Dynamic Service Provisioning: A User-Centric Approach

Gabi Dreo Rodosek[α] and Lundy Lewis[β]
[α]*Leibniz Supercomputing Center, Barer Str. 21, 80333 Munich, Germany*
E-Mail: dreo@lrz.de
[β]*Aprisma Management Technologies,121 Technology Drive, Durham, New Hampshire, USA*
E-Mail: lewis@aprisma.com

The paradigm shift to service management opens a new dimension in management. In this paper we address the issue of dynamic service provisioning. The vision is that users would be able to review a set of services or service packages, select or customize a package, whereupon a service provider checks the request, clicks an "execute" button, and the services are turned on, monitored and managed. Such a vision is hard to achieve. A first step towards dynamic service provisioning is to develop a framework for a service provisioning system (SPS). The key issues of the framework are (i) the concept of a service template, and (ii) the idea of looking at a service template as a case and applying case-based reasoning methods

Keywords: Dynamic service provisioning, service template, case-based reasoning

1. Introduction

Service level management (SLM) is recognized as a vital element of a "new economy" that is based on information technology (IT). As a representative example, the goals of a recent high-tech convention in Hawaii (HiTechHawaii@2000, October 2000) were (i) to address the state's shift from a manufacturing based economy to the "new economy" that is service-based driven and delivered by IT, (ii) to explore ways in which enterprises will need to redefine their business processes to meet the challenges of the new economy, and (iii) explore ways by which to implement SLM policies and procedures.

The paradigm shift to service management opens a new dimension and new problems in managing both service provider networks and enterprise networks. In this paper we address the specific problem of service provisioning. Our concept of a "service" is purposely ambiguous, much like the two sides of a single coin. First, "service" is defined with respect to the user's or business executive's point of view (e.g. see [11]). Second, it is defined as a function of underlying components such as networks, computer systems, and software applications. This approach is in contrast to the current method of defining services and quality of service (QoS) in pure "network-centric language" such as availability, packet loss, throughput, and network latency. We address the problem of dynamic service provisioning for services which are not configured individually, but are made available to a mass of customers.

The problems addressed in the paper are the definition of a user-centric "service language" and a method of mapping such language onto a "network-centric language." By following a top-down approach towards developing a framework for dynamic service provisioning, the framework contributes as well to user-centric service level agreements (SLA).

The paper proceeds as follows: Section 2 provides a sketch of requirements for a SPS, a corresponding use case model, and related work. Section 3 identifies the key concepts of a service template, including structure and examples. Section 4 introduces the design of a SPS. Section 5 discusses

O. Festor and A. Pras (Eds.) : 12th International Worshop on Distributed Systems: Operations and Management DSOM'2001 Nancy France, October 15-17, 2001.

the application of case-based reasoning methods to a SPS in order to improve the provisioning and management steps with real experience. Finally, Section 6 concludes the paper.

2. SPS Requirements and a Use Case Model

Current practices in service provisioning among service providers include (i) on-site installation and configuration and (ii) "drop-ship" and "swivel chair" provisioning. The first approach is manually demanding, since service providers cart devices to customer premises and wire/hook them up on site. The second approach represents some improvement because devices are partially pre-configured and sent to customer premises via a commercial shipping company. Customers are given minimal instruction that render the devices reachable and manageable from the service providers network operation center (NOC). Thus, the service can be set up and adjusted from the NOC.

The downside of the second method is the "swivel" aspect, meaning that service providers have to use multiple management systems for different kinds of devices, systems, or applications to render a service operable. The latter problem is familiar in both industry and research.

2.1 SPS Requirements

A third method is to use an automated SPS [8]. Importantly, the vision of service providers is to have an integrated management system whereby services can be set up from a single management console. For example, it is expected that service providers will provide a catalogue of services that can be reviewed on-line. Customers may select a service package with various options such as availability guarantees, security coverage, and response time. These are well known in the industry as SLAs. Further, the vision is that after customers would review the set of services or service packages via the web and select a package, then the service provider would verify the order request with respect to technical issues. Technical issues include whether the thresholds as specified in SLA are achievable and the question of whether the required resources for the provision of the requested services are available. Such verification is done mostly manually with tool support. After the order has been approved, the provider initiates service provision and service management procedures.

The service provider is expected to have a record of existing services and an indication of their consumption of resources. Thus, during the technical verification, the service inventory and the utilization of the required resources will have to be consulted before the new or modified service is activated. Next, it is expected that services will be monitored for possible faults. Since a service depends upon underlying devices, computer systems and applications, it is expected that the SPS will be integrated with systems that monitor and manage those components. Finally, it is expected that the SPS will provide bills at the end of the month.

2.2 Use Case Model

From these high-level requirements, we can sketch a use case model as shown in Fig. 1, where ovals indicate use cases and the stick figures indicate a class of users. Note that a user can be inanimate, e.g. an existing operations support system (OSS). A solid arrow between a user and a use case expresses the relation "uses the system for the purpose of" while a dotted arrow indicates the relation "provides (partial) means for."

The service provider uses the SPS for the purpose of setting up a library of service templates. Both the service provider and user use the SPS for the purpose of finalizing a service contract, where the template provides the means for doing so. Service Activation is provided for by (i) finalized contract, (ii) a current library, and (iii) an existing OSS. Observe that the service provider uses the SPS for the purpose of Creating Service Templates and producing Service Reports and Billing.

However, the Service Reports and Billing function depends upon (i.e. the converse of "provides partial means for") the Finalize Service Contract and Service Monitoring and Control functions, the latter of which depends upon existing OSSs. Following other relationships in Fig. 1, the reader should be able to see the inner structure of the SPS.

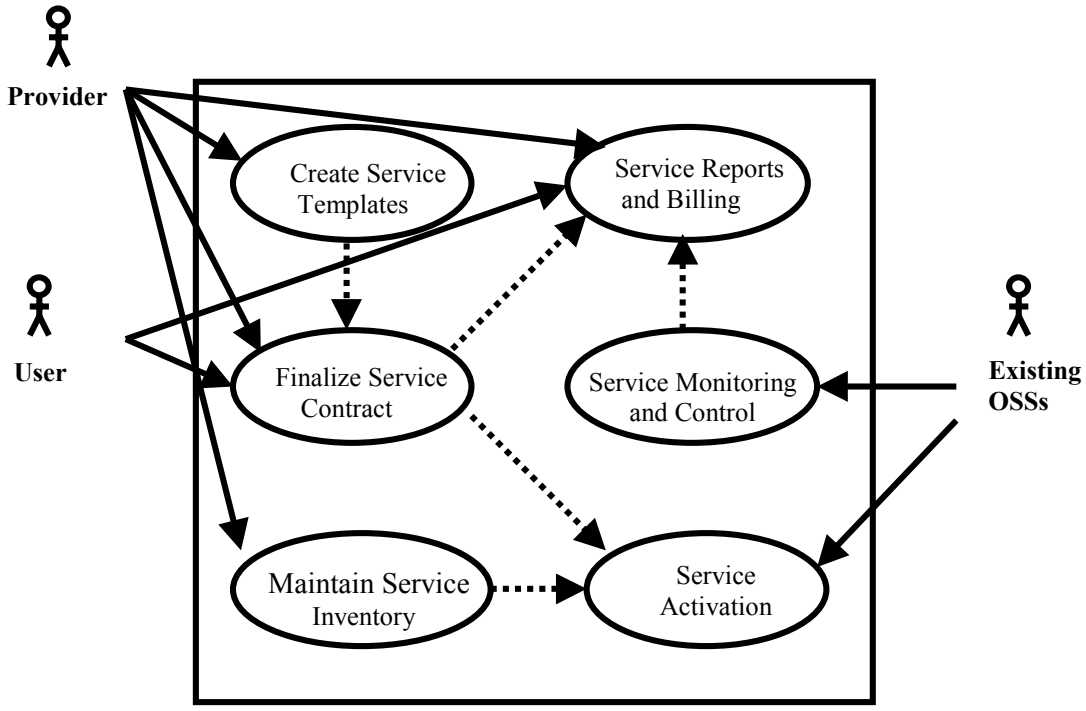

Figure 1: A use case model for a SPS

2.2 Related Work

Most existing work on SLM focuses on SLAs, QoS parameters in SLAs, and measurement techniques (e.g. [3 - 6, 9]). Further, most work expresses the relation between a service and underlying supporting components in a standard dependency graph. This relation is addressed also in [7]. Issues in automated service provisioning are discussed briefly in [2, 4], and a SPS using the Java Bean framework is discussed in [10]. However, this work doesn't address the problem of a user-centric, semi-automated or automated approach to dynamic service provisioning.

A conceptual framework and a methodology for SLM is described in [1] based on the general software development paradigm. One starts by defining a business process, then defining services upon which the business process depend, then defining service parameters and their range of possible values, and finally marking a particular value (i.e. a service level) that reflects acceptability of service performance. Next, one selects the network components upon which the service depends and constructs a function that maps measurement values of component parameters into the service parameters. However, the important issue of service provisioning isn't included in the discussion.

3. Service Templates in Support of an SPS

In order to derive a framework for service provisioning, it is necessary to start with the identification of two roles: the user and the service provider. See Fig. 2 (and refer back to the Use Case Model in Fig. 1). The user orders and subscribes services with special QoS parameters, as provided by the service provider. To identify the main parts of a SPS, it is necessary to address three central aspects:

(a) the definition of user-centric services and accompanying quality of service information to users in the form of SLAs,
(b) the gathering of necessary service-related information from resources in order to gauge SLAs over time, and
(c) the means by which to configure and control resources such as network devices, end systems, or applications with respect to selected services and the SLA.

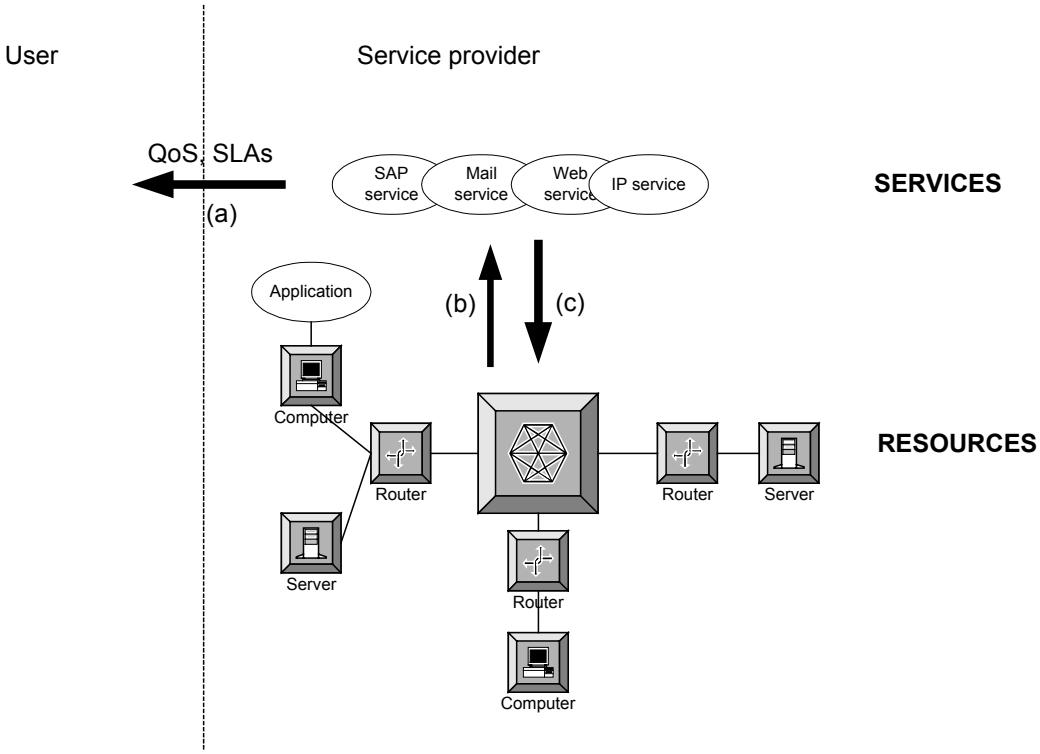

Figure 2: Three aspects (a), (b), and (c) of service provisioning

3.1 Bottom-up vs. Top-down Construction of Quality of Service Parameters

An analysis of QoS parameters is the first step towards a dynamic service provisioning system. Current SLAs in the domain of networking are mainly service provider centric (e.g. usually in terms of reachability, packet loss, response time, error rates etc.) even though the services they support include E-commerce, Web or Email services. Thus, service providers refer to Internet connectivity services (layer 3 and lower) instead of higher-level, user-centric services. However, from the business viewpoint it is important to know that emails are correctly delivered to the recipient within a certain time interval, or that hits on a business web site are reliable and fast enough to hold the user's attention.

The current practice among service providers regarding the definition of SLAs is bottom-up. Service providers typically start with network-centric parameters and then try to mould them into service-centric parameters. This procedure is represented with (b) in Fig. 2. The reason for this approach is that most current management tools are device-centric and the service representation is missing. For example, management tools such as Concord's Network Health, InfoVista, Aprisma's Spectrum, and HP's OpenView provide the means to define network-centric metrics and collect their values in an historical database. One exception is the product Continuity built by ICS in Germany. Continuity adds a layer of abstraction over the Spectrum management system that shows representations of services and corresponding collections of devices upon which services depend in a standard dependency graph. Continuity, however, doesn't cover the service provisioning problem.

Our approach in this paper is the opposite of the current trend among service providers. It is purposely top-down. We start with service-centric definitions and parameters and try to decompose them into network-centric parameters. Thus, we concentrate on points (a) and (c) in Fig. 2. We argue that by following a top-down approach the resulting framework will make it easier to realize dynamic service provisioning, user-friendly SLAs, user-friendly service reports, and business/network alignment. In this regard, we follow a common practice in general software engineering: Consider user-based requirements up front and let them influence software development throughout the design and implementation process.

Now, main components of an SPS include the following (see the Use Case Model in Fig. 1):

- Service Templates (i.e. a catalogue of services viewable by consumers),
- Service Contracts (SLAs),
- Service Activation and Modification,
- Service Inventory,
- Tight integration with underlying element and network management systems,
- Service Monitoring and Control (reactive and proactive), and
- Service Billing.

All the components of a SPS can not be discussed in requested detail in this paper. Therefore, our discussion is focused on the components which need to be addressed first and are basic for the framework, viz. the identification of the service template structure.

3.2 On Service Templates

From the discussion so far, we can identify two parts of a service template:

- a user part and
- a provider part.

To derive the structure of a service template in a satisfactory detail, it is necessary to analyze the use case model in Fig. 1 and the central aspects of a SPS in Fig. 2. From the user perspective, it is important to have a service model with the agreed quality in the Finalize Service Contract use case. In addition, a user wants reports about the provided quality of services as indicated in the Service Reports and Billing use case. From the service provider's perspective, it is important to know how to set-up, monitor, and manage services so that SLAs are not violated. This suggests that the service template should include a language that reflects the service from the provider's point of view in addition to the customer's view.

Thus far in the industry, there have been somewhat vague descriptions of what the structure of a service template should be, although most service providers acknowledge that it is a crucial aspect of an SPS.

As a result of our previous discussion, we identify the following dimensions of a service that need to represented in a service template:

- **Functionality**:
    - Defines the service functionality as perceived by a user at a service access point (SAP) (e.g. sending/receiving mails by using a mail client)
    - Specifies the internal building blocks of a service (its subservices)

- **Quality**: Defines the QoS parameters of a service
- **Management**:
    - Defines how a service has to be managed (i.e. how a service is instantiated on the infrastructure of a provider and the way that network and systems management platforms have to monitor underlying components)
    - Defines management information that users can access over a Customer Service Management (CSM).

Within a service provider domain all three dimensions are relevant and are part of the service management (SM) operations of a service provider. From the user's view, *Functionality* refers to the service functionality that a user can access at a SAP (e.g. send an email over the Netscape mail client). *Quality* refers to the specification of user-centric QoS parameters (e.g. hit rate, response times) with service levels. *Management* from the user's point of view refers to the CSM [5] application. CSM is the management interface between a user and a provider, where a user can access management information about the subscribed services.

In terms of a CSM a user should get information periodically about the quality of the subscribed services. The next step of CSM is to enable the user to actively subscribe to new services online, select QoS parameters and their values, and request further quality reports. Thus, a user configures an "order" for a service with his requirements regarding quality and information. The service provider needs to realize this "order" with a SPS.

From the provider's view, *Functionality* refers to the provision of a service by identifying the subservices upon which a service depends as well as the resources through which the service is provided. Resources, however, are not part of a service template. The reason is to clearly distinguish between the description of services and the instantiation on the resource layer, for example to "move" services dynamically across the infrastructure. Quality of service refers to the monitoring and aggregation of so-called Quality of Device (QoD) parameters as observed by management tools. Management in the context of a service provider refers to the configuration of management tools to monitor the resources through which services are provided. Additionally, policies which describe the (enterprise specific) ways services are provided and operated are of relevance as well.

The service provider part of the service template is hidden from the user. However, the service provider can view the entire template. A collection of service templates would be contained in a service provider's template library, where each template includes a service name and a description, the resource requirements for the service, and a procedure for instantiating the service. A collection of services described in this way would make up a service template library.

A service template is a key concept in service provisioning, and in terms of workflow it is the entity that initiates the service provisioning process. The structure of a service template is as follows:

Service template: ***service***

User part

Functionality*: the name of the service with the description of its functionality as accessible by a user*

User-centric QoS parameters:*: QoS parameters which represent a particular user's view with the specification of intervals within which a user can select values*

CSM: *configuration of reports about the provided quality, type of report (online, offline) etc.*

Provider part

Functionality*: specification of the required subservices*

Provider-centric QoS parameters: *QoS parameters which represent the service quality from the operating view in addition to the provision of user-centric QoS*

Provisioning steps*: configuration of resources*

Management steps*: the activation of resource monitoring tools i.e., network and systems management tools)*

Policies*: description of enterprise-specific issues related to the operation of services*

3.3 Examples of Service Templates

To illustrate the structure of a service template, let us consider the following two simplified examples. Assume that it is possible to subscribe to a user help desk (UHD) service to support the resolution of problems of the subscribed services. A simplified service template for a user help desk (UHD) service would look as follows:

Service template: UHD service

User part:

Functionality: support the resolution of problems
User-centric QoS parameters: problem resolution time [1-24h], first response time [1-24h]
CSM: monthly reports on the average resolution time, online reports on the average resolution time

Provider part:

Functionality: database and workflow subservice (e.g. trouble ticket system), system subservice, network connectivity subservice
Provider-centric QoS parameters: average resolution time
Provisioning steps:
Configure user account (user name)
Configure escalations, priorities of tickets (problem resolution time, first response time)
Check number of available (floating) licenses, check server load
Configure CSM application with respect to the requested reports
Management steps:
Configure network management platform to check network connectivity
Configure systems management platform to check system resources (e.g., server load)
Configure database and trouble ticket systems management
Policies: service is made available 24 hours a day, 7 days a week

Another example is the simplified service template for a service Internet Access.

Service template: Internet access

User part*:*

Functionality: Internet access
User-centric QoS parameters: bandwidth, availability, throughput, utilization at the service access point
CSM: online reports on throughput, utilization, availability of the router interface, monthly reports

Provider part:

Functionality: connectivity subservices on layer 2 for access and backbone lines, provided by an external provider
Provider-centric QoS parameters: availability of the whole backbone
Provisioning steps:
Configure an interface of a router with requested bandwidth
Configure user account (e.g., database)
Configure CSM application to provide the requested reports
Management steps:
Configure network management platform
Configure monitoring tools for end-to-end flows
Policies: service is provided 24 hours a day, 7 days a week

Thus, a provider would provide a service template for several services (e.g. Web service) with the appropriate parameters, e.g. access time for a web site and monthly reports on access statistics to a user. A user would now select a service template for, say a Web service, and configure the available parameters and order a service. In case the provided parameters and services do not meet the user requirements, a negotiation with the SP needs to start (i.e. as indicated with the Finalize Service Contract use case). At that time a service provider needs to check his resources to see whether it is possible to provide the service.

4. Design of a Service Provisioning System

Assume the following scenario based on the Use Case Model in Fig.1. A user orders a new service with specific user-centric QoS parameters over the CSM application. Afterwards, as depicted in the interaction diagram in Fig. 3, a service request (i.e. instantiated service template) is generated by a workflow management system. First, a service provider performs a verification of contractual issues with respect to SLAs (i.e. whether the service order request is within the agreed parameters). If the verification is not successful, the request is abandoned. At this step a negotiation phase between a service provider and a customer may start regarding the desired QoS parameters, the values, thresholds, penalties etc. This negotiation phase is not analyzed in the sequence diagram.

As part of the technical check, which is performed afterwards and may be supported by a service planning application (SPA), it is determined whether the service request can be granted or needs to be rejected with respect to the available resources. In case the request is granted, the provisioning steps and the management steps as defined in the service template are executed. The provisioning and management steps are specified in terms of actions with parameters (e.g. configure(router)). In some cases such detailed specification is not possible, especially if several providers are involved in the service provision.

To achieve the granularity where actions can be translated directly into commands of management tools is a difficult task. The idea is to start with actions that are specified by experts and refine/improve these steps through real experience by the technicians or experts who execute the steps. Thus, it is a learning process where at the end the steps can be specified in terms of commands for management tools. If this is achieved, then it is possible to refer to an automated SPS. In case steps are specified in terms of start_database(host1), such a specification is already helpful for technicians who execute these actions. Furthermore, the step start_database(host1I) can be refined to concrete system commands (e.g. oracle {start | stop} on host1, starting or stopping processes of the Oracle database). To specify the steps in a sufficient granularity for an automated SPS in the first step is, however, a hard task due to the distributed realization of services upon resources and the associated complexity.

The implementation of a CSM application has been successfully shown in [5] where it is used now for four years nationwide. Trouble ticket systems (e.g. Peregrine's ARS) have also proved to be useful as a workflow management tool. The verification of the contractual and technical issues (e.g. resource allocation) is performed by a service provider manually or with supporting tools. The resource allocation problem can be supported by a *service planning application* (SPA), for example RiskWise from Parc Technologies [17] using the constraint-based approach for modeling IT services, as presented in [16]. The configuration of devices to provide a service (not shown in Fig. 3) may be realized via a network management platform such as Aprisma's Spectrum or HP's OpenView.

To enhance the provisioning and management steps of the service template with real experience, we identify case-based reasoning (CBR) as the appropriate method to address this issue and recognize *the service template as a case***.** The user part of a service template characterizes the service request (i.e. what service with what QoS parameters have been ordered), and the provider part, - the provisioning and management steps - identify the necessary workflow to realize the user's service order request.

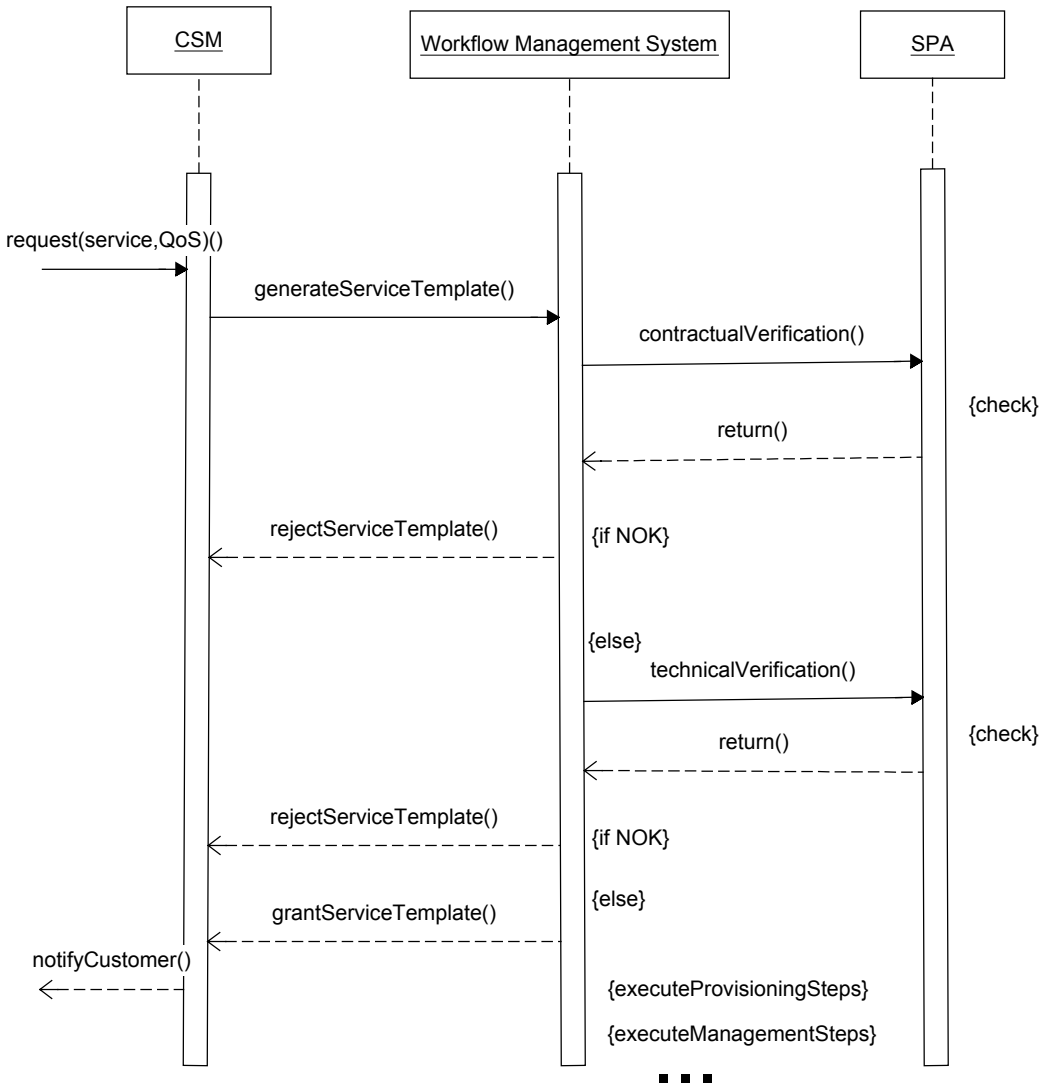

Figure 3: Sequence diagram of a SPS

5. Applying CBR to Service Provisioning

After finalizing a service contract, the next step is service provisioning, i.e. the next activities that occur after a user sends the order for a service to the SP via the CSM application. There are several aspects of service provisioning which need to be observed. One aspect is to check whether there are available resources to meet the user service order. The relevance of this issue is demonstrated especially with the example with the "Internet Access" service. After receipt of a user order, it is necessary to check whether there is enough bandwidth available on the system with a bandwidth management system. At this point, we see the tight integration with the network, system and application

management, i.e. the so-called "resource management." We can recognize that service planning, especially the aspect of checking available resources, can be seen as a "resource allocation" problem.

Another important issue is the configuration of resources to activate the service. For example, an SP may use a systems management platform to configure and monitor systems-related parameters (e.g. server load) and use a network management platform to monitor the connectivity between clients and server.

Consider the workflow aspect of an SPS, viz. the service provisioning steps. We propose the idea of thinking of a *service template* as a *case* and applying the case-based reasoning paradigm to service provisioning.

Case-based reasoning (CBR) is a well-known automated reasoning paradigm that issued from the AI community in the 1980s. In the network management domain, CBR has been applied to fault management and trouble-shooting (e.g. see [12 - 15]). The insight in this work is to look at a trouble ticket as a case, and then to apply CBR concepts of a case library, alternative case views, case retrieval, case adaptation, case execution, and library updating to the task of network fault diagnosis and repair.

Now, with respect to service provisioning, consider that the user view is much like an ordering form. When the user fills out the form (i.e. requests a service), the most similar template is retrieved from the template library (i.e. the case library in terms of CBR) and adapted to the user's request. The type of adaptation suited for this is parameterized adaptation. Thus the retrieval and adaptation of the template is the second iteration of the case, and now looks much like a work order. The work order is handed over to a field technician who follows the procedure described in the case for setting up the service. If problems arise, the technician notes them in the case. Thus, the third iteration of the case (i.e. the results of the execution of the case) becomes an "experience" and is entered back into the case library. The next time around, when a new user places an order, the most similar case is retrieved. The retrieved case might well be a different case depending on the technician's notes and additional parameters.

The main benefit of using the CBR paradigm is to improve the provisioning and management of services with real experience. Due to the dependencies between services and the fact the provision of services vary from user to user (and from SP to SP), the procedure for service provisioning becomes complex and unstructured, and those are the kinds of problems that CBR systems are designed to address. Although a well-seasoned field technician may become quite adept at setting up services for a particular SP, the problems are well-known, including employee churn and the introduction of new technicians on job.

Consider now a simple example. The main steps of service provisioning and management are described in an initial service template, and then adapted to make up a work order. Afterwards, during the configuration and management of several service requests, the service provisioning/management steps are enhanced by additional steps. Suppose the plan for particular service at Service Activation time is:

Given parameters x and y in a work order W for Service S, then
  Perform A
  Perform B
  Perform C

A field technician might find that the procedure is inadequate because an additional parameter z, not included in the work order, renders the plan unworkable. The technician might compose a new procedure as follows:

Given parameters x, y, and z in W for S, then
  Perform A
  If y then perform B1
  If y and z then perform B2
  Perform C

Clearly, the procedure is improved with the technician's experience. Also, it is expected that further experiences with S will enhance the knowledge required to realize S. In the CBR literature, this is often called incremental planning [15].

6. Conclusions

The paradigm shift to service management opens a new dimension in management and also new problems to deal with. Service provisioning is certainly one of the most important but also difficult ones. The proposed framework for a service provisioning system is a step towards the vision that a user is able to review a set of services, select an appropriate service, configure quality parameters, whereupon the SP checks the request, clicks an "execute" button, and the services are turned on, monitored, and managed. We believe that the key issue of a service provisioning system and service management in general is the concept of a service template. In the paper we proposed a generic service template structure which can be refined or instantiated for several purposes. Service provisioning and fault management were two of them. Further, we proposed to apply case-based reasoning methods to improve the service provisioning and management steps with experience.

Acknowledgements

The authors wish to thank the members of the Munich Network Management (MNM) Team and Aprisma'a Research Department for helpful discussions and valuable comments on previous versions of the paper. The MNM Team, directed by Prof. Dr. Heinz-Gerd Hegering, is a group of researches of the Munich Universities and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its webserver is located at http://wwwmnmteam.informatik.uni-muenchen.de.

7. References

[1] *Service Level Management for Enterprise Networks,* L. Lewis, *Artech House*, ISBN 1-58053-016-8, 1999.

[2] *SLA Management in Federated Environments,* P. Bhoj, S. Singhal, S. Chutani, *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, Boston, IEEE Publishing, 1999, pp. 293-308.

[3] *Determining the Availability of Distributed Applications,* G. Dreo Rodosek, T. Kaiser, *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, IEEE Publishing, 1997, pp. 207-218.

[4] *Management of Federated Services*, P. Bhoj, D. Caswell, S. Chutani, G. Gopal, M. Kosarchyn, *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Management*, San Diego, IEEE Publishing, 1997, pp. 327-340.

[5] *An ODP Enterprise Specification of CUSTOMER SERVICE MANAGEMENT for connectivity services*, M. Langer, M. Nerb, *Proceedings of the 3rd International EDOC Conference*, Mannheim, 1999.

[6] *Monitoring Quality of Service across Organizational Boundaries*, R. Hauck, G. Vogt, *Proceedings of the 3rd IFIP/GI International Conference on Trends towards a Universal Service Market* (edited by C. Linnhoff-Popien and H.-G. Hegering), number 1890 in Lecture Notes in Computer Science, Springer, 2000.

[7] *CIM*, Core model, DMTF, 1998.

[8] *Telecommunications Magazine*, Special Issue on Service Creation. January 2000.

[9] *Managing Application Services over Service Provider Networks: Architecture and Dependency Analysis*, G. Kar, A. Keller, S. Calo, *Proceedings of NOMS 2000* (edited by J. Hong and R. Weihmayer), IEEE Publishing. 2000.

[10] *Building a Service Provisioning System using the Enterprise Java Bean Framework*, S. Sengul, J. Gish, and J. Tremlett, *Proceedings of NOMS 2000* (edited by J. Hong and R. Weihmayer). IEEE Publishing, 2000.

[11] *It Ain't What You Charge, It's the Way that You Do It: A User Perspective of Network QoS and Pricing,* A. Bouch, M. Sasse, *Proceedings of Integrated Network Management VI* (edited by. M. Sloman, S. Mazumdar, and E. Lupu). IEEE Publishing. 1999.

[12] *Troubleshooting Network Faults Using Past Experience,* Melchiors, C. and L. Tarouco, *Proceedings of IEEE/IFIP Network Operations and Management Symposium* (ed. By J. Hong and R. Weihmayer), IEEE Publishing, 2000.

[13] *An Automatic Fault Diagnosis and Correction System for Telecommunications Management*, Penido, G., J. M Nogueira, and C. Machado, *Proceedings of Integrated Network Management VI* (edited by M. Sloman, S. Mazumdar, and E. Lupu), IEEE Publishing, 1999.

[14] *Managing Computer Networks: A Case-Based Reasoning Approach*, L. Lewis, *Artech House*, 1995.

[15] *Using case-based reasoning to acquire user scheduling preferences that change over time*, K. Sycara, D. Zeng, and K. Miyashita, *Proceedings of the Eleventh IEEE Conference on Artificial Intelligence Applications* (CAIA '95), Los Angeles, IEEE, Feb. 1995.

[16] *A CSP Approach to IT Service Management*, G. Dreo Rodosek, Th. Kaiser, R. Rodosek, *Proceedings of Integrated Network Management VI*, edited by M. Sloman, S. Mazumdar, and E. Lupu). IEEE Publishing. 1999.

[17] *RiskWise*, White paper, Parc Technologies (www.parc-technologies.com)