Using Autonomic Principles to Manage Converged Services in Next Generation Networks

John Strassner
Fellow and Director of Autonomic Networking,
Motorola Research Labs
1301 E Algonquin Rd., Schaumburg, IL 60196
john.strassner@motorola.com

Abstract

Network resources will always be heterogeneous, and thus have different functionalities and programming models. This adversely affects interoperability. Seamless Mobility is one example of how next generation networks and devices require a new approach for network management. This letter is an abstract of my ICAC 2006 keynote, which describes how our new FOCALE architecture can be used in such an environment. This approach establishes a common "lingua franca" 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. FOCALE is a semantically rich, context-aware, policy-based architecture for orchestrating the behavior of heterogeneous and distributed computing resources.

Keywords—autonomic computing, data model, information model, machine based learning, machine based reasoning, ontology, semantics

1. Introduction

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 [1] [2] – their desire to incorporate best of breed functionality prohibits the sharing and reuse of common data. There are five principle reasons for this [3].

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 [2]. This requires an increase in intelligence in the system, which is where autonomics will come in.

If autonomic computing, as described in [2] [4] [5] [6] [7] 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, FOCALE, which stands for Foundation-Observation-Compare-Act-Learning-Reasoning. This architecture develops a novel control loop by integrating modeling, knowledge engineering, and policy management

This paper describes some research in Motorola Labs to examine how autonomic principles and mechanisms can be used by network developers, administrators, and technicians to be able to better reconfigure network devices. This includes user devices, such as mobile phones and PDAs, in addition to network infrastructure. This project is part of a larger program aimed at bringing autonomic management operations to mobile wireless systems.

The remainder of the paper is structured as follows. Section II redefines the concept of convergence in the context of Seamless Mobility and Next Generation Networks. Sections III and IV explain the concepts

driving Seamless Mobility and Autonomic Networking, respectively. The differences between the traditional definition of autonomic computing and the new definition of autonomic networking are discussed in Section IV. Section V describes Motorola's FOCALE architecture, while Section VI explains FOCALE's "model-driven everything" approach. Section VII discusses the applicability of the FOCALE architecture to Seamless Mobility, while Section VIII provides conclusions and future work.

2. Convergence Redefined

Convergence is no longer about running voice, video, data, and other services on a single network. In the context of Seamless Mobility, this means that three key areas are required to move in union or uniformly to enable the different environments that users operate in to provide a seamless experience to the user. **Service Convergence** enables the end user to access the same service using one or more devices through one or more access mechanisms, **Device Convergence** enables the end user to access multiple services across multiple access mechanisms through a single device, and **Network Convergence** provides a unified network realizing multiple services using one or more devices through one or more access mechanisms.

Clearly, there can be other types of convergence, such as Billing Convergence (a unified billing system that provides a single bill for multiple services using one or more devices), Security Convergence (a unified system in which security exists end-to-end, independent of layers, sessions, applications, and protocols), Management Convergence (a unified system in which management of the system can be done in a consistent way, independent of devices, protocols, etc.), and others. Convergence in these and other areas is crucial in order to provide the user with a truly seamless experience.

3. The Vision of Seamless Mobility

People don't live in Zones, Categories, or Segments - People move about, figuratively and literally. The motivation for Seamless Mobility is simple to see. Businesses want anywhere, anytime communications to provide enhanced productivity to their workforce. This takes several forms, including the ability to respond more rapidly to customers, accelerate product time-to-market and spark business innovation. Consumers are equally eager for personalized services that make it easy to access and share digital content when, where and how they want it. Network operators seeking an edge in a changing marketplace are exploring new approaches to delivering this content in a timely and cost-effective manner.

Motorola's vision of Seamless Mobility [13] is shown in Figure 1 below.

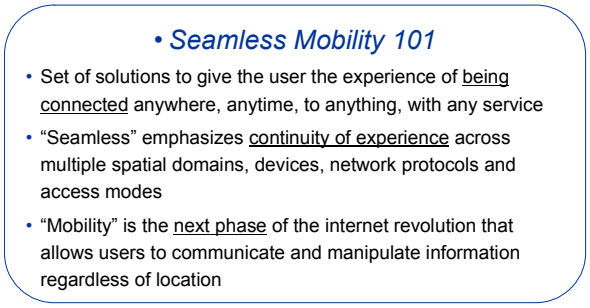


Figure 1. Motorola's Vision of Seamless Mobility

As can be seen, the goal of Seamless Mobility is to provide simple, uninterrupted access to the type of information desired at any current time independent of the device and media used. This information is always available, independent of, time, place, and most importantly, network and device.

The **interaction** of people and components creates interest and value, not the individual components themselves. This drives Seamless Mobility to become an experiential architecture. This means that the current *context* of what the user is doing should be known to the network, so that services that the user desires in this context can be optimized. We have developed a novel context model that provides a first step in solving this difficult problem that is part of our FOCALE [8] architecture; this will be explained in more detail in Section 0.

More importantly, Seamless Mobility is not "just" the handover between wired and wireless devices! Seamless Mobility environments will be user-centric, enabling the user to communicate anywhere using anything, regardless of media and protocol.

However, for the system administrators and operators, this represents a large challenge, due to the different protocols and functionality (e.g., processing capabilities and data display) of each device. Imagine several different applications:

- an automobile with different sensors that can detect a speeding car that will likely run a red light, or detect the presence of emergency vehicles approaching, or even something mundane as congested traffic

- a military installation, where a variety of sensors (thermal, video, location, and so forth) provide different information in a multiplicity of formats, content, relevance, and importance
- a think tank, where different people using a variety of inputs collaborate dynamically

How do we remove the barriers between these different sources of information and enable them to communicate, share, and exchange data seamlessly?

Motorola's Seamless Mobility architecture is shown in Figure 2 below.

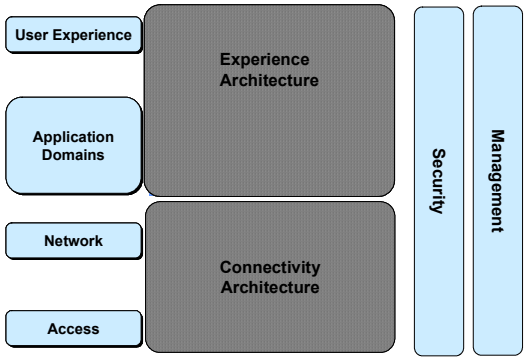


Figure 2. Motorola's Seamless Mobility Architecture

This paper explores the management function of this architecture.

4. Salient Features of Autonomic Networking

The name "autonomic" was deliberately chosen by IBM [4] to invite comparisons to biology. This section will first review the principles of autonomic computing, and then see how these apply to autonomic networking.

4.1 Autonomic Computing

IBM's definition of autonomic computing is: "Autonomic computing is an approach to self-managed computing systems with a minimum of human interference. The term derives from the body's autonomic nervous system, which controls key functions without conscious awareness or involvement. Autonomic computing is an emerging area of study and a "Grand Challenge" for the entire I/T community to address in earnest." [17]

Autonomic computing is, in general, a type of computing model in which the system exhibits self-healing, self-configuring, self-protecting, and self-managing properties. It is designed to mimic the human body's nervous system. Just as the autonomic nervous system acts and reacts to stimuli independent of the individual's conscious input, an autonomic computing environment operates organically in response to the input it collects.

In [4], the vision of autonomic computing is described as follows: "Systems manage themselves according to an administrator's goals. New components integrate as effortlessly as a new cell establishes itself in the human body. These ideas are … elements of the grand challenge to create self-managing computing systems."

In an autonomic computing architecture, the basic management element is a control loop, shown in Figure 3. This manages the monitoring, analysis, and actions taken on a set of predefined system policies.

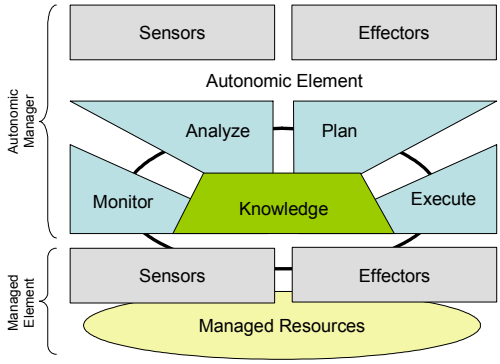


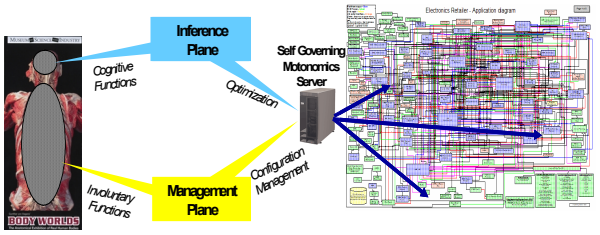
Figure 3. IBM's Autonomic Control Loop

The monitor portion collects, filters, and reports on data collected from the element. This is fed to the analysis portion, which analyzes and learns about the element being managed. The acquisition of knowledge is critical, as it enables future situations to be predicted and problems prevented. The planning portion uses predefined policies that establish the system's goals and objectives. The execute portion controls the commands issued, and determines whether the commands worked as desired.

This control loop is connected to sensor and effector "touchpoints". Sensors provide the mechanisms to collect data on the state of the element, and effectors are the software commands and/or APIs that change the state of an element (i.e., they act or alter the configuration of the element from the data provided from the sensors).

4.2 Autonomic Networking

The vast majority of current work focuses on IT issues, such as host systems, storage and database query performance. We have learned from these activities, but are instead focusing on the application of autonomic principles to networking. Figure 4 illustrates the autonomic nervous system analogy applied to autonomic networks.



- *Technical* complexity: human body ⇔ technology, devices
- *Business* complexity: macro-economics ⇔ e- and m-Commerce
- *Behavioral* complexity: social interaction ⇔ service composition
- *Operational* complexity: healing ⇔ anti-virus, configuration management

Figure 4. Autonomic Networking Analogies

The picture on the left in Figure 4 shows the autonomic nervous system (courtesy of Body Worlds), while the picture on the right is a block diagram of an Electronics Retailer application, courtesy of IBM. The first, and most important, analogy is that complexity is everywhere. You see four main types of complexity in this slide. Autonomics is first and foremost, a way to manage complexity.

The idea is for the autonomic system to perform tasks that relieve human administrators from time consuming manual functions (the involuntary functions performed by the autonomic nervous system), freeing administrators to work on more difficult and strategic tasks, such as network planning and optimization (the cognitive functions performed by the human brain). Our research views this as defining two new planes on top of the data and control planes. The management plane coordinates the functions of multiple control planes, and corresponds to the involuntary functions handled by the autonomic nervous system. The inference plane uses machine-based learning and reasoning algorithms to direct what the management plane should do, and corresponds to the higher-level cognitive functions.

The emphasis on adaptation to system, environmental and user changes requires a change in the definition of autonomics when applied to networking. We use the concept of **self-governance**, as opposed to self-management. The difference is simple but critical: a self-managing system is one in which it is assumed that the pre-defined policies that are applied to a system are capable of instructing the system what to do, no matter what changes in the system, the environment, or in user needs occur. The problem is that policies cannot be written that take every possible change into account. Hence, we use the concept of self-governance: the system is still pre-loaded with policies, but policies are changed in response to changing business needs, environmental conditions, user demands, and most especially, context. This enables the system to dynamically adjust the resources and services that it provides to the needs of its users, taking into account any applicable environmental changes.

These changes give rise to Motorola's definition of autonomic networking, which is as follows:

An ***autonomic network*** is a self-governing system, where the governance model is expressed using policies. Policies can be supplied by humans or derived by machines. Self-governance is accomplished through the use of self-knowledge to model the capabilities of the system and the constraints placed on the system as a function of context. This takes the form of a closed control loop – one in which the system senses changes in itself and its environment, analyzes those changes to ensure that business goals and objectives are still being met, plans changes to be made if business goals and objectives are threatened, executes those changes, and observes the result. This control loop is augmented by a self-learning process, which enables the system to develop greater knowledge of itself and its environment, both by experience as well as by incorporating new knowledge. Autonomic systems can be dynamically and automatically built from reusable autonomic elements, rearranged as necessary to provide new functionality. Autonomic systems are used to manage business, system, technical, and operational complexity. This also enables progressively greater adaptive systems that can dynamically respond to current and future needs.

In order to realize an autonomic network, a set of technologies are required, including:

- policies are used to express self-governance
- if environmental conditions, user needs, or business requirements change, governance policies will change accordingly
- self-governance requires self-knowledge
  - a system cannot be reconfigured unless its functionality is known!
  - models are used to represent current state and desired behavior
  - models can dynamically change, which requires code to be dynamically generated
- closed control loop ensures that changes cannot violate the goals of the system

- Self-learning and –reasoning processes enable an autonomic system to dynamically adapt to changes, recognize past situations and apply past solutions pro-actively, and most importantly, incorporate new knowledge

These technologies are integrated in a novel architecture, which we call FOCALE. This is described next.

5 Motorola's FOCALE Architecture

FOCALE stands for Foundation, Observation, Compare, Act, Learn, and rEason [8].

5.1 The Basic FOCALE Control Loops

We start with the same basic approach: monitor data from a managed resource, and then analyze the data. At this point, the FOCALE architecture takes a different approach. FOCALE assumes that the system being governed has a set of orchestrated actions which can be modeled as a set of state machines. Hence, the analysis of data is done expressly to determine the *actual* state of the managed resource, so that it can be compared to the *desired* state of the managed resource. If the two states are equal, then all is going according to plan; if they aren't, then corrective action needs to be taken.

Note that there are at least two different control loops in this approach – one that represents steady state monitoring, and one that is used when a behavioral problem, manifested as a managed entity being in the wrong state, is discovered. This is shown in Figure 5 below.

The reason for the two control loops is flexibility. If the decision is to reconfigure one or more devices, then it is unlikely that a basic control loop designed for simple monitoring is capable of managing reconfiguration actions. This will be developed further in the following sections.

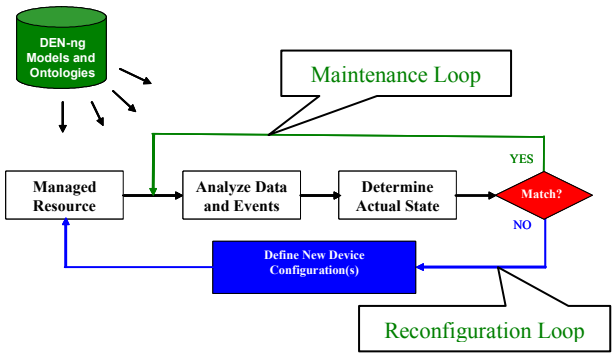


Figure 5. The Basic FOCALE Control Loops

5.2 Model-Based Mediation

One cannot assume that a managed resource, such as a subnet or network, is made up of uniform devices that all speak the same language and offer the same functionality. Clearly, it is not feasible to force the autonomic manager to speak multiple vendor-specific languages! Hence, we introduce a model-based translation layer that translates vendor-specific data and commands into a common "autonomic language" and vice-versa. This is shown in Figure 6. This approach uses the principle of abstraction to relate different vendor-specific data and commands to a single "lingua franca". This common language is based on the DEN-ng [14] information and data models, augmented with ontologies that provide a rich set of relationships between vendor-specific terms and the common lingua franca. This will be explained more in section V.0, Semantic Fusion.

FOCALE uses a "blade" architecture. A set of common interfaces and functionality is collected into a model-based translation **engine**, and custom **blades** are developed that use the functionality of the engine to translate one or more sets of vendor-specific commands and data into a common form that is represented in XML. Each blade uses the DEN-ng models and other functionality provided by the model-driven translation engine. Similarly, abstract commands defined by the autonomic manager are translated into vendor-specific commands by this same engine.

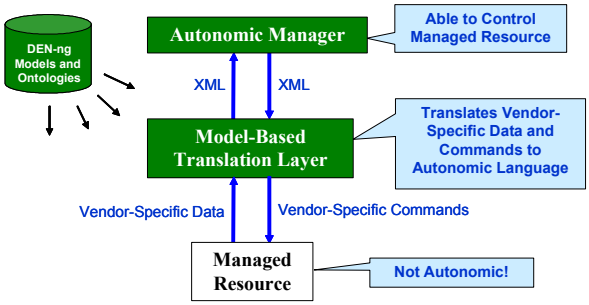


Figure 6. Model-Based Translation Layer

5.3 The Autonomic Manager

The Autonomic Manager is responsible for constructing the control loop out of reusable components, and then managing the operation of the control loop. It is shown in Figure 7 below.

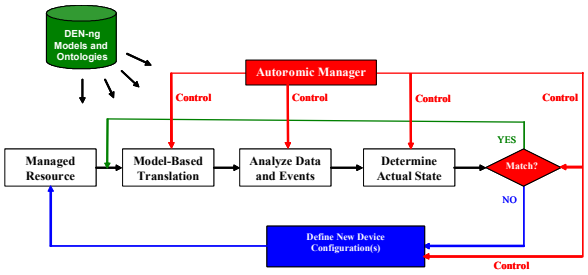


Figure 7. The Autonomic Manager

Unlike other autonomic managers that have been described, the FOCALE autonomic manager **adapts** the structure, components, and functionality of the control loop according to the current needs of the system. This is represented in Figure 7 above by the set of control functions that connect the autonomic manager to each component in the control loop. Note that the autonomic manager can also change the structure of the control loop; this is not shown for the sake of simplicity.

5.4 Policy and Context

In order to achieve the above autonomic manager functionality, the autonomic manager needs to be made aware of the outside world. This is done through the novel context-driven policy component of FOCALE.

This design assumes that policy *changes* as a function of context, which is a requirement for Seamless Mobility applications. For example, a user may have different Service Providers for work vs. home communication; when the user switches context from work to home, a different policy governing communication usage should be loaded.

We use [14] as a source for the following formal policy definitions. **Policy** is 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. A **Policy Rule** is an intelligent container. It contains data that define how the Policy Rule is used in a managed environment as well as a specification of behavior that dictates how the managed entities that it applies to will interact. The contained data is of four types: (1) data and metadata that define the semantics and behavior of the policy rule and the behavior that it imposes on the rest of the system, (2) a set of events that can be used to trigger the evaluation of the condition clause of a policy rule, (3) an aggregated set of policy conditions, and (4) an aggregated set of policy actions. For flexibility, the DEN-ng model defines three clauses (a Policy Event clause, a Policy Condition clause, and a Policy Action clause) that aggregate individual and groups of **Policy Events**, **Policy Conditions**, and **Policy Actions**. Each of these three clauses are treated as atomic objects that are in turn aggregated by a Policy Rule. A Policy Event defines the necessary occurrence or combination of occurrences that are used to trigger the evaluation of the Policy Condition clause. A Policy Condition defines the necessary state and/or prerequisites that define whether the actions aggregated by that same Policy Rule should be performed. This is signified when the Policy Condition clause associated with a Policy Rule evaluates to TRUE. (Note that in the DEN-ng policy language, an alternative set of Policy Actions can be defined that are executed when the Policy Condition clause evaluates to FALSE.) A Policy Action defines the necessary actions that should be performed if the Policy Condition clause evaluates to TRUE.

Most importantly, the effect of the Policy Action clause is to apply a set of actions to a set of managed objects, and have the effect of either **maintaining an existing state**, or **transitioning to a new state**, of that set of managed objects.

Therefore, there is a fundamental relationship between context and policy. This is shown in Figure 8 below. [15]

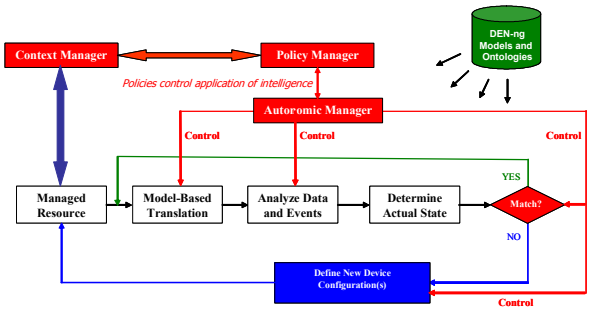


Figure 8. Context-Driven Policy Control

The generic DEN-ng context model is shown in Figure 9 below [15]. This preliminary context model is unique, in that it relates Context to Policy to Management Information. At a high level, this model works as follows: Context determines the working set of Policies that can be invoked; this working set of Policies defines the set of Roles and Profiles that can be assumed by the set of ManagedEntities involved in defining context. Significantly, this model also defines the set of management information that is used to determine how the Managed Element is operating. As shown in Figure 9, this is provided using one or more *roles* to represent specific functional aspects of the managed entity or entities.

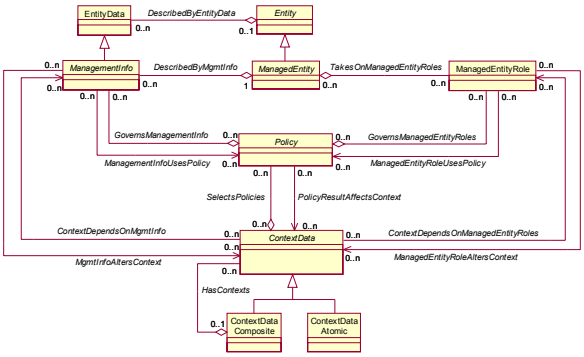


Figure 9. Simplified DEN-ng Context Model

Each of the entities in the above model provides feedback to the entity directing it. For example, the SelectsPolicies aggregation defines a given set of Policies that should be loaded based on the current Context. The association PolicyResultAffectsContext enables Policy results to influence Context. For example, if the execution of a Policy succeeds, then the current Context will change according to the actions of the policy; alternatively, if the execution of the Policy fails, then the current Context will also change, but not in the way that is desired by Policy.

5.5 State Automata

The above requires a machine-readable form of relating context to policy. Note that in Figure 9, both Context and Policy are related to management data. Specifically, Policy is used to define which management information will be collected and examined (via the GovernsManagementInfo aggregation); this management information affects policy using the ManagementInfoUsesPolicy association. This enables the autonomic manager to compare the current state of entities being monitored to the desired state of those entities. Since management information can also affect context, the two associations ContextDependsOnMgmtInfo and MgmtInfoAltersContext are used to represent these dependencies.

Behavioral orchestration is then done by using context to determine the set of applicable policies for the autonomic manager to use, which in turn governs the operation of the control loop. This is shown in Figure 8 above.

5.6 Semantic Fusion

As mentioned previously, autonomic networking is complicated by the fact that heterogeneous devices having vendor-specific languages and programming models. This is shown in Figure 10 below. The cognitive dissonance between the two data models A and B is caused by the differences in their structure, datatypes, protocols used to create, read, update, and delete data in the structures, and most importantly, different concepts and terms used in the definition of management data.

These problems are resolved by augmenting the facts in our information and data models with additional semantic terms and relationships from one or more appropriate ontologies. Algorithms for determining if two objects are semantically equivalent [9] are then used to build cognitive similarity relationships between the objects. This then enables one or more objects to be used in place of or even in addition to objects derived from sensed data. This provides critical data for our machine-based learning and reasoning algorithms, as explained in the next section.

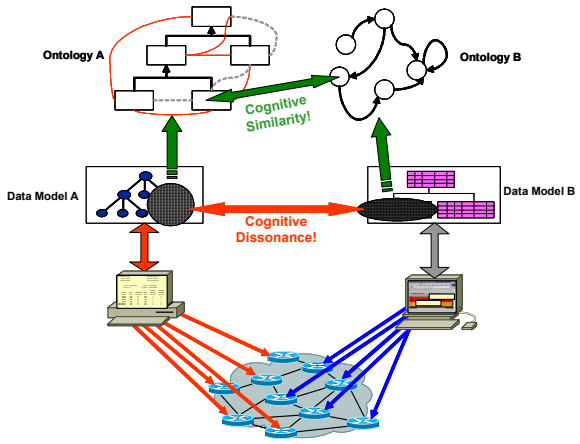


Figure 10. Semantic Fusion

5.7 Machine-Based Learning and Reasoning

The utility provided by our machine-based learning and reasoning algorithms is to incorporate new and learned behavior and data into the knowledge base. Information and data modeling facilitates machine learning by organizing the features, dependencies, behaviors and functionality of the system, as well as details regarding aggregation and connectivity, into a formal structure that can be easily managed and queried. When this is combined with a reasonable set of axioms derived from architectural and design experience, a reasoning engine may be applied that can determine system behavior.

One of the tasks of an autonomic system is to continually monitor the set of states of the system, ensuring that the system is behaving according to plan. When management data or events signal that this is not the case, the autonomic system will often have to create one or more hypotheses to help it understand what went wrong and how to fix what went wrong.

Hypothesis generation maps the data to be explained into the set of all possible hypotheses rank-ordered by plausibility [10]. We use the information model to derive the hypothesis space, since the information model defines objects and their relationships. Note that the use of ontologies helps this process, since it gives us more terms and relationships to use. If the resulting hypothesis space is too large, then the use of one or more falsification techniques aids in providing a "critic" function [10], which helps reject unsuitable hypotheses.

Hence, we add ontological comparison, learning, and reasoning functions to the FOCALE block diagram. Ontological comparison has already been described in Figure 10 above. Since the learning and reasoning functions can affect all parts of the control loop, we introduce a semantic bus to interconnect all control loop components, producing Figure 11 below [16].

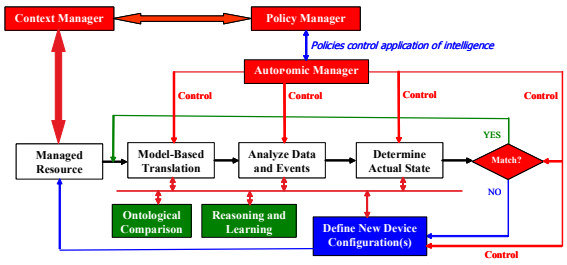


Figure 11. FOCALE with Learning and Reasoning

6 The FOCALE Model-Driven Everything Approach

Figure 12 shows a conceptual view of how FOCALE will be deployed. This figure represents a powerful notion for Seamless Mobility. It is a model-driven-**everything** approach, in that models are used for analysis, design as well as implementation and testing. A running system is governed via organization-specific business policies and processes using a common model and business process definition. Using a model-driven approach enables pieces of behavior to be designed and reused. More importantly, it enables the principles of structured, object-oriented analysis and design to be used to define the behavior and interaction between the components of a system.

The Business Process Model at the top shows a model of business processes, such as those defined by the eTOM [11] or ITIL [12], in a typical swim lane representation. Each of the tasks and activities shown in the Business Process Model can be linked to one or more fragments of the DEN-ng information and data models. (For the sake of simplicity, Figure 12 does *not* show additional links to other components shown in previous figures, such as the set of ontologies being used.

Different information from diverse sources (shown conceptually on the left of Figure 12) is stored in the knowledge base of the autonomic system. One or more state machines are built that orchestrate the behavior of the system. This is typically translated to a business process model that feeds a workflow engine. For the sake of simplicity, Figure 12 only shows a mapping

from DEN-ng to the business process model; clearly, other mappings are also needed.

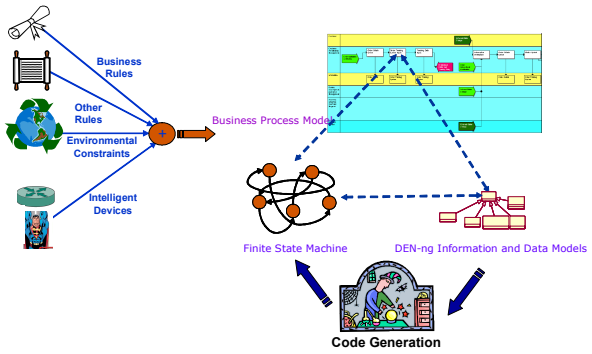


Figure 12. Model-Driven Everything Approach

At this point, we can build the control loop shown in Figure 11. It operates as follows.

Sensor data from the Managed Element is analyzed to determine if the current state of the Managed Element is equal to its desired state. If it is, the process repeats. If it isn't, then the autonomic manager examines the sensor data and tries to understand the cause of the problem.

The autonomic manager first tries to understand the data that it has received. This consists of a set of steps that start with examining the models and ontologies to develop knowledge about the received data (the remaining steps are complex and beyond the scope of this paper; indeed this first step is complex, but often be sufficient to discover the problem). This knowledge is then fed to a set of machine-based learning and reasoning algorithms that reason about the received data. For example, if the data represents a problem, then the algorithms try and determine the root cause of the problem; finding a cause, actions are then issued, which are translated into vendor-specific commands by the MBTL, and applied to the appropriate Managed Elements. Note that this may include Managed Elements that were not the cause of the problem, or were not being monitored. The actions are driven by appropriate modifications to the set of state machines that are being used to govern the system's behavior. Machine learning algorithms monitor all of these processes, and augment the knowledge base with any new knowledge learned in the remediation process.

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.

7. Applicability to Seamless Mobility

This section will describe five different use cases that each requires increased intelligence in devices as well as networks for their realization. Each of these use cases is representative of a particular aspect of Seamless Mobility.

A common feature of each of these five use cases is that autonomic networking will provide the *distributed intelligence* and collective governance to enable devices as well as networks to learn and predict user needs and preferences based on context. The intelligent interaction mechanisms of the autonomic network, coupled with device capabilities, will be able to reason about the services and devices available and dynamically construct the behavior needed to satisfy user request.

7.1 Driving Intelligent User Interaction

Currently, most human-device interactions are driven by the specific user interface of the device. The device simply responds to commands from the user. In a Seamless Mobility world, the device will understand the contextual needs of the user and the environment that the user is operating in to enable the user to easily accomplish what the user wants **without** being aware of the specific functionality and user interface of the device.

Not all devices have the capability (in terms of room for resources) to be autonomic, and not all devices *need* to be autonomic. Seamless Mobility does not require devices to be autonomic-*capable*; rather, it simply requires that devices be made *aware* of their surroundings. This can be accomplished with more sophisticated forms of mediation and communication than are currently used. For example, adding a communications interface to a refrigerator doesn't make the refrigerator autonomic, but it does make that refrigerator able to communicate with an autonomic system. By adding sensors inside the refrigerator, the autonomic system can "govern" the "state" of the refrigerator by sensing if certain contents are there in sufficient quantity or not. This can in turn be customized by adding in context (e.g., no need to refill milk since the people using the refrigerator will be gone vs. place an order for milk to be delivered to the home remotely via a phone or PDA).

Similarly, other devices, such as mobile phones, can be made more intelligent by correlating their context to the current environment and needs of the user. The autonomic system will govern this interaction

through its context-aware policy-based governance mechanisms.

7.2 Improving Content Handling

The modern world offers increased type of content, each with different requirements. This is exacerbated by the diversity of end-user devices, such as mobile phones and set top boxes, as well as by businesses that serve content over different media. Some content is free, while other content must be protected. Some content is unsuitable for certain audiences, while some content is or contains malicious software that can damage users in a number of different ways.

Seamless Mobility is a world in which the **experience**, not the individual content, is what is important. This requires distributed intelligence to make this goal a reality. The autonomic network will be responsible for connecting different enablers of the content handling experience, such as digital rights management, to the specific content that is desired, independent of media and device used. Autonomics can drive the convergence of content, computing, and communications to make life easier and better for the end-user.

7.3 Enabling Diverse, Real-Time Communications

Real-time communications is not just about voice. Rather, it is about providing a rich, multimedia experience to the end-user. This ***blending*** of content and real-time communications into rich interaction with people and devices has taken important leaps, such as Push to Talk and Push to View / Video, where people can benefit from a picture-augmented communication experience. In a Seamless Mobility environment, it is not just Push-To-Anything, but rather communicating to the *right* people, groups and devices. In other words, people want the ability to communicate using multiple media types with anyone, anywhere, using any device and media.

This will develop into a society-driven shared interaction experience, where multiple individuals and groups interact seamlessly using rich media. This in turn requires increased intelligence, not just in devices, but in the network itself, to coordinate the handling of different content. Group profiles and preferences will be used to signal context changes and provide the basis for policies that govern the set of services and resources that are available to a person or group of people at any given time.

7.4 Seamless Network Manageability

As described earlier, many of today's network management problems lie in the use of stovepipe architectures that focus not on user services, but rather on the device. In essence, current EMSs are focused on enabling the myriad capabilities of a device to be manipulated. There is no concept of a user or service in this approach, much less business needs. For example, what can you learn about a user's SLAs from a SNMP alarm?

Furthermore, each system in a stovepipe architecture defines data its own way, to suit its own needs [1] [2] [3]. Hence, sharing and reusing data are impaired. Worse, it is difficult to correlate information from different stovepipe systems, and knowledge cannot be easily shared.

Some current approaches to solving these problems use network-aware middleware to "tunnel through the stovepipe". This enables selected applications and processes to better communicate with each other, and enables a phased evolution to take place towards autonomic computing.

Efforts such as that described in this paper focus on the future, where autonomics is used to enable the business to drive the services and resources that the network should provide. This features identity and policy management, supported by knowledge engineering, to understand the current *context* of the task(s) being performed. Note that the learning and reasoning portions of the FOCALE architecture enable the autonomic system to learn from usage patterns as well as management data and events that are reported to it. Autonomic computing transitions the end-user experience from "the device dictating to the user" to "the user describing what tasks he/she want to perform, and the network reconfiguring to supply the services and resources necessary to perform that task".

8 Conclusions and Future Work

We've seen that complexity is everywhere, *now*. It takes many forms, and will only get more complex in the future, because users want more functionality in so many different ways (e.g., new ways to access data, new devices to use, and so forth). We must avoid previous mistakes, and make our management systems simple and easy to use. We must strive towards invisible management, as management functions

themselves are intimidating and require too high a skill set.

Autonomic computing manages complexity through the use of context-aware policy management, combined with knowledge engineering techniques. The FOCALE architecture uses both a model-based framework as well as a semantic framework in conjunction for representing the characteristics and behavior of managed elements. Using an analogy, the models provide simple grammatical elements, such as nouns and verbs, while the semantic tools supply the grammar rules needed to form proper sentences. More importantly, the use of these two frameworks enables new information to be integrated.

Autonomics is more than "just" self-management. Virtually every example of self-management today uses a statically defined set of rules (and in rare cases, models) to define how the system should behave. The problem is that this approach cannot work when the needs of the user and/or the environment that the user is operating in change. Clearly, one cannot simply change policies and device configurations at will – business rules and processes will be violated. Hence, we introduce a self-learning and -reasoning closed control loop that enables the autonomic system to adjust to changing needs and conditions while obeying some fundamental rules, and optimizing the goals of those rules.

Acknowledgements

This article is the IEEE Computer Society – Task Force on Autonomous and Autonomic Systems Jun/Jul 2006 – Letter released through the TFAAS Newsletter – Issue 5 @ http://tab.computer.org/aas/.

The article is based on a keynote talk given at the 3rd IEEE International Conference on Autonomic Computing (ICAC 2006) held June 2006 in Dublin, Ireland.

References

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

[2] Strassner, J., Kephart, J., "Autonomic Systems and Networks: Theory and Practice", NOMS 2006 Tutorial

[3] Strassner, J., "Knowledge Management Issues for Autonomic Systems", In: TAKMA 2005 conference

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

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

[6] 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

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

[8] Strassner, J., Agoulmine, N., Lehtihet, E., "FOCALE – A Novel Autonomic Computing Architecture", LAACS 2006, June 2006.

[9] 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.

[10] Josephson, John R. and S. G. Josephson. 1996. Abductive Inference: Computation, Philosophy, Technology. Cambridge. UK: Cambridge University Press.

[11] The eTOM franework documents are available at: http://www.tmforum.org/browse.aspx?catID=861&linkID=31098

[12] The ITIL framework documents are available at: http://www.itsmf.org/publications

[13] For Motorola's vision on Seamless Mobility, please see: http://www.motorola.com/content.jsp?globalObjectId=6611-9309

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

[15] Strassner, J., Carroll, R., VanderBaan, K., and Van der Meer, S., "Context-Aware Policy Management", to be submitted to MUCS 2007.

[16] Strassner, J. and Menich, B., "Providing Seamless Mobility for Wireless Networks Using Autonomic Networking", to be submitted to ICAC 2007.

[17] Please see: http://www.research.ibm.com/autonomic/overview/faqs.html