Towards a Context-Aware Information Model for Provisioning and Managing Virtual Resources and Services

Yeongrak Choi[1], Jian Li[2], Yoonseon Han[1],
John Strassner[1], and James Won-Ki Hong[1,2]

[1] Division of IT Convergence Engineering, POSTECH, Pohang, Korea
[2] Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea
{dkby,gunine,seon054,johns,jwkhong}@postech.ac.kr

Abstract. The demand for new and innovative services, and especially for personalized services, continues to increase. In such systems, different services may compete for the same set of shared resources. Hence, the *mix* of services and resources that each has often results in conflicting demands on shared resources, and is becoming increasingly difficult to manage. Autonomic systems that provide virtual resources and services can provide important management benefits to ensure that the needs of different services can simultaneously be met. This paper describes the requirements for provisioning and managing virtual resources and services, and extends the DEN-ng information model to architect such systems.

Keywords: Information Model, Autonomic Systems, Provisioning, Virtual Resource Management, Service Management, Context Awareness.

1 Introduction

Information technology systems are getting more complicated to manage. This is caused by many factors, such as an increased number of technologies and the increased complexity of individual devices that are used. This paper is mainly concerned with one type of business complexity – the ability to manage different services that each has different needs and diverse business objectives. This places two demands on the network and server infrastructure: (1) how to accommodate the various needs of different services that are sharing the same resources, and (2) how to accommodate the increasing amount and diversity of management and operational information. For instance, personalized services aim to provide customers with services that are tailored to their tasks and needs. While this is attractive for customers, it places a significant management burden on the provider of such services, who must try to accommodate different variations of the same service that are delivered to different customers.

One way to achieve this is to utilize virtual resources to create personalized virtual services. Virtualization has many benefits. For example, it can isolate different machines and applications that have maintenance and security requirements,

R. Brennan, J. Fleck II, and S. van der Meer (Eds.): MACE 2010, LNCS 6473, pp. 100–112, 2010.

decoupling hardware, operating system, and application dependencies. It can simultaneously provide availability, reliability, and backup policies that are targeted to the needs of different (and possibly isolated) machines and applications [15] [16]. By dynamically allocating the resources of physical servers among different applications, the business requirements of different applications can be met. However, virtualization technologies demand careful management. In order to use virtualization technologies effectively, policy management can be used to define which applications receive which virtual resources.

FOCALE [1] is a novel autonomic architecture for providing *context-aware* policy management [2]. In this approach, context is used to select the applicable working set of policy rules that are applicable for a given context; these policy rules are used to govern services and resources offered by the system being managed. This enables FOCALE to select new policy rules to govern the system (if appropriate). FOCALE uses separate maintenance and adjustment control loops to manage functionality. FOCALE uses the DEN-ng information model [3] to define context and policy management [4]. In this approach, knowledge from models is augmented with knowledge from ontologies to capture semantics and better understand the behavior of entities in the system through machine-based learning and reasoning.

Our modeling approach intends to fill the gap between managing virtual resources and the autonomic architecture to provide personalized services. When diverse applications, which are used to run personalized services, are running on our autonomic architecture, virtualization technologies simplify the actions of matching applications to available resources. Our model also addresses the investigation of the link between virtual resources and their services. In cloud computing, Infrastructure as a Service (IaaS) provides accessibility to a set of virtual resources that are tightly coupled to a given service, which can be used to form a building block to implement personalized services [17].

This paper focuses on extending the DEN-ng information model for representing and managing the provisioning of virtual and non-virtual heterogeneous resources and the services that they support. In provisioning, this model helps to decrease the effort of configuring the corresponding virtual and non-virtual resources; this in turn simplifies defining, deploying, and managing services. Moreover, this model shows how to reduce management complexity by providing unified management mechanisms for both virtual and non-virtual resources. This enables the existing model objects in DEN-ng, such as objects that represent the roles of people and devices, to be used to manage virtual as well as non-virtual resources and services.

The organization of the paper is as follows. Section 2 addresses key modeling requirements of virtual resources and services provisioning and management to provide personalized services. Section 3 explains the enhancements made to the original DEN-ng 7.0 model. Section 4 presents a use case which describes the applicability of our proposed model. Section 5 describes related work, and Section 6 discusses conclusions and future work.

2 Modeling Requirements for Provisioning and Management

In this paper, we define “provisioning” as a set of processes to deliver services (and optionally, resources) to customers according to certain service level agreements. The

Information Technology Infrastructure Library (ITIL) [5] and the enhanced Telecom Operations Map (eTOM) [6] define high-level "best practices" that describe processes and process workflows that include provisioning and related functions; however, they do not describe lower-level aspects of how to provision infrastructure components. The ITIL and eTOM documents are inherently high-level because different organizations use different business processes and workflows to accomplish the same task. This can be remedied by defining a common set of objects to represent concepts corresponding to different concerns, ranging from business concepts such as Service Level Agreements (SLAs) to network concepts such as protocols and device configurations. The OMG's Model Driven Architecture (MDA) initiative [7] can be used to define and generate code for the different objects required to create, deploy, and manage the provisioning process. This results in the following key requirements that an information model must satisfy to achieve this.

An information model must be able to represent entities in an extensible manner. The use of software patterns [8], such as the role-object pattern [9] and the policy management pattern [3], can be used to provide an extensible framework that can represent these and other variations while avoiding class explosion.

An information model must be able to clearly differentiate between entities and metadata that can describe various aspects and behavior of a managed entity.

An information model should be able to represent various policy rules for governing system operation. Typically, many policy rules will be applicable to a given situation; hence, it is critical to select the correct set of policy rules for a given context in order to define appropriate system behavior and services. A very powerful mechanism for doing this is to use context to select the set of policy rules that are applicable at a given point in time.

3 Design of the Enhanced DEN-ng Information Model

This section defines extensions to the DEN-ng 7.0 information model that define a more flexible and feature-rich approach for representing provisioning and managing virtual and non-virtual resources and services. This paper describes specific extensions to the Resource domain, which is part of a larger effort to represent and manage virtualization technologies.

3.1 Introduction of VirtualResource into the DEN-ng Model

Fig. 1 illustrates our extension of the Resource domain of DEN-ng to accommodate virtualization. We used the following three steps to develop this extension: 1) define a set of initial classes using applicable software patterns, 2) extend this model by connecting elements in this model to applicable parts of the rest of the DEN-ng model, and 3) represent the complex interaction between resources and other important entities specifically for provisioning.

In order to model virtual resources in a technology-neutral way, we decided to build our model based on the existing DEN-ng information model. In DEN-ng 7.0, the definition of a resource was: "A physical and/or logical entity that is of interest to the managed environment". The DEN-ng 7.0 Resource class had three subclasses: PhysicalResource (i.e., an object that you can pick up and hold), LogicalResource

(i.e., objects that cannot be picked up and held), and CompoundResource, which enables managed entities to have both physical and logical resource characteristics and behavior (e.g., a node in a topology diagram that has physical and logical components).

To maintain compatibility with the existing DEN-ng model, we defined a model for virtual resources with the following changes. First, we changed the hierarchy of Resource to include the VirtualResource and NonVirtualResource subclasses. Second, we changed the definition of Resource to encompass both of these two subclasses; its new definition is: "A Resource is any virtual or non-virtual component or system. A Resource contains a set of physical and/or virtual entities that are of interest to the managed environment. A Resource may represent a limited or critical entity that is required by other entities." Third, we used the original DEN-ng definition of Resource as the new definition of NonVirtualResource, and moved the original Resource hierarchies under NonVirtualResource. Fourth, we made a new definition for VirtualResource, which is: "A VirtualResource is an abstraction that decouples the physical manifestation of a Resource from its logical operation". Note that we are ignoring for the moment the experimental definition of virtual resource present in this version of DEN-ng; this version was built by the AutoI FP7 team [10], and will be discussed further in section 4. Finally, we are ignoring CompoundResource for the sake of simplicity.

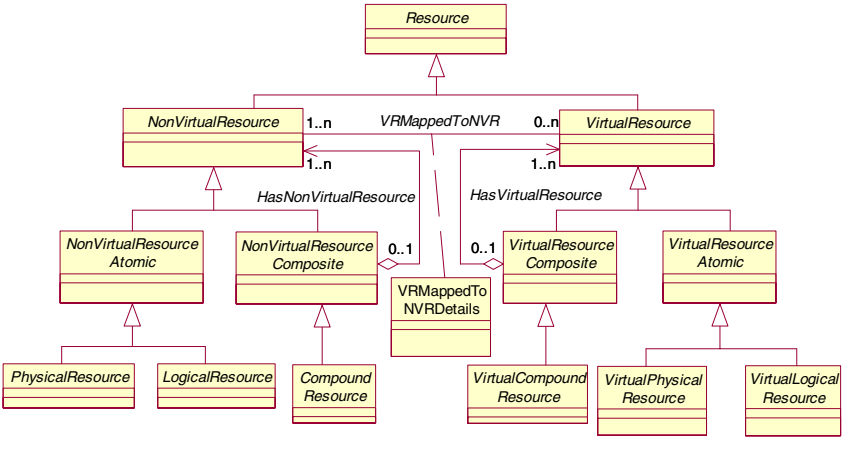

Fig. 1. Modified Top-level Resource Model

Now, we add an association between NonVirtualResource and VirtualResource, which models the dependency that VirtualResources have on their non-virtual hosts. This is shown in Fig. 1. The composite pattern is a powerful and extensible way to define graph and tree structures that represent part-whole hierarchies [8]. The inherent extensibility that the composite pattern provides implies that it can also be applied to the NonVirtualResource hierarchy to enhance its functionality. Therefore, we apply the composite pattern to both VirtualResource and NonVirtualResource, as shown in Fig. 1.

When we apply the composite pattern to the original (now non-virtual) Resource, we need to move its original three subclasses (PhysicalResource, LogicalResource, and CompoundResource) to one of the new subclasses introduced by the composite pattern. Both PhysicalResource and LogicalResource can be atomic classes, whereas CompoundResource is inherently composite (because it contains at least one physical and at least one logical Resource). Therefore, we make PhysicalResource and LogicalResource subclasses of NonVirtualResourceAtomic, and CompoundResource a subclass of NonVirtualResourceComposite, as shown in Fig. 1. Note that VirtualPhysicalResource was modeled as a subclass of VirtualResourceAtomic. This class represents the physical concepts of a Resource that has been virtualized (e.g., virtualized CPU, memory, hard-disk, and etc.). In addition, it describes different types of virtual hardware that constitutes a virtual product.

3.2 Management Using Context-Aware Policy Rules

We have introduced the VRMappedToNVR association, which models the dependency between a NonVirtualResource and the set of VirtualResources that are created from it. However, the interaction between these two different types of Resources is very complex, and additional classes and relationships must be introduced in order to properly model its semantics. Moreover, modeling the interaction without proper patterns may highly constrain its scalability as well. In order to solve these two problems, we utilized the policy pattern [2] [3].

Fig. 2 illustrates how context-aware policies are used to manage subclasses of VirtualResources. In Fig. 2, the policy pattern avoids various anti-patterns as well as a tendency to represent policy rules as “special cases” that cannot be shared or reused. Since various policy rules could be applied to govern how virtual resources are created and managed, a pattern is a particularly useful way to model this feature, since it provides a template for implementing a reusable solution to a commonly occurring design problem. By applying the policy pattern, we not only separate the representation of policy from its content, but we enable the semantics of the association to be adjusted according to the nature of the policy. By adding an association from PolicyRuleStructure to VRMappedToNVRDetails, policy related DEN-ng classes such as PolicyRuleStructure (this is the superclass of different types of policy rules, including event-condition-action, goal, and utility function policies [3]) and ManagementPolicy (this is the superclass of policy rules that are bound to a particular target entity [3]) can be used to define policy rules in an extensible fashion [2]. For example, different metadata can be applied to different types of policy rules to ensure that they are applied to specific types of managed entities.

Policy rules can then be associated with context to change the governance mechanisms in accordance with changes in context. In DEN-ng, context is modeled using two main classes: Context and ContextData. The latter represents different aspects of context, while the former represents a complete aggregate of context aspects that together represent a complete whole. For example, a virtual service may be made supported by different virtual devices; the set of resources that each virtual device provides would be a collection of ContextData, and the virtual service would be modeled by Context.

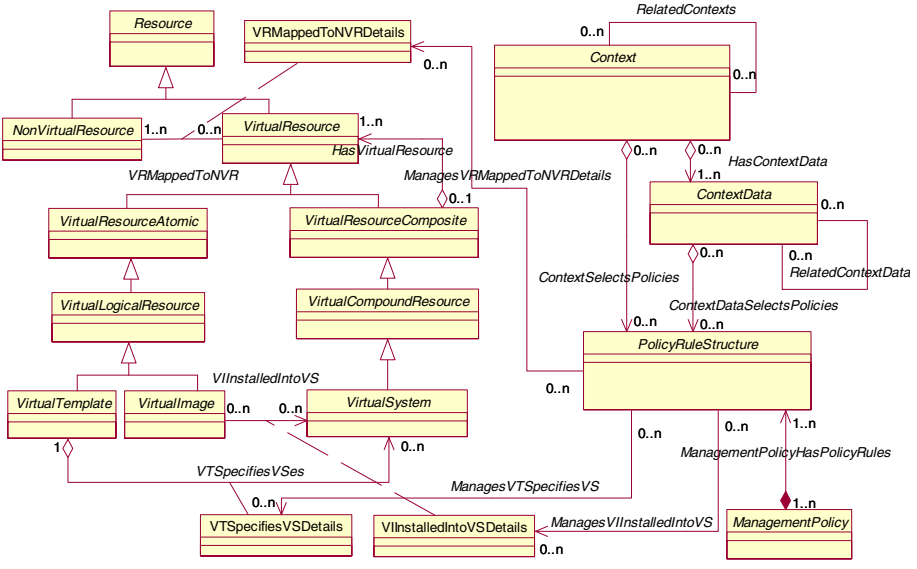

Fig. 2. The Use of Context-Aware Policies for Managing VirtualResource Subclasses

The basic DEN-ng loop uses Context and ContextData to select the appropriate set of policy rules to realize the set of governance operations required. Policy rules select roles that entities play; this is used to define functionality as well as to define the set of management and operational data that should be monitored by the control loop. Hence, context changes can be detected by changes in the state of the entities being managed; this (as well as other context changes) is used to adjust the policy rules that are used to define the services and resources that are offered at any given time.

3.3 Interacting PersonRole with VirtualResources

Fig. 3 shows the interaction of PersonRole and the subclasses of VirtualResources, such as VirtualTemplate, VirtualImage and VirtualSystem. Note that several classes and associations are not shown to make this figure easier to read.

The PersonRole concept is reused in its entirety from DEN-ng, which has a rich model for representing different types of interactions between people, devices, and services. We extended the concept of VirtualResource to support provisioning virtual environments using the VirtualSystem, VirtualImage, ImageRepository, VirtualTemplate, and TemplateRepository classes and their subclasses.

A VirtualSystem (VS) is a software implementation of a system, such as a computer, that contains PhysicalResources as well as LogicalResources. In this model, we modeled the VirtualSystem as a subclass of VirtualCompoundResource, because this concept must contain both VirtualPhysicalResources and VirtualLogicalResources.

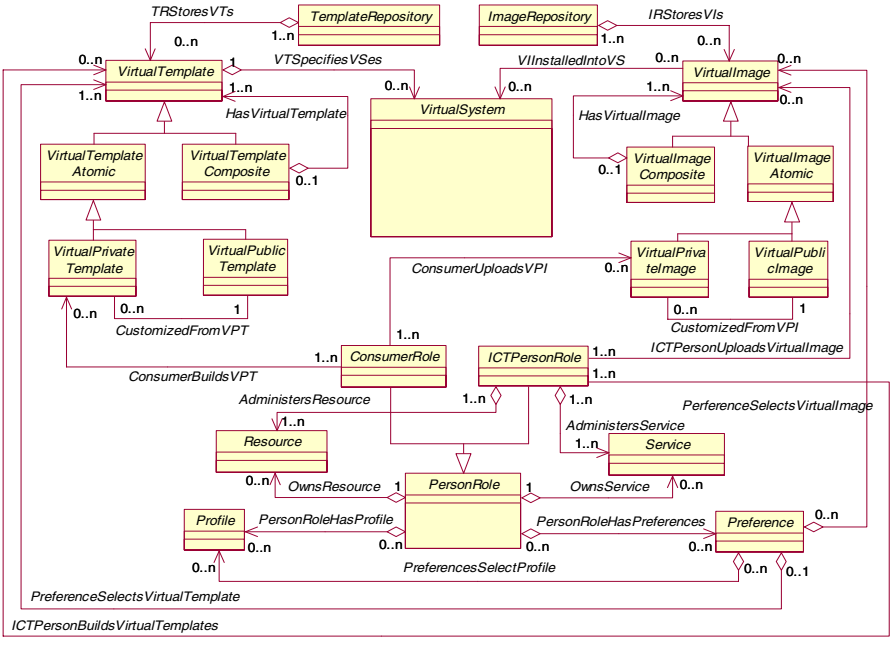

Fig. 3. Interaction between VirtualImage, VirtualTemplate and PersonRole

A VirtualImage (VI) is a file on a physical disk that is interpreted by a Virtual Machine Monitor (which is also known as a Hypervisor) as a repository [11]. The VirtualImage holds the file system where the OS will run from. VMMs enable multiple OSs to run concurrently on a host computer.

A VirtualTemplate (VT) is a virtual hardware specification and configuration description of a VirtualSystem. A VirtualTemplate may contain meta information that helps a VMM to generate a VirtualSystem, such as: 1) type of processors to be assigned to the VS, 2) the amount of memory the VS requires, 3) the location of the VI that contains the OS that runs on the VS, 4) virtual IP and MAC addresses to be assigned to the VS, and 5) other information that might be used for billing or other purposes.

There are an important set of associations between VIs and PersonRoles as well as between VTs and PersonRoles. A virtual *public* template (or image) is a VT (or VI) that is defined by an administrator for use by different types of ConsumerRoles. In contrast, a virtual *private* template (or image) is a VT (or VI) that has been personalized to suit the needs of a specific ConsumerRole. The full model realizes the CustomizedFromVPT (and the CustomizedFromVPI) associations as association classes, and then uses the policy pattern in a manner similar to that explained in the previous section to govern which set of policy rules control which set of changes a given ConsumerRole can make to a public VT (or VI).

VirtualImages may contain other VirtualImages; the same applies to VirtualTemplates. We used the composite pattern to model the creation of hierarchies of configurations for both VIs and VTs.

A PersonRole represents the part played by a Person in a given context. This includes the definition of any appropriate characteristics and behavior that it entails. There are two main types of PersonRoles: (1) ICTPersonRole and (2) ConsumerRole. ICTPersonRole represents PersonRoles that are involved in the design, development, management, and maintenance of information and communication technologies. ConsumerRole represents PersonRoles that can use, consume, or subscribe to Products, Services, and Resources. Since the roles for ICTPersonRole and ConsumerRole are different, their relationship with VirtualTemplates and VirtualImages are different. We use the role-object pattern, which enables a component object to be adapted to different needs through transparently attached role objects [8] [9], to represent these differences. This pattern is especially useful in separating the intrinsic and contextual characteristics and behavior of an entity. For example, ICTPersonRole could have been granted all privileges on building VirtualTemplates and VirtualImages no matter whether they are public or private; in contrast, ConsumerRole only has the permission to build his own private VirtualTemplate and VirtualImage. The use of private VirtualTemplates and VirtualImages enables the service configuration to be customizable according to Customers profiles and preferences.

3.4 Providing Services with VirtualResources

VirtualResources also play a key role of supporting the delivery of Services. In the original DEN-ng, PhysicalResource and LogicalResource are associated with ResourceFacingService, an abstraction that defines the characteristics and behavior of a particular Service that is not directly seen or purchased by customers but is nevertheless required for the proper operation of the service. (This is in contrast with CustomerFacingService, which is an abstraction that defines the characteristics and behavior of a particular Service that a customer purchases, leases, uses and/or is directly aware of in some other way). Similarly, VirtualResources are also associated with ResourceFacingServices.

Fig. 4 illustrates how the subclasses of VirtualResources are associated with ResourceFacingServices. The aggregation VPRHostsRFS defines the set of VirtualPhysicalResources that are required for this particular ResourceFacingService to function correctly. This is a *passive* relationship, and is used primarily to define which PhysicalResources are required to host a given service. The VLRImplementsRFS aggregation defines the set of VirtualLogicalResources that are required for the particular ResourceFacingService to function correctly. This is an *active* relationship, and defines the set of LogicalResources that are required to enable a ResourceFacingService to function. With these two aggregations, this information model can represent how a VirtualSystem hosts and implements a ResourceFacingService, because its super class VirtualCompoundResource is composed of a set of VirtualPhysicalResources and VirtualLogicalResources.

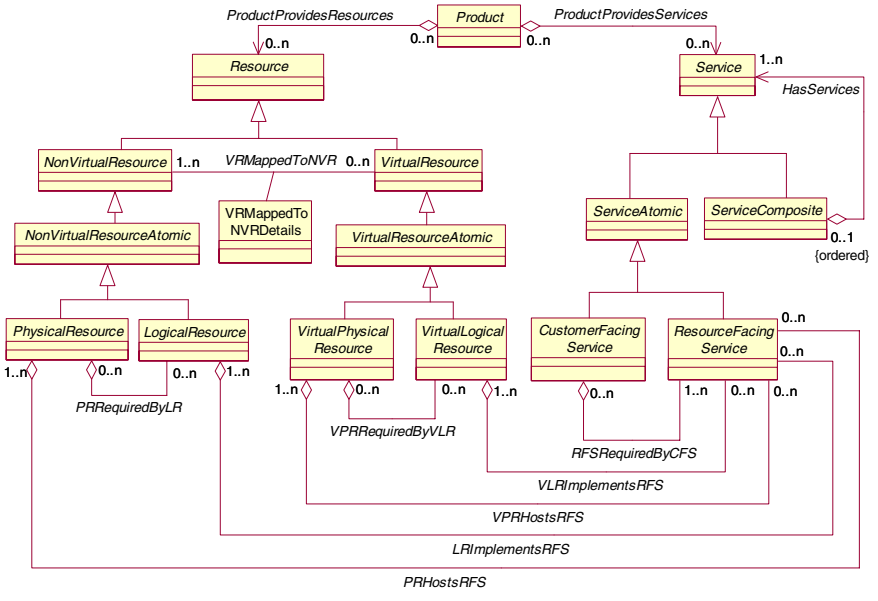

Fig. 4. Interactions between Services and VirtualResources

4 Applicability to SLA-Aware Services with Virtual Resources

The use case scenario depicted for the applicability of the proposed model presents the ability to satisfy the need of personalized services with virtual resources that are governed by one or more service level agreements. The scenario demonstrates how the contracts between a provider and the user of personalized services are not violated when virtual resources are provisioned and delivered into customers.

In DEN-ng, a service level agreement (SLA) is defined as a type of agreement that represents a formal negotiated agreement between two parties designed to create a common understanding about products, services, priorities, responsibilities and so forth. An SLA is one way to define the service level that is expected for a service, and can thus be used for personalized services as well. A DEN-ng SLA specifies a set of appropriate procedures and target metrics between parties to achieve and maintain specified goals, such as Quality of Service. However, it is difficult to reflect the characteristics of both non-virtual and virtual resources responding to personalized services into a contract. Although virtualization has interoperability advantages by abstracting the physical details of heterogeneous resources, some SLA statements may need to refer to specific resources. In addition, SLA terms are interpreted differently for different contexts. For example, when the security level is described for providing a personalized service, the context of the person is critical. The same person may be involved in tasks that have very different security requirements (e.g., work vs. entertainment). In addition, individuals will have different privacy requirements that are again influenced by context. To address these problems, our

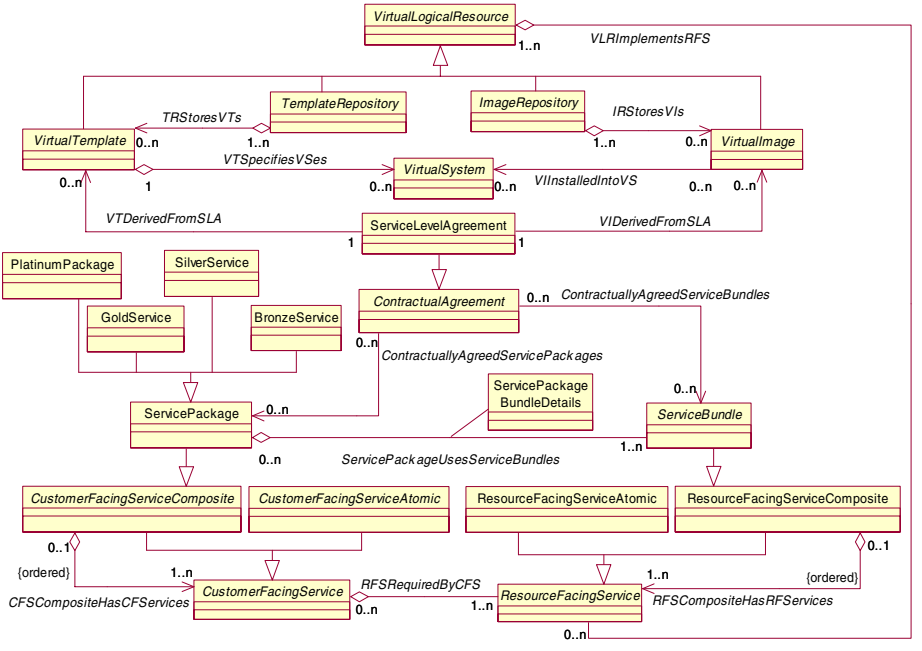

Fig. 5. Providing SLA-aware personalized services with virtual resources

modeling approach helps to identify items and functions and relate them to virtual and non-virtual resources, and then define how both are mapped to the same SLA.

Fig. 5 addresses how this use case can be applied using our proposed model when a VirtualSystem is provisioned based on VirtualTemplates and VirtualImages. For providing personalized services, a provider can define a set of services, where each service has different characteristics for various user-specific applications. For instance, the provider can define four different concrete service classes, such as premium, gold, silver and bronze service, to describe unique aspects of different personalized services. These classes are subclasses of CustomerFacingService. VirtualTemplate and VirtualImage are used to provision the appropriate virtual logical resources for a particular set of services. In Fig. 5, the ServiceLevelAgreement class (which represents an SLA) is defined as a subclass of the ContractualAgreement class. VirtualTemplate and VirtualImage are generated in accordance with ServiceLevelAgreement by considering interests of both users and the provider. Contractual agreement details of the ServiceLevelAgreement are described using the ServicePackage and ServiceBundle classes. A ServiceBundle class represents a collection of specifications of ResourceFacingServices; each ResourceFacingService defines a particular *class of service* that in turn specifies the Quality of Service (QoS) that this ResourceFacingService should receive. Each ServiceBundle interacts with one or more VirtualLogicalResources. In contrast, a ServicePackage contains a set of applications that are to be collectively treated according to a given SLA, and uses a set of ServiceBundles (one for each application in the ServicePackage) to deliver the application-behavior required. For example, a GoldService user and a SilverService

user may use the same application, but the QoS for the GoldService user will be higher than the QoS for the SilverService user. These relationships strengthen the provisioning of personalized services with virtual resources even when the contract is changed between the provider and customers. When there are modifications of an SLA due to the changes in business requirements or customer needs, the associations of SLA related classes can identify which changes should be applied to fulfill the contract.

5 Related Work

[3] provides a comparison of the DEN-ng, DMTF CIM [13], and TMF SID [14] models. There are five key reasons for choosing DEN-ng. First, the CIM and the SID do not have a context model, whereas DEN-ng has a well-developed context model. In network management, there will be hundreds of thousands of policies of different types (e.g., business, system, and device configuration); at any given time, most will not be applicable. Furthermore, as business needs and environmental conditions change, virtual resources and services need to change as well. Context is therefore critical for both selecting the set of policies that are applicable at any given time as well as determining the specific set of virtual resources and services required. Second, the CIM and the SID do not have a metadata model, whereas DEN-ng has a very well developed metadata model. This means that the CIM and the SID cannot differentiate between data that describes the intrinsic characteristics and behavior of a managed entity versus data that describes how that managed entity is used; this limits extensibility. Third, the CIM does not use software patterns, and the SID has only a small fraction of the patterns used in DEN-ng compared to DEN-ng. This also limits the extensibility of the model. Fourth, the CIM and the SID do not provide any mechanisms to orchestrate behavior, such as a finite state machine, which DEN-ng does provide. This is crucial, as it enables state and actions to be taken to be directly translated from the model into generated code. Finally, unlike the CIM and the SID, DEN-ng models an entity in a *holonic* fashion – as a complete entity as well as a collection of different aspects of an entity; this provides a powerful and extensible way to represent and manage behavior.

The AUTOI project [10] models the virtualization of network resources and services. The objective of the AUTOI project is to define a set of management overlays to control virtualized network resources and services defined by the mapping of business requirements to network functionality. The core idea is to use a common language in a machine-understandable form to orchestrate heterogeneous resources and services, and translate it to a set of Domain Specific Languages to control each actual resource and service. The AUTOI model uses an extension of the DEN-ng information model as the common language to capture and express all necessary concepts in the environments. The main benefit of the AUTOI model is a natural, continuous linkage and orchestration of business goals, virtual resources, and services. However, this model is limited in functionality and extensibility compared to the model described in this paper. Specifically, the VirtualResource class of AUTOI is defined as a subclass of Resource, which is not technically correct since it is not a Resource according to the DEN-ng definition of “resource”. While relationships are

defined between PhysicalResource and VirtualPhysicalResource (and similarly for their logical counterparts), these relationships are very high-level in nature and do not provide enough details to properly model the provisioning of virtualized services. Hence, our proposal would be to use the AutoI framework, but replace the existing AutoI model with our model.

6 Concluding Remarks

Although there have been many different virtual resources and services already deployed, managing these virtual resources and services is still a very complicated process. To our knowledge, no detailed model exists that can be used to provision and manage virtual resources and their services. This paper has briefly described such a model, and more importantly, has shown how it can be used to *personalize* services. Our modeling approach helps to represent virtual resources and their services by defining a set of classes and representing associations with software patterns, which provides inherent extensibility. For future work, we will continue to develop this model, and prototype how VirtualResources and VirtualServices are managed using ontologies and first order logic. The advantage of adding ontologies and first order logic is that in a manner similar to FOCALE, we could then treat data corresponding to the model as *facts* that could then be sent to the ontologies, where inferences could be produced using first order logic. Once this is done, we can realize a set of detailed network scenarios, deploy this model and its associated ontologies into the FOCALE architecture, and then prototype how managing VirtualResources and VirtualServices could be used for provisioning and managing more innovative and customized services with VirtualResources and VirtualServices.

Acknowledgments. This work was partly supported by the IT R&D program of MKE/KEIT [KI003594, Novel Study on Highly Manageable Network and Service Architecture for New Generation] and WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

References

1. Strassner, J., Agoulmine, N., Lehtihet, E.: FOCALE – A Novel Autonomic Networking Architecture. International Transactions on Systems, Science, and Applications (ITSSA) Journal 3(1), 64–79 (2007)
2. Strassner, J., de Souza, J.N., Raymer, D., Samudrala, S., Davy, S., Barrett, K.: The Design of a Novel Context-Aware Policy Model to Support Machine-Based Learning and Reasoning. Journal of Cluster Computing 12(1), 17–43 (2009)
3. Strassner, J.: Introduction to DEN-ng, Tutorial for FP7 PanLab II Project, January 21 (2009), http://www.autonomic-communication.org/teaching/ais/slides/0809/Introduction_to_DEN-ng_for_PII.pdf
4. Strassner, J.: Based Network Management. Morgan Kaufman, San Francisco (2003)
5. ITIL, http://www.itil-officialsite.com/home/home.asp

6. TMF, eTOM, http://tmforum.org/BusinessProcessFramework/1647/home.html
7. OMG, MDA, http://www.omg.org/mda/
8. Gamma, E., Helm, R., Vlissides, J.: Design Patterns-Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (November 2000)
9. Bäumer, D., Riehle, D., Siberski, W., Wulf, M.: The Role Object Pattern, http://hillside.net/plop/plop97/Proceedings/riehle.pdf
10. AUTOI (Autonomic Internet, an FP7 project), http://ist-autoi.eu/autoi/index.php
11. OpenNebula, http://www.opennebula.org/documentation:documentation
12. Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., Waldbusser, S.: Terminology for Policy-Based Management. RFC3198 (November 2001)
13. DMTF, CIM Schema: Version 2.24, http://www.dmtf.org/standards/cim/cim_schema_v2240
14. TMF, SID Schema, http://tmforum.org/InformationFramework/1684/home.html
15. Federica (Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures) project, http://www.fp7-federica.eu
16. Reservoir fp7 project, http://reservoir-fp7.eu/index.php
17. IAAS framework project, http://www.iaasframework.com