Policy Based Monitoring of a Web-Based Service

Adrian Baldwin, Marco Casassa Mont
Internet Business Management Department
HP Laboratories Bristol
HPL-98-76
April, 1998

E-mail: [ajb,mcm]@hplb.hpl.hp.com

service management, system management, policies, monitoring

The Internet provides an infrastructure for deploying and delivering business critical services either within a corporate Intranet as out sourced services or even end user services such as shops selling consumer products. For such services to be successful it is essential that they have a user focused management system to ensure that the end user experiences a reliable and secure service. This paper addresses issues associated with service level monitoring and diagnosis of a potentially complex web site and a tool is described that allows for the generation of service level events as well as periodic analysis.

The service administrator needs to focus on the requirements and functioning of the service so that the effect on service provision of any changes or faults in the computer system are clear. A graphical representation of a service based on a hierarchical graph structure has been used so that the administrator can present their model of the service, navigate through the service and have significant events mapped onto their service model. The graph hierarchy allows the service to be successively decomposed into a number of sub-services or user interactions until a desired level of granularity has been reached. The graph structure is then used to represent the links between various parts of a service.

'Policies' are associated to the graphical service components in order to describe contexts and constraints regarding the correct functioning of that part of the service. For examples policies may describe the system configuration necessary for a particular part of a service; or may describe performance requirements; or even describe potential security violations. Policies are validated against data provided by an underlying information system within the management tool. This information system supports retrieval, and logging of system information and its association to a particular part of the service. An event notification mechanism allows the policy monitoring system to re-evaluate a particular policy when the information it depends on changes: failure to comply with a policy is detected and displayed on the service graph. In such a context, the service administrator can run a diagnosis script to find out details of the failure and the same mechanism can analyse system information, for example, performing security checks or driving performance trends.

Internal Accession Date Only

Policy based monitoring of a web-based service

Adrian Baldwin and Marco Casassa Mont

Hewlett Packard Laboratories
Internet Business Management Department
Filton Road, Stoke Gifford,
Bristol BS12 6QZ
UK
ajb@hplb.hpl.hp.com, mcm@hplb.hpl.hp.com

Abstract

The Internet provides an infrastructure for deploying and delivering business critical services either within a corporate Intranet as out sourced services or even end user services such as shops selling consumer products. For such services to be successful it is essential that they have a user focused management system to ensure that the end user experiences a reliable and secure service. This paper addresses issues associated with service level monitoring and diagnosis of a potentially complex web site and a tool is described that allows for the generation of service level events as well as periodic analysis.

The service administrator needs to focus on the requirements and functioning of the service so that the effect on service provision of any changes or faults in the computer system are clear. A graphical representation of a service based on a hierarchical graph structure has been used so that the administrator can represent their model of the service, navigate through the service and have significant events mapped onto their service model. The graph hierarchy allows the service to be successively decomposed into a number of sub-services or user interactions until a desired level of granularity has been reached. The graph structure is then used to represent the links between various parts of a service.

'Policies' are associated to the graphical service components in order to describe contexts and constraints regarding the correct functioning of that part of the service. For examples policies may describe the system configuration necessary for a particular part of a service; or may describe performance requirements; or even describe potential security violations. Policies are validated against data provided by an underlying information system within the management tool. This information system supports retrieval, and logging of system information and its association to a particular part of the service. An event notification mechanism allows the policy monitoring system to re-evaluate a particular policy when the information it depends on changes: failure to comply with a policy is detected and displayed on the service graph. In such a context, the service administrator can run a diagnosis script to find out details of the failure and the same mechanism can analyse system information, for example, performing security checks or deriving performance trends.

1 Introduction

The Internet or Intranet provides a powerful mechanism for the delivery of distributed services; but if the mechanism is to be successfully used to support business level services it is essential that they can be effectively secured and managed. This paper describes a tool designed to support a system administrator in ensuring the server side of the application runs in a reliable fashion. The Internet is a highly service orientated system in that the user sees the information or the transactions they are performing and very little of the underlying systems (particularly at the server side). It is therefore essential that the administrator can manage the systems and view system events and problems in terms of the service being provided. This paper concentrates on this paradigm switch by providing a visual tool to link an abstract service view to the underlying system.

A prototype management tool, see Figure 1-1 for a general architecture, allowing the administrator to decompose and represent the structure of a service in terms of a hierarchical graph has been developed. Logic based policy constraints are used to describe the correct functioning of each part of the service by describing relationships that must exist between bits of system data to ensure that the service is maintained. A management provider system is used to obtain detailed system information or link to a management system such as Open View. An event system exists such that when a piece of system information changes the policy constraint can be re-evaluated and an error flagged if the data change leads to a problem with service delivery. Further pieces of system information can be associated with each piece of a service and an analysis tool can be used to walk through the policies and data performing analysis tasks such as service diagnosis. The paper describes the ideas behind the service model and system information provision before going on to discuss how policy concepts can be used to link the underlying systems to the service description.

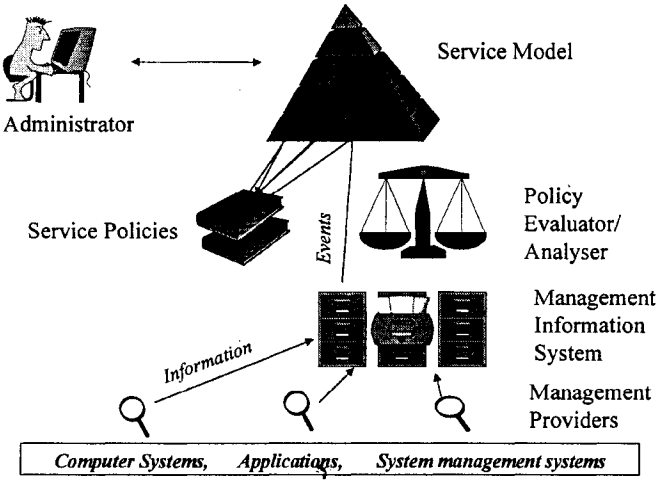

Figure 1-1
The overall architechture of the service management system

2. Service Model

To manage a service it is necessary to have a model of the service from the users point of view thereby allowing the users expectations to be understood; a necessary precursor to effective service management. A simple web service could be viewed as a single entity that delivers web pages to the user and the users expectations are based on their ability to access the pages within certain time limits. As the service becomes more complex from either the user or providers point of view it is useful to start decomposing it into a number of sub-services each of which can have specific management aims.

Take the example of a web based electronic shop selling and supporting a wide range of products. The service could be sub-divided just like a large store into areas containing different types of products; equally there is a split between product information and advice, purchasing of products and the support of products sold. Initially the administrator may choose to model the different product areas of the shop

indicating that these pieces are run in a separable way. The administrator could then further subdivide the service into pieces associated with the way the products are sold; such as product information, advice, support and purchasing. There is a clear navigation path through the service at this point from product information to advice and through to purchasing. The level of detail to which the decomposition occurs is also a factor for the administrator to control and will depend on the granularity of the management requirements.

The service model is represented as a hierarchical graph structure where the service or sub-services are represented by graph nodes and the navigation between the pieces of the service are represented by the graph arcs. The hierarchical aspects of the graph allow different decomposition depths to be visualised and even allowing different parts of the service to be viewed at different levels of detail. Take the example of the Internet shop, say selling computers, software and printers. At the highest level a single node graph would represent the shop as a whole. This top Internet shop node can be expanded and replaced with nodes representing each type of product being sold. In turn one of these nodes can be expanded to reveal the services within that department, see Figure 2-1. This type of graph structure provides a visual image of the service on which management information can be projected such that the administrator can focus on some areas whilst only seeing a more abstract management view from others

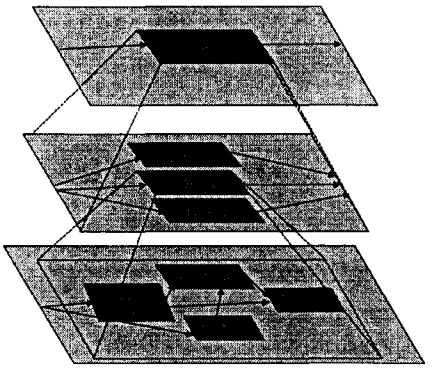

Figure 2-1
An example of the hierarchical graph structure used in representing service structure.

3. *Managing the service*

The service graph provides a visualisation of the service model as defined by the administrator allowing pieces of the graph to be associated with relevant management information. Each sub-service can be associated with various data sources containing management information relating to the provision of that portion of the service. Such a system provides a service-orientated way of organising management data allowing the administrator to browse through information related to each part of a service.

The next stage in producing an effective management tool is to allow the administrator to state their expectations for each portion of the service in terms of the associated management data or the status of the component sub services. These

management expectations are described as detailed constraints on the state of system information and can therefore be viewed as functional policies. The prototype only performs policy based monitoring and does not attempt to enforce the policies (Barruffi *et al* 97). Instead when a policy is found to be broken an event is flagged to the administrator via the service graph structure and the administrator is expected to perform any necessary action. Details about the policy representation and monitoring are discussed in a later section.

4. Information Provision

The service model along with the policies can be viewed as a management description of a service and to monitor the extent to which the management requirements are being met it is necessary to have an information system. A complex service will be constructed using many different system resources such as applications, databases and servers. The service management tool aims to allow the administrator to pull together information from each individual component or from an underlying management system into a standard format. A management provider interface provides a standard way to plug in interfaces to each available source of management information.

The management provider is a thin layer sitting between a system providing management information and the service management console. The management provider should provide a UI into the information source thus ensuring flexibility by keeping system specific functionality outside of the core management console. The management provider should provide its data in one of two standard forms, a table or a table generator (allowing references to dynamic sets of tables). These tables can be updated by the management provider as the contained information changes. Each service node interested in the information contained in a table registers with that table and it will be notified whenever the management provider changes the table's contents. Ideally it would be possible to log the contents of selected tables in a database to allow for historical analysis.

Consider the web shop manager who is running their service on an NT infrastructure and wants to use WBEM to access application and system data. The administrator would choose an appropriate service node and start to associate information from the WBEM system. This would call the WBEM management provider which uses its own UI to allow the user to select the data. The management provider is responsible for passing the information to the service management system and generating events when ever the information changes. They may also use SNMP and DMI management providers or even a specific provider written to interface with a their particular web server.

5. Policy

The term policy is used here as a description of the desired functioning of the system. More accurately the term functional policy may be used to refer to the low level policies being monitored in contrast to much of the policy work (Wies 94, Sloman 93, Goh 97] concentrating on specifying policies derived from business goals and strategy. Ideally the policy based management system would support the overall policy transformation cycle deriving highly specific policies from top-level business

goals. This full policy refinement cycle is considered beyond the scope of this paper with the focus here being on low level functional policies grounded on system information. Even at this level the policy specification should have a number of features describing the policy such as its motivation, production processes and peoples' responsibilities. The work described here focuses on the very narrow computational aspects of policy relating to monitoring; that of the context for activating a policy and a policy constraint (compliant condition) describing a set of valid system states.

The work here takes a slightly different overall slant in that the service is considered in isolation from a global enterprise strategy. This paper assumes that a number of the IT functions can be separately specified as individual services which may can be managed separately or outsourced. Or that the service, such as the Internet shop will be an integral part of the business but again the IT management could be outsourced. This emphasis on service management is aimed at ensuring that the requirements, aims and costs of the IT systems can be matched to the business processes they are supporting. The service policies may then be derived from the overall business aims and goals; however, the service layer adds an extra layer in the policy transformation cycle.

5.1 Policy Template

With this in mind these computational elements of the policy are described using a logic (Prolog like) language allowing policies to be written in a declarative style such that the policy writer need not consider how the policies are used in management. A policy template has been used to add some structure to the information within the policy both to increase clarity and to aid the policy evaluation. This template is defined as a logic predicate allowing hierarchical policies to be defined in a recursive manner. The first element in the template generates a domain of subjects to which the policy applies; this takes the form of a logic goal that binds a domain variable with a list of subjects. The remainder of the policy forms a list of policy blocks each containing a context, policy constraint and action. Each policy element contains a language description along with a logical goal allowing an easier view or human explanation of the policy to be presented to the user rather that the logic goal.

The domain definition can simply generate a list of single entities to which the policy should apply but can also generate tuples containing entity references whose relationships must be checked by the policy constraint. In most cases the policies are related purely to system specific entities, however, if policies want to talk about users or 'types of users' then role concepts can be included into the domain specification.

The context controls when the policy is activated according to some set of system features. As such the contexts provide a mechanism for policy selection and can aid the implementation of time and usage dependent service level agreements. For example a context may state that the policy is active between 9 and 5 o'clock Monday to Friday; alternate policies may be applicable at different times; or allow a policy to specify a response time when there are less than 10 users.

Typically the policy constraint would be applied to each subject defined in the domain; but a series of predicates are provided to support various expressions over the domains components such as at "least n must be working". It is also possible to gather

data from each subject in the domain and test the accumulation of the data allowing expressions such as the 'total access time for all components must be less than 10 seconds'. The information system described in the previous section should ideally present a comprehensive description of the system state. The policy constraint is built on top of this system state description allowing various bits of information to be constrained such that the management requirements are described.

An action field is included in the template so that simple scripts can be run when a policy failure occurs. Such scripts may perform some basic repair actions but the required repair can be too complex and a script could summon assistance or in the case a breach of security policy perhaps shut the system down.

A simple example of this type of performance policy for accessing a set of static pages on a web server would look like:

```
Policy (
        domain ( findfiles ( "docroot/pages", "html", _domain ) )
        [ Policy Block (
                context ( between ( 9am , 5pm ) ),
                constraint ( forall ( _domain, _url,
                                        [totalAccessTime ( _url , _accessTime ) ,
                                        less ( _accessTime, 10 ) ] )
                action ( )
        .... ] ) .
```

Here the domain constraint finds all the web files that are being constrained. The first policy block has a context specifying it is active between 9am and 5pm and the constraint specifies that the total access time for each url obtainable from the domain must be less than 10 milliseconds.

A general language has been provided to allow the administrator to be able to express the widest possible range of policies allowing complex management requirements to be specified. Obviously the administrator should not be expected to program and as such there is a need for policy composition tools which are discussed later. Along with a flexible language it is also necessary to provide a wide range of functions or predicates that express the highlevel concepts an administrator may need to express.

5.2 Policy and Service

The service model describes a way in which a service can be decomposed into various sub-services and visualised using a hierarchical graph. The policies provide a mechanism for describing the management aims for specific to part of a service by describing constraints on the values system data can hold. Each part of the service can have its own management requirements both from the service point of view as well as from the way in which the service is delivered. As such, a policy, or list of policies, is associated with each node in the service graph describing the desired service functionality. This means that policies associated with each service node can cover a wide variety of management objectives such as performance, security and reliability. The classification of policies could lead to different views on the service according to the role of a particular administrator; for example, a security administrator may only see violations of security policies.

In the example of the Internet shop discussed earlier the service model has a number of levels. At the top level of the Internet shop policies could specify a minimum time

for the delivery of a page of information using a policy similar to that shown in the previous section. At the leaves of the hierarchy there may be policies that describe the configuration of the systems used to form part of the service. For example, a policy could check that the web server providing the product purchasing system could access databases showing product prices and availability. If this service is further decomposed such that there is a 'get customer details' service then policies may check for events generated by the customer database server. Other policies may check event logs or interpret events collated through an event management system such as ManageX.

The service graph allows different levels of service description providing the ability to set different policy requirements. Take the example of the a simple printer service which consists of a number of printers and spoolers. Each individual printer can be thought of as an individual service and can be given constraints describing the correct working of that printer. At this level the administrator can see if each of their printers and spoolers is working. The administrator can group these printers into more abstract services and set different management policies that relate better to the objectives of their service. In this printer example the administrator may group together all their LaserJet printers and the appropriate spooling services and set management goals that at least half must be working. This means that the detailed service decomposition can show faults but higher level polices refering to these sub-services may filter these failures because they do not lead to an overall service failure.

5.3 Policy Writing

This system would require a great deal of expert knowledge to configure the monitoring for each individual service or even to change to service policies or deal with changes to the service. It is believed that this is a fundamental difficulty when trying to apply policy at any level of computation. Whilst it may be relatively easy to give a general description of the management aims for a service it will always be difficult to convert this description into a formal description mapping down to the system level. As such, it is believed that a number of tools should be provided to ease the specification process and try to hide as much of the language from the user as possible.

Much of the research on policy writing suggests that the process should be top down starting from the high level corporate objectives and gradually refining them into detailed low level functional policies. Whilst this is useful in deriving the objectives or policy specifications it is often useful to build the detailed functional description in a bottom up manner by looking at how to combine available data with supporting predicates. The original system data comes in the form of tables with the logical predicates performing selections, joining data sources and processing data to form new tables. In this way it is envisaged that a policy definition tool could allow the policy writer to select from the basic information tables and select from a set of operations that either generate new tables or act as binary predicates for the table as a whole. In such a way it is believed a simple low level policy writing tool very similar to database querying wizards could be created.

It is inevitably the case that although many services differ in detail and content they are often formed from common components and have very similar functionality. This should allow mechanisms to be created that lead an administrator through the service

creation process suggesting various service components and default policies. For example, in the web domain the structure of the web pages and easily be derived as well as the point where links are made to backend systems (such as databases and transaction servers). If these backend interfaces are done using standard methods (eg WebDB) it may be possible to derive more about the service structure. For a particular domain it should be possible to provide specialised predicates providing information useful in setting service policies as well as adaptable default policy examples to ease the authoring process.

5.4 Policy Evaluation

The policy evaluation mechanism is based on a Prolog system which is tightly coupled with the service model and information systems. Initially when a policy is set it is easy to prove the associated goal to ensure it is maintained. The policy evaluator will first find the elements in the domain, then follow through each of the pieces in the policy block checking the context is valid before proving each policy constraint. When there is a failure in proving one of these policy constraints then the service node to which it is associated is turned red showing that for that portion of the service there is a failure to meet the policy. Alternately if the service is working the node is turned green. The system could be extended to show levels of failures for certain types of policies.

The structure of the service graph means that a service's policies can be written in terms of the correct working of each of the decomposed sub-services as well as further system level policy constraints. This means that changes in the status of a service node must be passed up the service hierarchy and each policy should be re-evaluated until the status of the service remains unchanged.

The information system has an associated event system which is used to refresh system information thereby allowing the re-evaluation of policies as the state of the system changes. As information is associated with a service node the node registers its interest with the information system. The management provider is expected to handle all the relevant events, or if no event mechanism exists pole the source, and update the data table representation. In turn the data table tells all interested service nodes that their data has changed. These service nodes can then re-evaluate their policies; or just those pieces of the policies reliant on the changed data.

6. *Analysis*

The policy based monitoring system provides a mechanism for alerting the administrator to service failures and allowing them to examine data associated with various parts of the service. The system also allows an analysis script to walk over the service model, policies and information allowing more complex management questions to be addressed. It is a simple step to extend the policy evaluator so that other analysis programs can be executed allowing a wide range of functionalities to be plugged in to the management system.

This type of analysis mechanism could, for example, be used to support diagnosis tasks. The administrator is presented with a graphical view of the service and they can see when problems occur due to a colour change in a particular service node. This only indicates that there is a problem in delivering the service at that level and the

administrator needs further tools to identify the way in which service delivery is failing. A graphical tool can allow the administrator browse through the policies associated with that service. Alternately a diagnosis algorithm can be used to walk through the policies and find report on the policy failures and even initiate system level diagnosis (Baldwin *et al* 97) to pin-point the fault.

7. Discussion

Many system management tools provide mechanisms for collecting and filtering data and alerts but the information remains at the detailed level of the systems. With an increasing need to reduce management costs it is essential that management is related to the delivery of services which have real value. The management console described here aims to support service management by projecting information from the system view to an abstract user orientated view of the service. Ideally the management information should be derived from service design tools and available to a variety of management and support tools.

The work presented here starts by considering a users view of a service and how the overall structure of the service can be modelled using a simple hierarchical graph structure. This graph provides a skeleton onto which various pieces of management information can be associated. The use of ideas from policy based management allows each part of the service to be associated with simple statements describing the management requirements. These are implemented as logic statements describing valid states that the system can exhibit and still deliver that portion of the service. It is believed that the combination of policy and the service model provide a powerful abstraction for service level monitoring which could be expanded to other management tasks.

The downside of any flexible management system which aims to provide an abstract view of detailed system data is that considerable effort needs to be applied to developing the mapping between these views. The work described here has been based on ideas of using graphical structures to model a service and consideration has been given to easing the process of implementing low-level policies by abstracting basic data structures. Further work needs to be carried out both to develop ideas of policy authoring and to demonstrate that the trade off between configuring the management system and easing the longer-term usage.

The discussion has presented the tool as a management console providing a single view of a service. With a highly complex service or an organisation providing a large number of services it would be interesting to provide different limited views of the service to different administrators. This could be done along service lines so that each administrator is responsible for pieces of the service (ie nodes in the graph) and this could be expanded to a more federated system where pieces of a large service are provide by different organisations. An alternate breakdown could be based on management issues so that the policies are classified (eg security, performance, function) and each administrator would manage a speciality for a number of services.

9. Summary

A service management console has been described which projects system management data onto an abstract service view. This has been achieved using the idea of policy which at a high level should specify the management aims of individual or composite components of a service. The prototype uses a detailed policy constraint to allow detailed system information to be linked to the service view allowing the administrator to have simple visual cues showing service failures. An analysis mechanism also allows the administrator to carry out a detailed diagnosis, or other analysis as a first step to fixing the problem and maintaining service quality. It is believed that such a system provides the administrator with information highly related to the management requirements of the service they are running thereby emphasising service delivery whilst easing the routine monitoring tasks.

10. References

A. Baldwin, C. Bartolini, G. Di Vitantonio, K. Eshghi., (1997) "A Novel Algorithm for Fault Diagnosis in Internet-Based Services". *4th Workshop of the HP OVUA.* Madrid Spain.

R. Barruffi, E. Lamma, P. Mello, M. Milano, (1997) "Application of planning techniques for system configuration tasks". *4th Workshop of the HP OVUA.* Madrid Spain.

C. Goh. (1997) "A Generic approach to policy description in system management" *in Proceedings of the 8th IFIP/IEEE International workshop on distributed systems: Operations and Management.*

M. Sloman (1993) Specifying Policy for Management of Distributed Systems" *in Proceedings of the IFIP International workshop on distributed systems: Operations and Management.*

R. Wies (1994) Policies in Network and System Management – Formal Definition and Architecture–. Journal of Network and System Management. **2**(1).