The Design of an Autonomic Communication Element to Manage Future Internet Services

John Strassner[1,2], Sung-Su Kim[1], and James Won-Ki Hong[1]

[1] Department of Computer Science and Engieering, POSTECH, Pohang, Korea
{johns,kiss,jwkhong}@postech.ac.kr
[2] Telecommunications Systems & Software Group, Waterford Institute of Technology, Ireland
jstrassner@tssg.org

Abstract. Future Internet services will have vastly different requirements than the current Internet. Manageability, which has been largely ignored, will have the dual role of controlling capital and operational expenditures as well as enabling agile reorganization and reconfiguration of network services and resources according to changing business needs, user demands, and environmental conditions. Autonomic systems have the potential to meet these needs, but important architectural extensions are required. This paper examines the core building blocks of an autonomic system, and compares our approach to existing approaches with respect to Future Internet networks and networked applications. A special emphasis is placed on business driven services.

Keywords: Autonomic Architecture, Autonomic Element, Business Driven Device Management, FOCALE, Future Internet, Management.

1 Introduction

The current success of the Internet architecture has spurred advances in business, social, and technical communications. However, that simplicity is also the source of many of its inherent limitations [1][2][3]. Two of the most important are its architectural limitations and its inability to relate business needs to network services and resources offered. If a design for the Future Internet is to be *sustainable*, then it must not just be technically correct; it is far more important for it be economically motivating and extensible, so that it can support new business models and services.

There is currently a great deal of discussion concerning what type of design approach to take. There are three fundamentally different approaches for the design of the Future Internet that are now being pursued [4]. The first is an incremental, or "evolutionary" approach, and is evidenced by many solutions currently being applied to the current Internet that violate its architectural principles, such as Network Address Translators, Firewalls, and Virtual Private Networks. This approach is no longer sustainable [1]. The second is a revolutionary, or "clean slate", approach [5], which eliminates existing commitments, restraints, and assumptions, and starts with a new set of ideas. The third is a compromise between the above two approaches, and enables new ideas to evolve while simultaneously emphasizing backwards compatibility with the

C.S. Hong et al. (Eds.): APNOMS 2009, LNCS 5787, pp. 122–132, 2009.

existing Internet. This is very important to certain stakeholders, such as Internet service providers (ISPs), who have invested billions into their equipment and want to leverage those investments.

Unfortunately, most communities ignore the management aspects of the Future Internet. For example, many researchers consider the Simple Network Management Protocol (SNMP) [6] or current command line interfaces [7] appropriate for Future Internet management, but such network management protocols are not even suitable for the current Internet [8].

In our previous work, we summarized the management requirements of the Future Internet [9]. In this paper, we continue this work and define architectural components for managing the Future Internet. Our work is motivated by the fact that current and future networks and networked applications have vastly different requirements; this means that a single architecture for the Future Internet cannot simultaneously meet these different needs. Therefore, we need a modular approach that is distributable, so that a single programmable and reconfigurable system can be built that can be repurposed to suit different application needs.

The rest of the paper is structured as follows. Section 2 briefly describes the salient principles of autonomic communications, and reinforces why this approach is well suited to meeting the management needs of the Future Internet. Section 3 concentrates on the *autonomic element* abstraction, and compares three different implementations of it. Section 4 compares the autonomic elements, and Section 5 presents our conclusions and outlines our future work.

2 Autonomic Communications and the Future Internet

This section first describes the components of an autonomic communications architecture, and then explains why this approach is well suited to meeting the management needs of the Future Internet.

2.1 Autonomic Communications Primer

The purpose of autonomic computing is to *manage complexity*. The name was chosen to reflect the function of the autonomic nervous system in the human body. By transferring more manual functions to involuntary control, additional resources (human and otherwise) are made available to manage higher-level processes.

The fundamental management element of an autonomic computing architecture is a control loop, as defined in [10][11]. Sensors retrieve data, which are then analyzed to determine if any corrections to the managed resource(s) being monitored are needed (e.g., to correct “non-optimal”, “failed” or “error” states). If so, then those corrections are planned, and appropriate actions are executed using effectors that translate commands back to a form that the managed resource(s) can understand. If the autonomic network can perform manual, time-consuming tasks (such as configuration management) on behalf of the network administrator, then that will free the system and the administrator to work together to perform higher-level cognitive functions, such as planning and network optimization.

2.2 Future Internet Requirements

The current Internet design emphasized simplicity, and has inherent manageability, scalability, mobility, flexibility, security, trust and other limitations. The European FP7 program is addressing these shortcomings. Challenge 1, called Pervasive and Trustworthy Network and Service Infrastructures, recognizes that the current Internet architecture was not designed to cope with a wide variety and ever increasing number of networked applications, business models, and environments. Europe is advocating four *different* Internets: an Internet of Things, Services, Media, and Communities [12]. While the number of foci of Internets is not important, the splitting of different services that require different networks proves our hypothesis that a *single* architecture is not optimal for managing the Future Internet and its applications.

2.3 Meeting the Needs of the Future Internet Using Autonomics

However, just because four Internets are specified does not mean that four different architectures are required. First, this would greatly complicate sharing resources and services between these networks. Second, it would also make their management very difficult. Service and content providers, network operators, and many other constituencies would be adversely affected, because each would want to offer multiple services from two or more Internets to their subscribers. Hence, we believe that building a single framework that can support each of these needs, as well as future needs, is an important approach.

This is the reason that autonomics are inherently aligned with the Future Internet: autonomic technologies provide a means to reduce complexity. A typical scenario for the Future Internet is one in which a user requires *context-aware services*. This means that the devices that are being used should adapt the services and tasks that they are performing in accordance with changing user needs and environmental conditions. This adaptation should ideally be done with as little manual intervention as possible, which is the focus of Ubiquitous Computing. Mark Weiser said "In such a world, we must dwell with computers, not just interact with them….Interacting with something keeps it distant and foreign…Dwelling with computers means that they have our place, we have ours, and we co-exist comfortably [13]." That paper epitomizes many of the fundamental goals of ubiquitous systems, but especially the goal that the person now has to *think less* about his or her tasks and the environment because *unnecessary work has already been done or summarized by the computer(s).*

Components of network management applications are typically chosen because they are the "best of breed" in performing a particular type of function. The obvious problem is that many different additional applications all share different views of the same object. Therefore, any new management approach must deal with the inherent heterogeneity of programming models and data representation for that domain. The FOCALE [14] autonomic architecture was designed to meet these and other requirements. In particular, FOCALE solves the above heterogeneity problem by defining an internal vendor- and technology-neutral language that can be mapped to vendor- and technology-specific languages (and vice versa) [11][15].

3 The Autonomic Element Abstraction

An *Autonomic Element* (AE) is a fundamental abstraction that describes how autonomic systems manage resources and services. This section will compare and contrast three very different AE designs.

3.1 The Autonomic Element of IBM

Fig. 1 shows IBM's AE [10]. The AE provides a common management façade to be used to control the resource by using a standardized set of interfaces. The autonomic manager implements the control loop. Sensors monitor the resource, while effectors send commands to the managed resource. The set of sensors and effectors forms the management interface that is used for communication between autonomic elements and the environment. The monitor stage collects, aggregates, filters, manages and reports on sensor data. The analyze stage examines the collected data to determine if its goals are being met, as well as to learn about its environment and help to predict future situations. The plan stage organizes the actions needed to achieve its goals. The execute stage controls the execution of a plan.

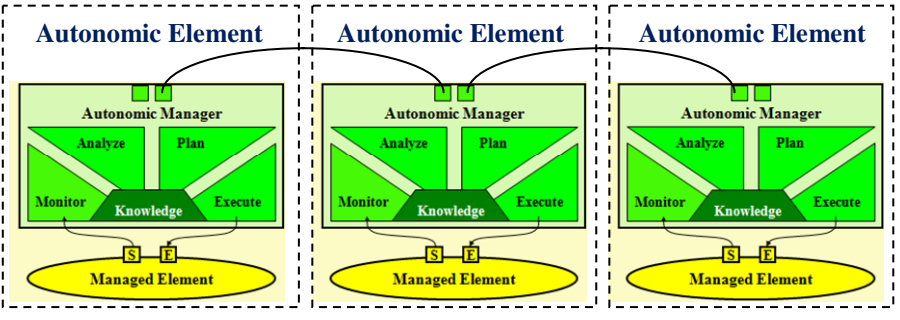

Fig. 1. Conceptual Diagram of IBM's Autonomic Element

3.2 The AE of CASCADAS

Fig. 2 shows the control loop from the FP7 CASCADAS program [16], which defines AEs as *distributed components* that can *autonomously self-organize* and provide *specific user communication services*. This flexibility enables them to self-adapt according to both social and network contexts. AEs have a "common" part that communicates using a set of standard interfaces that is also exposed to the outside world, and a "specific" part that uses dedicated interfaces to provide component-specific functionality (hence, only other AEs that need those functions will implement these dedicated interfaces). AEs use message-based communication to collaborate and provide aggregated and/or composited functions. The former defines a set of AEs that loosely collaborate to provide a service but ensure that each AE remains independent; the latter constructs a new AE that is the composition of each component AE.

Specific interfaces contain semantic descriptions of the set of features that a given AE can perform, and the conditions and actions required to accomplish each feature. The use of semantic features is not provided in the IBM approach. This enables the CASCADAS AE to use semantic routing (i.e., a recipient is addressed using its semantic properties) instead of using its logical or physical address. Each AE also has a "self-model", which defines the possible states and their associated transitions; this self-model is *published* to other components using a specific protocol that can describe the semantics of each state. The self-model can be *dynamically adapted* using the Facilitator, in response to (for example) context changes. Such changes are analyzed using the Reasoning Engine, and the Facilitator will adjust the model using one or more actions.

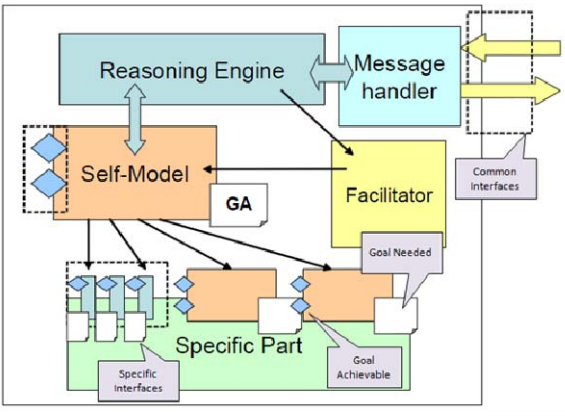

Fig. 2. The AE of CASCADAS

AEs use "plans", which defines the behavior of an AE. Features that an AE can provide are "achievable goals", while functionality that an AE requires are "needed goals". The Reasoning Engine implements the self-model, and the Facilitator is responsible for changing the self-model state machine when a feature exhibits different behavior than what was expected.

A unique approach in the CASCADAS architecture is that it does *not* continuously monitor the in-depth state of the entities that it is managing, but rather uses a restricted set of signals that provide an abstract view of the current state of the managed system. This enables the CASCADAS AE to take action only when needed.

3.3 The Autonomic Communication Element of FOCALE

Fig. 3 shows a simplified version of the FOCALE AE, which is made up of a set of *distributable components* connected by an enhanced enterprise service bus (ESB) [17] that supports simple as well as semantic queries. An ESB is an event-driven message broker. Our implementation is a distributed *content-based* message and retrieval broker; the difference between it and standard ESBs is that it can be used to orchestrate content, whereas standard ESBs are limited to orchestrating *messages.* Hence, the FOCALE ESB is really an enterprise *content* bus (ECB). The FOCALE Autonomic

Manager uses the ECB to orchestrate behavior. It can support different types of knowledge acquisition and distribution (e.g., push, pull, and scheduled) and perform common processing (e.g., semantic annotation, filtering and storage) before content is delivered to components. This enables components to register interest in knowledge in a more precise fashion, and thus reduce messaging overhead.

Sensor data is analyzed using the Observe component, which translates vendor-specific data into a vendor-neutral form. Translation is necessary, because in general, each device uses its own language to express its management data; hence, vendor-specific data must be translated into a common vendor-neutral form. This is done using a combination of information/data models and ontologies [15], where input data is matched to structures and patterns defined in the models and ontologies. This approach is unique to FOCALE.

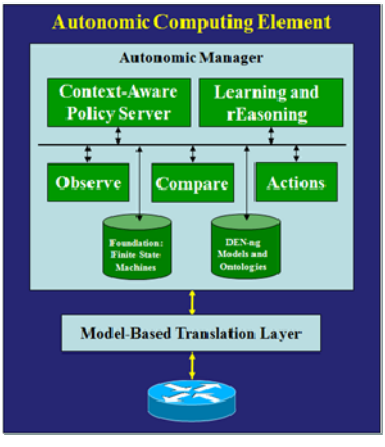

Fig. 3. Simplified Version of the FOCALE AE

The Compare component determines the current state of the managed entity from the normalized data and then compares it to its desired state. If the two are equal, the loop continues; otherwise, appropriate actions are computed and executed. This state-based approach is similar to that used in CASCADAS; however, in FOCALE, a set of adaptive control loops are implemented. An outer, or macro-control loop, is used to ensure that only policy rules applicable for the context at hand are used; an inner, or micro-control loop, is a variant of the Observe-Orient-Decide-Act (OODA) control loop [18], and is used to govern the functionality of the managed entity. Adaptation uses context-aware policy rules to determine both the specific components of the control loop as well as how each component functions. For example, a simple threshold comparison could be changed to a semantic relatedness test depending on the type of data, state, and context. This is shown in Fig. 4.

Since the action may be directed at a different managed entity (which, for example, could have been the root cause of the observed problem), the control loops could be changed (e.g., the original loop could be suspended or terminated and a new loop created). This can be done programmatically as well as through the use of different

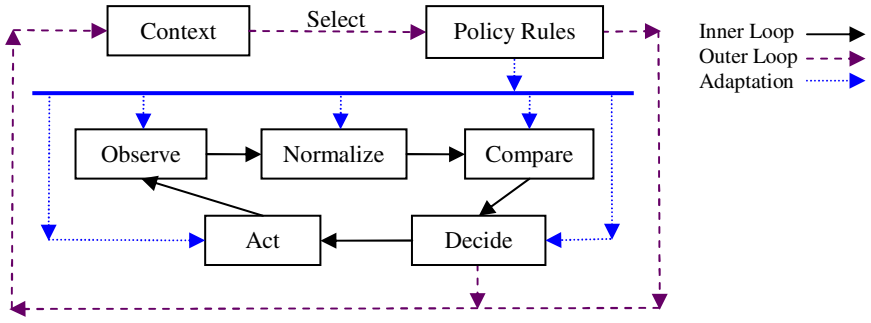

Fig. 4. FOCALE Control Loops

types of learning and reasoning algorithms. This is unique to FOCALE. FOCALE does not publish its model, as that would require external entities to fully understand the complex relationships between different managed entities that FOCALE governs.

Business goals and objectives can be directly related to the system using context-aware policy models [19][20] that use the Policy Continuum [21][22] to relate the needs of different constituencies (e.g., business, network, and programming people) that use *different terms* to each other. This is another unique aspect of FOCALE. For example, it enables business people to describe services using business concepts, such as revenue; this can then be translated to a form that network engineers can use and implement (e.g., traffic classification and conditioning configuration commands). This adaptation is reinforced by the self-governing nature of FOCALE, in that the system senses changes in itself and its environment, and determines the effect of the changes on the currently active set of business policies.

4 Autonomic Element Comparisons

This Section compares the strengths and weaknesses of the three AEs described; a summary is provided in Table 1.

The control loops of the three approaches are very different, and are summarized in the first 3 rows of Table 1. IBM has a single, pre-defined control loop, whereas CASCADAS and FOCALE both use adaptive control. FOCALE is the only approach that uses multiple control loops, and does so to separate the effect of large changes, such as those from applying new policy rules, from small changes, such as fine-tuning of parameters controlled by a set of policy rules. Both FOCALE and IBM use policy rules to govern control loop functionality.

IBM and FOCALE use a well-specified form of policy, which is lacking in the CASCADAS approach. As seen in the next two rows of Table 1, FOCALE uses a *set* of policy languages, which enables different constituencies to express policies in terms that are natural to them. IBM is limited to a single policy language.

The next 5 rows of Table 1 describe the respective knowledge bases of each AE. Both CASCADAS and FOCALE use dynamic knowledge bases, which IBM doesn't. Data scalability is limited in the IBM approach by the monitoring stage – if it cannot

Table 1. Comparison of IBM, CASCADAS, and FOCALE AE Approaches

Aspect	IBM	CASCADAS	FOCALE
Static or Adaptive Control	Static	Adaptive	Adaptive
Multiple Control Loops	Single	Single	Multiple
Policy Driven Control Loop	Yes	No	Yes
Policy Languages	One	None	Multiple
Policy Languages for Different Constituencies	No	No	Yes
Dynamic Knowledge Base	No	Yes	Yes
Data Scalability	Limited	Scalable	Scalable
Accommodates Heterogeneous Data	Limited by Common Base Event	Specific plugins can be added for new data	Yes, through model-based translation
Data Complexity	Limited to data that can be instrumented	Specific plugins can be added for new data	Uses patterns, models, ontologies to parse data
Data Semantics Encoded	No	Yes	Yes
AE Complexity	High	Low	Low
AE Components Distributed	No	Yes	Yes
Uses Ontologies	No	Could in Future	Yes
Semantic Matching	No	Yes	Yes
Self-Organizing	No	Yes	Yes
Uses Self-Model	No	Yes	Yes
Publishes Self-Model?	No	Yes	No
Model-Based Supervision	No	Yes	Yes
State-Driven	No	Yes	Yes
Adaptable State Machine	No	Yes	Yes
Autonomic Behavior - Loosely or Tightly Coupled	Tightly Coupled	Both	Both
Context-Aware	No	Yes	Yes
Supports Emergent Functionality	No	Yes	Yes
Communication Mechanism	Pre-Defined Interfaces	Messages and interfaces	ECB messaging and interfaces

keep up with the data that is being sent to it, the entire system suffers. In addition, the IBM approach is limited to data that is pre-defined in its static knowledge base, and offers no way to encode data semantics. This is one of the key limitations of the IBM architecture, as the knowledge repository is populated with data defined at development time, and cannot accept new or changed data discovered at runtime. In contrast, CASCADAS supports plugins for accommodating new data. Specific interfaces contain semantic descriptions of the set of features that a given AE can perform, and the conditions and actions required to accomplish each feature. FOCALE uses a combination of models and ontologies to generate software patterns to understand and learn from data.

In the IBM architecture, the AE combines the resource to be managed with the autonomic manager. Thus, self-awareness (of the AE) and awareness of the external environment of the AE are each limited. In contrast, both CASCADAS and FOCALE were designed as distributed systems from the start.

The IBM AE does not use semantics. CASCADAS uses semantics to determine decisions and also to route knowledge. FOCALE is similar, except that it uses both models and ontologies to represent different types of knowledge.

The IBM AE is not self-organizing; it cannot adapt to change, as its behaviors are controlled by the plan and execute parts using statically defined knowledge. Both FOCALE and CASCADAS use models and semantics, along with machine-based learning and reasoning, to organize their components to adapt to change. This enables them to support *emergent functionality.*

IBM does not provide state automata for management, whereas both FOCALE and CASCADAS do. In addition, both FOCALE and CASCADAS enable parts of their models to be changed. A unique feature of the CASCADAS AE is the publishing of its "self-model", which defines the possible states and their associated transitions. The self-model can be *dynamically adapted*. FOCALE does not publish its model, since it uses a complex governance model.

Contrary to both the IBM and FOCALE approaches, CASCADAS does not have a well-developed notion of policy; rather, it discovers deviations from the desired state of an entity and takes corrective actions to return that entity to its desired state. FOCALE is similar to CASCADAS, except it uses context-aware policies, and has a set of control loops that adjust the functionality of the autonomic manager according to changing context.

5 Summary and Future Work

This paper describes the key role that autonomic systems can play in managing the Future Internet and its applications. An AE enables a communications façade to be used to control managed resources and services that may or may not be autonomic using a common set of governance mechanisms.

Three different types of AEs were described and compared. CASCADAS and FOCALE offer significant advantages over the IBM approach in terms of scalability, ability to handle complex data, self-organization of the components of the AE, and communications flexibility. FOCALE has several unique features that provide advantages over CASCADAS, such as its ability to use different types of policy rules authored by different constituencies (e.g., network as well as business people), ability to accommodate heterogeneous data, use of an ECB, and more powerful control mechanisms that provide additional functionality.

FOCALE uses a model-based approach and emphasizes dynamic code generation. This reduces operational expenditures by removing the human from the configuration and monitoring processes in most circumstances, as changes to devices are dynamically constructed and applied under FOCALE's supervision. This emphasis in code generation drives the use of models and ontologies, from analysis and design through implementation and deployment. As such, a related goal of FOCALE is to build a knowledge framework that enables *reusable libraries of behavior* to be built that are

then *matched based on their semantics*. This combination of extensions helps FOCALE based applications realize the important vision of Weiser's *invisible management* in Ubiquitous Computing systems.

Future work will concentrate on building different FOCALE applications as specified in the World Class University research program for Ubiquitous Health and Ubiquitous Environment applications. We will report our results as soon as possible.

Acknowledgments. This work is sponsored in part by the WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0). This work is also partially sponsored by Science Foundation Ireland under grant number 08/SRC/I1403 (FAME).

References

1. Clark, D., Sollins, K., Wroclawski, J., Katabi, D., Kulik, J., Yang, X., Braden, R., Faber, T., Falk, A., Pingali, V., Handley, M., Chiappa, N.: NewArch: Future Generation Internet Architecture, NewArch Final Technical Report, http://www.isi.edu/newarch/
2. Blumenthal, M., Clark, D.: Rethinking the design of the Internet: the end to end arguments vs. the brave new world. ACM Transactions on Internet Technology 1(1), 70–109 (2001)
3. Feldmann, A.: Internet clean-slate design: what and why? ACM SIGCOM Computer Communication Review 37(3) (2007)
4. Kim, S., Won, Y., Choi, M., Hong, J., Strassner, J.: Towards Management of the Future Internet. Accepted for publication at the 1st IEEE/IFIP Workshop on Management of the Future Internet, Long Island, NY, USA, June 5 (2009)
5. McKeown, N., Girod, B.: Clean slate design for the internet, April 2006 Whitepaper (2006), http://cleanslate.stanford.edu
6. Harrington, D., Preshun, R., Wijnen, B.: An Architecture for Describing Simple Network Management Protocol Management Frameworks, RFC3411 (December 2002)
7. Cisco, http://www.cisco.com/warp/cpropub/45/tutorial.htm
8. Schonwalder, J., Pras, A., Martin-Flatin, J.-P.: On the future of Internet management technologies. IEEE Communications Magazine 41(10), 90–97 (2003)
9. Kim, S.-S., Choi, M.-J., Hong, J.W.: Management requirements and operations of Future Internet. In: Ma, Y., Choi, D., Ata, S. (eds.) APNOMS 2008. LNCS, vol. 5297, pp. 156–166. Springer, Heidelberg (2008)
10. Kephart, J., Chess, D.: The Vision of Autonomic Computing. IEEE Computer 36(1), 41–50 (2003)
11. Strassner, J.: Autonomic Networking – Theory and Practice. In: 2008 IEEE Network Operations and Management Symposium (NOMS) Tutorial, Salvador Bahia, Brazil (2008)
12. Future Networks and Services – Developing the Future of the Internet through European Research, ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/futint-book_en.pdf
13. Weiser, M.: Open House. In Review, the web magazine of the Interactive Telecommunications Program of New York University (March 1996)
14. Strassner, J., Agoulmine, N., Lehtihet, E.: FOCALE – A Novel Autonomic Networking Architecture. ITSSA Journal 3(1), 64–79 (2007)

15. Strassner, J.: Enabling Autonomic Network Management Decisions Using a Novel Semantic Representation and Reasoning Approach, Ph.D. thesis (2008)
16. Manzalini, A. (ed.): Deliverable 1.1, Report on state-of-art, requirements and AE model, CASCADAS Project, January 9 (2007)
17. Christudas, B.: Service-Oriented Java Business Integration: Enterprise Service Bus Integration Solutions for Java Developers. Packt Publishing (August 2008)
18. Boyd, J.R.: The Essence of Winning and Losing, June 28 (1995)
19. Strassner, J., de Souza, J., Raymer, D., Samudrala, S., Davy, S., Barrett, K.: The Design of a Novel Context-Aware Policy Model to Support Machine-Based Learning and Reasoning. Journal of Cluster Computing 12(1), 17–43 (2009)
20. Strassner, J., de Souza, J., van der Meer, S., Davy, S., Barrett, K., Raymer, D., Samudrala, S.: The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking. Journal of Network and Systems Management 17(1) (March 2009)
21. van der Meer, S., Davy, S., Davy, A., Carroll, R., Jennings, B., Strassner, J.: Autonomic Networking: Prototype Implementation of the Policy Continuum. In: 1st IEEE International Workshop on Broadband Convergence Networks, pp. 1–10 (2006)
22. Davy, S., Jennings, B., Strassner, J.: The Policy Continuum – A Formal Model. In: 2nd International IEEE Workshop on Modelling Autonomic Communications Environments, Multicon, Berlin. Multlicon Lecture Notes, vol. 6, pp. 65–78 (2007)