Towards the Development of an Autonomic Network Management Architecture

John Strassner[1,2]
[1]Telecommunications Systems & Software Group
Waterford Institute of Technology
Carriganore, County Waterford, Ireland
jstrassner@tssg.org

James Won-Ki Hong[2]
[2]Dept. of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, Korea
{johns,jwkhong}@postech.ac.kr

***Abstract*—The current Internet did not define any inherent management constructs and mechanisms; such concepts were added *after* networking standards and architectures were constructed. This also influenced network management for other types of networks. This paper is the first in a series that explores concepts for a new autonomic approach to network management that can be used for current and next generation networks as well as for the Future Internet.**

Keywords-autonomic management; Future Internet; network management; next generation network

I. INTRODUCTION

Management of the current Internet is based on OSI [1][2], TMN [3][4], and/or IETF principles [5][6][7], all of which were defined after the network functions that were to be managed were implemented, thereby complicating their management. There are also numerous studies of the Future Internet. The majority of these list network management as a problem; however, the vast majority of these studies do not offer any detailed analysis of the nature of the problem nor any concrete ideas on how to solve it.

Network management can be defined as all activities needed to operate communication networks and services in a secure and effective way. This includes the initialization, monitoring, modification, shutting down, and administration of functions that govern the services and resources provided by the network. The fundamental problem with the current Internet is that there is no standard management architecture that can be used to ensure interoperable communication and governance between heterogeneous management applications. This is needed because of the proliferation of vendor-specific devices that implement common functions and services in different ways, using different programming models having diverse formats and languages for representing management data. In general, each application brings a new concept for managing resources and services, and hence redefines similar concepts. For example, a configuration management application and a billing application can each define the concept of a "user" in different ways, having different attributes and datatypes; this complicates the exchange of management data and, more importantly, understanding the significance of those data.

Since a common management framework does not exist, middleware has been commonly used to "glue together" different management applications. This is fraught with problems, as middleware is fundamentally about distribution of information, not management.

There are two distinct problems in managing current networks: (1) there is no way to embed management data and monitoring data or functions in the network communication functions, and (2) since there is no standard architecture, there is nothing to stop different vendors from implementing management applications and functions in their own ways. This prohibits the sharing and reuse of common management information, and gives rise to stovepipe applications that redefine the same data and concepts for their own use [8].

This paper proposes that autonomic principles and systems can be used to manage next generation networks as well as the Future Internet, and is organized as follows. Section 2 provides requirements for essential network management functions. Section 3 describes how autonomic principles can be applied to network management. Section 4 introduces a set of extensions to the FOCALE Autonomic Architecture to realize the concepts described in this paper. Section 5 presents a summary and future work.

II. NETWORK MANAGEMENT REQUIREMENTS

Current telecommunications network management operate using *layered* principles, requiring the management of each technology layer, along with any management information that requires multiple layers to cooperate on common tasks. Inter-layer interactions are different for different applications, which further complicate management.

The above in turn assumes a constant context. Situations such as the network undergoing a malicious attack, as well as benign environments that seek to provide context-aware, personalized services that change according to user needs and/or environmental conditions, need more sophisticated management approaches. In order to change the services offered in response to context changes, an entity must be able to understand its own behavior as well as that desired of it. Most existing autonomic systems use a control loop to collect information regarding the state of the entity being managed. The problem with this approach is that sensing and interaction

is directed towards the controlled element itself and not towards the environment of the system. However, there is a more fundamental problem: most control loops have limited intra-loop interaction mechanisms, and tend to perform actions in a sequential manner (e.g., the monitor-analyze-plan-act cycle of [10]). This means that there is no opportunity for data being monitored to be properly oriented towards changing goals. As we observe mismatches between what was desired and what actually happened, we have to change our orientation to provide new corrective actions.

This context-aware ability is one of the key hallmarks of the Future Internet. Context ushers in additional concerns that are beyond the technical operation of a network, including economic and social considerations. For example, the vast diversity in business, system, and technical requirements makes it very difficult to correlate concepts from different constituencies. Self-management, in the form of autonomics, offers a way to overcome the complexities of network management described above and in more detail in [9][11].

In order to realize self-management, every network and network component must have the ability to manage all or part of its own functionality. Note that *a node could still help manage other nodes as long as a core set of functionality is working*. This enables network management to be distributed among applications that are internal and external to the network. Internal management applications do not have to concern themselves about global state and other factors, such as business requirements – they instead ensure that the network infrastructure is performing as expected, and provide critical operational and performance data to external management applications. These external applications can then relate these data to different application-specific requirements.

III. AUTONOMIC NETWORK MANAGEMENT DESIGN GOALS

The preceding section has shown the need for a paradigm shift in network management. For example, how does an SNMP (Simple Network Management Protocol) alarm relate to the Service Level Agreements (SLAs) of a given customer? If we cannot relate fundamental network data to business concepts, how can we make the network responsive to changing business demands? [8][11] The IETF has launched several "evolutionary approaches" to improving SNMP, all of which "have failed or had limited market acceptance. The IETF accepted this failure recently." [28]

There have been a variety of interesting (but incomplete) proposals that brush on the subject of network management. [12] describes a "role-based architecture" (RBA) that, instead of using layers, organizes communication in functional units referred to as "roles". Note that the use of the term "role" is not the same as that used in software patterns; roles in the RBA are not even hierarchically organized. The motivation for RBA was to address the frequent layer violations that occur in the current Internet architecture. This subject is covered in more detail in [13]. The SILO [14] approach also introduces a non-layered design based on "silos" of services assembled on demand and specific to a given application and network environment. In contrast to RBA, the goal of SILO was to facilitate "cross-layer" interactions. The following subsections briefly describe the salient features of our new architecture that were designed to meet these and other challenges. Each section will define if this feature is an original, enhanced, or new feature of our new architecture. A high-level block diagram of the FOCALE architecture is shown in Figure 1, with callouts so the reader can relate the following subsections to the architecture.

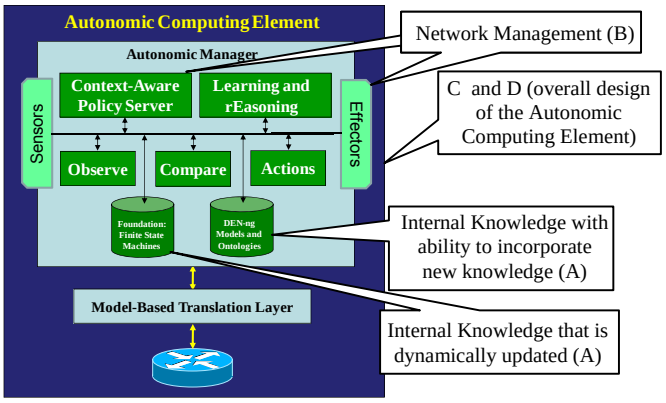

Figure 1. The FOCALE Autonomic Architecture

A. Internal and External Knowledge

An autonomic network node requires knowledge, both of itself as well as of its neighboring nodes, systems, and environment that it is interacting with. From a management point of view, some of this knowledge must be inherent, and some must be provided externally. For example, if a device is to be able to "configure itself", then it should have an inherent understanding of what each configuration command does, and how the execution of that command will affect both its state as well as the state of the system in which it is operating in. Current management approaches do not have this knowledge. However, this was a basic design premise of FOCALE [22].

This requirement has profound effects on the design and use of management data and applications. The design of most network device languages and their associated data [11][15] are not structured to provide commands that are free of side effects or have multiple purposes. More importantly, the underlying language is neither formal nor complete, which means that machine-based learning and reasoning cannot be done. As another example, external knowledge is often coupled with internal knowledge to provide application-specific behavior. Unless both forms of knowledge can be translated into a form where their syntax and semantics are similar, they cannot be combined. (Note that we are *not* recommending Yet Another Programming Language! We are instead recommending the mapping of vendor-specific languages and data to a common form, which is an extension to the FOCALE approach [11].)

B. Core Internal Network Management Functionality

Each node must have a common set of management functions. While the functions listed below are part of our original architecture, we have enhanced that architecture to remove the requirement that all nodes use the same protocol; this decouples the protocol from the task that it is performing, and instead uses it as a means to communicate data and commands. This is in stark contrast to current networks, which

almost always use a vendor-specific language and a set of protocols to configure the device and either that language or SNMP [16] to obtain management data from the device. The use of multiple protocols is a new feature of our approach.

The following capabilities must be supported for a network node to interface with other network nodes, systems, and applications: (1) a set of common communication facilities (shown as sensors and effectors in Figure 1 to avoid clutter), (2) a means for the node to describe its capabilities (provided by the Finite State Machines, Models, and Ontologies), and (3) what functions the node can perform at the current or a given future time (provided by the Finite state Machines and the Learning and Reasoning components). The first two are part of our original approach, while the third is an extension to our model-driven architecture that takes the form of exchanging appropriate portions of the internal model of the node.

The following capabilities must be supported to obtain operational and management data: (1) self-inspection (this is directed by the Context-Aware Policy Server (CAPS)), (2) a set of standard responses to a set of standard queries (provided by the CAPS), (3) using a standard way to inspect its environment and providing a standard way to communicate the results of the inspection (through the Model Based Translation Layer), and (4) problem reporting in a standard format (through the Sensors and Effectors). These were part of our original design.

C. Core External Network Management Functionality

There are a number of common principles that have been used to build distributed systems; these were all part of the original FOCALE design. These include (1) abstraction, (2) reusability, (3) software components, (4) software contracts, and (5) discoverability. There are others, of course, such as statelessness, but these are beyond the scope of this short paper.

1) Abstraction is the process of reducing the information content of a concept or an observable phenomenon in order to retain only the information which is relevant for a particular purpose or mechanism. We use the DEN-ng information model [17] to provide a detailed object-oriented information model (i.e., a model that is independent of platform, language, and protocol) that describes entities and their relationships in the Unified Modeling Language (UML) [25].

2) Reusability enables a design to be repurposed. Our approach uses sets of application-independent reusable objects that function as a library for building application-specific solutions. This goes beyond the traditional use of building objects that reuse classes and class attributes, and instead creates behavioral libraries whose corresponding code can be generated at runtime.

3) Software Components [19] are reusable software elements that offer a set of pre-defined services that enable components to interact in an expected fashion. Software components encapsulate their implementation, and offer a standard way to access services. They represent a standard unit of deployment and versioning.

4) Software Contracts [20][21] enable each component to communicate in a standard way based on specifications of mutual obligations. Component interaction is specified using at least three fundamental mechanisms: pre- and post-conditions (to ensure that all requisite behavior before and after the contract has executed are satisfied) and invariants (characteristics of the components that do not change during the life of the contract). This enables contracts to specify semantics that other mechanisms, such as interfaces, may not be able to (especially in a standard way).

5) Discoverability is the ability to find objects of interest that were not previously known. Our architecture must be able to translate different types of data dynamically into a form that can be stored and later reasoned about.

D. Management Lifecycle

Given the changing demands of businesses and users, as well as the introduction of new devices and technologies, the management process must not be conceived of as a fixed solution. Rather, it is a process that can continually evolve as long as it has defined rules that govern its creation and extension [21], and enables solution-agnostic, entity-centric (as opposed to task-centric) processes to be used. This is a new extension to the FOCALE approach.

E. The Role of Metadata

One of the unique design features of DEN-ng is that it has a well-developed metadata hierarchy [8][24]. In DEN-ng, there are only three subclasses of the Top class (which shows a strict use of classification theory): Entity, Value, and Metadata. The rich metadata hierarchy of DEN-ng increases flexibility in modeling concepts by separating characteristics and behavior that is inherent to an Entity from ways in which that Entity is used (e.g., the current *role* that it is playing). The use of metadata also avoids embedding explicit references to other Entities, thereby simplifying the configuration and provisioning of Entities and the Services that they provide. This feature has evolved as the DEN-ng model has evolved.

F. There Can Never Be "One" Internet!

There can never be "one" Internet. Fundamentally, there are different applications whose traffic has different needs. Therefore, the design of FOCALE has concentrated on building a *reusable* and *programmable* framework that can be applied to different scenarios. This is a complex subject that is beyond the scope of this paper. However, the following observations can help the reader appreciate the scope of this problem.

a) Does it really make sense to try and build mission-critical applications, along with applications that require differentiated service, on an infrastructure that cannot support anything other than best effort service?

b) Converged networks, which were the rage of the 90s and early 2000s, are struggling to meet the demands of new services, let alone meet new requested availability, security, prvacy, anonymity, and various other legal constraints.

c) There is a growing differentiation between different networks having different technologies or uses, such as core, access, and distribution networks, fixed versus wireless versus ad-hoc networks, and networks devoted to different services.

d) How are economic considerations transferred into networks? Clearly, converged networks make this very difficult. Examples include:

- Effective pricing and business models for bundled versus personalized services
- Efficient data monitoring and accounting for bulk versus personalized tariff schemes
- Integration of policy management and pricing/tariffing approaches
- Automated auditing of application-level and third party service consumption
- Making services aware of changing contexts

IV. A New Autonomic Network Architecture

Current network management optimizes data plane traffic by applying control plane mechanisms as needed. The problem with this is that manageability is an afterthought, making it impossible to configure network services and resources to support the realization of business objectives, particularly across heterogeneous networks with different ownership and possibly conflicting objectives. Furthermore, different control plane mechanisms are used for different networks, (e.g., wired vs. wireless) that cannot be easily coordinated. Due to space limitations, we will only describe extensions to our approach that was described in [9].

FOCALE [22] is a model-driven [27] architecture that can dynamically generate code to (re)configure managed elements. This is critical in autonomic systems, which seek to manage the ever-growing complexity of configuration by automating as many management operations as it is safe to do so. Automation is performed by machines that can generate configuration changes, instead of humans that cannot respond fast enough and can make costly errors that are hard to detect. More importantly, there is a pronounced shortage of skilled humans that can perform these tasks.

Four of the most important architectural features of [9] were: (1) the explicit recognition of context and business goals and their translation using the Policy Continuum [23] (a mechanism for enabling policies for different constituencies to be related to each other), (2) support for multiple programming models and languages by using a novel combination of models and ontologies to build a vendor-independent language that vendor-specific data and commands could be mapped to, (3) true reuse of supporting tools and technologies, and (4) accommodation of emerging technologies, such as cognitive networking [11].

Our current work extends this approach in a number of ways. We now have a more powerful knowledge management representation, which is built on the combination of models and ontologies [11]. Conceptually, we construct a multi-graph, where nodes from model elements (e.g., UML classes and associations) are connected to nodes from ontologies by one or more semantic relationships. Our semantic relationship (for an example, see [26]) provides a ratioed result of how close the semantics of one element are to the semantics of another element. This is much more than a simple synonym relationship – rather, it uses formal logic to determine how close the *meanings of the two elements are*. An example is shown in Figure 2.

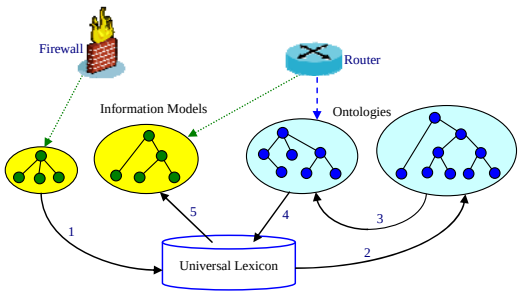

Figure 2. The FOCALE Knowledge Representation Process

The commands (and management data) of the firewall and router are both represented in an information model. Similar information are represented in associated ontologies. In step 1 of **Error! Reference source not found.**, a lexicon is used to relate data describing the firewall (in the information model) to a consensual set of terms, which are then used to form a mapping between terms in the information model and concepts in the ontologies. Specifically, model data describing the firewall are augmented with meanings from ontologies in step 2 using one or more linguistic relationships, such as synonyms or meronyms. For example, routers, firewalls, and switches are different types of devices, but share common functions, such as dropping and forwarding traffic. These generic terms (dropping and forwarding traffic) can be "attached" to our managed elements independent of vendor. Since network management represents a specific domain, we also define a set of custom relationships, such as "is similar to", to perform specialized mappings. An example of this type of mapping is to define the semantic similarity between a set of m commands from one vendor with a set of n commands from a different vendor.

Step 3 uses semantic relationships (an example of such an algorithm is given in [10]) to relate concepts describing the firewall to concepts describing the router. For example, the verb "accept" can be defined as both a synonym of the verb "forward" as well as an antonym of the verb "drop". Hence, the system now knows that the verb "accept" in the firewall language performs the same function as the verb "forward" in the router language (and also that it performs the opposite function of the verb "drop" in the router language). Step 4 then ensures that these relationships are entered in the lexicon, and step 5 uses these relationships to search for additional model elements (e.g., classes and associations) in the information model for the router. The process iterates as necessary. This knowledge can then be used to generate the appropriate CLI commands that are issued to reconfigure each device.

This is an important tool in both network management (e.g., for performing root cause analysis) as well as in associating business concepts (e.g., SLAs and revenue) to network resources and services (e.g., a fault that adversely affects the traffic in a VPN). Current model-driven architectural approaches, such as MDA [27], cannot solve this problem for two reasons. First, a model would not contain an association between an SLA and a SNMP alarm because this is not an

inherent characteristic of either the SLA or the alarm. Second, UML lacks the ability to reason, so it cannot infer such a relationship. Put another way, one could represent a relationship such as a synonym using a UML association, but UML has no ability to define the semantics of what a synonym means, and how it is distinct from other relationships. This was one of the points of [11]: use the power of models to generate code, but use the power of formal logic to reason about the code before we generate it. For example, our knowledge representation uses the semantic relationships between model elements and ontological elements to infer which SLAs for which customers are affected by a given alarm.

Another important extension is the support of advertisement and negotiation of capabilities in an extensible manner. Capabilities are a key abstraction of the DEN-ng model, and can be thought of as a "summary" of the salient characteristics and behavior of a managed entity. In autonomic networks, people and devices often need to negotiate capabilities to use in order to collaboratively fulfill certain tasks. Capability negotiation often involves multiple negotiation objects such as quality of service parameters, security mechanisms and algorithms, reliability, and other features, which translates to a computational expensive multi-dimensional negotiation problem. We have created a graph-based negotiation mechanism in which multiple features to negotiate are represented as objects; this then enables appropriate governance mechanisms, such as a utility function, to be applied to the object. There are two important advantages of this approach: (1), it converts a multi-dimensional negotiation problem into a much simpler one-dimensional negotiation problem, and (2) it enables the negotiation to be orchestrated using policy rules and/or utility functions. The utility function is a function that, given a set of capabilities as input, outputs a value representing the overall preference (or benefit) for the set of capabilities to be negotiated. The utility functions used in this invention are defined based on each negotiator's capabilities, policies and constraints. The advantage of using a utility function is that offers and counter-offers can reflect the negotiator's preference over one or a set of capabilities. In addition, multiple capabilities can be combined into a single negotiation. Finally, the utility function can use either discrete or continuous capability values.

Another major enhancement is the use of virtual machines, not just to provide functionality, but also to isolate problems. This enables the managed environment to emulate the problem so different solutions can be investigated without harming the rest of the network. This technique can be effectively used as long as the infrastructure supports creating and tearing down such virtual machines. Our approach is shown in Figure 3.

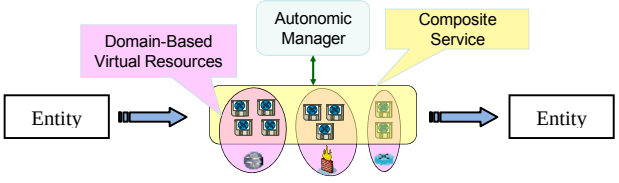

Figure 3. FOCALE Management of Virtual Services

Figure 3 shows a composite service that is constructed from multiple virtual devices that each offer different functionality. In the FOCALE approach, each device has a set of *roles* that describe the characteristics of a feature, such as a Service. Each device that supports this Service belongs to an administrative domain that the autonomic manager governs; in this example, each administrative domain consists of a set of virtual devices that are hosted on a set of physical devices. Hence, the Service is completely decoupled from the devices that are supporting it.

The fourth major extension is shown in Figure 4. In this figure, each of the five nodes represents a distinct state that has an associated configuration (e.g., commands to enable a port on a router). Behavior is orchestrated by controlling the set of state transitions to be applied. Our novel implementation uses Policies to determine the *cost* of the edges; this determines the the set of state transitions to be executed. In Figure 4, Policy 1 and Policy 2 ar executed, which causes Action A_1 of P_1 and Action A_2 of P_2 to set the cost of edges E_{ab} and E_{bc}. Meta-policies can be control which set of policies are applied to which set of edges in the graph, which enables policy to be used to prefer one path over other paths in the graph.

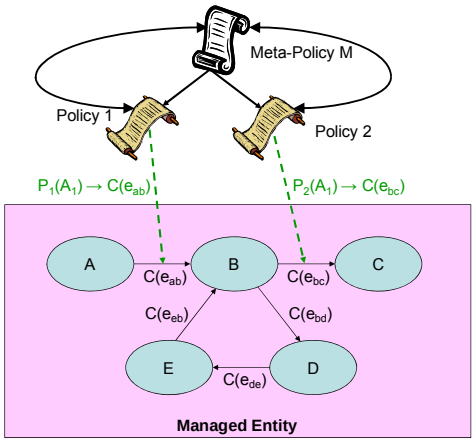

Figure 4. Policy-Based Orchestration in FOCALE

The advantage of this approach is that it is generally considered unsafe to implement a number of configuration changes without any intermediate backup of the configuration. Our approach enables context-sensitive semantics to be used to inspect the outcome of a policy and reason as to whether other changes should be made. Each policy is a set of objects in DEN-ng, which enables each policy component (e.g., events, conditions, and actions) to be related to a set of nodes in one or more ontologies. Hence, the above knowledge representation can be used to select ontological data to be used to reason about the policy and/or its components.

V. SUMMARY AND FUTURE WORK

This paper is the first in a series of papers that will describe a new approach to network management – one that uses an autonomic architecture that can manage existing networks, evolutionary changes to the current Internet, and revolutionary (i.e., clean-slate) approaches to the Future Internet.

The introduction pointed out that very little progress in network management has been made. Section II elaborated on this, and delineated difficult problems, such as the lack of association between business concepts and network resources and services, that lie ahead. Section III describes the design goals of our architecture, focusing on knowledge management, separating internal and external network management, providing an extensible set of core services that facilitate distributed management, governing the lifecycle of management operations, separating metadata from the behavior of an entity, and addressing the likelihood that the "Future Internet" is in reality a collection of "Internets". Section IV described how our previous work [9] was extended to include: (1) providing a powerful knowledge representation that can map vendor-specific data and commands to vendor-neutral forms, as well as can integrate diverse data from different sources to provide an integrated, contextually-sensitive view of the system and its environment, (2) a structured methodology to associate business requirements with network resources and services, (3) supporting the advertisement and negotiation of device and system capabilities in an extensible manner, and (4) using virtual machines to provide functionality as well as to diagnose problems.

Future work will be done in multiple areas. Our first will be exploring how management decisions can be made in the face of uncertain, inaccurate, and/or incomplete data. Important tasks include determining when management data is corrupted, taking into account probabilistic behavior, and detecting cheating and falsification of data.

ACKNOWLEDGMENT

This work is partially sponsored by Science Foundation Ireland under under grant numbers 03/CE3/I405 (AMCNS) and 08/SRC/I1403 (FAME). It is also partially sponsored 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).

REFERENCES

[1] ISO, ISO/IEC 7498-4 (ITU-T X.700), "Information Processing Systems – Open Systems Interconnection – Basic Reference Model", Geneva, 1989

[2] ISO, ISO/IEC 10040 (ITU-T X.701), "Information Processing Systems – Open Systems Interconnection – Systems Management Overview", Geneva, 1992

[3] ITU-T: "Series M: Principles for a Telecommunications Management Network", Recommendation M.3010, Geneva, February 2000

[4] ITU-T: "Series M: Generic Network Information Model", Recommendation M.3100, Geneva, April 2005

[5] D. Harrington, R. Preshun, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol Management Frameworks", RFC3411, STD0062, December 2002

[6] D. Levi, P. Meyer, B. Stewart, "Simple Network Management Protocol Applications", RFC3413, STD0062, December 2002

[7] M. MacFaden, D. Partain, J. Saperia, W. Tackabury, "Configuring Networks and Devices with Simple Network Management Protocol (SNMP)", Informational RFC, April 2003

[8] J. Strassner, "Autonomic Networking – Theory and Practice", 20th Network Operations and Management Symposium (NOMS) 2008 Tutorial, Salvador Bahia, Brazil, April 7, 2008

[9] J. Strassner, M. Ó Foghlú, W. Donnelly, N. Agoulmine, "Beyond the Knowledge Plane: An Inference Plane to Support the Next Generation Internet", IEEE Global Information Infrastructure Symposium (GIIS 2007), 2-6 July, 2007, pages 112-119

[10] IBM, "An Architectural Blueprint for Autonomic Computing", v4, June 2006

[11] J. Strassner, "Enabling Autonomic Network Management Decisions Using a Novel Semantic Representation and Reasoning Approach", Ph.D. thesis, 2008

[12] D. Clark, K. Sollins, J. Wroclawski, D. Katabi, J. Kulik, X. Yang, R. Braden, T. Faber, A. Falk, V. Pingali, M. Handley, N. Chiappa, "NewArch: Future Generation Internet Architecture", NewArch Final Technical Report

[13] J. Strassner, S. van der Meer, J. Won-Ki Hong, "Autonomic Management of Communications Networks Without Using the "L" Word", submitted to JSAC special issue on Advances in Autonomic Communications, 12 January 2009

[14] I. Baldine, M. Vellala, A. Wang, G. Rouskas, R. Dutta, D. Stevenson, "A Unified Software Architecture to Enable Cross-Layer Design in the Future Internet", Proceedings of 16th International Conference on Computer Communciations and Networks, pages 26-32, 13-16 Aug. 2007, Honolulu, HI

[15] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure of Management Information v2", RFC 2578, STD 58, April 1999

[16] D. Harrington, R. Preshun, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol Management Frameworks", RFC3411, STD0062, December 2002

[17] J. Strassner, "DEN-ng Model Overview", Joint ACF, EMANICS, and AutoI Workshop on Autonomic Management in the Future Internet, May 14, 2008

[18] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994, ISBN 0201633612

[19] C. Szyperski, "Component Software: Beyond Object Oriented Programming, 2nd edition, Addison-Wesley, 2002

[20] B. Meyer, "Applying Design by Contract", IEEE Computer Vol. 25 (10), 1992, pages 40–51

[21] S. van der Meer, J. Fleck, M. Huddleston, D. Raymer, J. Strassner, W. Donnelly. "Manageability of Autonomic Software Artifacts using Contracts and Traceability", Proceedings of the 2nd IEEE International Workshop on Modelling Autonomic Communications Environments (MACE 2007), October 29-30, 2007. San Jose, USA

[22] J. Strassner, N. Agoulmine, E. Lehtihet, "FOCALE – A Novel Autonomic Networking Architecture", ITSSA Journal, Vol. 3, No. 1, May 2007, pages 64-79, ISSN 1751-1461

[23] J. Strassner, "Policy Based Network Management", Morgan Kaufman, ISBN 1-55860-859-1

[24] J. Strassner, J.N. de Souza, S. van der Meer, S. Davy, K. Barrett, D. Raymer, S. Samudrala, "The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking", Journal of Network and Systems Management, Volume 17, Issue, 1, March 2009

[25] Object Management Group: *OMG Unified Modeling Language Specification*, OMG, Version 1.5, March, 2003

[26] A. Wong, P. Ray, N. Parameswaran, J. Strassner, "*Ontology Mapping for the Interoperability Problem in Network Management*", IEEE Journal on Selected Area in Communications, Vol. 23, No. 10, October 2005, pages 2058 – 2068

[27] www.omg.org/mda

[28] J. Schönwälder, A. Pras, J.Martin-Flatin, "On the Future of Internet Management Technologies", IEEE Communications Magazine, October 2003, pages 90-97