis, Vol. 3, 582-585, 1999.
9. Smith, R. G. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver." IEEE Transactions on Computers C-29, 12 (December 1980).
10. Woods, S.G. and Barbacci, M.R. Architectural Evaluation of Collaborative Agent-Based Systems (CMU/SEI-99-TR-025) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 1999.