Using Service Models for Management of Internet Services

Deborah Caswell, Srinivas Ramanathan
Internet Systems and Applications Laboratory
HP Laboratories Palo Alto
HPL-1999-43
March, 1999

E-mail: [caswell,srinivas]@hpl.hp.com

service management,
Internet services,
service models,
root-cause diagnosis

As Internet services grow in complexity, Internet Service Providers (ISPs) are finding out that ad-hoc methods that they have employed thus far to monitor and diagnose their services are not sufficient to provide acceptable service quality to their subscribers. In this paper, we demonstrate how service models can be used by ISPs to effectively manage their service offerings. A service model encapsulates a human expert's knowledge of a service, its components, and its inter-dependencies with other services. In addition, using on-going measurements, a service model tracks the health of the different services and their components. By traversing a service model top-down, an operator can not only assess the overall health of a service, but also easily correlate the health of all the services and service components to determine the root-cause of any problems that may occur. By minimizing the time and effort needed to diagnose problems, service models enable ISP operators to efficiently resolve problems that occur in an ISP environment.

Since each ISP system is unique in many respects, unique service models have to be crafted for each of the services in every ISP system. Handcrafting customized service models requires an enormous effort and time on the part of a human expert, a luxury that few ISPs can afford. In this paper, we describe a methodology for constructing customized service models for a target ISP system with minimal human intervention. This methodology relies on a service model creation engine that composes a custom service model for an ISP system using a pre-defined service model template specification and automatically discovered information about the target ISP system. We describe a prototype implementation of this methodology and present an example of a service model obtained from a real-world ISP system. Although described in the context of ISP systems, the concepts described in this paper are applicable for the management of networks and services in enterprise systems as well.

Internal Accession Date Only

1 Motivation

Increased competition and lower revenues have meant that Internet Service Providers (ISPs) are often under-staffed, and the few experts who possess all the domain knowledge necessary to rapidly diagnose and fix problems rapidly are often overworked. As a result, these experts are often unavailable to deal with many of the problems that occur. Problem resolution is often left to relatively inexperienced operators who resort to ad-hoc methods for diagnosing problems. Consequently, ISP subscribers have often experienced poor service with frequent outages [1].

The emergence of new, more complex Internet services is exposing the inadequacies of the unscientific approaches that ISPs have adopted for managing their service offerings. From being simple client-server applications, Internet services are expanding to include complex inter-dependencies on a core set of infrastructure services such as the Domain Name Service (DNS), the Network File Service (NFS), and an authentication service. A problem in one of the service components may ripple through and affect the performance of another service. Correlation across services is essential to be able to accurately diagnose problems and to determine their root-causes [2].

In this paper, we demonstrate how a service model that encapsulates a human expert's knowledge of a service, its components, and its inter-dependencies with other services can be used by ISPs to effectively manage their service offerings. The service model is structured as a graph in which nodes represents the different service components, and the state of a node reflects the health of its corresponding service component. Edges connecting nodes in the service model represent dependencies between the respective service components that the nodes represent. By traversing a service model top-down, an operator can not only assess the overall health of a service, but also easily correlate the health of all the services and service components to determine the root-cause of any problems that may occur. Consequently, the service model paradigm represents a way of equipping lower skilled operators to effectively handle problems that occur in an ISP environment. For the experts too, service models can provide a way of minimizing the time and effort that they need to expend diagnosing problems.

Since each ISP system is unique in many respects (e.g., configurations of servers, types of application servers, service offerings, organizational structure, inter-service dependencies, etc.), unique service models have to be crafted for each of the services in every ISP system. However, handcrafting service models that are customized to a target ISP system requires an enormous effort and time on the part of a human expert, a luxury that few ISPs can afford. Consequently, one of the primary challenges in the pervasive use of service models is the ability of a management system to be able to define custom service models for a target ISP system with minimal human intervention.

In this paper, we describe a methodology for automatically constructing customized service models. This methodology relies on a service model creation engine that uses a pre-defined service model template specification together with automatically discovered information about the target ISP system to compose a custom service model for the ISP system. We describe a prototype implementation of this methodology and present an example of a service model obtained from a real-world ISP system. Auto-discovery techniques that are key elements of the automatic service model construction methodology are discussed in detail elsewhere [3].

The rest of this paper is organized as follows: To set the context, Section 2 presents an overview of a typical dial-up ISP system. Section 3 introduces the service model concept. Section 4 describes the methodology for automatic construction of service models. Section 5 describes a prototype implementation of this methodology. Finally, Section 6 summarizes the contributions of this work and enumerates potential avenues for future research.

2 ISP System Architecture

Figure 1 depicts a typical ISP system that supports modem dial-in access to residential subscribers and offers web hosting services for business customers. There are two key components of this system: the Points of Presence (POP) sites and a server farm. These two components are elaborated next.

- ***Points of Presence (POP):*** In order to allow subscribers to connect to the Internet, an ISP establishes strategically located sites that house equipment to which subscribers connect through the telephone network using dial-in modems. These locations are referred to as Points of Presence (POPs). The key components of a POP are modem banks that are collections of modems used to handle incoming subscriber calls. Each modem bank is associated with a terminal server, which is a device that handles communication of packets from the POP site to other locations in the ISP system. A subscriber dialing to a POP connects to one of the modem lines in the POP. The terminal server associated with this modem line first performs subscriber authentication and then assigns an IP address to the subscriber PC.

- ***ISP Server Farm:*** A server farm includes servers that support applications such as Web, Email (Pop3 [4] and SMTP [5]), and News that subscribers access. Web sites that the ISP hosts for business customers are also located in the server farm. Infrastructure services such as authentication during login [6], domain name service (DNS) [7], content storage via the network file system (NFS) etc., are also offered via servers at this site. To ensure the scalability of their services, ISPs often replicate Web, Email, and News content across a number of servers. A common technique for balancing load among these servers uses the round-robin scheduling capabilities offered by the DNS service [7]. In this technique, replicated servers that support the same service (Web, Email, or News) are grouped together and assigned a single domain name in the DNS database. A DNS server that receives a request for this domain name returns the IP address of one of the servers in the group corresponding to the domain name. A round-robin scheme is used by the DNS server to choose the IP address that it must return.

  Since subscribers typically access services using domain names rather than IP addresses, the ISP's use of a group of replicated servers is often invisible to the subscribers. In this paper, we refer to the service provided by a group of servers functioning cooperatively as a *service group*.

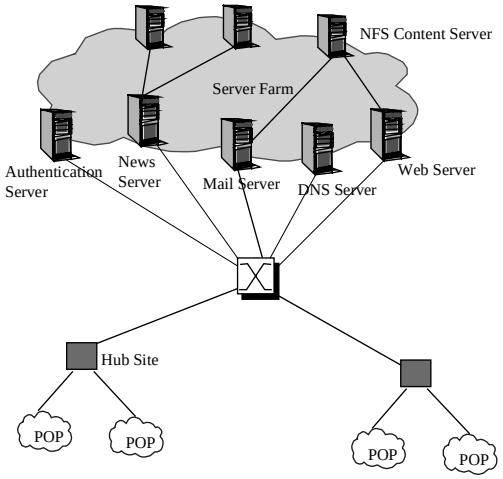

Figure 1: A typical dial-up ISP system

3 Service Models

3.1 Motivation

To meet subscriber expectations, ISPs have to measure and manage the quality of their service offerings. There are several products and services available today that ISPs can use for measuring the quality of their services. For instance, the Internet measurement service from Inverse Networks emulates a typical subscriber accessing the ISP system from several locations (POP sites) and reports metrics such as dial-up availability, Email delivery times, Web access times etc. [8]. Another service from Keynote Systems measures the performance of an ISP's web sites from different locations on the Internet [9]. ISPs can also use software applications such as *WhatsUp* [10] and NOCOL [11] to monitor the availability of their services. Using these products and services, ISPs have access to a wealth of data about the availability and performance of their services.

Although no doubt important, service-level measurements alone are insufficient for effectively managing an ISP system. Our prior work [2] has demonstrated the value of monitoring various infrastructure services (such as DNS, NFS, authentication services, etc.), the network elements (terminal servers, routers, etc.), and the servers in an ISP system, and of correlating these measurements with the service-level measurements to diagnose problems. However, manual correlation takes a lot of time and effort. More importantly, it requires a great deal of expertise and experience to draw meaningful conclusions from the correlation of the results of different, heterogeneous measurements. Service models simplify the task of correlation of measurements to deduce the root-causes of problems. A service model encapsulates the knowledge of a human expert – by reflecting the dependencies that may exist among services, and among service components (servers, hosts, networks, etc.), a service model represents the structure of a service. By traversing the service model and using simple logic to interpret the state of the different services and service components represented in the model, a human operator can easily identify the root-cause of a problem.

3.2 Definition

A service model has three main components:

- *Service topology representation:* In order to facilitate problem diagnosis, the service model encodes the topology of a service as a directed graph. Nodes in the service model graph represent components that are involved in providing the service (e.g., a host machine, an application server, network links, etc.). Alternatively, the nodes may be other services that affect the state of the service under consideration (e.g., the NFS service that provides access to a subscriber's mailbox may affect the *Read Mail* service which refers to a subscriber attempting to access his/her mailbox from the ISP's mail servers). Since ISP personnel are often assigned individual domains of responsibility (e.g., a mail expert is responsible for the ISP's mail servers alone; he/she does not deal with the network links connecting subscribers from the POP sites to the server farm, even though the status of the network links impacts the mail service as perceived by subscribers), the service model nodes can also be used to group services and service components that fall under the same domain of control. Edges between nodes in the service model indicate dependencies between the nodes that they connect. There are different forms of dependencies that are captured by the edges of a service model:
    - *Execution dependency:* The performance of an application server process executing on a host machine depends on the status of the host.

- *Link dependency:* The performance of a service offered over a network link depends the status of the link.
- *Component dependency:* An example of this dependency occurs in the case of a web service that is provided collectively by multiple "front-end servers" (FESs). While a single domain name is associated with the web service, round-robin DNS scheduling is used to map subscriber requests one of the FESs. In this case, the status of the web service depends on the status of the service provided by the individual FESs; i.e., the web service has component dependencies on the service provided by the different FESs.
- *Inter-service dependency:* As the name indicates, dependencies of this type occur among services. For instance, the mail service of an ISP depends on an authentication service for verifying a subscriber's identity and on an NFS service to store the subscriber's mailbox.
- *Organizational dependency:* Services and/or servers may be mapped to different domains of responsibility. A dependency of this type is an organizational dependency.

Since a single host machine can support multiple services (e.g., Web and Email), the same host node can appear in service models that represent different services. The collection of service models for all services provided by an ISP is therefore a directed acyclic graph (rather than a tree).

- *Measurement topology representation:* A service model also maps the different measurements being made in an ISP system to nodes in the service model graph. The node to which a measurement maps depends on the semantics of the measurement (i.e., which logical node(s) are targeted by the measurement) and on the location(s) from which the measurement is made. In the simplest case, each measurement directly indicates the status of one of the nodes in the service model. Otherwise, composite measurements, which are combinations of measurements, have to be defined and mapped to the different nodes in the service model.
- *State representation:* Each of the nodes in the service model has an instantaneous state associated with it. The state of a node reflects the health of the service or service component that it represents. A number of policies can be used to compute the state of service model nodes: for instance, one policy computes the state of a node based on all of its measurements. Another policy assigns weights to the different measurements based on an estimate of their reliability (gauged by a domain expert). The states of the measurements associated with a node are determined by applying baselining and thresholding techniques to the measurement results [12].

The use of dependency graphs to facilitate problem diagnosis is not new. For instance, Hellerstein proposes a measurement navigation graph (MNG) in which measurements are represented as nodes and the relationships between the measurements are indicated by directed arcs [13]. The relationships among measurements are used to diagnose problems. A key difference between an MNG and a service model is that the MNG only represents relationships among measurements. An ISP operator still has to understand the details of the measurements (when, where, and how each measurement is performed) and their relationships to the different service components (e.g., if the TCP connection time to a web server is high, which service components are faulty – could it be the web application server? the host machine? the network? the DNS service?). The service model goes one step further in relating the measurements to services and service components. Since state is maintained for each of the services and service components, an ISP operator can easily determine the faulty components in the event of a problem. Since it is defined based on how a service is configured, a service model is easier and more natural to construct than a measurement navigation graph. Another key

difference is the structuring of a service model to account for the organizational structure of an ISP, so that ISP operations personnel only view the states of services that they are responsible for.

Another form of dependency graph is the bayesian network model that has been extensively used for medical diagnosis for several years and has more recently been used for fault management in the networking domain [14]. Given a set of observed symptoms, a bayesian network model attempts to determine the possible cause(s) of the observed symptoms using a dependency graph. Nodes in the dependency graph represent symptoms and possible causes, and arrows represent steps that may be taken in order to determine the cause(s) of an observed symptom. Many prior efforts have focussed on heuristics for computing the minimum number of steps that must be taken in a bayesian network model to determine the potential cause of a problem. The utility of the bayesian network model in resolving problems is critically dependent on the computation of conditional probabilities that are assigned to edges in the graph, which link symptoms with causes. This information can be obtained only after significant analysis of and experience with a problem domain. Hence, much of the published literature in this area requires that the bayesian network model of a service or network be generated manually. Not only is constructing a reliable bayesian network model for a problem domain an extremely tedious task, it is also more natural to construct a service model that describes the topology of a service than to construct a bayesian network model that relates observed symptoms to possible causes.

Next, by means of an example, we illustrate how a service model can be constructed for an ISP service and how it can be used for problem diagnosis.

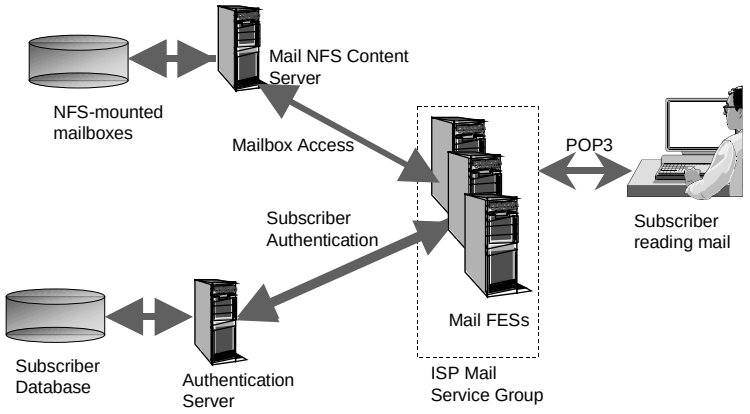

Figure 2: Topology of the Read Mail service of an ISP

3.3 A Service Model Example

3.3.1 Service Topology

In this section, we consider one of the services offered by an ISP's Email system – the *Read Mail* service, which refers to a subscriber accessing his/her mailbox from the ISP's mail system. Figure 2 depicts a service topology for this service. Using a browser that supports the Post Office Protocol - Version 3 (Pop3) [4], a subscriber accesses his/her mail. Internal to the ISP system, the subscriber request is received and processed by one of many mail servers (referred to as front-end servers in the figure) that constitute a mail service group. Before the subscriber can access his/her mailbox, the mail server that handles the request contacts an authentication server to verify the subscriber's identity.

Following authentication, the mail server retrieves the subscriber's mailbox from a back-end content server using the NFS service and transmits the mail messages back to the subscriber terminal.

3.3.2 Measurements

There are several active and passive measurements that can be made to assess the health of the different components involved in supporting the Read Mail service. While active measurements generate artificial workloads to emulate typical subscriber requests, passive measurements typically rely on instrumentation built into the service components to track their status. Suppose that a measurement station (MS) is installed in the ISP server farm and that measurements are made using agents executing on the MS and on the different ISP hosts. From [2], the different measurements that characterize the Read Mail service are:

- Read Mail service quality measurement: This active measurement made from the MS in the server farm emulates a subscriber accessing a mailbox and measures the total response time for retrieving a fixed set of messages. In addition, this measurement also breaks down the total response time into sub-components such as the time taken for TCP connection establishment[1], the authentication time, etc.
- Network performance measurement: This active measurement assesses the throughput that is achievable over a TCP connection operating over the network links connecting the MS in the server farm to the POP sites.
- Host status measurements: Passive measurements of CPU and memory utilization, as well as TCP connection and packet traffic to the mail servers, obtained from agents executing on the mail servers provide information about the status of the mail server hosts.
- NFS service measurements: Passive measurements of NFS statistics (number of calls, timeouts, etc.) obtained from the mail servers and NFS servers are used to assess the status of the NFS service on which the Read Mail service depends.

While the active measurements attempt to assess the service quality as viewed from subscribers connecting to the POP sites, the passive measurements provide resource utilization and traffic statistics that are critical for problem diagnosis.

3.3.3 Service Model Construction

Figure 3 depicts the service model for the Read Mail service supported by an ISP system that conforms to the service topology and measurements considered above. While the nodes in the service model graph represent the different services and service components, the arrows feeding the nodes represent measurements. The root of the service model graph (node N_1 in Figure 3) represents the overall health of the Read Mail service as assessed by a measurement station located in the ISP's server farm (i.e., without considering the state of the network links from the server farm to the POP sites). There is no direct measure of the overall health of the Read Mail service. Instead, the state of the service is inferred based on the states of the different mail front-end servers - FESs (node N_2 in Figure 3) that together enable the Read Mail service. Direct active measures of availability and performance, and passive measurements of TCP statistics to the Pop3 port (110) contribute to the status of the Read Mail service for each of the FESs.

[1] The Pop3 protocol uses the Transmission Control Protocol (TCP) as the transport protocol.

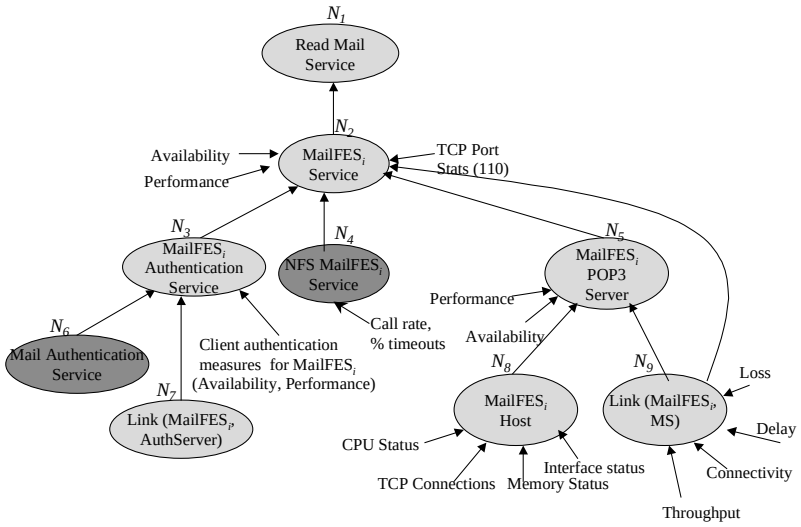

Figure 3: Service model of the Read Mail service of an ISP

The next level of the service model reflects the dependencies of the service provided by a mail FES on the Pop3 server executing on the mail FES, the authentication and the NFS services used by the mail FES. Considering the Pop3 server first (node N_5 in Figure 3), the health of the server is measured based on the ability to establish a TCP connection from the measurement station to the mail FES as part of the active Read Mail service quality measurement and the time taken to establish the connection. In turn, the health of the Pop3 server depends on the health of the mail FES host (node N_8 in Figure 3). Since the availability and performance of the Pop3 server are measured by an active measurement from the measurement station, the results of these measurements can be biased by the state of the network link connecting the measurement station and the mail FES. This measurement dependency is reflected in Figure 3 by link between nodes N_5 and N_9.

Focusing next on the authentication service dependency, we note that it is possible that the authentication service itself (node N_6 in Figure 3) may be healthy but a specific mail FES may be failing authentication (e.g., because the network link between the mail FES and the authentication server is down). Consequently, it is necessary to assess the states of the authentication service from the point of view of each of the mail FESs (node N_3 in Figure 3). The state of the mail authentication service as observed by a mail FES is determined based on the Read Mail service quality measurement for this FES, which tracks the authentication delay as one of the components of the overall response time for the service. In turn, the mail authentication service for a FES depends on the state of the central mail authentication service (node N_6 in Figure 3). The service model for this service is not expanded in Figure 3. The NFS service dependency is handled in the same way as the authentication service dependency. The darker shading on the NFS service node in Figure 3 indicates that the NFS service model is not presented here in detail.

As alluded to earlier, Figure 3 represents the service model for the Read Mail service in isolation. To represent the end-to-end service, a service model must take into account the state of the DNS service used by subscribers to resolve the ISP mail domain to one of the mail FESs, and the state of the network links between the different POP sites and the ISP server farm. Clearly, since the different POP sites use different routes to connect to the server farm, a subscriber's

perception of the end-to-end service may vary depending on the POP site that the subscriber is using to connect to the ISP system. Figure 4 depicts the service model for the end-to-end Read Mail service.

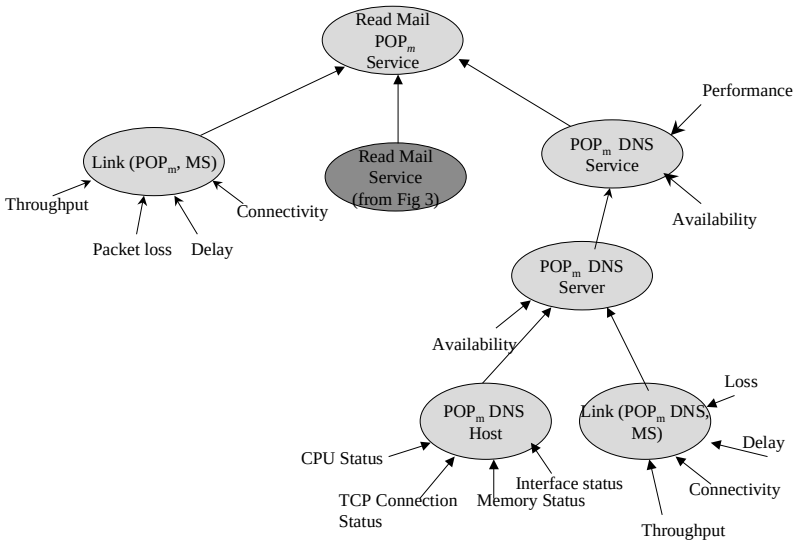

Figure 4: Service model for the Read Mail service as perceived by a subscriber accessing the ISP via POP_m

3.4 Uses

There are several ISP management functions that can exploit the capabilities of service models. For operational monitoring, a service model indicates the status of the different components providing a service. When a failure occurs, the service model indicates which components have been affected by the failure. Moreover, by traversing the service model graph top-down, an operator can determine the root-cause of the failure. For example, in Figure 3, when a problem is noticed with the overall Read Mail service, an operator can traverse the service model graph to determine first whether the problem is caused by a specific mail FES or whether all of the mail FESs are experiencing a problem. Assuming the former scenario, moving down the service model graph, the operator can further determine whether the problem is related to authentication failure, NFS failure, or a failure of the Pop3 application server. Continuing this same process, the operator can determine the cause of a problem.

The services and service component nodes in the service model can be organized based on domains of responsibility, so that ISP operations personnel need only monitor and diagnose the services that fall in their domain of responsibility. In the Read Mail service example in the previous section, an Email operations expert who is responsible for the mail service and mail servers uses the service model depicted in Figure 3, since the expert is mainly interested in the state of the Email services and servers alone. The authentication and NFS services are included in the service model representation, since these services can adversely impact the Read Mail service. In contrast, the links between the server farm and the POP sites are not included in the model since they do not affect the Read Mail service from an Email expert's perspective.

The customer support function, which relates to the handling of trouble calls from subscribers, also benefits from the existence of service models. Taking the subscriber's perspective, customer support personnel require an end-to-end view of the service (e.g., Figure 4). Furthermore, unlike operations personnel, the customer support personnel are only

interested in knowing the services that are affected by a failure, not in knowing what the root-cause of the failure is. Consequently, only the service model nodes that reflect the health of subscriber-visible services (some services such as DNS and NFS are not subscriber-visible) are of interest to customer support. Instantaneous information about the health of a service enables customer support personnel to efficiently filter incoming subscriber trouble calls.

4 Automatic Construction of Custom Service Models

4.1 Motivation

Since individual services offered by an ISP have different dependencies, they have different service topologies. Consequently, each ISP service has a unique service model. Even when different ISP systems support the same service, there are several considerations that must be taken into account while constructing service models:

- *Uniqueness in configuration:* Each ISP system is unique in its configuration. The number of servers used may vary depending on an ISP's subscriber-base. Different systems may use different schemes for load-balancing among servers. Different operating platforms may be in use (Unix, Windows NT) influencing the type of services deployed (e.g., DNS vs. WINS). Even when the same service is supported by ISP systems that use the same number and type of servers, the inter-service dependencies may vary. For instance, in the example of Figure 3, one ISP system may choose to use the same authentication server for all of its mail FESs, while another ISP system may choose to use different authentication servers for each of the mail FESs. The service topology in these two cases is different.
- *Differences in organizational structure:* The delegation of responsibilities to operations personnel can be different from one ISP to another. An ISP's organization structure may be dependent on several factors including the geographic locations of the ISP server farms, the skill-set of the operations personnel, the range of services offered by the ISP, the scale of the ISP system, etc., all of which are specific to each ISP. Since the service model captures the organizational structure of ISP operations, the service model will vary from one ISP to another.
- *Scale of operation:* Since large ISP systems have hundreds of POP sites and application servers (for the hundreds of thousands of subscribers), it is impractical to expect the service model to be entirely defined manually.
- *System complexity:* Defining a service model that captures all of the existing dependencies requires a domain expert with intricate knowledge of the ISP system. The number and types of dependencies that need to be captured in defining a service model are large enough to make the process of defining a service model time-consuming.
- *System evolution:* To deal with growth and increasing subscriber demands, ISP systems are constantly being reconfigured. Because of the effort involved and time required to redefine the service models, it is unreasonable to expect that the service models will be manually updated after every reconfiguration.

While the uniqueness of an ISP's system configuration and organization structure motivates the need for generating custom service models for each system, the scale, system complexity, and evolutionary considerations motivate the need for being able to automatically generate service models with minimal human intervention. The next section describes a template-driven approach for automatically generating custom service models.

4.2 Template-Driven Approach

4.2.1 Phases of a Service Model

The fundamental idea behind this approach is to decompose the lifetime of a service model into three distinct phases:

- *Service model template:* The first phase in constructing a service model is the specification of a template for the service model. A service model template is a generic specification of the service topology and measurement topology for a service. Depending on the service being modeled and the service components that are likely to be involved, the template defines nodes of various types (hosts, servers, network links, services, etc.), their associated measurements, and indicates the dependencies among the nodes. Default state computation rules are specified for each node in the template. The template is not specific to any ISP system in the sense that it does not specifically reference any hosts, servers, network links, or other services and service components in the ISP system.

- *Service model instance:* The service model instance maps services and service components such as hosts, servers, network links, etc., that exist in an ISP system with nodes in the service model template specification. In doing so, the service model instance also specifies a set of measurements that must be made against each of the services and service components in the ISP system. The inventory of services and service components in an ISP system may either be auto-discovered or may be manually specified for creating the service model instance. To minimize the amount of customization that is necessary to establish a service model instance for an ISP system, the default state computation rule to be used for each node in the service model instance is derived from default values specified for each node type in the service model template. Unlike a service model template, a service model instance is specific to an ISP system. We refer to the process of constructing a service model instance using a service model template specification as the *instantiation* of a service model. The relationship between service model template and instances is analogous to the relationship between a schema and records in databases.

- *Service model view:* As is evident from the discussion in Section 3, the different management functions that benefit from service models may be interested in different subsets of nodes and edges of a service model instance. For example, the operations personnel of an ISP are interested only in services and service components that fall in their domain of responsibility (Figure 3). In contrast, customer support personnel require an end-to-end view of the service (Figure 4). Even when different management functions are interested in the same subset of nodes and edges of a service model instance, they may require that different rules be used for state computation.

  A service model view defines the subset of nodes of a service model instance that is of interest to a management application, and the method for evaluating the states of the nodes of interest. The measurements associated with the nodes in the service model instance are common across different views. However, the rules applied to compute the state of a node based on the states of the measurements and the states of other nodes are different for different views. As mentioned above, service model views are ideal for representing the organizational responsibilities in an ISP system. A key advantage of using views (rather than service model instances) to represent organizational responsibilities is that when reorganizations of ISP operations staff occur, only the views need to change. That is, rediscovery of the ISP system and reinstallation of measurement agents is not necessary since the service model instance is unaffected.

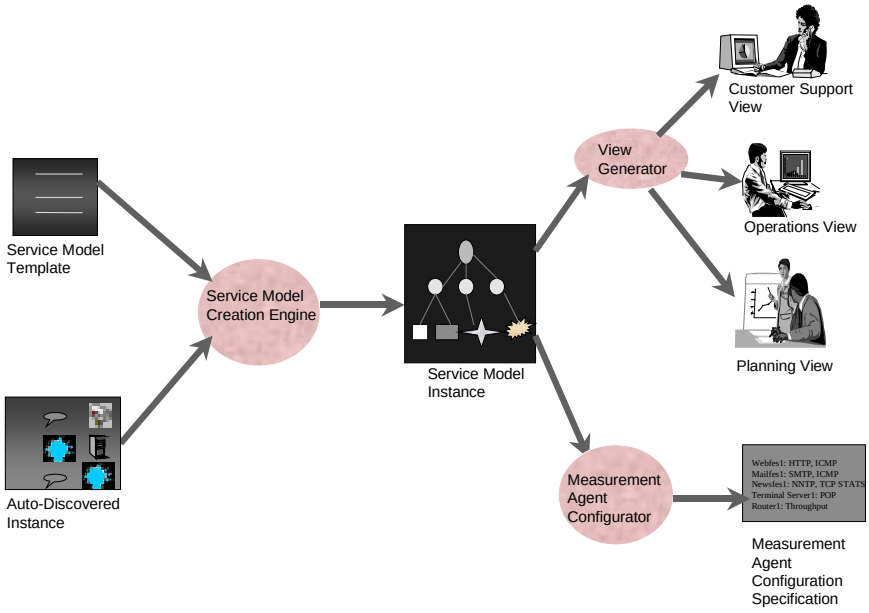

Figure 5: The creation and use of a service model instance from a service model template

4.2.2 Service Model Creation Process

Figure 5 depicts the service model creation process. The various steps in this process are:

1. *Handcraft a service model template:* For each service in the ISP system, a service model template must be specified. The template has to be handcrafted, typically by a domain expert.

2. *Obtain discovered information:* Information about the services and service components that exist in an ISP system may be auto-discovered. We refer to the store representing auto-discovered information as the *discovered instance*. In [3], we discuss auto-discovery techniques that can discover the services and service components in an ISP system, as well as the inter-dependencies among them. [3] also discusses the notion of using a discovery template to drive the discovery process, thereby making discovery extensible to support new service and service component types that may be added to an ISP system over time.

3. *Instantiate the service model:* A service model creation engine is used to create a service model instance based on the service model template specified in step 1 and auto-discovered instance generated in step 2. As we shall see later, the service model template itself encodes instructions that the service model creation engine uses to map sections of the service model template with appropriate sections of the discovered instance, so as to generate a service model instance. In the ideal case, all the discovered information is available prior to instantiation. In many cases however, discovery has to be performed in multiple phases [3]. In such cases, instantiation may happen partially after each phase of discovery. The main advantage of doing so is that the partially instantiated service model can serve to guide what information needs to be discovered in the subsequent phases of discovery. Considering just the first phase of discovery, in Section 5.4**,** we present a detailed design for a service model creation engine. Enhancing this design to handle multi-phased discovery is a topic for future research.

4. *Construct measurement agent configuration:* The service model template (step 1) associates different types of measurements with specific types of service model nodes. The service model instance generated in step 3 indicates which measurements need to be made on or against each of the service components and services in the ISP system. A measurement agent configuration engine parses the service model instance specification and determines the measurements that must be made for each host. The mapping of measurements to hosts thus obtained can be used to install and operate measurement agents on the different hosts in the ISP system.

5. *Configure different views of the service model:* A view specification language (not discussed in this paper) can be used to compose different service model views from the service model instance.

At each step of the above process, a user can customize the service model creation.

5 Prototype Implementation

In this section, we describe our prototype implementation of the service model construction approach described in Section 4. We present example specifications for a service model template, an auto-discovered instance, and a service model instance, and a detailed design for the service model creation engine. In the process, we highlight design issues that come up when translating the service model construction approach of Section 4 into a practical implementation. The syntax used for the specifications in the following sections is only an example. Other specification and schema definition technologies such as Common Information Model (CIM) being proposed in the Desktop Management Task Force [15] can also be used to implement the concepts described in the following sections.

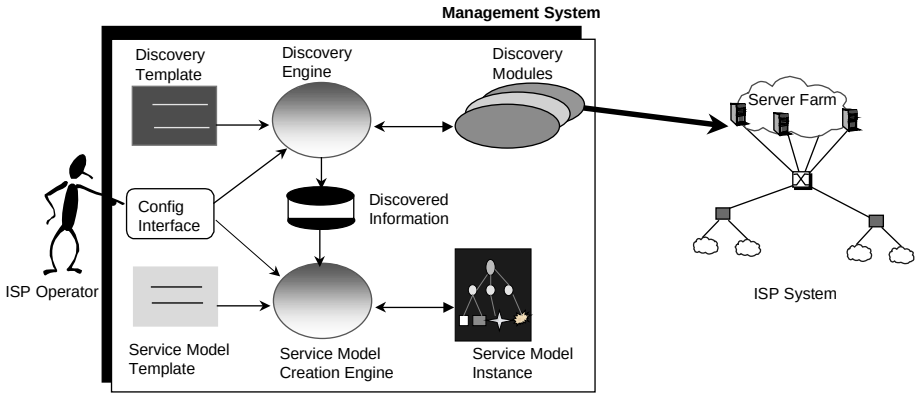

Figure 6: Architecture of the prototype system

5.1 Overview

Figure 6 presents an overview of our prototype system. Our prototype has two main components - an auto-discovery engine and a service model creation engine. While the auto-discovery engine is responsible for generating a discovered instance by discovering the services and service components and their inter-dependencies in a target ISP system, the service model creation engine is responsible for generating a service model instance using the discovered instance and a service model template. The auto-discovery process is implemented in an extensible manner. In this approach, the types of services and service components to be discovered are specified in a discovery template. The discovery template

includes specifications of discovery modules to be invoked to discover different services and service components. The discovery engine processes the discovery template, invokes the appropriate discovery modules, interprets the output of these modules, and stores the output as a discovered instance that is made accessible to the service model creation engine.

The discovery engine supports a configuration interface, using which ISP operations staff can control and customize the discovery process. Through the configuration interface, an ISP operator can restrict the discovery to specific IP address ranges or host name patterns (see Figure 7). The configuration interface also serves as a way for an operator to specify naming conventions that the ISP uses (e.g., for naming terminal servers and POP sites). In addition, the operator can specify a categorization of services and service components based on the ISP's organizational structure, geographical location of the servers, differences in business functions of the services (e.g., web servers that an ISP hosts on behalf of the business customers vs. local web servers that the ISP offers for providing access to the dial-up customers), etc.

In our prototype system, the discovery engine itself distributes categorization information to the discovery modules at the time of their invocation. The discovery modules restrict the discovery they perform to abide by these specifications. Furthermore, the discovery modules use the category specifications to determine the categories to which different services and service components belong and record this information as part of the discovered instance. As we shall see later, the service model creation engine exploits this information to generate a customized service model instance.

```
Hosts               exclude   ip        10.1.204.1-10.1.205.1
# exclude all hosts with IP addresses in this range.  These hosts represent subscriber home PCs
WebServers          category  name      .*.com.fakeisp.net  WebHost
# servers with names of the form *.com.fakeisp.net are Web hosting servers
TerminalServers     extract   name      max[0-9]{3}<PopSite>[0-9]t.fakeisp.net
# Terminal servers must match the naming pattern above. Extract the POP site name from the terminal server's name
```

Figure 7: An example configuration specification

Next we delve into the details of the prototype, starting with example specifications of the discovery instance and the service model template. Since it is not critical to the discussion of the service model instance construction, the discovery template specification is not discussed in detail here.

All the templates and instances in our prototype are specified using the INI file format that is commonly used to represent the content of system files in Microsoft Windows-based personal computer systems. As per the INI format, a template or an instance is organized as different sections, each of which is specified by a unique string enclosed within a pair of square brackets ("[<*section name*>]"). Within each section, a number of variable-value pairs are specified.

5.2 Discovered Instance Specification

Figure 8 depicts a discovered instance of an ISP system. Although this discovered instance was automatically generated by the discovery engine of our prototype, in general, the discovered instance can also be manually specified. The target ISP system in this example has six hosts, one of which supports an SMTP mail server. Two of the hosts support Pop3 mail servers, three others support Web servers, and one host supports an NFS server. The Pop3 mail servers are configured to form a service group.

The discovered instance is organized as sections, each section representing a service or a service component. Variable-value pairs in each section indicate various attributes of the corresponding service or service component. For example,

in Figure 8, there is a section for each of the six hosts in the ISP system. The attributes of a host (say 10.174.173.23) indicate its *ipAddress*, *hostName*, *state*, and *category*.

Following the categorization specified in Figure 7 that all web servers whose names match the pattern **.com.fakeisp.net* are servers that host web sites for commercial customers of the ISP, the discovered instance indicates that there are two hosted web sites, namely, *www.xyz.com.fakeisp.net* and *www.abc.com.fakeisp.net.*

```
; the list of hosts in the ISP system
[10.137.196.52:Host]
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net:smtp.fakeisp.net
; this host has two names by which it is referred
state=Alive
; the host is alive (i.e., responding at the network level)
category=
; the configuration specification does not specify any categorization for hosts

… Similar specification for mailfes22.fakeisp.net(10.137.196.54) www.fakeisp.net (10.137.196.58), www.xyz.com.fakeisp.net(10.137.196.69),
www.abc.com.fakeisp.net(10.137.196.70)

[10.174.173.23:Host]
ipAddress=10.174.173.23
hostName=nfs.fakeisp.net:dns1.fakeisp.net
; this host is referred to by two names
state=Alive
category=

; SMTP mail servers in the ISP system
[10.137.196.52:Smtp_Mail]
ipAddress=10.137.196.52
; ipAddress of the host on which the SMTP server is executing
hostName=mailfes21.fakeisp.net:smtp.fakeisp.net
category=

; Pop3 mail servers in the ISP system
[10.137.196.52:Pop3_Mail]
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
; relatedSmtpServer refers to the SMTP server that must be contacted to send mail to this Pop3 server
category=
[10.137.196.54:Pop3_Mail]
ipAddress=10.137.196.54
hostName=mailfes22.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=

; Web servers in the ISP system
[10.137.196.58:Web]
ipAddress=10.137.196.58
hostName=www.fakeisp.net
serverType=NCSA/1.5
category=InternalWeb
; based on the configuration specification, the discovery module has determined this web server to be an internal server
[10.137.196.69:Web]
ipAddress=10.137.196.69
hostName=www.xyz.com.fakeisp.net
serverType=Apache/1.2
category=WebHost
; based on the categorization specification, the discovery module has determined that this web server is a web site being hosted for xyz.com
[10.137.196.70:Web]
ipAddress=10.137.196.70
hostName=www.abc.com.fakeisp.net
serverType=Apache/1.2
category=WebHost
; based on the categorization specification, the discovery module has determined that this web server is a web site being hosted for abc.com

; NFS Server
[10.174.173.23:NFS]
ipAddress=10.174.173.23
hostName=nfs.fakeisp.net:dns.fakeisp.net
category=

; DNS Server
[10.174.172.23:DNS]
ipAddress=10.174.172.23
hostName=nfs.fakeisp.net:dns.fakeisp.net
category=

; Pop3 mail service groups - discovered by querying the DNS service
[pop3.fakeisp.net:Pop3_MailServiceGroup]
serviceGrpName=pop3.fakeisp.net
serviceGrpComponentsList=10.137.196.52:10.137.196.54
; both of the above IP addresses have Pop3 servers executing on them

; a list of categories determined previously. This information is based on the configuration specification and is used to generate a customized service model.
[InternalWebServer:Category]
categoryName=InternalWeb
[WebHostedServer:Category]
categoryName=WebHost
```

Figure 8: An example discovered instance specification

5.3 Service Model Template Specification

Figure 9 presents a representative example of a service model template specification. Each section of the template represents a type of node in the service model instance and contains a series of instructions for creating the node. The service model creation engine processes sections of the service model template one at a time, attempting to match the section with appropriate sections of the discovered instance. The different instruction types supported in our implementation of the service model template are:

- *Match*: Most sections of the service model template begin with a match criterion. The match criterion indicates how the service model creation engine should find the discovered instances that are relevant to the service model template section under consideration. For instance, the match criterion corresponding to the host node's specification (*SM-Host*) in the service model template indicates that the corresponding discovered instances are those of type *Host*. The match criterion is specified as a regular expression that is matched against section names of the discovered instance. For each section of the discovered instance that satisfies the match criteria, a corresponding node is instantiated in the service model instance.

  Each service model template section can match discovered instances of at most one type. When a discovered instance satisfies the match criterion, all of its attributes are available for reference in the subsequent instructions of the service model template section. For instance, the specification "*hostName=<hostName>*" in the *SM-Host* section of the example in Figure 9 indicates that a service model node of type *SM-Host* inherits its *hostName* attribute from the discovered instance that it matches.

  The absence of a match criterion in a section of the service model template indicates that there is only one instance of this type. For instance, there is a single root node - the *ISP Read Mail* service in Figure 9. This node has as its components all of the services and service groups pertaining to the Read Mail service.

- *InstanceKey*: Each node in the service model instance must be referred to by a unique name. In our implementation, the *InstanceKey* variable in the service model template specification determines how a service model instance node is named. The *InstanceKey* is specified in terms of attributes (which are enclosed within angled braces "< >") of the discovered instances that match the service model template section.

- *Components*: The *components* instruction specifies the parent-child relationships in a service model instance. Various types of dependencies – inter-service dependencies, component dependencies, link dependencies, etc. - are captured by this specification. The components list of a node can refer to either

  - Specific nodes: In this case, the exact name of a node (*instanceKey* in our implementation) is specified to refer to a specific node in the service model instance. For example, a node of type *SM-WebServer* has a component node of type *SM-Host*, wherein the *ipAddress* attribute of the *SM-Host* node is the same as that of the *SM-WebServer* node.

  - All nodes of a specific type: To indicate that all nodes of a specific type must appear in the components list, the service model template specification of Figure 9 uses regular expression syntax. For instance, the *SM-ISP-ReadMailService* node has as its children all nodes of type *SM-ReadMailService* (referred to as "**:SM-Read-MailService"*) or *SM-ReadMailServiceGroup* (referred to as "**:SM-ReadMailServiceGroup").*

  - All nodes of a specific type that have a specific attribute value: An example of this specification is the components specification in the *SM-Category* section in Figure 9. The components of a node of type *SM-Category* are all nodes

in the service model instance that have the category attribute set to the same value as the category name of the *SM-Category* node. As we shall see in Section 5.6, this feature can be used to automatically generate service model instances that are customized to an ISP's organizational structure, so that ISP operators only get to view the states of services that they are responsible for.

Component specifications are not provided for sections of the service model template that generate leaf nodes in the service model instance.

- *Measurements:* This instruction specifies a list of measurements that must be targeted at the corresponding node in the service model instance. By processing the measurement specifications of nodes in the service model instance, a measurement agent configuration module (see Figure 5) can determine the agents that must be scheduled for execution against each component of the discovered instance.
- *State computation rule*: This instruction governs how the states of the nodes in the service model instance are computed. By default, the state of a node is determined based on the states of all of the measurements associated with the node and the states of all of its dependencies (children nodes) in the service model graph. The service model creation engine may support additional state computation policies. For instance, a *measurementsOnly* policy indicates that only the states of the measurements associated with a node must be taken into account in determining the state of that node.
- *Attribute value settings*: Each service model node may inherit attributes from the discovered instance that it refers to. Our service model template syntax also allows for hierarchical aggregation of attributes. This is demonstrated in the *append* construct used for defining category attributes for the service model nodes (see the specification of the *category* attribute in the *SM-Web* section in Figure 9).

```
[SM-Host]
; Host Node in the service model
match=[*:Host]
; for each discovered Host
instanceKey=<ipAddress>:SM-Host
measurements=TCP-ConnectionRate(<ipAddress>),VMstat(<ipAddress>),IFstat(<ipAddress>)
; these are measurements of the host
; next inherit attributes from the discovered instance
hostName=<hostName>
; the hostName attribute of a SM node is the hostName attribute of the discovered instance that it matches
ipAddress=<ipAddress>
state=<state>
category=<category>
; state computation for this node follows the default rule

[SM-Web]
; SM node for a web server
match=[*:Web]
; match all discovered Web server instances
instanceKey=<ipAddress>:SM-Web
components=<ipAddress>:SM-Host,<ipAddress>-msIP:SM-Link
measurements=HTTP-TCPConnectionTime(<ipAddress>)
; next copy all the attributes from the server discovered instance
stateComputationRule=measurementsOnly
; the state of this node is determined based on the states of its
; direct measurements alone (i.e., the states of its dependencies do not matter)
serverType=<serverType>
hostName=<hostName>
ipAddress=<ipAddress>
category=append(<category>,(<ipAddress>:SM-Host?<category>))
; to compute the category for this node, append the category of the discovered instances that
; match this section to the category specification of the children nodes of this node in the service
; model instance

[SM-WebService]
; a web service node
match=[*:SM-Web]
; there is a web service node corresponding to each web server node
instanceKey=<ipAddress>:SM-WebService
measurements=HTTP-Availability (<ipAddress>), HTTP-TotalResponseTime(<ipAddress>), HTTP-DnsTime (<ipAddress>)
components=<ipAddress>:SM-Web, <<ipAddress>,Web>:SM-NFS, <<ipAddress>,Web>:SM-DNS
; NFS and DNS service dependencies will be determined by phase 2 discovery
category=<category>
```

```
[SM-WebServiceGroup]
match=[*:WebServiceGroup]
instanceKey=<serviceGrpName>:SM-WebServiceGroup
components=list(<serviceGrpComponentsList>):SM-WebService
category=<category>
category=append(list(<serviceGrpComponentsList>):SM-WebService?<category>)
; the category for SM-WebServiceGroup is obtained by appending the categories
; of each of its components in the service model instance

[SM-TopLevel-Web]
instanceKey=Web:SM-TopLevelWeb
components=*:SM-WebService, *:SM-WebServiceGroup
; components are all web services and all web service groups

[SM-NFS]
match=[*:NFS]
instanceKey=<ipAddress>:SM-NFS
components=<ipAddress>:SM-Host
measurements=NFS-TotalCalls(<ipAddress>), NFS-DupReqs(<ipAddress>), NFS-TimeOutPercent(<ipAddress>)
stateComputationRule=measurementsOnly
ipAddress=<ipAddress>
hostName=<hostName>

[SM-DNS]
match=[*:DNS]
instanceKey=<ipAddress>:SM-DNS
components=<ipAddress>:SM-Host, <ipAddress>-msIP:SM-Link
measurements=DNS-Availability(<ipAddress>), DNS-CacheHitResponseTime(<ipAddress>)
ipAddress=<ipAddress>
hostName=<hostName>
category=<category>

[SM-Pop3_Mail]
match=[*:Pop3_Mail]
instanceKey=<ipAddress>:SM-Pop3_Mail
components=<ipAddress>:SM-Host
measurements=POP3-TCPConnectionTime(<ipAddress>)
hostName=<hostName>
category=<category>
relatedSmtpServer=<relatedSmtpServer>
ipAddress=<ipAddress>

[SM-ReadMailService]
match=[*:SM-Pop3_Mail]
instanceKey=<ipAddress>:SM-ReadMailService
measurements=POP3-Availability (<ipAddress>), POP3-TotalResponseTime(<ipAddress>), POP3-AuthenticationTime (<ipAddress>)
stateComputationRule=default
components=<ipAddress>:SM-Pop3_Mail, <<ipAddress>,Pop3_Mail:SM-NFS>, <<ipAddress>,Pop3_Mail:SM-Auth>
category=<category>

[SM-ReadMailServiceGroup]
match=[*:Pop3MailServiceGroup]
instanceKey=<serviceGrpName>:SM-ReadMailServiceGroup
components=list(<serviceGrpComponentsList>):SM-ReadMailService
category=<category>

[SM-TopLevel-ReadMail]
instanceKey=ReadMail:SM-TopLevel-ReadMail
components=*:SM-ReadMailService, *:SM-ReadMailServiceGroup

[SM-Category]
match=[*:Category]
instanceKey=<categoryName>:SM-Category
components=*:SM-*?category=<categoryName>
```

Figure 9: An example service model template specification

While the discovered instance specification in Figure 8 can be either manually specified or automatically generated, the service model template has to be specified by a human expert based on the expert's knowledge of the topology of the ISP's services. While the initial generation of the service model template can be time-consuming, once generated, the service model template needs to be re-edited only when new services are added to the ISP system, or when changes are made to a service's topology. Furthermore, in cases when the service topologies of two ISP systems are similar, the service model template designed for one system can be re-used in the other system. Using the discovered instances for the two systems, a service model creation engine that uses the same service model template can generate two different service model instances that are specific to the two ISP systems being managed.

5.4 Service Model Creation Engine

The service model creation engine incorporates the logic for processing a service model template specification together with the discovered instance to generate a service model instance. There are different choices for the order in which a service model creation engine processes a service model template:

- *Sequential processing:* Assuming that the sections in the service model template are placed in the order in which they need to be processed, the service model creation engine processes the sections of the service model template in sequence. Sequential processing enables a simpler design of the service model creation engine. However, this approach burdens the template developer with having to manually determine the placement of a section in the service model template based on the order of processing. Moreover, since processing typically starts from the host nodes onwards, this approach may result in a number of service model host nodes that do not have parent nodes in the service model instance graph (these nodes correspond to hosts that do not support any services). The service model instance created must be processed further to remove such "orphaned" nodes.
- *Hierarchical processing:* Alternatively, a service model creation engine can use the *components* specifications in the different sections of a service model template to determine the order for processing its sections. Since the components list specifies the dependencies of a service model node, before processing the corresponding section, the service model creation engine must process all of the sections corresponding to the node types in the components list for the node. Based on this rule, sections with no components specification are processed first. Sections whose component lists have been processed earlier are considered next, and so on. Although more complex to implement than the sequential approach, the hierarchical approach does not result in any orphaned nodes in the service model instance.

Due to the simplicity of its implementation, the service model creation engine of our prototype system incorporates the sequential processing approach.

5.5 Service Model Instance Specification

Figure 10 depicts a service model instance specification for the service model template in Figure 9 and the discovered instance in Figure 8. The service model instance is also organized as sections, with each section corresponding to a node in the service model instance. The name of a section (enclosed within square brackets) indicates the unique name associated with the corresponding service model instance node. Variable-value pairs defined in a section indicate the attributes of a node, the measurements that must be made corresponding to the node, the children of the node in the service model instance, and the rule that must be used to compute the state of the node. Unlike the service model template, the service model instance pin-points specific dependencies that exist among services supported by different hosts. Moreover, the service model instance specification can also be used to identify the measurements that need to be made for each of the nodes.

The discovered instance in Figure 8 represents information that can be determined by taking an external view of the ISP system. Consequently, the discovered instance does not capture all the dependencies that exist in the ISP system. For instance, the discovered instance does not indicate the identity of the NFS service that the Pop3 mail server executing on host 10.137.196.52 depends on. Such dependencies have to be discovered using a second phase of discovery that takes an internal view of the ISP system. Since the service model instance in Figure 10 is determined based on the discovered instance of Figure 8, the dependencies of some of the nodes in the service model instance are not completely specified (these dependencies are indicated by strings enclosed within angled brackets (<>) in the *components* description in Figure 10). An additional phase of discovery and another phase of instantiation (not discussed in this paper) are necessary to complete the generated service model instance.

```
[10.137.196.52:SM-Host]
; host node in the service model
measurements=TCP-Connection-Rate(10.137.196.52), VMstat(10.137.196.52), IFstat(10.137.196.52)
; measurements that need to be made for hosts
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net
state=Alive
category=

...  similar specifications for the other hosts -  10.137.196.54, 10.137.196.58, 10.137.196.69, 10.137.196.70, and 10.174.173.23

[10.137.196.58:SM-Web]
; 10.137.196.58 is supporting a web server
components=10.137.196.58:SM-Host,10.137.196.58-msIP:SM-Link
measurements=HTTP-TCPConnectionTime(10.137.196.58)
stateComputationRule=measurementsOnly
ipAddress=10.137.196.58
hostName=www.fakeisp.net
serverType=NCSA/1.5
category=InternalWeb
[10.137.196.69:SM-Web]
components=10.137.196.69:SM-Host,10.137.196.69-msIP:SM-Link
measurements=HTTP-TCPConnectionTime(10.137.196.69)
stateComputationRule=measurementsOnly
ipAddress=10.137.196.69
hostName=www.xyz.com.fakeisp.net
serverType=Apache/1.2
category=WebHost
[10.137.196.70:SM-Web]
components=10.137.196.70:SM-Host,10.137.196.70-msIP:SM-Link
measurements=HTTP-TCPConnectionTime(10.137.196.70)
stateComputationRule=measurementsOnly
ipAddress=10.137.196.70
hostName=www.abc.com.fakeisp.net
serverType=Apache/1.2
category=WebHost

[10.137.196.58:SM-WebService]
; specification of a web service node
components=10.137.196.58:SM-Web, <10.137.196.58,Web>:SM-NFS, <10.137.196.58,Web>:SM-DNS
; the web service has dependencies on the web server, an NFS service, and a DNS service. The identities
; of the NFS service and DNS service that the web service relies on are not available from external discovery.
; hence, the service model has place-holders for these dependencies. A second phase of discovery is need to complete
; the service model instance.
measurements=HTTP-Availability(10.137.196.58),HTTP-TotalResponseTime(10.137.196.58),HTTP-DnsTime(10.137.196.58)
category=InternalWeb
; category information available in the discovered instance is carried forward
[10.137.196.69:SM-WebService]
components=10.137.196.69:SM-Web, <10.137.196.69,Web>:SM-NFS, <10.137.196.69,Web>:SM-DNS
measurements=HTTP-Availability(10.137.196.69),HTTP-TotalResponseTime(10.137.196.69),HTTP-DnsTime(10.137.196.69)
category=WebHost
[10.137.196.70:SM-WebService]
components=10.137.196.70:SM-Web, <10.137.196.70,Web>:SM-NFS, <10.137.196.70,Web>:SM-DNS
measurements=HTTP-Availability(10.137.196.70),HTTP-TotalResponseTime(10.137.196.70),HTTP-DnsTime(10.137.196.70)
category=WebHost

[Web:SM-TopLevelWeb]
; the root of the web service tree
components=10.137.196.58:SM-WebService, 10.138.196.69:SM-WebService, 10.138.196.70:SM-WebService


...  specification of [10.174.173.23:SM-DNS] and [10.174.173.23:SM-NFS] (omitted here to simplify the example)

 [10.137.196.52:SM-Pop3_Mail]
; node representing a POP3 mail server on host 10.137.196.52
components=10.137.196.52:SM-Host, 10.137.196.52-msIP:SM-Link
measurements=POP3-TCPConnectionTime(10.137.196.52)
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net
relatedSmtpServer=mail.fakeisp.net
category=

...  similar specification for [10.137.196.54:SM-Pop3_Mail]

[10.137.196.52:SM-ReadMailService]
; Read mail service node corresponding to the Pop3 mail server on 10.137.196.52
measurements=POP3-Availability(10.137.196.52), POP3-TotalResponseTime(10.137.196.52), POP3-AuthenticationTime(10.137.196.52)
components=10.137.196.52:SM-Pop3_Mail, <10.137.196.52,Pop3_Mail>:SM-NFS, <10.137.196.52,Pop3_Mail>:SM-Auth
; dependencies of the read mail service are the Pop3 server, an NFS service and an authentication service that is used by
; the Pop3 server. Component specifications with angled brackets (<>) indicate that some components have not been
; completely discovered. In the above instance, the NFS and authentication service dependencies have not been specified in
; the discovered instance of Figure 8. Based on this service model instance, a second phase of discovery can be triggered
; so as to complete the specification of the components list.
category=

...  similar specification for [10.137.196.54:SM-ReadMailService]

[pop3.fakeisp.net:SM-ReadMailServiceGroup]
components=10.137.196.52:SM-ReadMailService,10.137.196.54:SM-ReadMailService

[ReadMail:SM-TopLevel-ReadMail]
components=pop3.fakeisp.net:SM-ReadMailServiceGroup,10.137.196.52:SM-ReadMailService,10.137.196.54:SM-ReadMailService

[InternalWeb:SM-Category]
components=10.137.196.58:SM-WebService,10.137.196.58:SM-Web

[WebHost:SM-Category]
components=10.137.196.69:SM-WebService,10.137.196.70:SM-WebService,10.137.196.69:SM-Web,10.137.196.70:SM-Web
```

Figure 10: An example service model instance specification

The discussion of the service model construction approach in Sections 5.2-5.5 has focussed mainly on the initial construction of a service model instance. In practice, discovery and service model instantiation must happen on an on-going basis, since new servers and network elements may be added to an ISP system from time to time. By comparing the newly generated service model instance with a previous instance, the measurement agent configurator (see Figure 5) can determine additional measurements that need to be made to adapt to the configuration changes in the ISP system.

5.6 Customizing the Service Model to an ISP's Organizational Structure

In order to be useful, a service model instance must be customized to an ISP's organizational structure, so that ISP operations personnel only view the status of services and service components that are of relevance to them. A straightforward approach to customize a service model instance is to manually edit the service model template to explicitly include nodes that capture the organizational dependencies. For instance, in the example in Figure 10, the web service nodes are grouped into two categories – *InternalWeb* and *WebHost.* Each of these categories may be managed by different operators. To accommodate this case, the service model template can be modified to define nodes that represent the individual categories and dependencies that indicate the components of the different categories. Since the organizational structure varies from one ISP to another, this approach would require that *each* ISP edit the service model template to match their organization structure. Editing the service model template to define the categories and the component relationships can be a tedious task, especially when the target ISP system has hundreds of hosts.

The alternative approach we propose enables customization of a service model instance to an ISP's organizational structure without requiring extensive changes to the service model template. The key idea behind this approach is to allow an ISP to specify their organizational structure using the configuration interface discussed in Section 5.1. The service model template is pre-specified to exploit the configuration specification and generate a service model instance that is custom to each ISP system. The main attractiveness of this approach is that an ISP operator has to primarily edit the configuration specification, a much simpler task compared to modifying the service model template.

The main steps in this approach, which is demonstrated in Figure 7-Figure 10 are:

- Through the configuration interface, an ISP operator specifies ways in which the discovered instances are to be categorized. For instance, in the example in Figure 7, the configuration specification indicates that the ISP uses the naming pattern **.com.fakeisp.net* to identify web servers that are hosted for businesses. Web servers that do not match this pattern are internal web servers that are used by the ISP's residential customers.
- Each of the discovery modules then uses the configuration specification to determine a categorization of the instances they discover. For instance, the web server discovery module assigns the server executing on the host 10.137.196.58 to a category named "*InternalWeb*", whereas it assigns the server executing on the host 10.137.196.69 to another category called "*WebHost*". The category associated with a service or service component is recorded as part of the discovered instance.
- The category information contained in the discovered instance is used by the service model creation engine to *automatically* construct a service model instance that represents the ISP's organizational structure. To enable this, the service model template in Figure 9 has a category section (called *SM-Category* in our implementation) that generates a node in the service model instance for each category in the discovered instance. As a result, there are nodes corresponding to the categories "InternalWeb" and "WebHost" in the service model instance of Figure 10. The

components specification in the *SM-Category* service model template section maps all the services and service components that are associated with a category to the corresponding node in the service model instance. For instance, all service model instance nodes that have their category attribute set to WebHost are children of the WebHost category node.

As can be seen from the service model instance in Figure 10, by merely specifying the categorization of services and service components using the configuration specification in Figure 7, an ISP operator can derive a service model instance that is customized for the ISP.

6 Conclusions

In this paper, we have proposed the concept of service models as a way to equip lower-skilled ISP personnel to effectively handle problems that occur in an ISP environment. For the experts too, service models can provide a way of minimizing the time and effort that they need to expend diagnosing problems. Since each ISP system is unique in many respects, customized service models have to be crafted for each of the services in each ISP system. We have described a methodology for constructing customized service models for a target ISP system with minimal human intervention. Future work in the service management area includes multi-phase auto-discovery techniques, investigation of baselining and thresholding techniques for state computation of service model nodes, and automated root-cause analysis techniques that exploit the relationships embedded in a service model.

References

[1] R. Wetzel. Customers rate ISP services, PC Week, November 1997, http://www.zdnet.com/pcweek/sr/1110/10isp.html

[2] S. Ramanathan and C. Darst. Measurement and management of Internet services using HP Firehunter. *In Proceedings of the IFIP Integrated Management Conference*, May 1999.

[3] S. Ramanathan, D.L. Caswell, and S. Neal. Auto-Discovery techniques for Internet services management, *Hewlett-Packard Laboratories Technical Report*, Jan 1999.

[4] J. Myers and M. Rose. Post Office Protocol – Version 3, *Request for Comments RFC 1939*, May 1996.

[5] J. Postel. Simple Mail Transfer Protocol, *Request for Comments RFC 821*, August 1982.

[6] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote authentication dial in user service (RADIUS), *Request for Comments RFC 2138*, April 1997.

[7] P. Albitz and C. Liu. *DNS and BIND*. O'Reilly and Associates, 1992.

[8] J. Poole and A. Doan. Inverse profile checks up on ISP performance, *InfoWorld*, April 28, 1997, pp. 68.

[9] J. Battey. Keynote Business 40 gauges Web-site performance, *InfoWorld*, Vol. 19, Issue 15, April 14, 1997.

[10] M. Avery. WhatsUp Gold enhances network supervision tasks, *InfoWorld*, Vol. 20, Issue 4, January 26, 1998.

[11] R. Enger and J. Reynolds. FYI on a network management tool catalog: Tools for monitoring and debugging TCP/IP Internets and interconnected devices, *Request for Comments RFC 1470*, June 1993.

[12] M.Thottan and C.Ji. Adaptive thresholding for proactive network problem detection. *In Proceedings of IEEE International Workshop on Systems Management*, Newport, Rhode Island, April 1998.

[13] J.L. Hellerstein. A comparison of techniques for diagnosing performance problems in information systems: Case study and analytic models, *IBM Technical Report*, September 1994.

[14] C. Hood and C. Ji, Intelligent Processing agents for network fault detection. *IEEE Internet Computing*, Vol. 2, No. 2, March/April 1998.

[15] The DMTF Common Information Model (CIM) subcommittee, http://www.dmtf.org/work/cim.html