Realizing on Demand Networking

John Strassner

Intelliden Corporation
90 South Cascade Avenue, Colorado Springs, Colorado 80903
john.strassner@intelliden.com

Abstract. The promise of e-business is being increasingly adopted. However, applications are getting more sophisticated, and often require conflicting resources and/or conflicting network configurations. Meanwhile, the environment in which these applications operate is becoming more complex. This paper describes what is needed to build On Demand Networking capabilities. It will be shown that two related problems need to be overcome. First, the services that the network provides are in general not related to how the business operates. Second, network management must undergo a revolutionary change – one that enables it to become autonomic. The result of this approach is a new genre of management applications that ensure that the network delivers the services asked of it by the business, on demand.

1 Introduction

Dr. Irving Wladawsky-Berger defines on demand computing this way: "An on demand business is one that can respond with complete flexibility to changing market conditions, customer demand or external threat, in real time, as they are occurring, because all its business processes are thoroughly integrated" [1].

This is certainly a compelling vision. Imagine, for example, an autonomic network, one which can self-heal, self-configure, self-protect and self-optimize according to changing needs and the changing environment. Unfortunately, this vision looks extremely unlikely given today's management environment. It is common practice to use multiple management systems from the same vendor to manage different devices manufactured by that vendor. Using heterogeneous devices exacerbates this problem, since different languages and programming models are introduced. In addition, there is no connection between management software to control network devices and services and applications that provide revenue. Since Operational Support Systems (OSSs) from best-of-breed applications, there are usually multiple management applications of different categories.

Most importantly, however, there is no standard way to tie business rules that describe how the organization operates to the network. Thus, there is no means to ensure that the network delivers the services that the business needs at any given time. Until this is solved, then it is impossible to have the network change the services that it provides to accommodate changing business needs, because it is unaware of business needs in the first place.

A. Marshall and N. Agoulmine: MMNS 2003, LNCS 2839, pp. 313–327, 2003.

This paper presents a new approach towards network management. This approach is based on two observations. First, current network management approaches are not enabling Service Providers or Enterprises to use business rules to develop and deploy device configuration changes. Solving this problem enables business rules to *drive* the configuration of the network. Second, any form of autonomic operation starts with a simple premise: the autonomic entity must *know itself*. The solution must leverage the knowledge of the system and its environment, and embed in the system the ability to react to changes in the environment. This paper proposes a variant of the OMG's Model Driven Architecture (MDA) [2] as one way to implement this "higher intelligence".

The organization of this paper is as follows. Section 2 defines terms used in this paper. Section 3 describes the current problems with network management and anticipates new problems that will arise, based on the ever-growing complexity of applications and user tasks. Section 4 provides a brief overview of the DEN-ng information model (Directory Enabled Networks new generation) which is being developed in the TeleManagement Forum (TMF). Section 5 describes the roles of policy and process management, and how the DEN-ng models support them. Section 6 examines the generic MDA approach. Section 7 describes specific enhancements made to MDA by this new On Demand Networking approach. Section 8 presents conclusions and future work. Finally, Section 9 lists references for this work.

2 Terminology

Definitions in this section are taken from [3].

2.1 DEN versus DEN-ng

DEN, or Directory Enabled Networks [4], is a *specification* of an object-oriented information model describing the entities in a managed environment, and how they are related to each other. It also specifies a *model mapping* to an (L)DAP implementation. The original DEN specification is *not* solely a mapping to an LDAP model, nor does it have anything to do with WBEM, as [5] states.

DEN new generation is the next version of the DEN specification. It is tightly bound to the TMF NGOSS architecture [6], and contains business, system, and implementation entity definitions. Its main purpose is to define the functionality and synchronize the relationships between Customers, Processes, Products, and Network Services and Capabilities. It is the model that is being used in the approach described in this paper.

2.2 Policy

A *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. (Note in particular the use of *state* in this definition, which is new – see [3] for more detail.)

2.3 Information Model versus Data Model

An *information model* defines the properties, operations, constraints, and relationships of managed objects. It is independent of any specific repository, software usage, platform, or access protocol.
In contrast, a *data model* is a concrete implementation of an information model that is bound to a particular platform. It includes data structures, operations, and rules that define how the data is stored, accessed and manipulated.

2.4 Model Mapping

A *model mapping* is a translation from one type of model to another type of model. Model mapping changes the representation and/or level of abstraction used to another representation and/or level of abstraction.

2.5 Business Driven Device Management (BDDM)

BDDM is defined as the ability to use business rules to manage not just the construction of configuration files and commands for a device, but also to enforce how the configuration of a device is created, verified, approved, and deployed. With business-driven device management, configuration management becomes the foundation for managing the entire network (see [7] for more detail).

3 Problems in Current Network Management Approaches

Current network management solutions, and especially OSSs, suffer from the inability to use business rules to translate business needs and processes into device configuration changes. This is being exacerbated by the ever-increasing complexity in applications and the desire to build one network to serve the needs of multiple people. This section will examine some of the problems in building an OSS.

3.1 Barriers to Operational Efficiency and Data Exchange

The three largest barriers to operational efficiency are: (1) fragmented business processes, (2) fragmented data, and (3) incompatible systems with weak levels of integration.
There are always business rules that govern how devices should be managed. Unfortunately, these business rules tend to be ignored, because of the *lack of correlation between business rules and network configuration*. For example, people shouldn't simply telnet into a router and start typing CLI to fix a problem. Yet they do, because business processes either aren't specified or are ignored. Unless the network is treated like any other business asset, it will always be handled differently, and it will always be "OK" to ignore business process. We need a paradigm shift –

one which *encourages* the user to "play by the rules". This means that we need to *automate* how business rules govern network configuration, because no matter how pretty and functional the GUI, people won't use it if they can find an alternative means to do the same job faster.

This paradigm shift requires a common information model to represent not just different managed objects, but also the interaction between these managed objects. (The question of why another information model should be built is answered in the next subsection.) Currently, data exists as isolated information islands, because there is no common information model that is used in building an OSS. This is because of the proliferation of stovepipe management applications, and is shown in Figure 1 below (which is taken from a real OSS in the industry). Note that the figure is supposed to look complicated.

Current systems, like that shown in Figure 1, are designed using stovepipe applications because stovepipe applications provide best-of-breed functionality. Unfortunately, each stovepipe application is designed with itself as the "center of the universe". This creates an interoperability nightmare, because objects and information are continually renamed and redefined. It leads, inexorably, to systems with an amazing amount of complexity, as shown in Figure 1. Such systems have a large number of *brittle* interconnections, which means changing any component in this OSS is a challenging task. Realizing that vendors won't discard existing applications just because a new information model exists, the approach is rather to build a common mediation layer between these disparate management applications. This in turn means that such mediation must be inherently extensible and automated to keep up with the changing business demands.

3.2 Why Another Information Model?

Before the advent of DEN-ng and the TMF's SID (Shared Information and Data) model, there were three main policy models in the industry: (1) the DEN model [8]), (2) the IETF model (as defined in [9], [10], [11], and [12], and (3) the DMTF model ([13]). Of these, DEN was used as the basis for both the IETF and the DMTF models. There are in addition other efforts, such as Ponder ([14]), that imply additions and/or changes to one or more of these models.

Policy models are critical to the approach proposed in this paper. However, they need to be integrated with models of other objects (e.g., users, services, and so forth) so that policy can be used to control these other objects.

With the exception of the DEN-ng model, the problem with all of them to date is fourfold. First, they all focus on modeling the current state of an object. While this is important, this does not describe the life cycle aspects of the managed object. Therefore, models have only been used in the design process, and not in the actual management process.

Second, current models can be categorized as either business or system models, but not both. This makes it impossible to implement BDDM, and ensures that the business and networking worlds will remain separate.

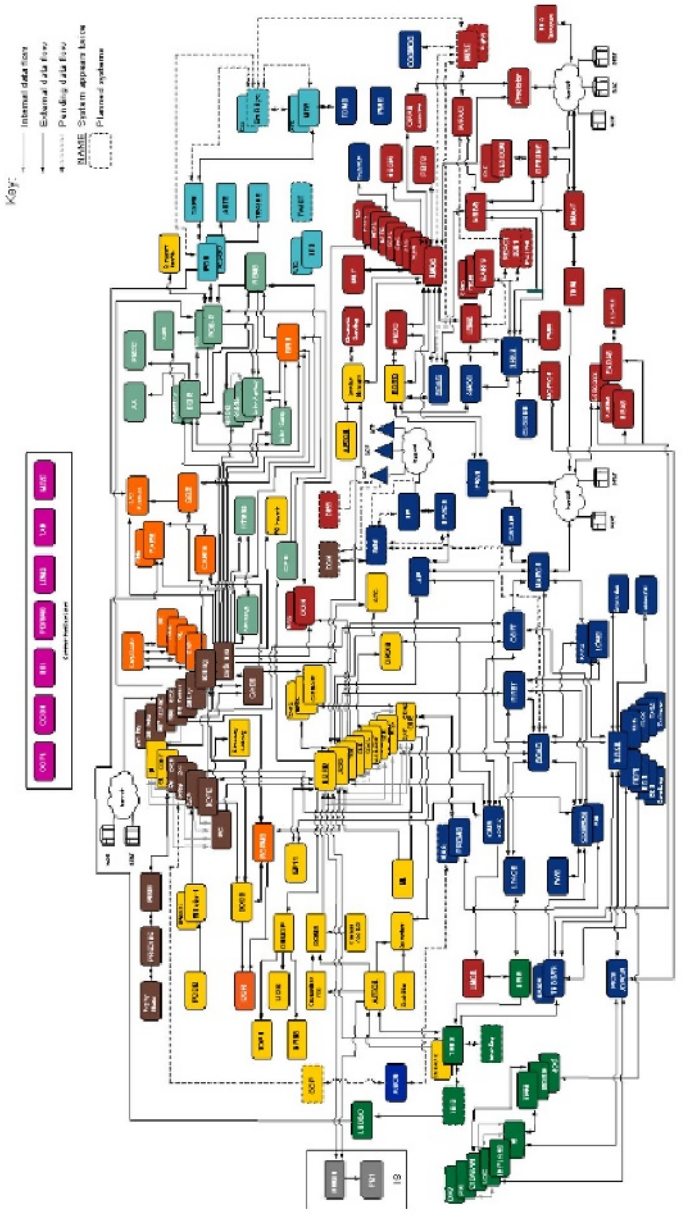

Fig. 1. A Traditional OSS

Third, previous models focus on policy as a domain that is only loosely integrated to other domains. In contrast, the main use case for the DEN-ng policy model was to define a policy model that was closely integrated with the rest of the managed

environment. This would enable the policy information model to define how policy interacted with the rest of the managed environment.
Finally, previous information models have been built without considering how to derive data models from them. Management data has too many diverse characteristics to enable all types of management data to be stored in a single repository. Model mappings provide formal methods to translate between models, and enable different repositories to be federated.

3.3 Sharing and Reusing Data

Stovepipe applications, such as those shown in Figure 1, cause many integration issues. A GUI shows its own application-centric view of the piece of the network that it is managing, not of the entire system. However, an application is simply a realization and manipulation of its underlying object model. Object models prescribe their own view of the world upon the applications and systems that use them. Unfortunately, this means that the same data will be needlessly redefined, or restructured, or worse, conflict with other management data, instead of being *reused*. This in turn implies that applications that could have used information from other applications must instead build a duplicate process to obtain that data. For example, a provisioning application requires up-to-date topological information as well as knowledge of which hardware is running which release of which operating system.

3.4 Using Business Rules to Drive the Configuration of the Network

There are two conflicting desires for network design. First, people want to use the same network to do more things, rather than have multiple networks doing specialized tasks. Second, more people are using networks with more sophisticated applications. The problem is that applications having vastly different resource utilization requirements need to peacefully coexist and run concurrently on the same network. Clearly, business rules must be used that decide which application gets priority usage of shared resources. In order to meet these requirements, the current focus of network management products must change to enable business rules to drive how the network is configured.
DEN-ng uses the concept of viewpoints (as defined in RM-ODP, in [15]). A viewpoint is an abstraction obtained by using a selected set of architectural concepts in order to focus on a set of particular concerns within a system. One can think of the DEN-ng models as organized along two dimensions – knowledge domain and viewpoint. For each knowledge domain (e.g., policy, resource, service, and so forth), three viewpoints (business, system, and implementation) abstract the entities and relationships in that viewpoint. This makes DEN-ng unique, as other information models don't have the concept of viewpoints. This makes it possible to connect the business, system, and implementation viewpoints, which enables business needs to be translated to implementation software and hardware. This is a prerequisite to meeting the needs of On Demand Networking.
Figure 2 shows a simplified UML diagram of how DEN-ng solves this problem.

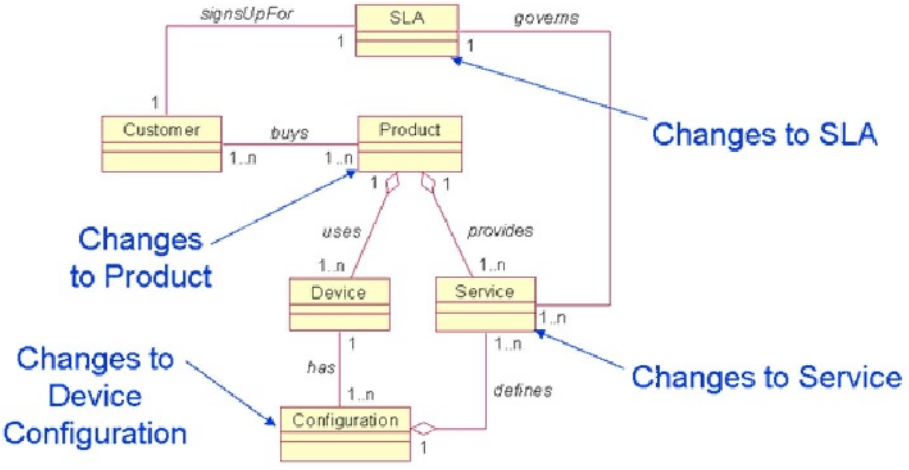

Fig. 2. Business Changes Must Drive Network Configuration

Previous solutions have not integrated multiple viewpoints. In contrast, DEN-ng links them firmly together by defining the (business) notion of a Product as containing Resources and Services. Figure 2 is a simple UML drawing that shows that a Customer buys one or more Products. (Note that the DEN-ng models are much more complex.) Each Product uses one or more Devices, and provides one or more Services. Note that Configurations exist for the Device and the Services that the Device provides. Changes to a Product, a business definition of a Service, or an SLA all affect the Devices and/or network definitions of Services in the Product, and hence are reflected in Device Configuration changes. Finally, the network is driven by business changes!

4 Overview of DEN-ng

The original goal of DEN was to define a set of network services that met the needs of a set of applications according to the policies that were applicable at the current moment. In essence, it proposed a *lingua franca* that enabled applications to express their needs in terms that the network could understand, and vice-versa.

The DEN-ng work is being done in the TeleManagement Forum (TMF – see www.tmforum.org) because the TMF had already started work on its NGOSS architecture, and because the TMF realized the value of having a set of models that met the requirements of the previous section.

DEN-ng, like DEN, is built around using a finite state machine model to describe and control managed entities (another key point that other information models do not have). DEN-ng defines three fundamental types of classes to support a finite state machine: (1) classes to model the current state of a managed entity, (2) classes to

model the changing of state of a managed entity, and (3) classes to control when the state of a managed entity is changed.
Conceptually, DEN-ng is built as a framework of frameworks, as shown in Figure 3 below.

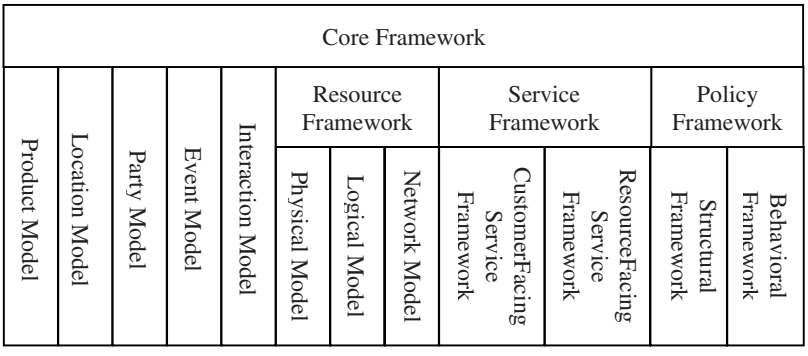

Fig. 3. Simplified View of DEN-ng

The Core Framework contains high-level entities and relationships that enable more specific domain models (even from other standards bodies and fora) to be integrated into a single cohesive whole. The Product model provides an abstraction for containing Resources and Services, and links both to other business entities, like Customer. The Location model enables semantics to be attached to locations. The Party model is an abstraction for individual and groups of users, along with organizations. The event model is based on the UML metamodel definition of event, and has been refined to include RM-ODP concepts of announcement and interrogation [16]. The Interaction model is based on several UML metamodel concepts, such as Collaborations, and is used to define the semantics of how entities interact with each other. The Resource Framework provides additional abstractions for representing physical, logical and network resources (each of which has their own detailed domain model). The Service Framework abstracts Services into two types – Services that a Customer is explicitly aware of, and Services that a Customer is not explicitly aware of. For example, a VPN is a service that a Customer can buy, so it is a CustomerFacingService. That VPN may use BGP to advertise routes, but (hopefully!) the Customer is blissfully unaware of BGP. Thus, BGP is a ResourceFacingService. Finally, the Policy model is divided into two domains. The Structural Framework contains a set of entities that can represent Policy across the Policy Continuum [3]. The Policy Behavioral Framework connects the representation of Policy to the other models in DEN-ng.
Note that the TMF has defined a Shared Information Model (SID) in [17] that is beginning to incorporate some of the DEN-ng models. Similarly, DEN-ng is heavily influenced by the SID. The SID is currently chartered just to work on the business model, whereas DEN-ng is focused on building out the business, system, and implementation viewpoints of the Resource, Service, and Policy domains. DEN-ng uses the SID work and then extends that work to the system and implementation viewpoints. These extensions are being proposed back into the formal SID work.

Unlike the other information models cited in this paper, DEN-ng and the SID use modern software techniques (such as Patterns [18], [19]) and various types of abstraction techniques, such as roles [20]. Two critical abstractions introduced by DEN-ng are capabilities and constraints. Capabilities are a means of abstracting the functionality of a device into a set of interoperable building blocks.

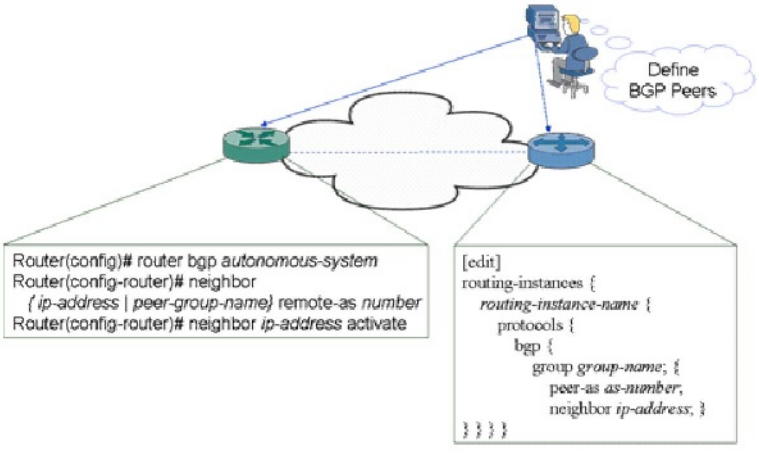

Fig. 4. Normalizing Different CLIs

Figure 4 shows an operator trying to accomplish the same task on two different routers. Even though the task is the same, the CLI is completely different. More importantly, the device on the left has different configuration *modes*, which are absent in the device on the right. Thus, the *programming model* for these devices is different. This makes it hard to build a single end-to-end service involving these two devices. Capabilities *normalize* the different functions of heterogeneous devices, so that a single control plane can be used to manage each device. This enables generic functions to be described independently of the programming model and type of device, which simplifies using heterogeneous devices.

The DEN-ng model also differs from other models in the level of granularity of information that it models. For example, in addition to high-level concepts like Product and Device, it contains very low-level concepts, such as a DeviceInterface, or the Configuration of a Device. This is because one of the important goals of the DEN-ng model is to be able to model configuration changes, as well as to show how policy can be used to control the deployment of these changes. Experience has shown that this mix of levels of abstraction is required to implement support for On Demand Networking, because the On Demand Network serves multiple users operating at multiple levels of abstraction.

Constraints are used to model restrictions on using certain device functions by the current environment. For example, one application may prohibit the use of a command, or restricting configuration changes to a particular time.

This combination of capabilities and constraints is essential for realizing the most fundamental of all requirements for building an autonomic network: *self-knowledge*.

The world will never adopt a single command language, or use a single programming model, or agree on a single platform. The only way to define knowledge is through a set of abstractions that can accommodate the different functionality and behavior of the elements of a system. The DEN-ng model is the first step in that direction.

5 The Holistic Combination of Policy and Process Management

Business process automation can be characterized by the separation of the expression and execution of business processes and services from the software that implements these business processes. This separation enables the application of business management techniques to the business processes that are implemented. There are two major efforts for standardizing business processes – the eTOM of the TMF [21] and ITIL [22]. The eTOM team in the TMF is currently producing a mapping between the eTOM and the relevant portions of ITIL.

Policy management is defined as the usage of policy rules to mange the configuration and behavior of one or more entities. The DEN-ng policy model [3] is designed as a class hierarchy that is used to represent the structure of policy rules as well as their semantics. This combination enables the DEN-ng policy model to be used as *reusable objects that control the state transitions of the managed objects of the environment*. This enables models to be built to represent the entire life cycle of the managed system.

Policies have traditionally been used primarily as a means to prioritize the allocation and access to resources by different applications. Commercial implementations, as well as the literature, have not defined *how* this prioritization occurs – it is just assumed that it has occurred, and is simplified into administrators listing policy rules in an *ad-hoc* fashion. The answer to this question lies in realizing that Policy implies the use of a methodology. Finite state machines have been chosen in DEN-ng because they are standardized in UML and they are a simple, well-known, and effective concept.

Currently, Process Automation techniques and Policy Management are separated. Instead, the approach defined in this paper suggests that the holistic combination of policy management and process management is what is needed. A simple, yet elegant, model ensues from this combination: policies are used to define goals to be achieved, which result in the selection of one or more processes. The execution of these processes are monitored, and their collective results analyzed to adjust (if needed) the set of policies that are active at any given time. Thus, a closed loop system is achieved. This is shown in Figure 5.

The above process starts with the important concept of separating the act of constructing a configuration change from the act of deploying that change. This realizes the basic fact that no two types of configuration changes are different. Sadly, most current approaches do not make this separation.

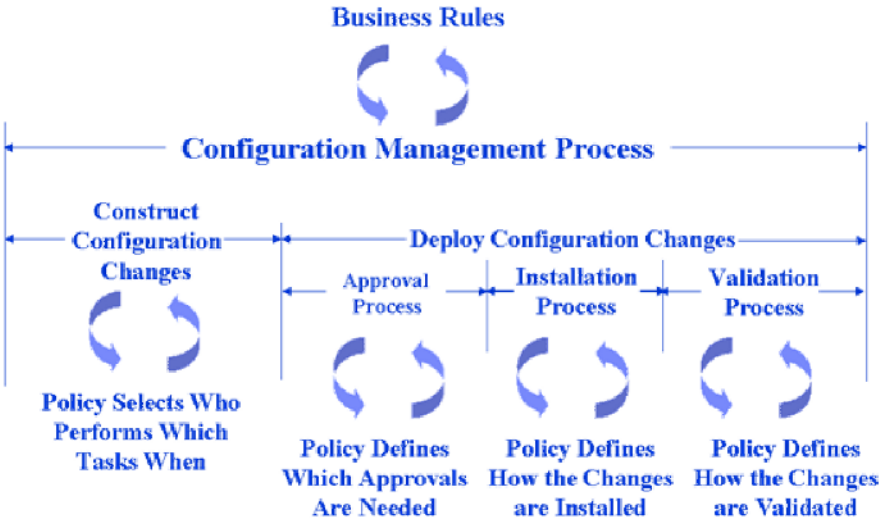

Fig. 5. Using Policy and Process Management for Resource Configuration

Consider two different configuration changes – a simple change of the SMTP address of a device, versus changing BGP peering relationships. These are very different in nature – the former is a simple and straightforward change, whereas the latter can be quite complicated, and can affect many different parts of the organization. However, even though the first change is simple, most organizations will define a time period for implementing this change. This is, of course, a *business policy*; the question is, how is it enforced in the network?

The latter change is more complicated to construct. In addition, this change could affect other devices and other services. Other (business) organizations will probably need to look at this change, since it could have significant adverse financial consequences if it is implemented incorrectly. Thus, this change should require multiple levels of technical and business approval. Unless all of these different reviews are related to each other, it will be impossible to relate device and service changes to external events, which could adversely affect the liability of the business. Again, how is this enforced in the network?

This paper proposes that policy and process management work together to control how a resource is (re)configured. The output of the business processes is monitored; the results of monitoring the business processes are used to adjust the set of policies that can be used at any given time. This closed loop system ensures a stable, repeatable means for adjusting the network to meet the demands of the organization.

6 The Generic Model Driven Architecture (MDA) Approach

MDA is defined by the OMG in [23]. The heart of the MDA effort is based on the use of open standards defined in the OMG – Unified Modeling Language (UML) [24]; Meta-Object Facility (MOF) [25]; XML Meta-Data Interchange (XMI) [26]; and Common Warehouse Meta-model (CWM) [27]. MDA uses these four standards to help separate business and application logic from their implementation. This results in better application reuse and increases the overall portability of an application that uses MDA. More importantly, it helps an organization implement their intellectual property in a modular fashion. A notable feature of MDA is that it addresses the complete life cycle of designing, deploying, integrating, and managing applications and their data.

UML is the key enabling technology for the MDA, as it enables every application to be based on a normative, platform-independent UML model. In the OMG approach, a platform-independent model (PIM) is mapped onto one or more platform-specific models (PSMs).

7 The New on Demand Networking MDA Approach

The MDA is a good approach. Nascent efforts in implementing the mappings are also good approaches. However, the scope of this paper is slightly different. It is not trying to solve the *general* PIM-PSM problem, nor is it trying to build *general* code generators. Instead, it is trying to use MDA techniques to build a new type of network management system – one that is driven by business requirements.

In order to connect the business world with the networking world, we need a way of representing information from the business and system viewpoints, and a method to tie them together. In order to implement this in a product, we need to tie the implementation viewpoint with the previous two viewpoints. Thus, these three viewpoints *jointly* affect the design of the models. The fact that there are three viewpoints means that abstraction must be used to encourage subject-matter experts in the business, system, and implementation domains to use the approach. The business entities are designed *in response to* standard process descriptions from the eTOM. The system view is the abstract specification of how these entities work together. The implementation view is the preparatory step towards implementing the system, and ensures that fundamental characteristics, such as the support of namespaces, are used. Experience has proven that building separate models to support these viewpoints cannot work, because it is impossible to keep the models synchronized. Rather, the analogy of a “database view” is used – information is hidden or shown as appropriate to support the current viewpoint.

Models are important to the On Demand Networking architecture, because they represent reusable abstractions that capture, in a formal way, the components used in the business, and how those components relate to the overall infrastructure as well as the managed environment. Thus, it is the model that drives the development of the application.

The key to using models for On Demand Networking is to use UML's inherent extensibility features to capture the unique semantics of networking. A *UML Profile* is a collection of model elements that have been customized for a specific domain or purpose by extending the metamodel using stereotypes, tagged definitions, and constraints. A *stereotype* is a model element that defines additional values (based on tag definitions), additional constraints, and optionally a new graphical representation. *Tag definitions* specify new kinds of properties that may be attached to model elements. The actual properties of individual model elements are specified using *Tagged Values*. *Constraints* express (typically) invariant conditions that must hold true for the set of entities that are being modeled. Note that constraints by definition do not have side effects, and therefore cannot alter the state of the system.
Figure 6 shows the code generation process used for DEN-ng.

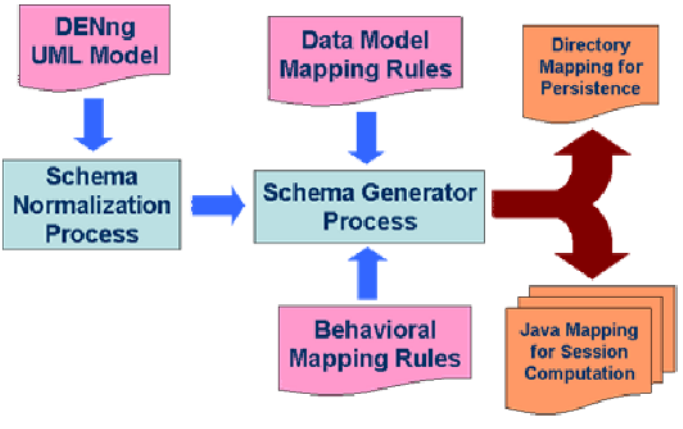

Fig. 6. Generating Code for On Demand Networking

Since DEN-ng contains business, system and implementation information, this process is independent of the type of model for which code is being generated. The UML source files are parsed and put into a canonical format as preparation for being mapped into a set of data models. This process uses one or more model mapping rule files to generate schemata for the appropriate target data models, along with rules that define precise semantics for the behavioral components of the UML models. We have to date successfully generated schemata for directories as well as for Java environments. More importantly, data coherency is maintained even though the two data models are significantly different, because they are both derived from the same information model.
It is important to understand why both of these data models are generated. Directories are for searching, and should only be used for storing entities that change much slower than the replication frequency of the directory. The Java entities are used for session computation, and are better suited to implement business rules and constraints specified in the model. They are used for per-session computation, with the result being persisted in an appropriate repository.

8 Conclusions and Future Work

This paper has presented a new approach to building support for On Demand Networking. This support is based on four key principles. First, the DEN-ng information model is used as a single means to represent management information. This choice is made because of its following unique characteristics: it is based on UML, it is inherently extensible through its use of patterns, it supports multiple levels of abstraction and multiple viewpoints, it uses a finite state machine, and it contains low-level information modeling (e.g., a "device interface") that other information models do not have. Second, the DEN-ng information model supports multiple mappings to different data models. This is mandatory to support the diverse types of management information that On Demand Networking requires. Third, the holistic combination of policy management and process management are used to mange entities in the On Demand Network. Conceptually, policies control state transitions of a managed entity, and processes are used to implement the goal of the policy. Finally, a variant of the OMG's MDA approach was defined. This variant was specifically tailored to meet the needs of On Demand Networking.

Future work will explore three different areas of research. First, more complicated network support will be built, to prove that this approach does indeed work generically. Second, the introduction of other autonomic elements, such as network devices or host systems, will be explored. Third, the concept of knowledge that spans multiple system elements for On Demand Systems will be examined.

References

[1] Please see: http://www7b.software.ibm.com/dmdd/library/techarticle/0302iwb/0302iwb.html

[2] Please see: www.omg.org/mda

[3] J. Strassner, "Policy-Based Network Management", Morgan Kaufman Publishers, ISBN 1-55860-859-1, to be published 2003

[4] J. Strassner, *Directory Enabled Networks,* Macmillan Technical Publishing, 1999, ISBN 1-57870-140-6

[5] The DMTF has redefined DEN to fit better to its CIM and WBEM efforts. The author does not agree with this redefinition. The DMTF definition is found here: http://www.dmtf.org/standards/standard_den.php

[6] TM Forum, NGOSS Architecture, TMF 053

[7] J. Strassner, "A New Paradigm for Network Management: Business Driven Device Management", SSGRRs Conference, August, 2002

[8] J. Strassner, "*Directory Enabled Networks*", Chapter 10, Macmillan Technical Publishing, ISBN 1-57870-140-6

[9] B. Moore, E. Ellesson, J. Strassner, A.Westerinen, "*Policy Core Information Model – Version 1 Specification*", RFC 3060, February 2001

[10] B. Moore, L. Rafalow, Y. Ramberg, Y. Snir, A. Westerinen, R. Chadha, M. Brunner, R. Cohen, J. Strassner, "*Policy Core Information Model Extensions*", draft-ietf-policy-pcim-ext-06.txt, November 2001

[11] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, B. Moore, "*Policy QoS Information Model*", draft-ietf-policy-qos-info-model-04.txt, November 2001

[12] S. Gai, J. Strassner, D. Durham, S. Herzog, H. Mahon, F. Reichmeyer: "QoS Policy Framework Architecture", February 1999
[13] Current work is in the DMTF Members only site, under: http://www.dmtf.org/apps/org/workgroup/policy/. The current release of CIM's policy model as of this writing is version 2.7.1, and is located at: http://www.dmtf.org/standards/standard_cim.php.
[14] Please see the following web page for detailed information about Ponder: http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml
[15] Open Distributed Processing Reference Model – Foundations, ISO/IEC 10746-2, 1996
[16] Open Distributed Processing Reference Model – Overview, ISO/IEC 10746-1, 1998
[17] TMF, "Shared Information/Data (SID) Model", GB922 and its Addenda (one for each domain, 10 so far)
[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", ISBN 0-201-63361-2, October 1994
[19] The Patterns Homepage contains a variety of useful links, and is at: http://hillside.net/patterns/
[20] D. Bäumer, D. Riehle, W. Siberski, M. Wulf, "The Role Object Pattern", can be downloaded from: http://jerry.cs.uiuc.edu/~plop/plop97/Proceedings/riehle.pdf
[21] F, "eTOM, the Business Process Framework", GB921, version 3.5, 2003
[22] The home page of ITIL is: http://www.itil-itsm-world.com/
[23] The home page of the MDA effort is: www.omg.org/mda
[24] The home page for the UML effort is: http://www.omg.org/technology/documents/formal/uml.htm
[25] The home page for the MOF effort is: http://www.omg.org/technology/documents/formal/mof.htm
[26] The home page for the XMI effort is: http://www.omg.org/technology/documents/formal/xmi.htm
[27] The home page for the CWM effort is: http://www.omg.org/technology/documents/formal/cwm.htm