An Architecture for Non Functional Properties Management in Distributed Computing

Pierre de Leuss[1], Panos Periorellis[1], Theo Dimitrakos[2] and Paul Watson[1]

[1]Newcastle University, School of Computing Science, Newcastle upon Tyne, NE1 7RU, U.K.
[2]Information Technology Futures Centre, British Telecommunications plc., U.K.
pierre.de-leusse@ncl.ac.uk

Abstract. One of the primary benefits of Service Oriented Architecture (SOA) [1] is the ability to compose applications, processes or more complex services from other services. As the complexity and sophistication of theses structures increases, so does the need for adaptability of each component. In recent years, a lot of effort has been put into improving the flexibility of these systems so that totally loose services can be integrated dynamically without imposing any architectural restrictions. This proposal presents a project to devise a novel model that aims at increasing the adaptability of the resources exposed through it by dynamically managing their non-functional requirements. To manage these non-functional properties, we aggregate the services required into what we define as a profile.

1 Introduction

In the past few years, inter-application integration has become one of the main interests in IT industry [2-5]. This has brought up the emerging growth of the Service Oriented Architecture (SOA) paradigm which has become the main reference in terms of distributed software architectures [5]. Service orientation is a design paradigm intended for the creation of solution logic units that are individually shaped so that they can be collectively and repeatedly utilised to a specific set of strategic goals and benefits associated with SOA and service-oriented computing [1].

In such environments where change can be frequent, adaptation to contextual changes is a strong requirement [6] but, due to its complexity, can be difficult to achieve. Indeed, the SOA maturity model described in [7] defines the ability to automatically react and respond to change as one of the main characteristics of its top level environments. Such changes can come from the need to adapt the non-functional behaviour (e.g. Security, QoS) of a service according to different contexts.

The Ph.D. will investigate the concept of service composition in the light of notions such as evolution of the environment the service is consumed by. Additionally, the Ph.D. will study the extent to which services can make use of the model we propose in order to increase their adaptability to accommodate change. From a practical point of view the notion of change may reflect alterations such as a change of identity providers or the need to answer to different security requirements by changing the authorisation system. Different aspects of service integration such as I/O parameter range, message protocol and others are also targeted. Our goal is to

prove that we can develop service oriented applications out of totally decoupled services capable of discovering and negotiating coupling parameters. The domain of application for this Ph.D. is that of web services. We will examine the notion of web service integration with the aim of developing an architectural model that handles business messages directed to and from web services. These composition tools will increase the adaptability of the capabilities exposed through them by dynamically managing their non-functional requirements. To manage these non-functional properties, we aggregate the functions required into what we define as a profile. Like a biochemical profile of blood presents its characteristics (e.g. blood potassium, levels of circulating enzymes), the profile we define represents the various non-functional properties of the service that uses it (e.g. security type, audit) and implements them.

In this project we attempt to leverage on the research carried out on service oriented computing and in particular elaborate on the research findings of the GOLD [8] and CARMEN [9] projects. In addition the Ph.D. will draw from the experiences and concepts developed in the area of cloud computing and build on the common ground. We elaborate on this later. The Ph.D. concepts will be demonstrated via a system built by the author. The system will demonstrate how decoupled services can negotiate their message requirements and adapt so that they can establish a communication channel. In addition to the departmental research the implementation of the prototype will make use of research carried out by the sponsor of this project on policy enforcement, trust and access control on the BT B2B Security Gateway [10, 11].

Having established the areas this project will be investigating, the rest of the proposal is structured as follows. In the first part of this proposal, the specifics of this particular work will be briefly described. Following this, different projects that are related to this research will be introduced and shortly discussed. In the next chapter, the technical objectives will be established. Finally, the project methodology and deliverables will be presented.

2 Challenges

The Grid computing vision encompasses an integrated infrastructure for the coordinated sharing of resources and problem solving in distributed environments [10].

One may say that the idea itself is very old in terms of computing science history and one of the most prominent ancestors of the Grid is Meta-computing. This term was first used in the early eighties by L. Smarr [12]. The idea was to interconnect supercomputer centres in order to achieve superior processing resources. Following this I. Foster and C. Kesselman organised in 1997, a workshop entitled “Building a Computational Grid” [13]. The name grid came from the electrical power grid and illustrates the idea that computing resources should be as ubiquitous to access as electric power.

According to [14], the history of the Grid can be divided into three phases. The first stage, taking place in the nineties, saw the creation of systems to access high performance computing infrastructures. These systems were put in place and used by scientific communities mainly to run simulation or perform collaborative works.

During the second phase, the main focus was to develop common infrastructures for grid applications. The last period focused on providing more flexible access to the resources shared in Grid systems. The most prominent output was the adoption of Service Oriented Architecture (SOA), which is often labelled Open Grid Service Architecture (OGSA) [15]. This approach was used to extend the infrastructures developed previously to allow them to deal with large scale distributed collaboration. The concept of Virtual Organisation (VO) [16] was introduced to allow for an enhanced management of shared resources in such distributed collaboration.

In [17], three main types of Grid systems are introduced, each aiming at dealing with a different set of requirements. The first category, computational Grid, takes its roots in the creation of the concept of computing Grid itself and aims at providing its users with high performance computing. The second category, data Grid provides to its users a ubiquitous access to distributed data sources. Finally the last one, and the most recently exposed, is described as the service Grid, which instead of providing computation or data resources allows to share specific functions defined and exposed as services.

Projects such as GOLD investigated thoroughly the above ideas in relation to both Grid computing and virtual organisations. The project's research findings demonstrated how far the notion of decoupling could be pushed given the diversity of emerging web services standards. It also brought forward an interesting set of questions relating to the composition of services without any prior knowledge of each other in an environment where they can negotiate the composition semantics between them without any internal intervention (internal service code and architecture would remain intact).

Cloud computing focuses on these researches questions and proposes a loose architectural framework to support the above research question. The Cloud computing concept envisages a world where applications are built by users on demand via their browser. There are a number of services currently on the Internet that demonstrate cloud computing concepts. The following paragraph discusses cloud computing in more detail.

The increasing demand for distributed computing resources (e.g. data, computation and service) is now completed by additional types of resources such as application (e.g. Operating System) or storage management. This augmentation in complexity, matched with the growth of its usage has pushed the already growing requirement for more flexibility and adaptability to a new level. In this context, the concept of Cloud computing has been pressed to be an innovative way to approach these issues.

The name of Cloud computing implies that resources comes from "the cloud", a public network (e.g. the Internet), rather than from a specific identifiable system.

Although the community has only recently started to investigate this paradigm, certain attributes can already be put onto the Cloud.

The first attribute that defines the notion of Cloud is that the underlying complexity of the systems and their connectors should not only be hidden from the end user, but for the most part to the technical users as well (e.g. developers, architects). This concept was already introduced by Grid computing, in which a user would have been presented a friendly interface. Within the Cloud paradigm, the developer should also be presented with such options as most management issues should be handled automatically by the systems.

The Amazon Simple Storage Service (S3) [18] provides a web services interface that can be used to store and retrieve data from any systems with an Internet access. This illustrates how in the vision of Cloud computing a relatively simple interface can hide a complex system.

The second characteristic which has been recently introduced in Grid and has been pushed forward in Cloud computing is that the computing resources should not be restricted to computation power or data access. As introduced above Cloud computing is an attempt to define computing resource as a wider concept, including the like of OS, and other complex applications.

A good example to present this specificity is the set Web OS projects [18]. These projects aim at providing standard OS features on a browser or through any Internet enabled device. Furthermore they attempt to leverage on their Web orientation to offer their users with collaboration tools and seamless access to distributed file systems.

The scale of the problem and its complexity however are now larger and the technologies now seem to allow for this type of widely distributed and more automated architectures (e.g. autonomic computing architectures). This creates the need for different approaches and generates a new set of challenges such as automatically managing such levels of adaptability in a secure and reliable way. This research question forms the backbone of this Ph.D. research. As we explained earlier we are investigating the concept of composition for the purpose of enabling services in a cloud to establish a communication with each other through a process of automatic discovery and negotiation.

Following is, in summary, the list of the research challenges addressed by this Ph.D.

- This profile management model must allow adaptability in response to changes of the non-functional requirements of the resource exposed through it. For instance if different authentication semantics are required for a specific interaction, the system should allow this adaptation while still providing the other services required (e.g. authorisation, audit).
- This has an impact on the way the consumed services must be presented and it affects the management of the composition's lifecycle. Indeed in opposition to traditional compositions where the need to adapt comes from changes in its components, here we attempt to adapt to external changes. This could have an impact on the way the profile's components are aggregated and the profile managed.
- Moreover, the model needs to be adapted to the fact that the infrastructure composition is built to react to message interception.
- Finally, there must be a procedure to determine that the profiles are safe for the resource exposed through it to use.

3 Objectives

The added value of such a system is that while allowing the communication between services without imposing any architectural constraints it enables a safe and rapid exposition and consumption of any service. At the same time, the system maintains

the responsibility for message delivery, ensures the integrity of the message and the safety of the infrastructure services aggregation.

The Ph.D. is expected to advance the current state of web service integration, by providing mechanisms which are essential for securing process driven coordination of services composition. We expect to leverage on the above capabilities of web service transaction mechanisms and combine them with the flexibility of specifications for expressing such policies and communicating them securely via a standardised format in order to achieve the following technical objectives.

The overall aim of the PHD is to investigate the extent to which it is possible to dynamically handle the non functional properties of a resource exposed on a network.

The specific objectives are to:

- Survey existing methods (technologies and models) that could lead to such dynamic management and to identify the types that are most appropriate in this context as well as the most suitable way to use them. The particular interest points here are non-functional properties management, rapid adaptation, dynamic composition and distributed systems integration.
- Propose an architecture which will allow to enable a safe integration of the resource with the non-functional properties handlers as well as the resource's user without imposing any architectural constraints or as less as possible. This architecture will answer to the above challenges by:
    - Manage a resource's non-functional properties effectively. Allow for rapid change and possibly adaptation.
    - Propose an architecture which will enable integration as seamlessly as possible. Without needing extensive additional work on both resource and consumer sides.
    - Control the quality and security of the profile. In some cases including an authorisation mechanism in a profile might not be possible without using an authentication service. If this situation presents itself, the architecture should propose an answer.
    - Allow an easy integration of potential elements of the profile. Non-functional properties providers should be able to register their services adequately.
    - Guaranty an able management of the messages. If an existing profile isn't adequate anymore, the messages should be queued until they can be treated.
- Implement a prototype of the above architecture which is sufficient to demonstrate the validity of the concepts.

We follow an iterative approach to pursue these objectives. So far, in relation to the survey we have established a set of challenges, objectives and requirements. By looking into integration methods and Grid projects, we have drawn an architectural draft. Finally regarding the implementation, we have built a prototype which we plan to incrementally advance in relation with the progress of the Ph.D. The methodology is further developed in the Project Methodology chapter later in this proposal.

In the next paragraph we attempt to answer to the first set of objectives and elaborate on some related work. The composition methods introduced, at best, attempt

to adapt to changes in the services they consume. The autonomic computing models and projects are interesting for their approaches to high level of complexity and seamless usage for the end user but they aim to solve different problems and therefore are not entirely adequate in this context. Unlike these approaches, we aim to provide a solution that allows a seamless dynamic management of the non-functional proprieties of a resource exposed on the network.

4 Related Work

In [4], different existing approaches used in the IT industry to solve complex integration issues are discussed. In SOA, the abstraction layer provided by a broker model presents the advantage to allow decoupling the client from the service it consumes. This therefore matches this Ph.D. objective to allow for an integration approach as transparent as possible. However, it is not sufficient on its own.

Researchers in distributed computing security, and in particular in domains such that of Virtual Organisations (VO), are still trying to find a solution to the issue of providing flexible and reusable infrastructures. Several projects [8, 9, 19], have presented an abstract layer that aims to take care of certain aspects of the non-functional requirements without needing to modify the services exposed. This approach improves the control and management of these requirements as well as the reusability of the components put in place to satisfy them.

However, these abstracted layers are built according to models that do not allow for the adaptation to changes in non-functional requirements.

In the past years, many research teams have been trying to leverage on the service atomicity attribute to produce solutions that enable increased flexibility and adaptability. Amongst other results, this has given birth to technologies and structures that allow dynamic publication and composition of services [20, 21]. Several approaches have been explored, using semantic technologies [22, 23], formal models [24] or even agent based systems [25].

But these solutions respond to problems where a set of services are put together as a collection in order to stimulate specific functions (e.g. Purchase order). In the case of combining infrastructures dynamically to enable different non-functional behaviours, certain characteristics of the aggregation are different.

If resource R is what we aim to render more flexible through the use of profile P then R* = R + P is the resulting model where R* is the enhanced resource. But how to add profiles to a resource seamlessly and in the same time allow configuring and managing these profiles efficiently?

In response to the ever increasing complexity of computing systems, the autonomic computing paradigm was introduced in 2001 by IBM [26]. According to IBM, there are four fundamental aspects of autonomic computing. Self-configuration specifies that systems adapt automatically to dynamically changing environments. Self-healing defines that systems discover, diagnose, and react to disruptions. Self-optimizing characterises that systems monitor and tune resources automatically. Finally, self-protecting describes that systems anticipate, detect, identify, and protect themselves from attacks from anywhere. Many projects have been initiated using this

concept, including Self Managed Cells (SMC) [27], but these aim at solving different problems.

The architectures proposed for self-* systems in these efforts [28] are focused on large computational facilities or specific domains such as e-health; as such, their models are unsuitable to the types of message broker model that we are attempting to construct.

But the capacity of autonomic systems to self-* is a faculty that addresses the adaptation requirement of the model we aim to produce. The descriptive capacity of the works introduced above allows defining the grammar necessary to ensure that the profile's components can be made available transparently; the profile itself determined safe and the resource's owner to express the need that will define the profile. Finally, the broker model would insure that the enhanced resource can be exposed seamlessly.

In the next part, two different architectural models are introduced. These models were designed through experiments and aim to validate some of the requirements introduced above. Additionally, these models allow identifying the key components of such architecture and the ways they are managed as well as interact with each other.

5 Architectures

For the demonstration we have implemented a system that enables services with different types of security tokens to establish a communication through a process of negotiation and agreement, without imposing any constraints in either the service that acts as a client or as a server. In order to demonstrate the viability of such profile, we have used several security oriented services (e.g. PDP, STS) aggregated into a WSBPEL [29] composition.

Although the architecture is capable of negotiating other aspects of service integration such as I/O parameter range, message protocol and others we felt that security is a domain where researchers will see the immediate benefits of this.

From the experiments we have concluded that there are two different ways to architecture the composition and the way it interacts with its environment. The first option consists in an aggregation of the components required for the profile and the resource. We have named this model full profile as the profile is a composition of all the elements required. The other alternative, called managed profile, will see the composition being more independent from the resource and managed by independent entity. We explain this in more depth in the following paragraphs along with the pros and cons of both models.

5.1 Full Profile Architecture

This architecture defines the profile as a composition of all the components required together with the resource.

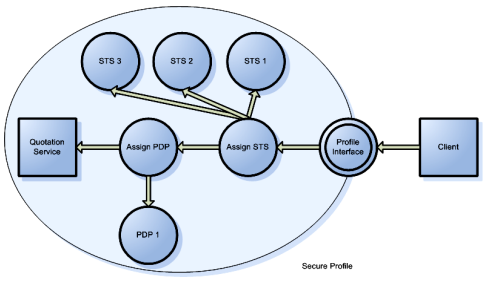

Fig. 1. Full profile architecture.

In figure 1, a simplified representation of the composition is presented. Please note that the two assign activities are not services, but simple operations done in the WSBPEL composition to copy elements from the message and forward to them to invoke the appropriate services according to their values.

This model has the advantage to be simple. If the different components are already configured and deployed, all that is required is to aggregate them to form the composition and expose the profile interface to the user. However, the purpose of the profile is to allow defining and changing the terms of the profile with more flexibility. As most components require configuration before being used (e.g. STS and PDP won't trust each others unless they have recognised themselves before hand) and lifecycle management thereafter, this model lacks this management capacity.

Additionally, with this architecture, the resource is directly linked to the profile. There are cases where the resource lifecycle is best managed separately.

5.2 Managed Profile Architecture

The experience gained with the previous experimentation allows defining a second architecture where both elements of components and resource lifecycles can be more precisely managed.

In this model the message broker acts as the resource interface for the client. The broker calls the appropriate profile manager which in its turn sends the message to the secure profile before, when relevant, sending it to the resource. We anticipate that on

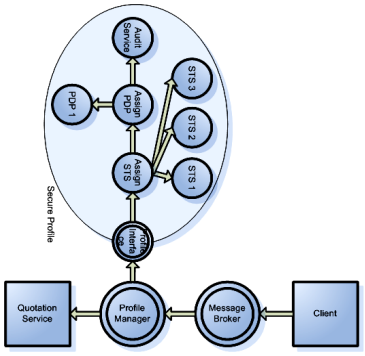

Fig. 2. Manage profile architecture.

the way back to the client, the message could take the same route, allowing the profile (or a different one) to handle additional requirements for the response.

In this experiment we assume that the resource owner has defined policies that allows for the PEP, PDP and STS to work together and allow the user to access the resource using the relevant token and query.

This experiment demonstrates the feasibility of composing a secure profile and introduces the need for a complex architecture that manages this profile. However if directly connected components such as these STSs and PDP can be composed in this way, this model will reach its limits when dealing with more complex aggregations. Additionally, issues related to how the different components can be configured to work together before they are composed is still to be investigated. Finally, both the resource owner and the client should be able to interact with the profile or some of its components (e.g. pushing policies). In further work, such condition will be investigated.

In the following chapter the core components, as defined through the two previous demonstrations and the prior literature review, will be introduced.

6 Core Components

The two previous stages have allowed defining a foundation for the model. In the following chapter the core components will be introduced and their functions presented.

6.1 Knowledge Management System

In IBM's vision of autonomic computing a knowledge source [30] is required to hold management data with architected syntax and semantics, such as symptoms, policies, requests for change, and change plans.

In our model, this core component is divided in three main parts. The first allows holding data about the individual profile's components and their details (e.g. role, identification, pre and post requirements, semantics used). The second allows ensuring the profile is sound by defining tights requirements. This can be achieved through the use of architectural models. For instance a secure profile should include authentication and authorisation. Finally the last part is dedicated to hold user's data such as custom policies.

6.2 Negotiation Broker

The client broker is the interface that allows resource owners to define their requirements in term of the profile's components needed. The request consists of a list of services required by the resource owner along with the operation type the profile is required for (e.g. request or response). The request is expressed using semantics that are provided by the knowledge management system. It consists of a list of components with potential constraints attached to them. These constraints could

include QoS (e.g. throughput, answer time) details for the components used as well as specific semantics to be used for certain operations (e.g. XACML [31], SecPAL [32]).

Alternatively or additionally, the resource owners can declare adaptability level constraints. This could be expressed in such a manner that all requests coming for certain partners in specific federations should be accommodated as best as possible. Or at the opposite that the choices made in terms of profile's components required should not be tempered with, or require the client's authorisation to go further.

Another area still to investigate is the case of a federation where the non functional requirements could be negotiated between two instances of such system. Such terms negotiation could allow for more transparent integration by increasing the compatibility between the resource's requester and the resource's interface as exposed for this particular usage.

6.3 Predication Engine

The role of the predication engine is to determinate the best possible way to achieve the profile requested. The decision making process is based on the requirements given by the user or by another system in the case of a federation, the capabilities held by the system along with their associated constraints and the architectural models stored in their registry.

The degree of automation of this activity would be directly related to quality of the data held in the knowledge management system.

6.4 Profile Manager

The role of this core component is to maintain the profile according to an agreement between this system and the resource owner regarding the profile's availability (and ultimately that of the resource it enhances).
On the top of the profile's lifecycle management, an important task that is attributed to this manager is to handle the adaptability of the profile. According to the type and quality of the data held in the knowledge management system, this model could be capable of identifying threats or miss usage and react to them by modifying the profile.

6.5 Message Broker

The message broker acts as an intermediary between the resource and the client. Upon reception of a message its role is to contact the relevant profile manager. It then forwards the response to the relevant destination.

In the next part, the methodology used to achieve this PhD's objectives will be presented.

7 Project Methodology

The project commenced with case studies of existing applications. This has mainly been achieved by studying how the GOLD middleware and the BT B2B Security gateway [9] are architected and the way they are used. The strength and limits of both projects have then been used to draw a first set of requirements.

These requirements were then used to implement a proof of concept demonstration which allowed validating the requirements previously gathered and completing them. Additionally this experimentation has allowed finding two different types of models. This stage will be completed by a further discussion of the experiment with the different participants of this work.

According to the results of the previous stages, in order to demonstrate the viability of the findings, a prototype implementation of the architecture will be constructed.

8 Deliverables

The expected outcome of this project is the creation of an architectural model allowing seamless integration of distributed resources through a transparent management of their non functional properties. Concretely, this model will be given form through a set of specifications as well as components with the different ways to implement them.

Additionally, a prototype will be implemented to demonstrate the validity of this model. This prototype will show the capability of resource exposed through a system implementing the model aforementioned to be integrated more transparently as well as being able to react to changes.

References

1. Erl, T., Service-Oriented Architecture Concepts, Technology, and Design. 2005.
2. Craggs, S., Raising EAI Standards. 2003.
3. Radhakrishnan, S., Integrating Entreprise Applications: Backgrounder. 2005.
4. de Leusse, P., P. Periorellis, and P. Watson, Enterprise Service Bus: An overview, CS-TR No 1037, in Technical Reports, S.o.C. Science, Editor. 2007, Newcastle University. (cited; Available from: http://www.cs.ncl.ac.uk/publications/trs/papers/1037.pdf)
5. Natis, Y.V., et al., Predicts 2007: SOA Advances. 2006.
6. T. Dimitriakos, e.a., Towards a Trust and Contract Management Framework for Dynamic Virtual Organisations. eAdoption and the Knowledge Economy, 2004. (cited; Available from: http://epubs.cclrc.ac.uk/bitstream/701/E2004-305-TRUSTCOM-OVERVIEW-FINAL%5B1%5D.pdf)
7. Bachman, J., S. Kline, and B. Soni. A New Service-Oriented Architecture Maturity Model. 2005.
8. Periorellis, P., et al., GOLD Infrastructure for Virtual Organisations. UK e-Science All Hands Meeting, 2006: p. 11-20. (cited; Available from: http://www.allhands.org.uk/2006/proceedings/papers/618.pdf)

9. Maierhofer, A., et al. Extendable and Adaptive Message-Level Security Enforcement Framework. in Networking and Services, 2006. ICNS '06. International conference on. 2006.
10. Dimitrakos, T., et al. Contract performance assessment for secure and dynamic virtual collaborations. in Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International. 2003.
11. Smarr, L. The Metacomputer: One from Many, NCSA. 1995 (cited; Available from: http://archive.ncsa.uiuc.edu/Cyberia/MetaComp/MetaHome.html).
12. Foster, I. and C. Kesselman, CRPC Joins National Research Alliance to Build Technology Grid. Parallel Computing Research Newsletter, 1997. 5(4).
13. De Roure, D., et al., The Evolution of the Grid, in Grid Computing: Making the Global Infrastructure a Reality, Wiley, Editor. 2003, Wiley. p. 65-100.
14. Prodan, R. and T. Fahringer. From Web services to OGSA: experiences in implementing an OGSA-based grid application. in Grid Computing, 2003. Proceedings. Fourth International Workshop on. 2003.
15. Foster, I. The anatomy of the grid: enabling scalable virtual organizations. in Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on. 2001.
16. Taylor, I. J. and A. Harrison, From P2P to Web Services and Grids. Peers in a Client/Server World. 2005: Springer.
17. Amazon.com: Amazon S3, Amazon Simple Storage Service, Unlimited Online Storage: Amazon Web Services. (cited 2008 25th April]; Available from: http://aws.amazon.com/s3).
18. Lawton, G., Moving the OS to the Web. Computer, 2008. 41(3): p. 16-19.
19. TrustCoM project. [cited; Available from: http://www.eu-trustcom.com/
20. Chafle, G., et al. Adaptation in Web Service Composition and Execution. in Web Services, 2006. ICWS '06. International Conference on. 2006.
21. Erradi, A., P. Maheshwari, and S. Padmanabhuni. Towards a policy-driven framework for adaptive Web services composition. in Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on. 2005.
22. Nagarajan, M., et al. Semantic Interoperability of Web Services - Challenges and Experiences. in ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06). 2006: IEEE Computer Society.
23. Li, Y., et al. Research on Reasoning of the Dynamic Semantic Web Services Composition. in WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. 2006: IEEE Computer Society.
24. Xua, D.-H., et al. A Formal Model for Security-Aware dynamic Web Services Composition. in Computational Science and its Applications, 2007. ICCSA 2007. International Conference on. 2007.
25. White, S., et al., Autonomic computing: Architectural approach and prototype. Integr. Comput.-Aided Eng., 2006. 13(2): p. 173-188.
26. Kephart, J. and D. Chess. The Vision of Autonomic Computing. Computer, 2003. 36(1):p.41-50.
27. Sventek, J., et al. Self-Managed Cells and their Federation. in 3rd Intl Conference on Mathematical MethodSt Petersburg, Russias, Models and Architectures for Computer Networks Security (MMM-ACNS 2005). 2005.
28. Sterritt, R., R. Sterritt, and M. Hinchey. Autonomic computing - panacea or poppycock? Autonomic computing - panacea or poppycock? in Engineering of Computer-Based Systems, 2005. ECBS '05. 12th IEEE International Conference and Workshops on the. 2005.
29. OASIS. Web Services Business Process Execution Language (WSBPEL). (cited 2008 30th April); Available from: www.oasis-open.org/committees/wsbpel/.
30. An architectural blueprint for autonomic computing, in Autonomic Computing. 2006.
31. Extensible Access Control Markup Language (XACML). (cited; Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
32. Moritz, Y.B., D.G. Andrew, and C. Fournet, SecPAL: Design and Semantics of a Decentralized Authorization Language. September 2006.