Adaptive Workflow = Web Services + Agents

Paul A. Buhler
College of Charleston
Department of Computer Science
66 George Street
Charleston, SC 29424, USA
pbuhler@cs.cofc.edu

José M. Vidal
University of South Carolina
Computer Science and Engineering
Columbia, SC 29208, USA
vidal@sc.edu

Harko Verhagen
Department of Computer and System Sciences
Stockholm University / KTH
Forum 100
16400, Kista, Sweden
verhagen@dsv.su.se

Abstract

Workflow management systems exactly enact business processes described in a process description language. Unfortunately, such strict adherence to the prescribed workflow makes it impossible for the system to adapt to unforeseen circumstances. In this paper we propose that workflow description languages and their associated design tools can be used to specify a multiagent system. Specifically, we advance the idea that the Business Process Execution Language for Web Services can be used as a specification language for expressing the initial social order of a multiagent system, which can then intelligently adapt to changing environmental conditions.

keywords: workflow enactment, coordination technology, multiagent systems, process description languages

1. Introduction

Advances in Information Technology (IT) are creating opportunities for business enterprises to redesign their information and process management systems. The refinement of service-oriented architectures and the emergence of web-enabled, semantically described services allow us to envision a future where these Web services become the next generation of enterprise components. This new enterprise software vision places new demands on software architectures because they will need to support computing with "dynamically-formed, task-specific, coalitions of distributed autonomous resources" [1, pg 99]. These changes are a logical consequence of the seminal work in coordination technology done by Gelertner.

It is now generally accepted that Gelertner was correct when he theorized that computation was orthogonal to coordination [2]. This orthogonality was implied by DeRemer, who wrote in 1976, "Structuring a large collection of modules to form a 'system' is an essentially distinct and different intellectual activity from the construction of the individual modules [themselves]" [3]. From these perspectives, a software system is viewed as an ensemble of coordinables and their orchestrated interactions. Coordinables are entities that function as independent units of computation. The coordinated interaction of the computational units produces the desired behavior of the system. Obvious parallels to workflow systems exist; the workflow activities are the coordinables and business processes coordinate their interaction.

Leymann asserts that workflow construction can be viewed as a two-level programming problem [4, pg 217]. His view is that the implementation of workflow activities is akin to traditional programming, or programming in the small. Activities encapsulate well-defined functionality that typically involves low-level data access routines and algorithmic processing. In contrast, the building of the workflow's process model is akin to programming in the large. The process model prescribes coordination rules by providing a means to express the sequencing of the activities and the flow of data amongst them.

We advocate the synthesis of Gelertner's and Leymann's points of view. We believe that the statements *applications = computation + coordination* and *workflow = activities + processes* are equivalent. This paper presents our vision that multi-agent systems are a required ingredient for the flexible enactment of enterprise workflows. Our view can be summarized by the aphorism *Adaptive Workflow = Web services + Agents*. In this context, the Web services provide the computational resources and the Agents provide the coordination framework. We propose the use of the Business Process Execution Language for Web Services (BPEL4WS) as a specification language for expressing the initial social order of a multiagent system.

2. Adaptive Workflow Enactment

Traditionally, workflow management systems have not been designed for dynamic environments requiring adaptive response. Currently, the need for adaptive workflow is being driven by the demands of e-commerce in both B2B and B2C space. Initial B2B automation activities were centered on Electronic Data Interchange (EDI) initiatives. More recent work in the B2B space has focused on the development and deployment of ebXML (electronic business XML). With both EDI and ebXML the collaborating business partners need to predefine the terms of their electronic interaction. As discussed by Jenz, these technologies enforce regulated B2B interaction and as such, they create closed communities of business partners. [5]. In comparison, views toward virtual organizations require flexible, on-the-fly alignment of business partners; in other words, adaptive workflow capabilities. These loose collaborations of business partners operate in open, non-regulated B2B/B2C scenarios [5]; pre-negotiated collaboration agreements are a hindrance in these environments.

As businesses integrate across organizational boundaries, it becomes important to separate the 'public' process logic from the 'private' business logic. The process logic specifies the order and conditions under which things get done; whereas, the business logic specifies what gets done. Business Process Management (BPM) software is an emerging classification of integration software that treats business processes as first-class entities.

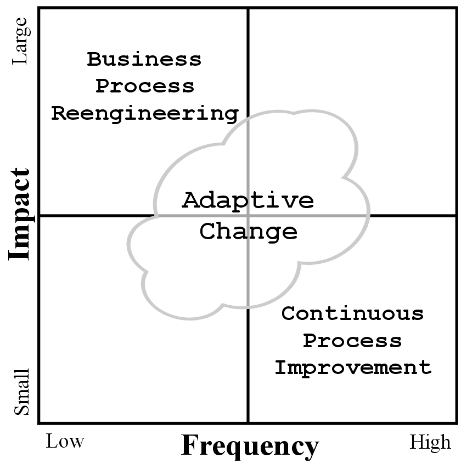
Large
Business Process Reengineering
Impact
Adaptive Change
Continuous Process Improvement
Small
Low
Frequency
High

Figure 1. Characterization of adaptive change

Adaptive workflows need to react to changing environmental conditions. Currently, businesses change their workflows through two primary mechanisms: Business Process Reengineering (BPR) and Continuous Process Improvement (CPI). Figure 1, adapted from [6, pg 239], illustrates the difference between BPR and CPI. BPR is the periodic analysis and subsequent redesign of the intra- and inter- business processes used by an organization. BPR is used to overhaul processes in order to create operational efficiencies that improve quality and save time and cost. Conversely, CPI focuses on continuous improvement through the application of small and orderly changes. Workflows are continuously examined in order to find ways to increase quality and reduce waste. Adaptive workflows respond to changing conditions through adaptive change. As shown in Figure 1, adaptive change should not constrained by measures of frequency or impact.

Current workflow initiatives have embraced the Web service model. Given the current state of technology, Web service based workflows typically are deployed behind corporate firewalls and are used for intra-organizational workflow. The reason for this is that Web service specifications are weak in regards to issues of security, transaction management, internationalization, et al. Inevitably, as standards evolve to address these deficiencies, workflows will transition from the domain of intranets to that of the Internet. This transition will be accompanied by a new set of problems.

When an intranet-based workflow system executes, it does so with a collection of services that are owned and managed by the same organization. In this environment, service interruptions are infrequent and typically scheduled due to consolidated system management. In contrast, Internet-based workflows must be designed for resilient operation as service partners periodically become unavailable due to decentralized system management and the lack of network service guarantees. The evolution from intra- to inter- net based workflows will increase the design and run-time complexity, since the coordination mechanism must become more fault tolerant.

3. BPEL4WS

Recently, IBM, Microsoft and BEA released a new Process Description Language (PDL) named BPEL4WS [7]. BPEL4WS represents the merger of IBM's Web Services Flow Language (WSFL) and Microsoft's XLANG. This merger has created the market consolidation necessary to make BPEL4WS the de-facto standard for expressing workflows consisting of Web services. IBM, Microsoft and others will be releasing retooled versions of their enterprise integration product suites that will use BPEL4WS as the PDL for workflows. IBM has recently released BPWS4J on their Alphaworks site [8]. BPWS4J provides a preview of the capabilities that will be released in WebSphere Studio and WebSphere Application Server. BPWS4J consists of an Eclipse based graphical editor and a workflow engine that are BPEL4WS compliant.

BPEL4WS was designed to combine the features of IBM's WSFL and Microsoft's XLANG. As such, it provides both graph-based and block-based control structures, making it capable of representing a wide range of control flows. Aalst has compared the expressiveness of several PDLs and has confirmed that BPEL4WS represents the union of WSFL and XLANG [9]. BPEL4WS can be used to describe executable business processes and abstract processes. Abstract processes are not typically executable but are used to create behavioral specifications consisting of the mutually visible messages exchanged between transacting parties executing a business protocol. BPEL4WS utilizes the following XML-based specifications: WSDL 1.1, XML Schema 1.0, and XPath 1.0. Importantly, WSDL is used to model both the process and the participating Web services. BPEL4WS is compositionally complete, which means that a composition of Web services is exposed as a Web service eligible to participate in other compositions [10].

Structurally, a BPEL4WS file describes a workflow by stating whom the participants are, what services they must implement in order to belong to the workflow, and what are the various orders in which the events must occur. The BPEL4WS process model is built on top of the WSDL 1.1 service model and assumes all primitive actions are described as WSDL portTypes. That is, a BPEL4WS description describes the orchestration of a set of messages all of which are described by their WSDL definitions.

4. Multiagent Systems

Multiagent systems emerged as a new research area in the early 1990's. It developed partly from distributed processing and partly from artificial intelligence and its modern incarnation as agents. Since its early days, one facet of multiagent systems research has been focused on coordination mechanisms. An example of such a system is TÆMS [11], which is a framework for the analysis and design of coordination mechanisms with a special focus on task dependencies and uncertainty about the environment. Systems like TÆMS can be seen as building upon the seminal work of Malone [12, 13]. Sichman further developed the analysis of dependency relations as an input for coordination between agents [14], [15]. In this work, coordination issues are dealt with at the level of the agent, resulting in systems that exhibit decentralized flow control. Central control however is not only contrary to the whole idea of agents, it is also of no use in building the adaptive systems we aim at.

An application area for multiagent systems research is the simulation of systems, specifically social systems. Here the focus is upon the interplay of individual decision-making and the resultant system level consequences. Concomitantly, consideration must also be given to how system level properties constrain the individual's decision-making ability. The intertwining of the individual and the system is commonly described as micro-macro linkage. By varying parameters, simulation experiments sweep the universe of possible configurations, attempting to find desired and sustainable system level behaviors. Such experiments often lead to new insights and understanding of the system's micro-macro linkage. Often these discoveries are unanticipated, due to the inherent complexity of the system being modeled.

Recently, several large corporations have successful used agent-based modeling and simulation to optimize their operations. Companies such as Procter & Gamble, Southwest Airlines, Merck, and Ford Motor Company have all benefited from agent-based simulations. In these simulations, software agents represent the individual components of the system. The agent's behaviors are modeled after their real-world counterparts. After validating the accuracy of the simulation, by comparing its performance to the real-world system, individual agent's behavior rules can be modified to assess the impact of the change on the system. Procter & Gamble claims that changes instituted in its supply chain, based upon the results of agent-based simulations, are saving the company $300 million dollars annually [16]. We believe that resilient workflows can be achieved by moving agents out of the simulation and into the actual executing system.

5. BPEL4WS for Multiagent Systems

In our earlier work [17], we established a relationship between Web services and agents. Our vision was that of using passive Web services as external behaviors for proactive agents. Huhns further distinguishes between Web services and agents. Some of the distinctions he provides are: Web services know only about themselves, they do not possess any meta-level awareness; Web services are not designed to utilize or understand ontologies; Web services are not capable of autonomous action, intentional communication, or deliberatively cooperative behavior [18]. In contrast, agents possess all of these capabilities.

Workflow enactment by a multiagent system can be viewed as an act of cooperative problem solving. "Cooperative problem solving occurs when a group of autonomous agents choose to work together to achieve a common goal" [19]. For cooperative problem solving to occur, an agent in the multiagent system must recognize that the best path to achieving a goal is to enlist the help of other agents. Social commitments arise when one agent makes a commitment to perform work for another agent. Thus for a multiagent system to engage in cooperative problem solving, the relationships between the agents must be discovered. Since BPEL4WS describes the relationship between the Web services in the workflow and if an agent represented each Web service, then the relationship between the agents would be known a priori. In other words, BPEL4WS could be used to establish the initial social order of the multiagent system.

5.1. BPEL4WS and FIPA

The work of The Foundation for Intelligent Physical Agents (FIPA) can be thought of as creating a component model that allows agents from heterogeneous origins to collaborate in open agent environments. The leading method for representing multiagent interactions is the FIPA Interaction Protocol (IP) standard. FIPA has defined several IP's that describe the most common agent interactions, such as auctions, iterated contract-net, purchasing, etc. Each IP is given a unique name. If an agent claims to be complaint with a certain IP then it must obey the published specifications. FIPA defines IPs in order to facilitate the development of new agents. That is, FIPA envisions a time when most agents are sophisticated enough to be able to carry out unscripted conversations and, in order to enable that goal, it provides a quick and easy way to develop simple reactive agents that can still participate in an agent society. IPs fulfill that role. FIPA IPs are described in the standard documents and are largely defined with the use of UML diagrams.

FIPA IP's are a close equivalent to workflow description languages, but with some differences. Specifically, BPEL4WS and FIPA IPs share many of the same goals. They are both languages for representing a series of structured communications among a set of actors. BPEL4WS uses the concept of a "partner" which is identical to the "roles" in IPs. Both of them have complex flow mechanism (flow, pick, while) which support all desired iterative behaviors. Still, there are many significant differences between them, including:

BPEL4WS uses WSDL PortTypes, which makes it easy to convert a BPEL4WS description into executable code since the atomic actions are all well-defined interfaces. In an agent-based approach, the WSDL information would be used to specify the interaction between the agent and the Web service.

BPEL4WS provides fault handlers. IPs do not have an explicit concept of fault handling. However, existing IPs could be extended to handle faults. These fault-handling messages would not be differentiated from the existing messages.

Since BPEL4WS was designed for Web services, and since Web services are stateless, the designers had to find a way to add state information to the ongoing workflow. They did this with the use of "containers" which provide persistent storage that holds past messages and can be queried to determine the current state of the workflow. This is in stark contrast to IPs, which avoid the complexities of explicitly modeling state because agents are stateful.

BPEL4WS has the concept of "links" which allow it to express precedence constraints that are more complex than those that can be expressed with UML. For example, a BPEL4WS description can express the fact that a certain message will be sent only after three out of a set of messages have been received.

Partners in BPEL4WS are assumed to be completely reactive. That is, the only actions they ever take are those prescribed by the BPEL4WS description and all those actions are reactions to other actions, namely, they are triggered by arriving messages and might depend on the partner's current state. Agents are generally assumed to be pro-active. That is, agents can take actions based on their own internal "deductions" about the world at large.

The last point reflects the key difference between BPEL4WS (and other workflow description languages) and multiagent systems. Workflows prescribe exactly what can be done by each of the participants at any moment in time. The participants, therefore, do not need to understand the whole workflow. They can be implemented as simple reactive agents. While this limitation makes it much easier to implement the agents it also eliminates the robustness and opportunistic behavior that are landmark advantages of multiagent systems.

For example, imagine a contract-approval workflow, which is instantiated every time a consulting company seeks to get a new contract approved. A new multi-million contract arrives which is very important to the company. As this contract works its way through the workflow it could get stuck if the accounting office fails to validate the budget on time. However, in a standard workflow description the accounting agent has no knowledge of the significance of this particular contract and might give some other contract precedence. The accounting agent could also be inoperable that day. It is at these times---when things go wrong---that pro-active agents can use their high-level understanding of the whole process in order to take the action that is best for the company.

There is much that multiagent researchers can learn from workflow descriptions. It seems clear that business demand predictable emergent processes. That is, when everything is working fine then everything should be working as the workflow stipulates. It is only when things break or change that agents have an opportunity to show superior performance over purely reactive processes. We envision a merger of these two complementary approaches. Multiagent systems can be built starting with a workflow description that defines the most common scenario and fault conditions. Once this basic system is tested and deployed the agents could be extended to understand the whole workflow so they can adapt to unforeseen circumstances, reduce unneeded work, and automatically handle the extension of the workflow description.

6. Multiagent Workflow Enactment as an Autonomic System

We believe that IBM's Autonomic Computing initiative provides an interesting vantage point from which to consider adaptive workflow. As noted in IBM's Autonomic Computing Manifesto [20], complexity itself is a byproduct of automation; workflow management systems by their very definition are the automation of a business process. One of the tenants of the autonomic computing initiative is to remove the complexity from the end-user and embed it in the infrastructure of the system. Sophisticated self-governing processes then manage the infrastructure. These processes possess several key characteristics; among them are: self-configuration, self-optimization, self-healing, and self-preservation. Each of these characteristics speaks to the need for adaptation that is designed to achieve specific goals.

Using multiagent systems for workflow enactment is the first step in creating an architecture that will allow the exploration of many fundamental questions. As noted in [21] autonomic systems will consist of autonomic elements that will have policy driven relationships with one another. If the BPEL4WS workflow description is interpreted as a strict policy statement, then a static enactment mechanism like BPWS4J would be appropriate; however, if interpreted as a policy guideline, multiagent enactment mechanisms provide a greater degree of flexibility. Some of the questions to be answered are:

How might the concept of adjustable autonomy be used to enable multiagent enactment across the spectrum of workflow types, from collaboration to production? In production workflows, multiagent implementation may provide execution-monitoring advantages; even without the agents possessing a high-degree's of autonomy. On the other end of the spectrum, agents that monitor the interaction of the participants in a computer-supported cooperative work scenario could potentially discover interaction patterns, formalize process rules and utilize their autonomy to enact elements of the ad-hoc workflow without manual intervention.

How might agents leverage a workflow design tool that can capture the business logic and rationale for service selection and flow? This meta-process information could latter be utilized by the autonomous agents for process redesign (self-optimization). Having a design specification for the multiagent system provides self-knowledge, which could be leveraged for self-optimization. For example, agents can use the workflow description to determine the impact of hypothetical changes, or use it, along with knowledge of available resources, to find under-utilized resources that can be exploited.

How might BPEL4WS be extended to allow the specification of multiple, functionally equivalent partners at each end of the service link? In a supply chain management scenario, the agents could use this information to tailor the workflow to deliver different QOS levels based upon cost, time or quality constraints (self-configuring, self-optimizing). Likewise, the list of partners might represent primary, secondary, and tertiary service providers; in the event of primary partner failure, the workflow could automatically engage the secondary partner (self-healing).

How might an agent's active monitoring of service invocation patterns be useful for the purposes of detecting/correcting inappropriate service access? (self-protection) Perhaps, agents could use a BPEL4WS process description to identify normal behavior and signal everything else as abnormal. Abnormal behaviors would have to be further analyzed to determine if they are a real threat or a legitimate deviation enacted by the agents in an effort to optimize the system's behavior.

How might the abstract process notion be useful as a specification that can be instantiated by agents? (self-configuring) An abstract process definition is non-deterministic and does not specify under what conditions each branch is chosen. As such, it can be used by agents to determine the set of "legal" actions and leaves the choice to the agent's reasoning. Once can envision the use of abstract specifications (if made very flexible) as very high-level system behavioral limits. The agents would then be free to implement any specific system behavior that falls within this space.

7. Related Developments and Future Work

Adaptive workflow capabilities, achieved through multiagent enactment mechanisms, will be influenced by developments related to: BPM software and PDL developments, Web services, the semantic web, and Agent-Oriented Software Engineering (AOSE). The pace of change in each of these areas is quickening as commercial entities strive to capture early market share and consortia like WfMC, BPMI, and W3C struggle to maintain their relevance. In the BPM solution space, this scramble is being driven by market analysis that predicts the BPM market will be worth $6.32 billion in 2005, up from $2.26 billion in 2001. Interestingly, evidence that establishes the need for self-configuring, self-optimizing BPM systems is found in this same research report, which shows that for every dollar spent on BPM software in 2001, three dollars were spent on related professional integration services [22].

In the domain of PDL development, we feel that BPEL4WS will become the de-facto standard as soon as Microsoft and IBM retool their product offerings for release in the first half of 2003. Both the WfMC and BPMI have release statements indicating that their own process description languages, XPDL and BPML respectively, are more capable than the BPEL4WS specification; however, they embrace BPEL4WS as a positive development for the BPM industry [23, 24]. BPMI asserts that BPML is a strict superset of BPEL4WS and therefore it provides a natural target for an eventual convergence of standards. Not to be outdone, in January of 2003, the W3C started the Web services Choreography working group and OMG anticipated the release of a RFP for the specification of a unifying business process definition metamodel.

Although not at the same frenetic pace, developments are also occurring in the space of Web services, the semantic web and AOSE. Regarding Web services, the WSDL and SOAP specifications are completing an update cycle. Currently, the semantic web initiative is transitioning ontology languages from DAML+OIL to the new Web Ontology Language (OWL). Notably, the field of AOSE is beginning to pay close attention to the Web service developments. FIPA has established a technical committee that is proposing an integration strategy which will allow FIPA compliant agents to interoperate with Web services.

On the academic front, several researchers are working at the intersection of agents and workflow. Specifically, [25-27] have written about the potential benefits of introducing agent technology into workflow enactment mechanisms. In [26, pg 575], Marinescu discusses the use of the Bond agent architecture to enact a workflow description captured in XPDL. Most closely related to our vision of using contemporary BPM tools and Web services for multiagent system design is the work described in [28]. In this paper, Korhonen, et al. describes the creation of a workflow ontology that is used to describe both agents and Web services. They hope to build a workflow enactment mechanism that can utilize the ontology to bridge the communications gap between agents and Web services.

As we look forward, we have much work to do to demonstrate an adaptive, multiagent-based workflow engine. Our current activities are focused on several fronts: the completion of a mapping of BPEL4WS constructs onto a multiagent system decomposition; understanding the interplay between BPEL4WS and other associated specifications, namely WS-Transaction, WS-Coordination, and WS-Security; and the definition of a generic Web service/FIPA-compliant agent interface. We anticipate having these tasks complete by the end of 2003.

8. Conclusion

In this paper, we have likely generated more questions than answers. In part, our goal has been to contextualize thoughts of multiagent systems as a workflow enactment mechanism. We predict that the landscape of enterprise integration will undergo dramatic changes in the next 3-7 years as Web services usher in a new era and BPM applications replace traditional EAI efforts. Agent-oriented researchers need to seriously investigate the use of workflow design tools and PDLs for producing multiagent system specifications. As we experiment in earnest, we anticipate answering many of the open questions we have raised.

9. References

[1] D. Garland, "Software Architecture: a Roadmap," presented at The Future of Software Engineering, Limerick, Ireland, 2000.

[2] D. Gelernter and N. Carriero, "Coordination Languages and their Significance," *Communications of the ACM*, 35(2), pp. 97-107, 1992.

[3] F. DeRemer and H. Kron, "Programming in the Large versus Programming in the Small," *IEEE Transactions on Software Engineering*, 2(2), pp. 80-87, 1976.

[4] F. Leymann and D. Roller, *Production Workflow: Concepts and Techniques*. Upper Saddle River, New Jersey: Prentice Hall PTR, 2000.

[5] D. E. Jenz, "The 'big boys' unite forces - What does it mean for you?," at http://www.webservices.org/index.php/article/articleview/633/1/24/.

[6] W. v. d. Aalst and K. M. v. Hee, *Workflow management : models, methods, and systems*. Cambridge, Mass.: MIT Press, 2002.

[7] CoverPages, "Business Process Execution Language for Web Services (BPEL4WS)," at http://xml.coverpages.org/bpel4ws.html.

[8] IBM, "BPWS4J," at http://www.alphaworks.ibm.com/tech/bpws4j.

[9] W. v. d. Aalst, "Don't go with the flow: Web services composition standards exposed," *IEEE Intelliegent Systems*, 18(1), 2003.

[10] J.-G. Schneider, M. Lumpe, and O. Nierstrasz, "Agent Coordination via Scripting Languages," in *Coordination of Internet Agents : Models, Technologies, and Applications*, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, Eds. New York, NY: Springer-Verlag, 2001, pp. 153-175.

[11] K. S. Decker, "TÆMS: A framework for analysis and design of coordination mechanisms," in *Foundations of distributed artificial intelligence*, G. M. P. O'Hare and N. Jennings, Eds. New York: Wiley-Interscience, 1996.

[12] T. W. Malone, "Modeling coordination in organizations and markets," *Management Science*, 33(10), pp. 1317-1332, 1987.

[13] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Computing Surveys*, 26(1), pp. 87-119, 1994.

[14] R. Conte and J. Sichman, "DEPNET: How to benefit from social dependence," *Journal of Mathematical Sociology*, 20(2-3), pp. 161-177, 1995.

[15] J. S. Sichman, "DEPINT: Dependence-Based Coalition Formation in an Open Multi-Agent Scenario," *Journal of Artificial Societies and Social Simulation*, 1(2), 1998.

[16] G. Anthes, "Agents of Change," *Computerworld*, January 27, 2003, pp. 26-27.

[17] P. Buhler and J. M. Vidal, "Semantic Web Services as Agent Behaviors," in *Agentcities: Challenges in Open Agent Environments*, LNCS/LNAI, B. Burg, J. Dale, et al., Eds. Berlin: Springer-Verlag, 2003.

[18] M. N. Huhns, "Agents as Web Services," *Internet Computing*, 6(4), pp. 93-95, 2002.

[19] M. J. Wooldridge, *Reasoning about rational agents*. Cambridge, Mass.: MIT Press, 2000.

[20] IBM, "Autonomic Computing: IBM's Perspective on the State of Information Technology," at http://www.research.ibm.com/autonomic/manifesto/.

[21] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, 36(1), pp. 41-50, 2003.

[22] S. Cowley, "BPM market primed for growth," in *InfoWorld*, September 23, 2002.

[23] WfMC, "Press Release," September 12, 2002.

[24] BPMI.org, "BPML|BPEL4WS: A Convergence Path toward a Standard BPM Stack," August 15, 2002.

[25] M. P. Singh and M. N. Huhns, "Multiagent Systems for Workflow," *International Journal of Intelligent Systems in Accounting, Finance and Management*, vol. 8, pp. 105-117, 1999.

[26] D. C. Marinescu, *Internet-based workflow management : toward a semantic web*. New York: Wiley-Interscience, 2002.

[27] M. Griss, "Software Agents as Next Generation Software Components," in *Component-based software engineering: putting the pieces together*, G. T. Heineman and W. T. Councill, Eds. Boston: Addison-Wesley, 2001, pp. 641-657.

[28] J. Korhonen, L. Pajunen, and J. Puustijarvi, "Using Web Services and Workflow Ontology in Multi-Agent Systems," presented at Workshop on Ontologies for Multi-Agent Systems, Siguenza, Spain, 2002.