Autonomic Web Processes

Kunal Verma and Amit P. Sheth

LSDIS Lab, Dept. of Computer Science, University of Georgia, Athens, GA 30605, USA
{verma, amit@cs.uga.edu}

Abstract. We seek to elevate autonomic computing from infrastructure to process level. Different aspects of autonomic computing – self configuring, self healing, self optimizing and self aware are studied for Autonomic Web Processes (AWPs) with the help of a supply chain process scenario. Existing technologies and steps needed to shorten the gap from current process management systems to AWPs are studied in this paper. The behavior of AWPs is controlled by policies defined by users. Sympathetic and parasympathetic policies are introduced to model short and long term policies. A key advantage for elevating autonomic computing to a process level is that the trade-offs can be more evident because the process components map more readily to business functions.

1 Introduction

The increasing complexity in computing models, as well as massive growth in computing resources has made efficient interaction of humans and information technology increasingly difficult [10]. The vision of autonomic computing [14] proposes a computing model analogous to the autonomic functioning of the human nervous system which regulates various human functions without conscious control of the human mind. Autonomic computing is characterized by systems with capabilities of self management of their resources based on policies. The field of autonomic computing has addressed some very important research issues like self adaptive middleware [16], autonomic server monitoring [20] and policy driven data centers [15]. In this paper, we propose to elevate autonomic computing from infrastructure level to the process level to create Autonomic Web Processes (AWPs).

We present AWPs as a natural evolution of autonomic computing from individual information technology resources to the business processes that govern the functioning of various businesses activities. Essentially, AWPs are self aware, self configuring, self optimizing and self healing processes that interact with the environment based on user specified policies. AWPs may be a more appealing way to benefit from autonomic computing. This is because it is inherently more difficult to define and measure tangible ROI for an infrastructure, but it can be more possible to do so since business functions that can be directly supported by AWPs or mapped to its components.

In this paper, we will build upon previous research on semantic Web processes [18], workflows and autonomic computing to create a framework for AWPs. One of the three process architectures presented in [22] termed "dynamic trading processes" shared the characteristics of AWPs such as self configuration and dynamism. We use a motivating scenario to discuss the potential advantages of supporting autonomic properties at the process level. We also briefly survey the current research and technological expertise for supporting each of the properties and try to outline enhancements to current state of the art to create AWPs.

Consider following examples:

- When there is a change in supplier's capabilities in highly reactive part procurement process of a computer manufacturer such as Dell. Currently delays in part deliveries lead to huge losses [13]. This is largely due to non responsive business processes that take time to react to the environment. Using an AWP would help the process to react to the situation with the help of declaratively specified policies. It is important to be able to model both the short term and the long term policies. A short term policy may want to re-order the part from some other supplier to reduce the immediate loss, but a long term policy might consider the previous order fulfillment history of the supplier, as well as, the relationship with the supplier. In order to capture such policies, we introduce the concept of sympathetic (short term) and parasympathetic (long term) policies.
- Where the manufacturer has already decided the suppliers, but a sudden change in foreign currency exchange rate (modeled as an external/environmental constraint/parameter), may make another set of suppliers more optimal. For example, Indian textiles became cheaper and the need to distribute risks became more important when China announced 2.5% devaluation of its currency and stopped linking it solely to US$.
- When market demands and buyer needs change suddenly. Consider the case of iPOD component manufacturers, before and after the announcement of iPOD Nano. Based on the popularity of iPOD Mini, a manufacturer of its component mini-drive could raise the cost or even be tempted to invest into new production lines to increase capacity. However, if the manufacturer does not very quickly react to the announcement and sudden popularity of the iPOD Nano which uses flash memory, it could face substantial losses.

An AWP must continuously try to self optimize and must have the ability to reconfigure the process. The rest of the paper is organized as follows. Section 2 provides some background information about the autonomic nervous system and autonomic computing. AWPs are defined in Section 3. The motivating scenario is presented in Section 4. Section 5 presents AWP Properties in detail. Finally, Section 6 outlines the conclusions and future work.

2 Background – Autonomous Nervous System and Autonomic Computing

In this section, we provide a brief background of the autonomic nervous system (ANS) and autonomic computing. The ANS is responsible for maintaining constant internal environment of the human body by controlling involuntary functions like digestion, respiration, perspiration, and metabolism, and modulating blood pressure [6]. All these functions are not voluntarily controlled by us (e.g., a person does not have direct control over blood pressure). At a high level of granularity, the ANS has four main functions [12]: 1) Sensory function – It gathers information from the outside world and inside the human body, 2) Transmit function – transmits the information to the processing area, 3) Integrative Function – processes the information and decides the best response and 4) Motor function – sends information to the muscles, glands and organs so that they can respond properly. It is divided into two subsystems- sympathetic and parasympathetic. The sympathetic nervous systems deals with providing responses and energy needed to cope with stressful situations such as fear or extremes of physical activity. It increases blood pressure, heart rate, and the blood supply to the skeletal muscles at the expense of the gastrointestinal tract, kidneys, and skin. On the other hand, the parasympathetic nervous systems brings normalcy in between stressful periods. It lowers the heart rate and blood pressure, diverts blood back to the skin and the gastrointestinal tract.

The vision of autonomic computing aims to make systems that simulate the autonomic nervous system by being more self managing. The objective is to let user specify high level policies and then the system should be able to manage itself, based on those policies. The following properties have been defined for autonomic systems [10] – self aware, self configuring and reconfiguring, self optimizing, self healing, policy based interaction with other components and self protecting.

A blueprint for autonomic architectures [30] was presented in [11]. It identifies the main entities in an autonomic system as – resources, touchpoints and autonomic managers. The resources are the entities that are managed by managers. Touchpoints are the interfaces by which the entities interact with the autonomic managers or other resources. A touchpoint has two sub components – sensors and effectors. Sensors are used to disseminate information about the resource by providing an interface for accessing the state of the resources. They also support event generation for sending events to the autonomic managers. Effectors provide interfaces which are used by autonomic managers to change state of resources. Another crucial aspect of autonomic computing is the representation and reasoning based on policies.

3 Autonomic Web Processes

AWPs are Web service based processes that support the autonomic computing properties of being self configuring, self healing, self optimizing, self aware, self protecting and self healing. The underlying backbone of AWPs will be based on autonomic infrastructure proposed by various autonomic computing researchers. Our aim is elevate these properties to the business process level, as the business processes are the backbone of the businesses and key to their competitiveness. Fig. 1 shows the benefits of autonomic computing at the infrastructure level and the process level. The benefits of autonomic computing at the infrastructure level are manifold. Human involvement is reduced in configuring infrastructure and recovering from failures. In addition, businesses are able to guarantee SLAs based on autonomic resources. We believe that these benefits can be leveraged in an even more efficient manner if the business processes that control the infrastructure were also autonomic. Hence, the benefits of autonomic computing would be magnified by reducing human involvement in configuring the processes and recovering from failures. In addition, the processes would be self optimizing and highly reactive to changes in the environment.

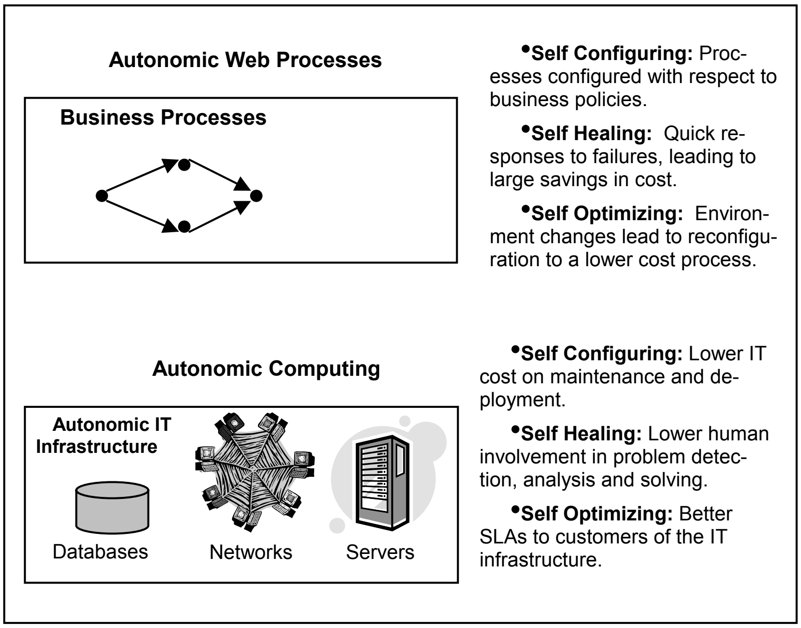
Autonomic Web Processes
Business Processes
•Self Configuring: Processes configured with respect to business policies.
•Self Healing: Quick responses to failures, leading to large savings in cost.
•Self Optimizing: Environment changes lead to reconfiguration to a lower cost process.
Autonomic Computing
Autonomic IT Infrastructure
Databases
Networks
Servers
•Self Configuring: Lower IT cost on maintenance and deployment.
•Self Healing: Lower human involvement in problem detection, analysis and solving.
•Self Optimizing: Better SLAs to customers of the IT infrastructure.

Fig. 1. Autonomic Web Processes and Autonomic Computing

4 Motivating Scenario

Consider the part procurement process of a computer manufacturer. The inventory management software (IMS) sends an order of a number of parts to the procurement module (PM). It is the IMS's job to decide the quantities and number of parts to be ordered. It is also responsible for deciding the amount of money to be spent on the whole process and/or for each individual part and setting required times for delivery. In addition, it may specify some compatibility issues between some quantities of the parts (e.g. ordering a certain quantity of a type motherboard requires ordering matching quantities of compatible memory, video cards, etc.). In other words, the IMS is responsible for setting the configuration parameters for the part procurement process.

We now introduce the part procurement process of the PM, which is responsible for actually procuring the parts from suppliers without violating the constraints set by the IMS. The PM has some more factors to consider like whether to order only from preferred suppliers, or to choose cyclically among its bag of suppliers [13]. Ideally, it should be able to optimally configure the part procurement process and then place the orders. Then it should monitor the orders for physical and logical failures and have the ability to deal with them. Physical failures are based on the supplier service going off-line, while logical failures might include delay in delivery or partial order fulfillment by suppliers.

In this paper, we will explore the autonomic aspects of the process shown in Fig. 2. The AWP properties that we will consider are as follows.

- *Self Configuring:* How can the process be self-configured without violating the constraints (policies) of the IMS and PM?
- *Self Healing:* Can the process use the policies to recover from physical and logical failures?
- *Self Optimizing:* Identifying points for the process to be notified of more optimal suppliers or currency exchange rates.
- *Self Aware:* Creating a comprehensive semantic model expressive enough to support the above mentioned AWP properties.

In order to support these properties, we propose four AWP components, the autonomic execution engine and three autonomic managers that support self configuring, self healing and self optimizing functionalities.

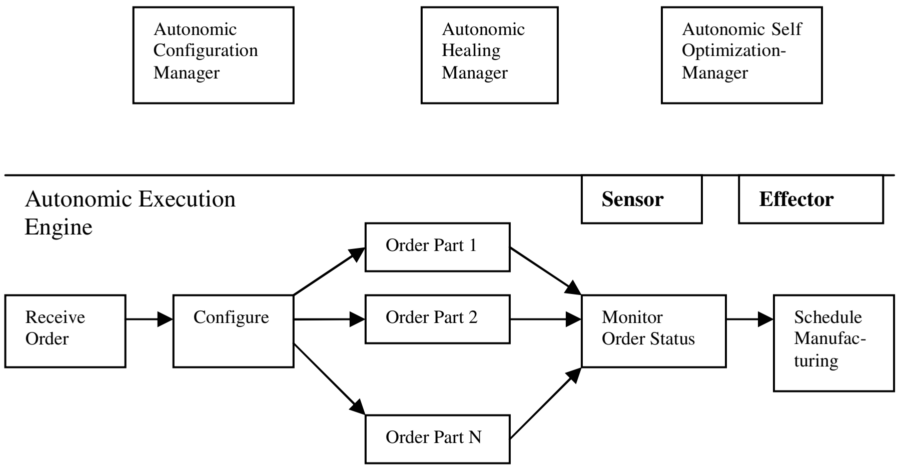
Autonomic Configuration Manager
Autonomic Healing Manager
Autonomic Self Optimization-Manager
Autonomic Execution Engine
Sensor
Effector
Receive Order
Configure
Order Part 1
Order Part 2
Order Part N
Monitor Order Status
Schedule Manufac-turing

Fig. 2. Autonomic Part Procurement Process

5 Defining Autonomic Computing Properties for AWPs

In this section, we describe different properties for AWPs. We start by explaining each property with the help of motivating scenario presented in Section 4 and then survey some of the research work relevant for supporting the property.

5.1 Self Configuring

An AWP must be able to configure itself on the basis on the user policies. For an AWP, configuration may include the following functions- discovery of partners, querying partners for quotes, negotiation with the partners, constraint analysis (non quantitative analysis, optimization using integer linear programming/genetic algorithms, etc.) and dynamic/runtime binding. For the motivating scenario in Section 4, self configuration refers to the optimal selection of suppliers of the process on the basis of the computer manufacturer's policies. The AWP configuration manager must be able to configure the process with respect to the policies. The policy language must be able to specify the goals of the configuration. In this case the goals of configuration are the following:

1. Identify supplier(s) for each part (discovery)
2. Retrieve quote from database/ Query suppliers for quotes (cost estimation)
3. Negotiate better prices if possible (negotiation)
4. Find optimal suppliers and quantities based on the policies (constraint analysis)
5. Configure the process with the optimal suppliers (dynamic binding)

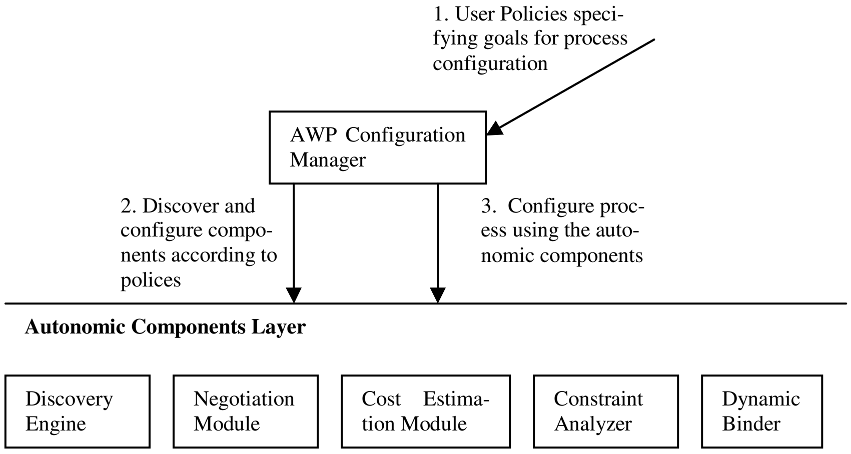
1. User Policies specifying goals for process configuration
AWP Configuration Manager
2. Discover and configure components according to polices
3. Configure process using the autonomic components
Autonomic Components Layer
Discovery Engine
Negotiation Module
Cost Estimation Module
Constraint Analyzer
Dynamic Binder

Fig. 3. AWP Configuration Manager

A high level overview of the AWP configuration manager is shown in Fig. 3. There are three steps to the configuration. The AWP sends the configuration module the goals for configuring it. Then the configuration module finds the required components for the tasks needed and configures them. (e.g., a certain protocol may be loaded for negotiation to configure the negotiation module). Finally, the process must be configured using the different components.

There has been noteworthy research in all modules mentioned for configuration. Semantic Web service which enhances the querying capabilities of UDDI has been discussed [19] [26] [24]. The process of requesting quote from suppliers in the electronics domain has been standardized by RosettaNet. Negotiation using game theory was discussed in [7] [9]. Constraint analysis has been discussed using integer linear programming [2], genetic algorithms and SWRL [28]. Dynamic binding capabilities for Web processes have been discussed in [25]. For creating an infrastructure for self configuration AWPs all the modules must be created as autonomic components and the interactions between them should be policy driven.

5.2 Self Healing

An AWP must be able to recover from failures. There could be two types of failures – system level failures and logical level failures. An example of a physical level failure is a supplier Web service failing during order placement. Logical failures include delay in delivery or the supplier fulfilling only part of the order. For either kind of failure, the AWP must be able to make an optimal choice based on existing alternatives. The choices could include replacing the supplier or canceling the order as a whole. Replacing the supplier could be costly, as there may be a long term relationship or some other parts' orders may have to be cancelled and re-ordered because of part dependencies.

The self healing behavior of an AWP should be governed by policies. In order to preserve the long term business policies, we propose to model the recovery policies as sympathetic policies (e.g., replace supplier after 5 retries or short term profit maximization) and the long term policies as parasympathetic policies (e.g., preferred supplier order cancellation should be avoided). The AWP framework should be able to reason on the policies and choose the most appropriate plan for healing. The self healing aspect of an AWP can borrow from the rich work on workflow transactions [21], compensation [4] and recovery [17]. Ideally, a cost based healing mechanism must be created for AWPs, which combines all the three models (transaction, compensation, recovery) with some optimization model.

5.3 Self Optimizing

An AWP must be able to optimize itself with changes in the environment. It must have the ability to monitor the changes in the environment and reconfigure itself, if there exists a more optimal configuration. As an example of change of the environment, consider the case where some of the suppliers are in different countries and the change in currency conversion rates can render an optimal process sub-optimal. In that case, the AWP must be able to change the suppliers by reconfiguring the process. Other changes in the environments include a supplier announcing a discount, the most favorable clause of a contract getting activated because the supplied offered a better deal to another buyer or a new supplier registering itself with the manufacturer.

As shown in Fig. 4, the self optimization manager has a number of listeners, which monitor the environment of the AWP. The entities and variables to be monitored are selected according to the user specified policies. In this case, there are two entities being monitored – currency exchange rates and supplier discounts. Fig. 4 shows a currency change event above the user specified threshold which is detected by a listener and sent to the self optimization manager. The self optimization manager generates a reconfigure event for the configuration manager, which performs analysis using different reasoning engines at its disposal. If a more optimal solution is found, it uses the effector of the execution engine to change the process configuration. The self optimization property creates a need for a new generation of process coordination (workflow) engines that are not only guided by control flow constructs but also by optimal execution based on the changing environment.

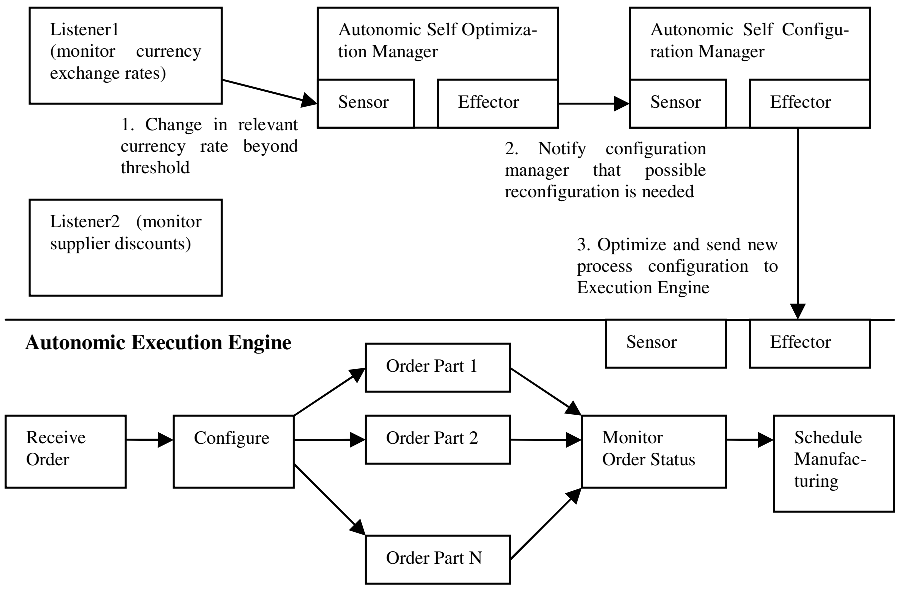
Listener1 (monitor currency exchange rates)
1. Change in relevant currency rate beyond threshold
Listener2 (monitor supplier discounts)
Autonomic Self Optimization Manager
Sensor
Effector
2. Notify configuration manager that possible reconfiguration is needed
Autonomic Self Configuration Manager
Sensor
Effector
3. Optimize and send new process configuration to Execution Engine
Autonomic Execution Engine
Sensor
Effector
Receive Order
Configure
Order Part 1
Order Part 2
Order Part N
Monitor Order Status
Schedule Manufacturing

Fig. 4. Self Optimization of AWP due to change in Environment

5.4 Self Aware

In order to achieve the autonomic computing properties in this section, an AWP must be aware of itself and its environment. This implies that there must be a comprehensive model of the AWP, the Web services, the environment and the policies that guide its operation. Given the already entrenched position of the WSDL and related standards, a truly extensible and upwardly compatible approach that preserves current investment in tools, techniques and training must be used to create model. Based on our experience with the METEOR-S [18] project, which deals with modeling the complete lifecycle of Web processes, we have concluded that no one approach is enough to capture all the intricacies of AWPs. We will build upon our broad classification of the semantics [23] required for this – data, functional, execution and non-functional semantics to outline the model.

The emerging field of the Semantic Web [29] proposes using description logics based ontologies (with the W3C recommended OWL language) to capture the semantics of data on the Web. While, this seems adequate to capture the necessary data semantics (inputs/outputs) of Web services, it is not adequate to capture the functional semantics of Web services (what the Web service does), where a different representation like horn logic based SWRL may be more adequate. The execution semantics focus on the behavioral aspects of Web services, the current state of Web processes and different approaches like task skeletons [5], Petri nets based YAWL [1] or different variants of temporal logic can be considered to represent the behavior of Web services.

The non-functional semantics include the policies, business rules, constraints, and configuration/reconfiguration parameters. While the logic based modeling languages are good for capturing qualitative aspects of business rules, and process constraints, they are not effective in capturing the quantitative constraints for process optimization, which can be represented using an operations research based technique like integer linear programming (ILP). For goal or utility based reconfiguration of processes [15] or decision theoretic planning models like Markov decision processes may be more adequate. Another important issue in non functional semantics is the ability to represent the policies at different levels - Business Level Policies, Process Level Policies, Instance Level Policies, and Individual Component Level Policies and have the ability to resolve conflicts between them.

For self configuration, the discovery phase would require functional, data and non functional semantics. All other phases – negotiation, constraint analysis and binding will be guided by policies (i.e. non functional semantics). For self healing the execution semantics which includes the state of process and transactional traits on the Web services will be required. In addition, the best plan for healing will be decided using the policies. For self optimizing, the entities in the environment to be monitored will be specified using policies. Both, self optimizing and self healing involve reconfiguration.

An important aspect of our approach is the ability to map our model to existing service oriented architecture standards [8] using the extensibility features, provided by the standards. This has been illustrated in our previous work in WSDL-S [3] [24], which adds data and functional semantics to WSDL and semantic extensions to WS-Policy [27], which proposes using OWL ontologies and SWRL rules to represent non-functional semantics of Web services using the WS-Policy framework.

6 Conclusions and Future Work

In this paper, we have a presented an approach for elevating autonomic computing to the process level. We have provided a brief outline of how an AWP can support self configuration, self healing and self optimizing properties. The contributions of this paper include:

- Defining and creating a framework for AWPs.
- Studying the applicability of current research for creating AWPs.

As we discussed earlier, there has been significant work done on autonomic computing, semantic and dynamic Web processes and workflows. AWPs are the logical next step in the evolution of all these fields, as it builds upon the work done in these vast and rich areas. As a first step towards creating AWPs, a comprehensive semantic model of all aspects of AWPs will have to be created. In future, we will demonstrate explicit need and use of the four types of semantics we have identified: data semantics, functional semantics, non-functional semantics and execution semantics [23].

We have also tried to outline some of the initial steps which will be needed to support the other AWP properties. We have provided initial discussions on how to model the first two examples mentioned in the introduction – autonomic supply chain recovery from failure with the sympathetic and parasympathetic policies and self optimization due to changes in environment with the help of the sensors, effectors and autonomic managers. We plan to implement these scenarios and test our hypotheses about the benefits of AWPs.

As the benefits from creating AWPs are manifold for both business and scientific processes, we aim to collaborate with our research partners in the industry and the academia to realize this vision. Our future work includes creating a research prototype that supports AWPs and creating a theoretical model to represent all aspects of AWPs.

Acknowledgements

We would to thank members of the LSDIS Lab and the METEOR-S project whose valuable insights and ideas helped us in writing this paper. In particular, special thanks go to John A. Miller, Karthik Gomadam and Prashant Doshi.

References

[1] Wil M. P. van der Aalst, A.r H. M. ter Hofstede: YAWL: yet another workflow language. Inf. Syst. 30(4): 245-275 (2005)

[2] R. Aggarwal, K. Verma, J. Miller and W. Milnor, "Constraint Driven Web Service Composition in METEORS," Proc. of the 2004 IEEE International Conference on Services Computing (SCC 2004), 2004, pp. 23-30

[3] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. Schmidt, A. Sheth, K. Verma, Web Service Semantics - WSDL-S, A joint UGA-IBM Technical Note, version 1.0, http://www.alphaworks.ibm.com/g/g.nsf/img/semanticsdocs/$file/wssemantic_annotation.pdf

[4] G. Alonso, D. Agrawal, A. Abbadi, M. Kamath, R. Günthör, C. Mohan: Advanced Transaction Models in Workflow Contexts. ICDE 1996: 574-581

[5] P. Attie, M.. Singh, E. A. Emerson, A. P. Sheth, M. Rusinkiewicz: Scheduling workflows by enforcing intertask dependencies. Distributed Systems Engineering 3(4): 222-238 (1996)

[6] S. Bakewell, The Autonomic Nervous System, available at http://www.nda.ox.ac.uk/wfsa/html/u05/u05_010.htm

[7] M. Burstein, C. Bussler, T. Finin, M. Huhns, M. Paolucci, A. Sheth, S. Williams, M. Zaremba, A Semantic Web Services Architecture, To appear in IEEE Internet Computing, 2006.

[8] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, S. Weerawarana: The next step in Web services. Communication of the ACM 46(10): 29-34 (2003)

[9] H. Davulcu, M. Kifer, I. V. Ramakrishnan: CTR-S: a logic for specifying contracts in semantic web services. WWW (Alternate Track Papers & Posters) 2004: 144-153

[10] IBM Autonomic Computing Website, http://researchweb.watson.ibm.com/autonomic/

[11] IBM Autonomic Computing Blueprint Website, http://www-03.ibm.com/autonomic/blueprint.shtml

[12] J. Johnson, Autonomic Nervous System, http://www.sirinet.net/~jgjohnso/nervous.html

[13] R. Kapuscinski, R.. Zhang, P. Carbonneau, Robert Moore, Bill Reeves, Inventory Decisions in Dell's Supply Chain, Interfaces, Vol. 34, No. 3, May–June 2004, pp. 191–205

[14] Jeffrey O. Kephart, David M. Chess: The Vision of Autonomic Computing. IEEE Computer 36(1): 41-50 (2003)

[15] J. Kephart, W.. Walsh: An Artificial Intelligence Perspective on Autonomic Computing Policies. POLICY 2004: 3-12

[16] V. Kumar, B. Cooper, K. Schwan, Distributed Stream Management using Utility-Driven Self-Adaptive Middleware, The Proceedings of the 2nd IEEE International Conference on Autonomic Computing, 2005.

[17] F. Leymann: Supporting Business Transactions Via Partial Backward Recovery In Workflow Management Systems. BTW 1995: 51-70

[18] METEOR-S: Semantic Web Services and Processes, http://lsdis.cs.uga.edu/projects/meteor-s/

[19] M. Paolucci, T. Kawamura, T. Payne and K. Sycara, Semantic Matching of Web Services Capabilities, Proc. of the 1st International Semantic Web Conference, 2002.

[20] C. Roblee V. B. George Cybenko, Large-Scale Autonomic Server Monitoring Using Process Query Systems, The Proceedings of the 2nd IEEE International Conference on Autonomic Computing, 2005.

[21] M. Rusinkiewicz, A. P. Sheth: Specification and Execution of Transactional Workflows. Modern Database Systems 1995: 592-620

[22] A. P. Sheth, W. M. P. Aalst, I. B. Arpinar: Processes Driving the Networked Economy. IEEE Concurrency 7(3): 18-31, 1999

[23] A. P. Sheth, "Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration," Invited Talk, Workshop on E-Services and the Semantic Web, WWW, 2003.

[24] K. Sivashanmugam, K. Verma, A. P. Sheth, J. A. Miller, Adding Semantics to Web Services Standards, Proc. of the 1st International Conference on Web Services, 2003.

[25] K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, J. Lee, On Accommodating Inter Service Dependencies in Web Process Flow Composition, Proc. of the AAAI Spring Symposium on Semantic Web Services, March, 2004.

[26] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar and J. Miller, METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management, 6 (1), pp. 17-39, 2005.

[27] K. Verma, R. Akkiraju, R. Goodwin, Semantic matching of Web service policies, The Proceedings of the Second Workshop on Semantic and Dynamic Web Processes (SDWP), (in conjunction with ICWS), Orlando, Fl, 2005.

[28] K. Verma, K. Gomadam, A. P. Sheth, J. A. Miller, Z. Wu, "The METEOR-S Approach for Configuring and Executing Dynamic Web Processes", LSDIS Lab Technical Report , University of Georgia, June 24, 2005

[29] W3C Semantic Web Activity, http://www.w3.org/2001/sw/

[30] S. White, J. Hanson, I. Whalley, D. Chess, J. Kephart: An Architectural Approach to Autonomic Computing. ICAC 2004: 2-9