Automated Generation of Resource Configurations through Policies

Akhil Sahai, Sharad Singhal, Vijay Machiraju
HP Laboratories
1501 PageMill Road, Palo-Alto, CA 94304, USA
Rajeev Joshi[1],
Jet Propulsion Laboratories, Pasadena, CA
{asahai, sharad, vijaym}@hpl.hp.com, rjoshi@jpl.nasa.gov

Abstract

Resource Management systems have been attempting to undertake automated configuration management. Automated configuration management involves considering user requirements, operator constraints and technical constraints of the system to create a suitable configuration, and to create a workflow to deploy it. In this article we propose a policy-based model that we have used for automating these configuration management aspects.

1. Introduction

Resource management systems have been trying to create systems that provide automated provisioning, configuration, and lifecycle management of a wide variety of resources. The current trend in utility computing is a step towards creating such automated resource management systems. HP's Utility Data Center product [1], IBM's "on-demand" computing initiative [2], Sun's N1 vision and Microsoft's DSI initiative, Grid initiative are examples of this trend. However, the resources that are available to these resource managements systems are "raw" computing resources (e.g., servers, storage, network capacity) or simple clusters of machines. The user has to still manually install and configure applications, or rely upon a managed services provider to obtain pre-configured systems from service providers.

Because every user's needs are different, it is usually not possible to create custom environments for every user—managed service providers rely on a small set of pre-built (and tested) application environments to meet each user's needs. However, this limits the ability of users to ask for applications and resources that have been specially configured to meet their needs. In our research, we are focusing on how complex application environments (e.g., an e-commerce site, oracle clusters) can be automatically "built-to-order" for users. In order to create a custom solution that satisfies user requirements many different considerations have to be taken into account. Typically, the underlying resources have technical constraints that need to be met in order for valid operations, e.g., not all operating systems will run on all processors, and not all application servers will work with all databases. In addition, system operators may impose constraints on how they desire such compositions to be created. For example, when resources are limited, only certain users may be able to request them. Finally, the users themselves have requirements on how they want the system to behave. Thus, automating the design, deployment and configuration of such complex environments is a hard problem.

In this paper we describe a model for generating specifications for such environments based on policies. Policies have been traditionally described as rules that change the behavior of a system [3] and policy based management has been viewed as an administrative approach to simplify management by associating certain conditions with actions.

In our model for resource composition [4], the complex environments themselves are treated as higher-level resources that are composed from other resources. Policy is embedded in the various resource types, specified by the operators of the resource pool, or by users as part of the requests for resources, and restricts the composition choices used when composing higher-level resources from the component resources. Unlike traditional policy systems, our policy model does not couple actions to constraints, and actions (workflows) needed for realizing the specification of the higher-level resource are automatically generated. By guiding the composition using policy, our model offers the following advantages over other methods of resource composition:

Component specification is easier than traditional approaches. Policy can be specified in a distributed and hierarchical manner by specifying the behavior of individual entities in the system. The designer only needs to specify constraints (as policy) that relate locally to the component(s) of interest, without worrying about global conflicts during composition. All policies that are

[1] Work performed while author was at HP Laboratories.

relevant are automatically combined to ensure that the system conforms to the relevant constraints.

The system designer no longer has to be concerned with *how* a given composition can be realized. Because configuration workflows are also generated as part of the specification, the designer only needs to specify the configuration actions available on individual entities in the system.

Updating components becomes easy. If a particular constraint on a component is modified, a new set of attribute values are computed by the policy system that would satisfy all policy constraints in the system, as well as the new configuration workflow that is needed to realize the system.

Adding or updating new entities or components is simplified. Since policy may be attached to any entity, new (or updated) entity instances and entity types can be introduced freely with their associated policies. These new policy instances are automatically considered in the policy management system when the new or updated entities are used.

In the next section of the paper we describe the policy-based model, which is followed by a section on application of the policy model for automating workflow creation.

2. Policy-based Resource Composition

When resources are combined to form other higher-level resources, a variety of rules need to be followed. For example, when operating systems are loaded on a host, it is necessary to validate that the processor architecture assumed in the operating system is indeed the architecture on the host. To ensure correct behavior of a reasonably complex application, several thousand such rules may be necessary if the construction of such applications is to be automated. This is further complicated by the fact that a large fraction of these rules are not inherent to the resources, but depend on preferences (policies) provided by the system operator or indeed, by the customer as part of the request itself.

In this section, we propose a policy-based model for combining resources which allows specification of such rules in a distributed manner. By capturing the construction rules as part of the specification of resource types, and by formalizing how these rules are combined when resources are composed from other resources, we provide a very flexible policy-based model for generating configuration specifications for complex resources.

In our model, we visualize policy as the entire set of strict (enforced) constraints that restrict allowable configurations of some target entity to those that satisfy some goal. Policies are therefore formulated as constraints on system composition (as opposed to conditions that arise as a result of system operation). In our model, resources and configuration activities are considered as the target entities. Each entity is characterized by a set of attributes and values taken by those attributes. For resource entities, the attributes represent configuration or other parameters of the resource that are meaningful for resource composition. For configuration activities, attributes represent if a particular activity needs to be triggered by the deployment system, and if so, parameters that are required for that activity.

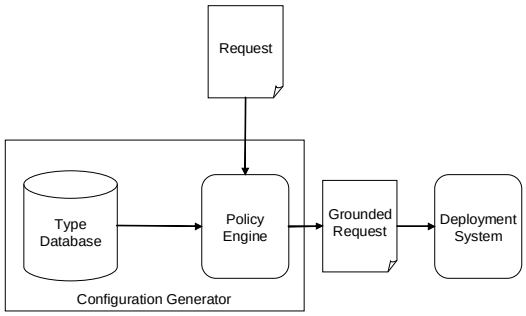

Figure 1: Resource construction process

Figure 1 shows the high level structure of the policy based configuration generator. The user creates a request (which may be minimally specific) for a composed resource. The configuration generator uses a type database and depending on the policies specified in the resource request and those associated with the resource types, generates a "grounded" request specification (i.e., a specification that is provably compliant with policy). The grounded request contains enough detail to allow a deployment system to instantiate the request. The policy engine treats the user's request and the corresponding policy constraints as a goal to be achieved. It uses a constraint satisfaction engine to select resource types and configuration activities, and assigns attribute values such that all of the policy constraints are satisfied.

Figure 2 shows the meta-model for construction policy. Construction policy is associated with both resources and activities that perform configuration operations on those resources. As part of creating the configuration specification, instances of resource types and activities are selected by the configuration generator such that the resulting model conforms to policy. The deployment system then uses that specification to initiate the appropriate activities to configure the resources to that specification.

Construction policy instances contain constraints that are defined using the attributes present in the associated resource type and activity type definitions. When a

resource request is grounded, the configuration generator ensures that all policy constraints specified for that resource are satisfied. Because resource types can be derived from other resource types, this implies that all constraints for all composing resources are also satisfied. Activity models contain attributes that describe if a particular activity needs to be triggered during deployment. They may also refer to attributes of the associated resources or other associated activities. By capturing dependencies between the activities in the policy specification, workflows or methods may be modeled as composite activities. Since the configuration generator creates the union of all (relevant) constraints when creating the resource specification, it can also accommodate a variety of operator and user level policies during grounding.

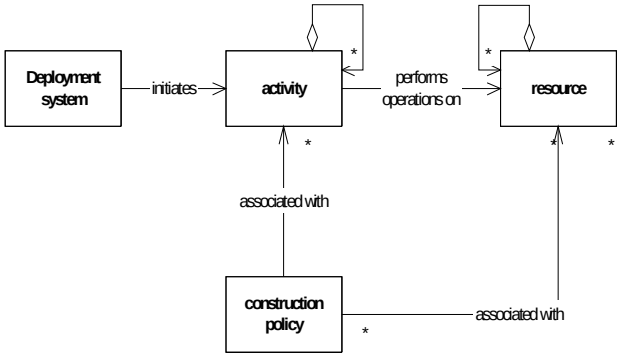

Figure 2: Relationship between policy, activities, and resources

The language that we use for describing policies is derived from the SmartFrog language [5]. Constraints contain first order predicates, arithmetic and logical operators, and other structural constructs.

5. Creating workflows for automated deployment

The deployment system has to execute a number of configuration activities to instantiate the composed resource. However, these activities cannot be executed in any arbitrary order. Just as the resources cannot be pre-composed (the composition depends on user requirements), the configuration parameters and the order of configuration cannot be pre-determined and provided to the deployment system. Depending on the exact composition, components may need to be configured differently and may need different work flows for configuration. Thus, for example, the configuration activities associated with an application server may change if the selected database server is different. Furthermore, depending on the composition, activities may need to be performed in different order. Some activities could be executed in parallel while others may need to wait for others to complete before proceeding.

Our Activity model is loosely based on the notion of task-graphs as implemented in Microsoft® Project. In our Activity model, configuration activities are modeled similar to resources, and are associated with the resources in the composition hierarchy. As the policy engine selects resources appropriate for a given request, it also selects the corresponding configuration activities. The Activity model is shown in Figure 3. An activity has set of attributes that determine when the activity will be performed. It has a duration, a startdate, enddate, and a mechanism to specify whether there is a deadline associated with the activity. It has a constraintDate and a constraintType that determines when the activity has to be executed. The constraintType could be either, As early As Possible (ASAP), As Late As Possible (ALAP), Finish No Earlier Than (FNET), Finish No Later Than (FNLT), Must Finish On (MFO), Must Start On (MSO), Start No Earlier Than (SNET), Start No Later Than (SNLT).

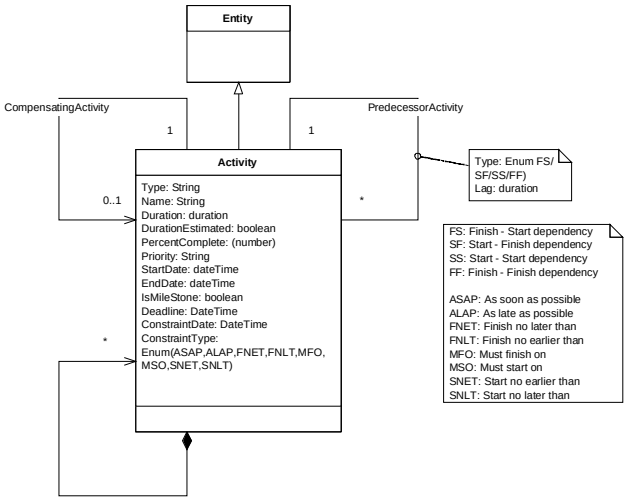

Figure 3: Activity Model

An Activity is made up of other activities. An activity may have a compensating activity and every activity may have a set of predecessor activities. Policies are associated with Activities and so the *pre-conditions* and *post-conditions* of an activity may be specified as policies. In our model, these pre-conditions and post-conditions may also be used to create a sequence of activities in a workflow. We have formalized the precedence through the type attribute defined in the association between an activity and its predecessor activities, which determines the order in which the activities are executed. These temporal planning based constructs enable creation of workflows.

- FS type means the predecessor activity is finished before the successor activity is started (sequence).
- SF means that the predecessor activity is started before the successor activity is finished
- SS means both the activities are started at the same time (parallel). Lag if present determines how much time after the predecessor activity is the successor activity started
- FF means both the activities must finish together (synchronize).

Policy constraints associated with each resource specify the associations between activities and predecessor activities of itself and its components. These predecessor activities have to follow the precedence relationships mentioned above with the successor activities. As a result of the constraint satisfaction a set of components are chosen along with a set of activities. These associations between the so chosen activities automatically establish ordering between the selected activities. A post-processor looks through all the enabled activities and using precedence relationships between the activities, creates a workflow. Figure 3 shows a part of a resource model for an e-commerce site with the associated activities. Note that where refinements exist (e.g.Oracle9i, IBMDB2 for a database) for resources, separate configuration activities may be associated with the refinements. The complete directed acyclic graph establishes the relationships between activities and their immediate predecessors.

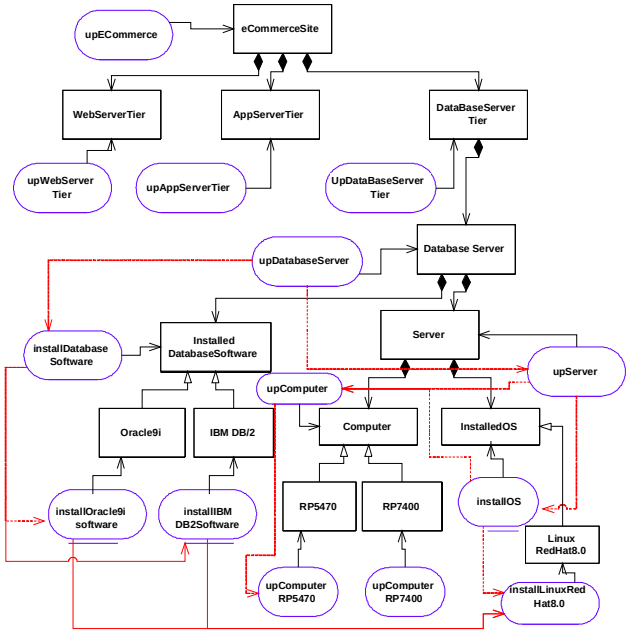

Figure 3: A resource model with associated activities

Figure 4 shows a part of the workflow generated when the model was used to instantiate an instance of an `eCommerceSite`. Note that the workflow contains both sequential and parallel activities as well as points where different activities need to be joined. Also note that it is not always possible to infer the sequence of activities from the composition hierarchy alone (e.g., the composition hierarchy does not show that the server has to be up before database software can be installed on it in order to instantiate the database server. Such sequences are determined from the dependencies specified as part of the activity model.

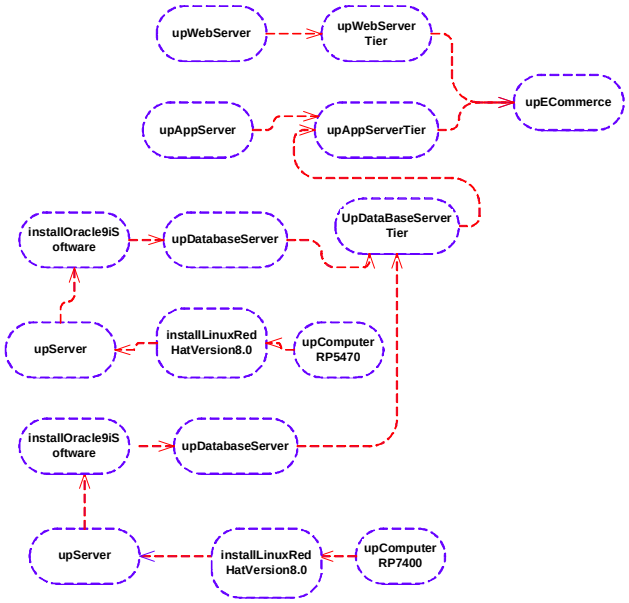

Figure 4: Part of the workflow generated when the e-commerce site is composed

Conclusion

In this article we have discussed how automated resource configuration may be undertaken using constraint satisfaction approach. The first-order logic and linear arithmetic based policy expressions are used to create system and workflow specifications of a system to be deployed.

References

[1] HP Utility Data Center (UDC) http://www.hp.com/enterprise
[2] IBM Autonomic Computing http://www.ibm.com/autonomic
[3] Nicodemos Damianou, Narankar Dulay, Emil Lupu, Morris Sloman: The Ponder Policy Specification Language. POLICY 2001: 18-38
[4] Sahai A, Singhal S, Joshi R, Machiraju V. Automated Policy-Based Resource Construction in Utility Computing Environments. In the proceedings of IEEE/IFIP NOMS 2004.
[5] SmartFrog http://www.smartfrog.org