FOCALE – A Novel Autonomic Networking Architecture

John C. Strassner[1], Nazim Agoulmine[2], Elyes Lehtihet[2,3]

[1]Autonomic Research Lab – Motorola Labs, Schaumburg, IL 60010, USA

[2] Networks and Multimedia Systems Group – University of Evry Val d'Essonne
Evry Courcouronnes, France

[3]Telecommunications System and Software Group
Waterford Institute of Technology, Waterford, Ireland

john.strassner@motorola.com, nazim.agoulmine@iup.univ-evry.fr, elehtihet@tssg.org

Abstract. *Network resources will* **always** *be heterogeneous, and thus have different functionalities and programming models. This can be solved through the combination of information models and knowledge engineering, which together can be used to discover and program semantically similar functionality for heterogeneous devices regardless of the data and language used by each device. This paper introduces FOCALE, a semantically rich architecture for orchestrating the behavior of heterogeneous and distributed computing resources. We apply the FOCALE architecture to Beyond 3G Networks as a case study.*

1. Introduction and Motivation

Current voice and data communications networks present a set of difficult management issues. The stovepipe systems that are currently common in OSS and BSS design exemplify this [Str04] – their desire to incorporate best of breed functionality prohibits the sharing and reuse of common data. There are five principle reasons for this [Str05]:

1. Separation of business- and technology-specific information that relate to the same subject
2. Inability to harmonize network management data that is inherently different
3. Inability to cope with new functionality and new technologies due to lack of a common design philosophy used by all components
4. Isolation of common data into separate repositories
5. Inability to respond to user and environmental changes over the course of the system lifecycle

The above problems are caused by the inability to manage the increase in operational, system, and business complexity [Str04a]. Operational and system complexity are spurred on by the exploitation of increases in technology to build more functionality. The price that has been paid is the increased complexity of system installation, maintenance, (re)configuration and tuning, complicating the administration and usage of the system. Business complexity is also increasing, with end-users wanting more functionality and more simplicity. This requires an increase in intelligence in the system, which is where autonomics will come in.

If autonomic computing, as described in [IBM01], [Kep03], and [Str02] is to be realized, then *the needs of the business must drive the services that the network provides*. This is the fundamental objective of our autonomic networking architecture.

The organization of this paper is as follows. Section 2 presents key terms used in this paper. A brief introduction to autonomic computing is described in Section 3. The proposed autonomic networking architecture (FOCALE) is presented in Section 4. Section 5 gives an overview of its application in the context of B3G networks. Finally, section 6 summarizes and concludes the paper.

2. Terminology

This section provides definitions of key terms used in this paper.

Component Name	Description
Autonomically Aware Managed Component (AAMC)	A component whose functionality can be autonomically managed through mediation; communication with other Autonomic Components must be done through mediation.
Autonomically Enabled Managed Component (AEMC)	A component that contains autonomic functionality, and hence can manage itself as well as communicate with other Autonomic Components.
Autonomically Unaware Managed Component (AUMC)	A component that has no inherent autonomic functionality, and cannot be autonomically managed (even through mediation).
Autonomic Manager (AM)	A component or system that orchestrates the behavior of other Autonomic Components in the system.
Autonomic Component (AC)	A component that is either an Autonomic Manager, autonomically aware or autonomically enabled
Autonomic Networking	The dynamic delivery of services and resources, as directed by business goals, wherein the nature of said services and resources *adapt* to the changing environment. Business goals use policies that use one or more closed control loops to adjust the set of services and resources delivered.
Data Model	The implementation of the characteristics and behavior of a component or system using a particular vendor, platform, language, and repository.
DEN-ng	A UML-compliant object-oriented information model that describes managed entities and their relationships. It has three salient characteristics: (1) it uses finite state machines to orchestrate behavior, (2) it is built using patterns and roles (making it inherently extensible), and (3) it consists of static as well as dynamic models.
Information Model	A representation of the characteristics and behavior of a component or system independent of vendor, platform, language, and repository.
Policy	A set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects.
Self-Governance	The ability of a system to conduct its own affairs.
Self-Knowledge	The ability of a system to model the characteristics and behavior of the users of the system, the environment in which the system operates, and the components making up the system, as well as the system itself.

3. Salient Points of Autonomic Computing

Autonomic computing is typically defined in terms of a set of self-functions, the usual being self-configuration, self-healing, self-optimization, and self-protection. These form building blocks that can be combined to realize more powerful functions. For example, current anti-virus (AV) software is a *reactive* mechanism. AV vendors issue a patch for problems, and then update subscribers. Even if the patch automatically downloads, some manual action is usually required (e.g., rebooting a machine). Furthermore, why burden the user with details about the patch, when the user does not understand the details or care? More importantly, it would be much better if the AV software knew what the user was currently doing, and did not interrupt the user if that task was critical. This reveals another important goal of autonomic computing – making the life of the user easier by changing the focus from a *computer-centric* to a *task-centric* model.

In order to implement any of these four self-functions, the system must first be able to *know itself* and its environment. Our approach supplies self-knowledge by using object-oriented information and data models augmented with ontologies. Deriving different data models from a *single* common information model provides data coherency and increased manageability [Str04a], [SID05]; ontologies provide inference mechanisms to *reason* using the facts defined in the models.

The technical aspects of controlling system complexity are well described in the literature. However, the business aspects of controlling complexity aren't. The TeleManagement Forum's NGOSS program [GB927], [TMF053], and [Str04b] has a well-articulated methodology to address the needs of different stakeholders (e.g., business, system, etc.) throughout the lifecycle of a solution. Previous and current efforts [Str04c] have addressed the potential link between NGOSS and autonomic computing, but have not addressed how to manage and integrate diverse knowledge present in the environment. This paper defines an architecture to address these needs.

One of the difficulties in dealing with complexity is accommodating legacy hardware and software. Current approaches to network configuration and management are technology-specific and network-centric, and do not take business needs into account. For example, SNMP and CLI are currently unable to express business rules, policies and processes, which make it impossible to use these technologies to directly change the configuration of network elements in response to new or altered business requirements [Str04a]. This disconnects the main stakeholders in the system (e.g., business analysts, who determine how the business is run, from network technicians, who implement network services on behalf of the business). The proliferation of important management information that is only found in different, technology-specific forms, such as private MIBs and the many different incompatible dialects of CLIs, exacerbates this problem. These are all symptoms of a more strategic problem: common management information, defined in a standard representation, is not available. There will most likely never be a single unified information model (just as there will never be one single programming language), but there must be an extensible modeling basis for integrating these diverse data. For this, the Shared Information and Data (SID) models [GB922] and DEN-ng [Str04a], [PBNM], [Ago04] are used as a unified information model; diverse knowledge is added to existing knowledge using a novel ontology-based approach, described section 4.5.

Autonomic operation of a system or component is achieved using a self-adjusting control loop [Str04, IBM03]. Inputs to the control loop consist of various status signals from the system or component being controlled, along with policy-driven management rules that orchestrate the behavior of the system or component. Outputs are commands to the system or component to adjust its operation, along with status to other autonomic elements. The approach used in this paper employs policy management [PBNM] to govern the autonomic control loop.

4. The FOCALE Architecture

The requirements behind the design of our architecture are:

- Accommodate legacy components
- Ensure that AEMCs can also be efficiently managed
- Provide a means to orchestrate the behavior of ACs
- Ensure that all ACs adjust their functionality in unison, according to policy
- Provide a means to reason about the environment and recommend or take appropriate actions, so that the underlying business goals are not violated and, hopefully, optimized

The following subsections describe the FOCALE architecture.

4.1. A Basic Autonomic Networking Control Loop

Figure 1 shows a basic autonomic networking control loop. Vendor-specific data from the Managed Resource is gathered, which is then analyzed to ensure that the Managed Resource is providing the appropriate services and resources at this moment in time. If it is, monitoring continues; if it isn't, then it is reconfigured and re-analyzed.

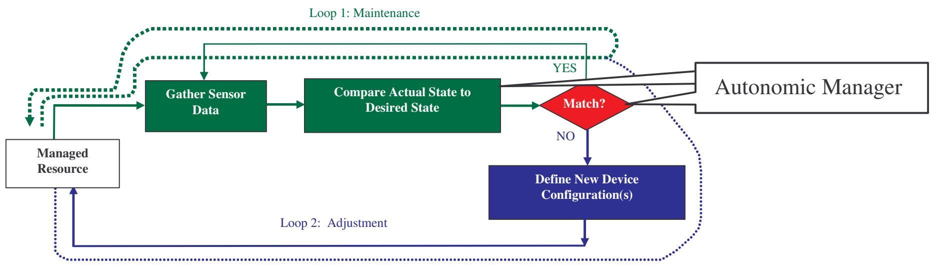


Figure 1. Simplistic Autonomic Networking Control Loop

The control loop involves comparing the current state of the Managed Resource to its desired state (which is pre-defined based on business goals [Str06b] [Str06d]. If this does not match, then the Managed Resource needs to be reconfigured. The AM is represented implicitly as the two functions "Compare Actual State to Desired State" and "Match". The reconfiguration process uses dynamic code generation based on models and ontologies. [Str06a] [Str06b] [Str06d].

Figure 2 shows a mediation layer that enables a legacy Managed Resource with no inherent autonomic capabilities to communicate with ACs. A model-based translation layer (MBTL), working with an AM, enables the Managed Resource to communicate with other ACs.

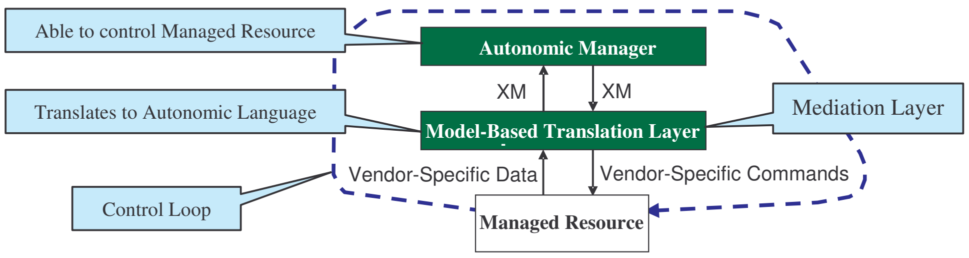
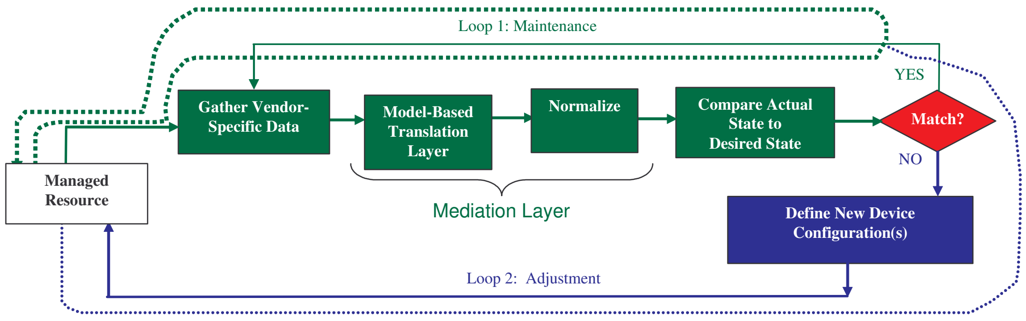


Figure 2. Generic Autonomic Control Loop for AAMCs

The motivation for the MBTL is as follows. Networks will be built out of heterogeneous devices that use different *languages and programming models*. For example, it is common to use an SNMP-based system to monitor the network, and a vendor-proprietary command line interface (CLI), because many devices cannot be configured using SNMP. Hence our dilemma: *there is no standard way to know which SNMP commands to issue to see if the CLI command worked as intended or not*. Furthermore, assuming that any device might have to communicate with any other device, there are *at least* n^2 programming models to manage. The MBTL solves this problem by accepting vendor-specific data and translating it into a normalized representation of that data, described in XML, for further processing by the AM. This reduces the number of input formats (n^2) to 1. The AM analyzes the data, and issues any reconfiguration actions required; these actions are converted to vendor-specific commands to effect the reconfiguration.

4.2. Managing Legacy Components

Figure 3 updates Figure 1 by including the mediation and normalization logic. This logic translates vendor-specific data from AAMCs to a single normalized form that the AM can use. It is also responsible for translating the single normalized form used by the AM into vendor-specific commands.



Loop 1: Maintenance
YES
Gather Vendor-Specific Data
Model-Based Translation Layer
Normalize
Compare Actual State to Desired State
Match?
NO
Managed Resource
Mediation Layer
Define New Device Configuration(s)
Loop 2: Adjustment

Figure 3. Adding Mediation into the Autonomic Networking Architecture

4.3. Managing Autonomic Components

Note that in Figure 3, the AM is limited to performing the state comparison and computation of the reconfiguration action(s), if required, *inside* the control loop. The placing of the control function outside the loop enables independent control over the constituent elements of the control loop. Figure 4 shows an enhanced architecture that uses *policy* to control each of the six architectural components shown in Figure 3. The Policy Manager is responsible for translating ***business requirements written in a restricted natural language*** (e.g., English) ***into a form that can be used to configure network resources***. [Str06a] This is done via the Policy Continuum [PBNM], [Str06b], which forms a continuum of policies in which each policy captures the requirements of a particular constituency. For example, business people can define an SLA in business terms; this is then transformed into architectural requirements and finally network configuration commands. Note that each policy of the SLA do not “disappear” in this approach – rather, they function as an integrated whole (or continuum) to enable the translation of business requirements to network configuration commands.

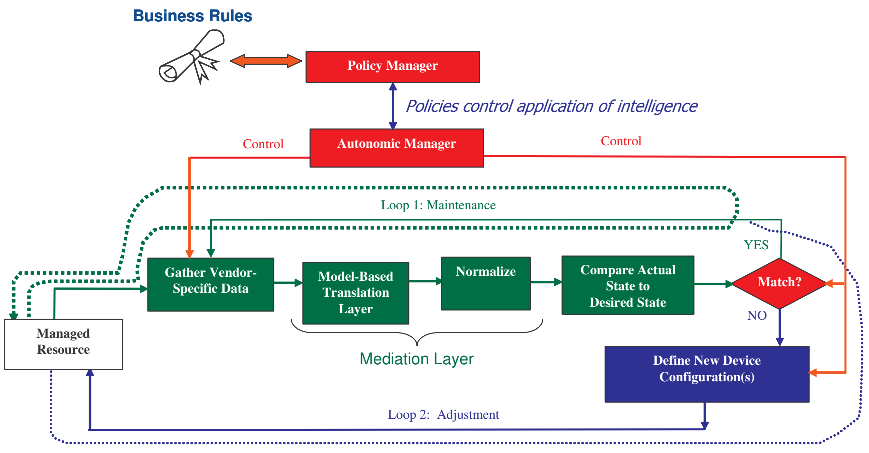

Business Rules
Policy Manager
Policies control application of intelligence
Control
Autonomic Manager
Control
Loop 1: Maintenance
YES
Gather Vendor-Specific Data
Model-Based Translation Layer
Normalize
Compare Actual State to Desired State
Match?
NO
Managed Resource
Mediation Layer
Define New Device Configuration(s)
Loop 2: Adjustment

Figure 4. Policy-Driven Autonomic Control Loop

Moving the AM from inside the loop to outside the loop (so that it can control ***all*** loop components) is a significant improvement over the state-of-the-art. For example, assume that computational resources are being expended on monitoring an interface, which suddenly goes down. Once the interface is down, it will stay down until it is fixed. FOCALE recognizes this, *stops monitoring this interface, and instead starts monitoring other entities that can help establish the root cause of* ***why*** *the interface went down*. This re-purposing of computational resources is directed by the AM, which controls each architectural component.

4.4. Orchestrating Behavior

The mission of any autonomic system is to adapt to change, according to its underlying business rules. This usually means that code must be dynamically generated, according to the current state of the system, to reconfigure one or more managed entities. OMG's MDA [MDA] initiative seems perfectly suited to this task. However, there are a number of problems with this initiative. [Ray06] The most important of these are:

- ➢ MDA is not an *architecture*, it is an *approach*
- ➢ While MDA formalizes the concept of a conceptual transformation pipeline,
  - it only *partially* bridges the semantic gap between different languages
  - it cannot in reality allow multiple languages to be used because MDA does not itself have a formal modeling language
  - it has no inherent semantic mechanisms that can be used for reasoning (which is a problem with UML), such as "is similar to", "is disjoint with", "is complement of" and "is a synonym of"
  - hence, it lacks the ability to build semantic translations between platform-independent and platform-specific models
- ➢ Even if QVT (Query-View-Transformation) is ratified, there are no OMG-compliant tools for
  - creating and storing MOF-compliant models
  - OCL-compliant compilers
  - OCL-compliant transformation engines

Hence, our approach uses the *fusion* of information models, ontologies, and machine learning and reasoning algorithms to enable semantic interoperability between different models. This will be explained in the next three sections.

4.5. Semantic Fusion

Most information and data models are designed as *current state models*. [PBNM] That is, they provide class and object definitions to model the current state of a managed entity, but do *not* provide mechanisms to model how that managed entity changes state over its lifecycle. The TeleManagement Forum (TMF) and 3GPP/2 are among the few standards bodies and fora that require UML-compliant models to be built (note that the Distributed Management Task Force, or DMTF) does *not* produce UML-compliant models, since it redefines MOF!); the IETF and ITU still do not use UML to define their models. Hence, while their artifacts (e.g., MIBs and ITU-T recommendations) define useful concepts, they cannot take advantage of the power of UML to represent their concepts (e.g., interaction diagrams as defined in UML to model behavior directly). Worse, they in effect build their own *proprietary* solutions that do not interoperate. Our architecture uses UML to avoid these problems.

However, an important point missed by most standards bodies and fora is that information models alone are *not enough* to capture the semantics and behavior of managed network entities. Information and data models are excellent at representing facts, but do not have any inherent mechanisms to represent semantics required to *reason* about those facts. For example, an SNMP alarm and a flashing red light in the Network Operations Center console can be intuitively related to each other by a *human*, but not by a *machine*, because the *nature* of these two data is fundamentally different. Furthermore, there is no ability to directly relate an SNMP alarm to the customers or SLAs that it may effect – this requires the ability to reason about the SNMP alarm and its relationship to other managed objects and their states in the environment. Therefore, we have *augmented* the use of information and data models with *ontologies*. Conceptually, ontologies function as an overlay to our information and data models. Our definition and use of ontologies [Str06c] is as follows:

> *An ontology is a formal, explicit specification of a shared, machine-readable vocabulary and meanings, in the form of various entities and relationships between them, to describe knowledge about the contents of one or more related subject domains throughout the life cycle of its existence. These entities and relationships are used to represent knowledge in the set of related subject domains. Formal refers to the fact that the ontology should be representable in a formal grammar. Explicit means that the entities and relationships used, and the constraints on their use, are precisely and unambiguously defined in a declarative language suitable for knowledge representation. Shared means that all users of an ontology will represent a concept using the same or equivalent set of entities and relationships. Subject domain refers to the content of the universe of discourse being represented by the ontology.*

This leads to the conceptual picture shown in Figure 5 below. This shows a typical network scenario – different devices having different data models are managed using different tools. This creates *cognitive dissonance* between data in the two data models – since there is no common vocabulary with established meanings defining the data and relationships, it is impossible to directly compare data from different sources, which in turn means that it is impossible to see if those data are related to each other. By using a set of ontologies to augment (not replace!) the facts represented in these data models, each fact can be mapped into a common vocabulary, which enables each fact to be augmented with appropriate semantics. This enables *cognitive similarity* between these different facts to be established. [Wong05] Once the AM establishes the cognitive similarity between different data, it can then form one or more hypotheses as to the nature of the received data, query the system to gather other information to prove or disprove these hypotheses, and then take appropriate action.

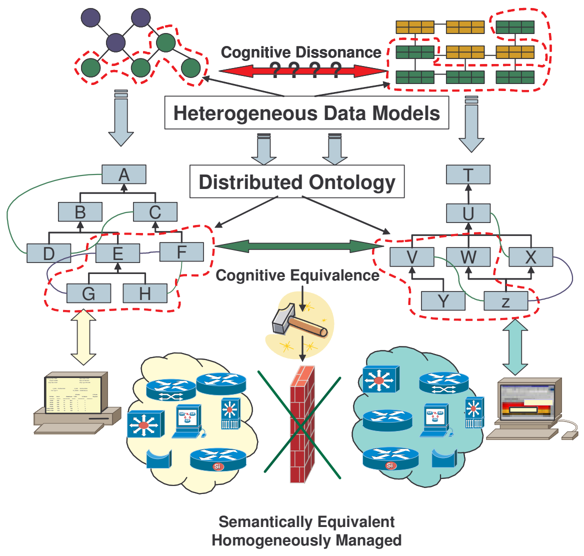


Figure 5. Fusion of Information/Data Models and Ontologies

4.6. The Resulting Architecture – FOCALE

FOCALE stands for Foundation, Observation, Comparison, Action, and Learning Environment. The first step in building FOCALE is shown Figure 6 below.

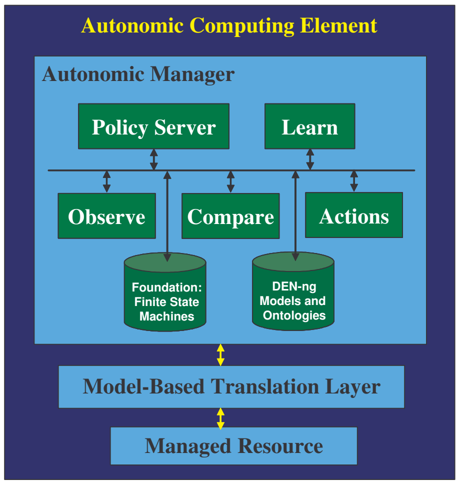
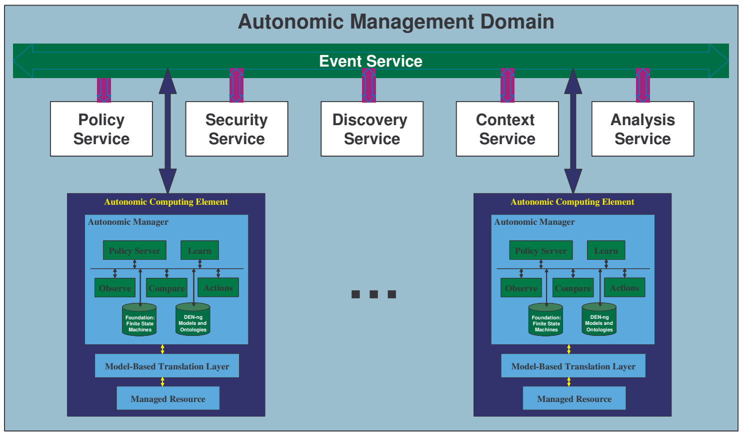

Autonomic Computing Element
Autonomic Manager
Policy Server
Learn
Observe
Compare
Actions
Foundation: Finite State Machines
DEN-ng Models and Ontologies
Model-Based Translation Layer
Managed Resource

Figure 6. FOCALE – the Autonomic Computing Element

This approach assumes that any ***Managed*** Resource (which can be as simple as a device interface, or as complex as an entire system or network) can be turned into an Autonomic Component. By embedding the *same* Autonomic Manager in each ACE, we provide uniform management functionality throughout the autonomic system (AAMCs and AEMCs). This is then expanded, first to a uniform Autonomic Management

Domain (Figure 7) and then to an Autonomic Management Environment (Figure 8), which consists of a set of Autonomic Management Domains.



Autonomic Management Domain
Event Service
Policy Service
Security Service
Discovery Service
Context Service
Analysis Service
Autonomic Computing Element
Autonomic Manager
Policy Server
Learn
Observe
Compare
Actions
Foundation: Finite State Machines
DEN-ng Models and Ontologies
Model-Based Translation Layer
Managed Resource
Autonomic Computing Element
Autonomic Manager
Policy Server
Learn
Observe
Compare
Actions
Foundation: Finite State Machines
DEN-ng Models and Ontologies
Model-Based Translation Layer
Managed Resource

Figure 7. An Autonomic Management Domain

FOCALE uses the following approach to make this transformation:

1. First, we use the DEN-ng information/data models and ontologies to build the foundation – a set of finite state machines (FSMs) that define the desired behavior of the system
2. Then, we form a mediation layer between the Managed Resource and the Autonomic Manager using the MBTL. This translates vendor-specific sensed data into a form that the Autonomic Manager can process; similarly, it enables the Autonomic Manager to define actions to be taken that are translated into vendor-specific commands.

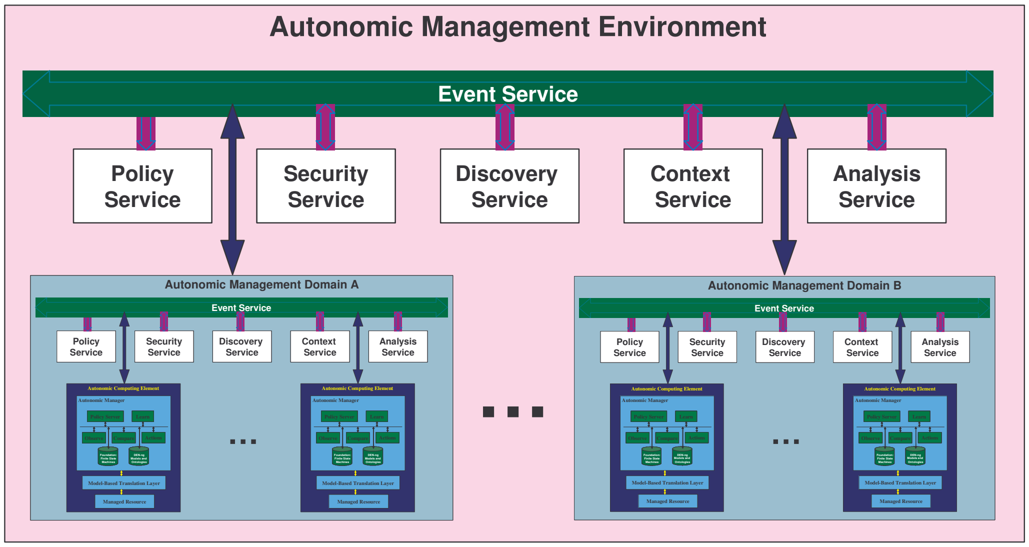


Figure 8. An Autonomic Management Environment

3. At this point, we can build the basic control loop shown in Figure 4.
   a. Sensor data from the Managed Resource is analyzed to determine if the current state of the Managed Resource is equal to its desired state. If it is, the process repeats.
   b. If it isn't, then the Autonomic Manager uses the ontologies to *reason* about the received data (using the models) to try and determine the root cause of the problem; it then issues actions, which are translated into vendor-specific commands by the MBTL, to the appropriate

Managed Resources. Note that this may include Managed Resources that were *not* the cause of the problem, or were not being monitored.

c. The cycle then repeats itself, except that in general the monitoring points will have changed to ensure that the reconfiguration commands had their desired effect.

4. One or more machine learning algorithms may be employed to gain experience from the environment, and to aid the reasoning process (these will be described in a future paper)
5. The policy server serves two purposes: (1) to control the action of each of the ACE components, and (2) to interface to the outside world (humans and machines, such as ACEs).

6. Applying FOCALE to Beyond 3G Networks

This section shows how FOCALE could be deployed in a complex environment such as beyond 3G (B3G) networks [3GPP05]. This environment enables users to access a portfolio of Internet Multimedia Subsystems (IMS) services from different access networks (e.g., different wired and wireless technologies (UMTS, Wifi, WiMax, xDSL) [UMTSFORUM'05]) that are owned by different operators. The deployed Management systems are likely to be heterogeneous even for the same technology, as operators are not necessarily using the same management system and, more importantly, not the same information model. Hence, supporting end-to-end quality of service is a big challenge for service providers. The management of this environment can became very complex, as the number of components to manage individually and globally can be large and complex, going well beyond the capacity of the human operators.

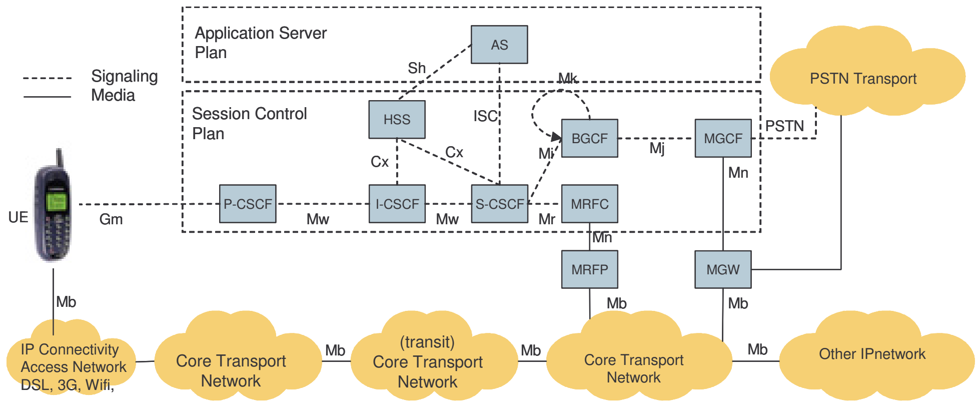


Figure 9. Components and Domains in a B3G Environment

Figure 9 shows multiple domains. For simplicity, we will consider only the Access Network and Mobile network domains, the Core Transport Network domain, and the IMS domain. Each domain will be managed by a particular operator management solution with its own tools and information model. The concepts of this paper are applied as follows:

- The autonomic management environment (AME) is responsible for ensuring end-to-end service assurance
- the autonomic management domain (AMD) is responsible for the autonomic management of a particular administrative domain
- the autonomic computing component (ACC) which is responsible for the autonomic management of resources (this domain can be recursively decomposed into other domains)
- The leaf domain can itself be decomposed in a number of ACCs that controls each individual component in the leaf domain

There are a number of ways to decompose the above environment into domains. For example, a bottom up approach, starting from the managed resource, can be defined, where the granularity is a function of the AC and the minimum size of the ACD. Then, each ACD can be grouped into an AMD; each AMD can similarly be recursively grouped into an AME. A provider can thus manage the appropriate entity (AC, ACD, AMD, or AME) that best corresponds to their real-world deployment. It is also possible to construct an AME to reflect a cross-domain system; however, the relation between AMDs can be very complex as only a limited behavior adaptation (through goal enforcement) can be permitted between the various AMDs (i.e., in case of a virtual service provider operating on the top of several network provider operators). Figure 11 gives an example of the FOCALE instantiation in this complex environment that shows how recursive process can be applied top-down and/or bottom up to build an end-to-end IMS services autonomic environment.

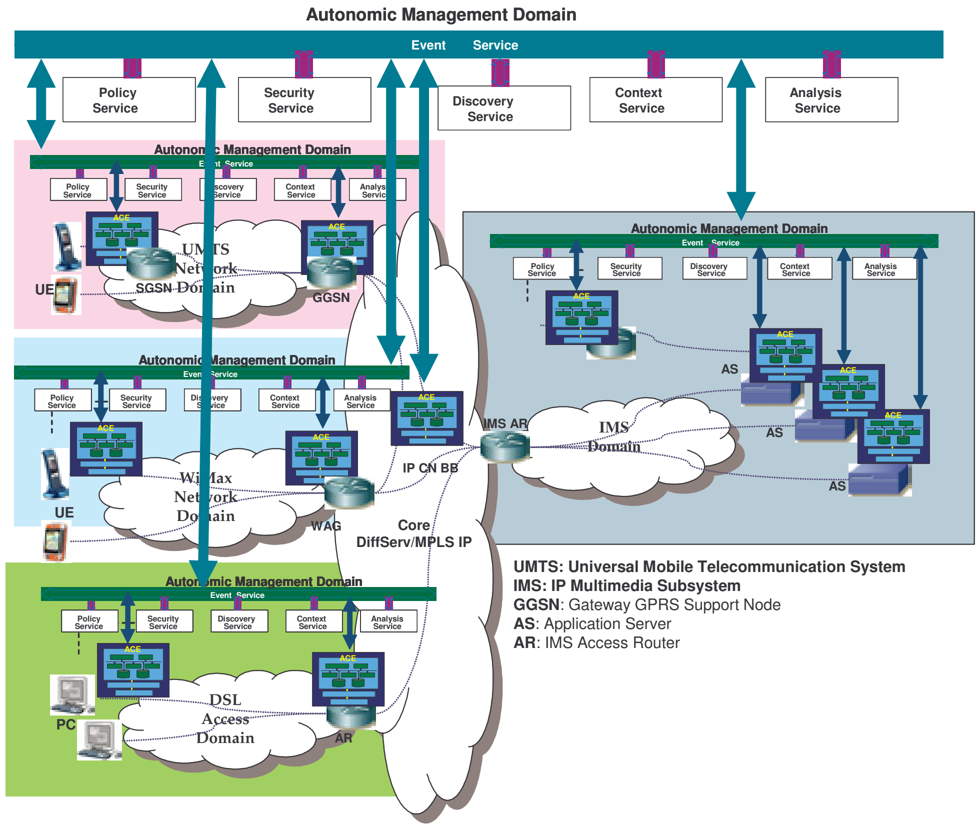


Figure 10: Applying FOCAL to mananing B3G networks

The recursive deployment of FOCALE architecture simplifies managing a complex B3G network environment. A bottom-up approach starts turning network equipment and servers into AMCs, and then grouping the AMCs into an AMD. The service provider can start with a top-down approach defining the high-level AME; the system then maps this definition to the desired behavior of the AMDs. The actions at this level will became goals at the lower level, and so forth, until we reach ACs. Then, each AC is transformed thought the MBTL process into models whose behavior is orchestrated through reconfiguration. Each operator at his level will define his own business goals to enforce in his autonomic networking domain. At the top level, the business objectives described in term of goals are enforced in the autonomic system and translated between different levels of the Policy Continuum. The Policy Continuum guides the transformation of goals into policies, which contain actions that are enforced at the appropriate control point. In this use case, dominant relationships can be found between autonomic domains inside the UMTS domain, while peering relations can be found in the case of the AME where the underlying AMDs could be operated by different operators (e.g., the AME operated by the service provider and the AMD operated by the UMTS operator, WiMax operator or xDSL operator).

7. Summary and Future Work

In this paper, we have proposed a novel autonomic networking architecture called *FOCALE. It introduces differents techniques to* seamlessly manage and integrate heterogeneous and distributed computing resources using two levels of control loops. We have proposed the use of knowledge engineering techniques to manage legacy as well as new (as in inherently autonomic) components and orchestrate their behaviour. Our solution uses Ontologies as a shared vocabulary between semantically equivalent components and Policies. Finally, we have shown an example of FOCALE deployment in a B3G network environment to facilitate and automate the management of complex and heterogeneous domains.

Our future work will focus on the design of a set of layered ontologies that integrate heterogeneous domain-specific languages, such as information models, data models and policy languages. We will also present an implementation of the Mediation Layer, based on our ontology, and validate our Model Based Transformation Layer in different real-world scenarios.

References

[Ago04] N. Agoulmine, J. Strassner, "Network Management: From SNMP to Policy Directories" Computer Science professional Journal (France), March 2004.

[GB922] TMF, "Shared Information and Data (SID) model". *GB922 and Addenda,* Jul 2004

[GB927] TMF, "The NGOSS Lifecycle Methodology", GB927, v1.3, Nov 2004

[IBM01] Autonomic Manifesto, www.research.ibm.com/autonomic/manifesto

[IBM03] IBM, "An Architectural Blueprint for Autonomic Computing", April 2003

[Kep03] Kephart, J.O. and Chess, D.M., "The Vision of Autonomic Computing", Jan 2003, In: www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf

[MDA] URL: www.omg.org/mda

[PBNM] Strassner, J., "Policy-Based Network Management", Morgan Kaufman Publishers, ISBN 1-55860-859-1, Sep 2003

[Ray06] Raymer, D., Strassner, J., Lehtihet, E., "In Search of a Model Driven Reality", submitted to Globecom 06

[SID05] Strassner, J. and Reilly, J. "Introduction to the SID", In: TMW University Tutorial, May 2005

[Str02] Strassner, J., "A New Paradigm for Network Management – Business Driven Device Management". In: SSGRRs 2002 conference (summer session)

[Str04a] Strassner, J., "Autonomic Networking – Theory and Practice", IEEE Tutorial, Dec 2004

[Str04b] Strassner, J., “Building Better Telecom Models Through the Use of Models, Contracts, and Life Cycles”, ECUMN 2004, Oct 2004

[Str04c] Strassner, J., and Twardus, K., “Making NGOSS Autonomic – Resolving Complexity”, In: Session TECH2, TMW, Nice, May 2004

[Str05] Strassner, J., “Knowledge Management Issues for Autonomic Systems”, In: TAKMA 2005 conference

[Str06a] Strassner, J., “Seamless Mobility – A Compelling Blend of Ubiquitous Computing and Autonomic Computing”, in Dagstuhl Workshop on Autonomic Networking, Jan 06

[Str06b] Strassner, J., Raymer, D., Lehtihet, E., Van der Meer, S., “End-to-end Model-Driven Policy Based Network Management”, In: Policy 2006 Conference

[Str06c] Strassner, J., “Knowledge Engineering Using Ontologies”, contributed chapter to Handbook of Network and System Administration, in preparation

[Str06d] Strassner, J., Kephart, J., “Autonomic Systems and Networks: Theory and Practice”, NOMS 2006 Tutorial

[TMF053] TMF, “The NGOSS Technology Neutral Architecture”, *TMF053 (and Addenda),* Jun 2004

[Wong 05] Wong, A., Ray, P., Parameswaran, N., Strassner, J., “Ontology mapping for the interoperability problem in network management”, Journal on Selected Areas in Communications, Oct. 2005, Vol. 23, Issue 10, page(s): 2058- 2068.