

A Policy-based Approach to Composite Service Assurance for Ubiquitous Computing

Kevin Carey, Brian Cullen, Dave Lewis, Vincent Wade
Knowledge and Data Engineering Group, Computer Science, Trinity College Dublin
{Kevin.Carey, Brian.Cullen, Dave.Lewis, Vincent.Wade}@cs.tcd.ie

Abstract

The provision of services in ubiquitous computing environments, or smart spaces, presents a number of unique management challenges. One of these challenges is to assure that the quality of service perceived by smart space users can be managed to meet user and provider goals. Such assurance will be complicated by the fact that many smart space services are composed, possibly dynamically, of a number of other services each of which must be managed in a coordinated fashion to achieve the desired level of Service. This paper provides an overview of the issues involved in managing such composite services within a smart space environment. The paper then outlines a proposed architecture that facilitates Policy Based Management of these services. A case study that illustrates how such services can be used, and the utility of the management system in using them, is then presented. Finally results from preliminary work conducted in this area are presented and discussed.

1 Introduction

Ubiquitous computing environments, or smart space, offer users within them functionality that consists of an aggregation of functions within and external to the space. This provides a high potential for flexibility in how these functions are assembled and how they behave collectively to provide the use with the functionality they want with an desired quality, e.g. in terms of response time, reliability or other task-specific quality aspects. Such flexibility also gives rise to complexity which needs to be managed. In this paper we assume that the elemental units of functionality available in a smart space may be modeled as services. Assembly of such functionality, including integration with external services, then becomes a problem of service composition, upon which existing process-oriented techniques can be brought to bear.

However, though a variety of service compositions could be designed from the set of elemental service available in a smart space, the utility of the composed service is highly dependent on the preferences and permissions of individual users and the context of the task for which they are using the aggregate service. Many such dependencies will relate to non-functional aspects of the service composition. These will be highly dependent on the varying allocation of resources to services in the composition. In addition, where elemental services are developed for a broad, open market, we can expect to see similar service differentiating themselves through different level of configurability and quality of service control they offer. Therefore, the desired behavior of the composition may be best achieved by modifying the behavior of elemental services, rather than changing the process flow that makes up the composition. It seems, therefore, that there are limits to the benefits of designing service compositions in advance for a smart space to address all possible combinations of user preferences, permission task context and current resource allocations. In this paper we examine the use of policy rules for modifying the behavior of a composite service within a smart space. We aim to illuminate the level to which the behavior of a service composition can be controlled by policy rules specified at the level of the service as perceived by the user and by the operator of the smart spaces resources, and the issues of policy refinement and policy conflict this raises. Policy Based Management Systems (PBMS) have been applied successfully to quality of service for telecommunication networks [1] and IP networks [2], but this paper extends this work to investigate an architecture for realizing such dynamic assurance of composite services accessed from a smart space.

This paper first provides some background in Quality of Service (QoS) assurance, policy-based management, web-services and service composition, before outlining our approach to combining these techniques for service assurance in smart spaces. A case study is presented illuminating how the proposed policy-based approach may be applied and preliminary results and further work are outlined.

2 Quality of Service

The term Quality of Service is a broad one that is used to describe the overall experience a user will receive when accessing a service. Put more formally it can be defined as “the set of quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application” [3]. For the Service provider the ability to provide such guarantees can be a key differentiator in a competitive marketplace. Additionally the different service levels can be priced in a way that increases the average earning per customer. It is not surprising, therefore, that a number of different QoS mechanisms have been developed to help ensure that the performance of a service meets the requirements of the service customer.

Much of the initial work on QoS systems concentrated on providing guaranteed bandwidth and controlled delay for IP based services; real time services and multimedia services e.g. Voice of IP (VoIP), video on demand (VoD). However as these services continued to increase in complexity, it soon became obvious that guarantees at the network level are not always sufficient. For example it has been shown that even for a simple web based service more time may be required to process the request on the server than transporting the data over the network [4]. Such services illustrate the need for management systems that can provide QoS assurances at both the application and network level. By coordinating such guarantees these management systems can then provide true service level performance guarantees.

In the remainder of this chapter a number of different QoS systems will be introduced. To facilitate this discussion these systems have been classified according to the type of services for which they can provide guarantees, e.g. for network services, application services or composite services.

2.1 Network Oriented QoS Systems

The *Integrated Services Framework* (IntServ)[5], facilitates the provision of different service levels for network traffic on the Internet using a signaling protocol, e.g. the Resource Reservation Protocol (RSVP). Guaranteed service provides guarantees on a fixed delay for real-time applications while Controlled-Load service provides a reliable service with no firm quantitative guarantee and the default best-effort service deteriorate as the network load increases. IntServ framework has a few drawbacks, in particular, the fact that it doesn't scale efficiently. This is due to its overhead and high requirements on the routers, such as, RSVP, admission control, MF classification, and packet scheduling. All the devices on the path should support the same signaling protocol in order to enforce the desired guarantees. The presence of a non-IntServ device is possible but effectively renders any guarantees meaningless.

The *Differentiated Services Framework* (DiffServ)[6] was designed to address issues and difficulties encountered in implementing and deploying IntServ. DiffServ is more scalable than IntServ due to the fact that the amount of state information for the services is proportional to the number of classes and not the number of flows. However, each router must be aware of how to treat each class and as a network grows, it would become increasingly difficult to ensure that all the routers adhere to the same policy without the aid of a management system. Like IntServ, the presence of a non-DiffServ device is possible but renders any guarantees meaningless[2]. Unlike IntServ, DiffServ is much easier to deploy, with only small changes on a network since it uses a pre-existing IP header field for classifying the packets.

Another means of providing network QoS is by using *Multi Protocol Label Switching* (MPLS) [7]. MPLS works by adding an extra header onto each packet with a single 4 byte label. Routing is

achieved by assigning each packet to a Forward Equivalent Class (FEC) based on the 4 byte label. Processing packets in this way increased the speed at which packets are processed. Additionally, 3 bits of the 4 byte label can be used to indicate the Class of Service (COS) so that a DiffServ type mechanism can be implemented to provide QoS guarantees. MPLS traffic needs to be processed by a MPLS-enabled device and the removal of the MPLS header can be computationally expensive.

2.2 Application Server Oriented QoS

Many of the Application Server QoS system architectures, e.g. *OMEGA* [8], *QUARTZ* [9] and *ERDoS* [10], don't have any native QoS enforcement or monitor mechanisms. Instead, these architectures rely on existing underlying management mechanisms. The advantage is a lighter architecture that is capable of being deployed on top of certain existing management systems. Of course, the guarantee provided by the overall system is only as strong as those provided by these underlying mechanisms.

Most Application Server QoS systems provide support that allows the user(s) to specify their desired level of QoS, an abstract level that is related to the service in question. To facilitate this, these systems must also provide a method by which these abstract requirements can be translated or mapped to the low-level parameters that can be understood by the underlying QoS mechanisms. *OMEGA* uses a fixed specification of this translation providing just a one-to-one bi-directional mapping between the abstract and concrete parameters. However the architecture also allows streams to be bundled together or split. *QUARTZ* uses 3 layers of translation units, the Application Filter Layer, QoS Interpreter Layer and System Filter Layer. *QUARTZ* provides support for one-to-one, one-to-many and many-to-many parameter translations and, as with *ERDoS*, these mappings are bi-directional. The *ERDoS* architecture provides information models used to capture the logical and physical units of work required to achieve the task, specified using *Logical Application Stream Model (LASM)*, and the logical units are mapped to the physical entities that actually performs the task. In the event of QoS degradation or resource failure, *QUARTZ* will attempt to automatically reconfigure the service to compensate for the change, if that is possible, otherwise like *OMEGA*, the system will notify the user of the problem and he must initiate renegotiation. *ERDoS* provides support for application adaptation, alternative action specified by the user, and graceful degradation. Graceful degradation is controlled using a benefit function that allows user to specify how to evaluate the effectiveness of a certain level of QoS.

Whereas the architectures introduced so far are all predictive. *User Service Assistant (USA)*[11] is an attempt to produce a non-predictive QoS system. In this system, the user specifies their requirements at run time and if, at any stage, they are unhappy with QoS they can specify new requirements. This approach avoids many of the complexities that other systems must address, e.g. how to behave in the case of service degradation. However it does require the user to specify their requirements in rather low-level parameters and provides no automatic management of the service.

3 Policy Based Management Systems for QoS Assurance

As the complexity of Internet services continues to increase the complexity of their management systems, including those for QoS management, must also increase. The management of such complex and distributed systems presents a number of significant challenges. Most notable among these challenges is how to ensure that the operation of the system matches the overall objectives of those using it. It was to address this, and other, problems that *Policy Based Management Systems (PBMS)* were developed[15][16][17].

Within a PBMS the desired behavior of the managed system is specified in a policy. A policy may be defined as a definite goal or method of action to guide present and future decisions [18]. But in simple terms, it can be defined as a set of rules that express and reach a desired behavior [19]. Policies can be specified at many different levels of abstraction from high-level business goals to policies for individual resources. Ideally, it should be possible for an organization to derive their system policies from their overall business policies.

The chapter is split into two sections: policy languages and their importance within a PBMS, policy architectures and how a PBMS could be deployed on a network.

3.1 Policy Languages

Separating policy from the system's implementation provides a number of benefits. Most notably, it allows the behavior of a system to be dynamically altered without changing the system components themselves [20][21]. Additionally, policy can also be used to introduce an element of intelligent behavior into the system. However, it is important to note that a PBMS is only as useful as the policy language that it supports. A number of different languages [22][21][23][24] have been proposed each intended to address a specific management area or problem, e.g. security.

Although each policy language takes a different approach there is a common theme running through many of them. In general, a policy is comprised of a number of policy rules that control the behavior of the system they are applied to [19]. Each rule within a policy consists of an event, condition(s), and action(s) tuple.

Although there is no clear classification of the policy languages currently available, in general they either address Security or Resource Management. The ones that address Resource Management that is of interest here, since they can be used to specify policies that enforce QoS. One such language is *Ponder* [21], which supports the concepts of *Obligation* policies that are event triggered and define the activities the subjects must perform on objects in a target domain. These policies can be used to construct resource management policies that support QoS requirements. The IETF/DMTF [23] have also produced a policy language for this area, which has become widely accepted and is now the unofficial standard within the area of resource management. Due to this other languages, including *Ponder*, have defined a mapping to the IETF/DMTF language [18].

3.2 Policy Based Management Architectures

Once the policy has been specified the PBMS must provide a framework in which it can be enforced. The IETF/DMTF [25][26] have jointly developed an architecture for a PBMS, depicted below, which is the most widely accepted approach within the area of resource management.

Although the diagram below presents a number of different entities the two main elements are the *Policy Decision Point* (PDP) and the *Policy Enforcement Point* (PEP) [26]. The basic interaction between these components begins with the PEP, which identifies a situation that needs a policy decision. It sends a request to PDP asking it what action to take. The PDP, which has access to the policies, will send back a policy decision to be enforced by PEP. PDP is likely to store its policies in a repository, such as a *Lightweight Directory Access Protocol* (LDAP) [20]. *Common Open Policy Service* (COPS) [27] protocol was developed by IETF to facilitate the interaction between PDP and PEP. In more complex systems, it is likely to encounter multiple PEPs and even a hierarchy of PDPs in order to have decision making as local as possible to reduce overhead. Using this surprisingly simple framework it is possible to construct very complex and powerful PBMS. Typically, these PBMS have provided QoS support at the network level [28][29][30][31] although some examples of systems that can operate at the application level are also available [14].

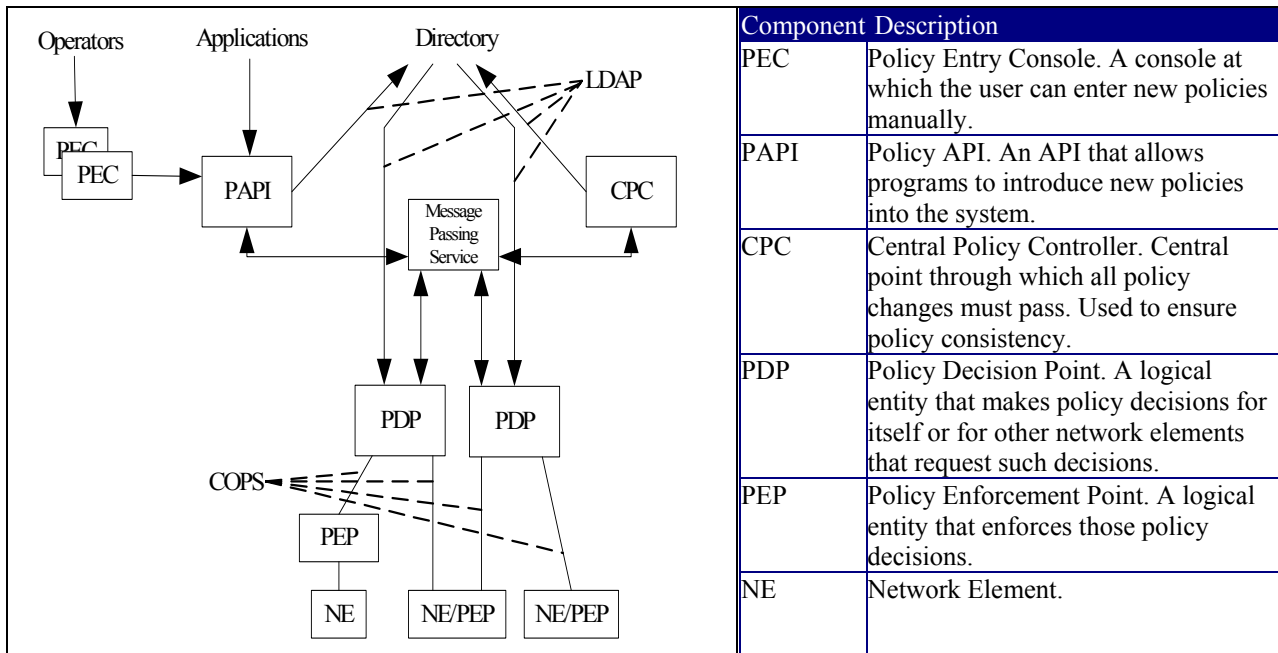


Figure 3.1. IETF/DMTF Policy Based Management System Architecture

4 Service Composition and Software Components

We aim to address the QoS assurance of services that are built from complex networks of systems, administered by different organizations and composed of software components sourced from different developers. In order to focus clearly on the relationship between the QoS assurance required for the overall service and the QoS assurance possible from the participating systems and components, a service oriented modeling approach is taken. Here, each of the contributing components or systems is modeled as a well-defined service, which is the means by which their functionality is composed to define the overall service. Such a composition of services provides the well-defined context required for addressing QoS assurance in complex, multi-organizational environments.

Service composition is the orchestration of a number of existing services to provide a richer composite service assembled to meet some user requirement in the form of a new service. A *composite service* is made up of several *constituent services* that are invoked according to some process flow model. Constituent services may be either *atomic services*, which cannot be subdivided, or other composite services. There are several languages that take a service-oriented approach to integration, however the *Web Service Description Language (WSDL)* is a W3C standard for defining services offered over the WWW [32]. WSDL web service definitions can be bound to different protocols, for instance SOAP [33], and advertised via UDDI directories [34].

While WSDL/UDDI/SOAP allows for the description, location and invocation of individual services in an interoperable manner, they do not support the description and enactment of service compositions. A range of web service composition languages has been proposed, e.g. BPEL4WS, PBML etc. However, as pointed out in [35], these offerings seem primarily bound to existing platforms and associated commercial interests and do not exploit the formal process semantics offered by Petri nets and process algebras. In a smart space environment, where service composition will need to be performed in an autonomous and dynamic way, such semantics are essential. In addition, these composition languages don't address the non-functional aspects such as reliability and response times that are key to QoS assurance.

The DARPA Agent Mark-up Language initiative is addressing the need for automated service composition by taking an ontology based approach to defining services and service compositions called DAML-S [36][37]. DAML-S is bound to WSDL, however, it allows for a richer expression of the semantics of input and output information than is possible with the XML scheme information

representation used in WSDL given that the service inputs and outputs operations as well as the pre-conditions to using the service and the effects of using the service are defined in terms of ontology elements. DAML-S provides some common semantic related to the resources used by a service and how they may be shared. This is an important abstraction in managing the assurance of a service, since what the QoS service is able to deliver, will depend on the sharing of such resources. DAML-S thus allows for automated reasoning about the compositions in which the service may be involved without the service designer having to explicitly address the requirements of that composition. This is needed in smart spaces where composite services must be formed from several services with no a priori knowledge of each other [38].

Service-oriented architecture such as those for web services and DAML-S do not address the structure of the software that implements atomic and composite services. However, there is a clear linkage with component-oriented software engineering techniques where atomic services can be implemented by component interfaces, as well as addressing the need for well defined interfaces (i.e. services), component-oriented architectures address the need for portability, configuration and administration of platform resources, typically through a set of common APIs, collectively know as a container. Application servers implement container APIs thus allowing third party components to be deployed and run without modification.

Our approach to assurance of composite services encompasses both the service-oriented view of service composition and a component-oriented view of how atomic services can be implemented. We assume the enactment of the composite service is through some decentralized workflow-like mechanism as described in [39]. A *semantic service description* is a DAML-S document bound to a WSDL service description. Several semantic service descriptions can be bound to the same WSDL service (and vice versa, though this case is not explored here). In addressing service composition, we assume both atomic service and composite services are described as semantic services.

Each particular atomic service is implemented as an interface to one or more components, potentially produced by completely separate developers. Though the WSDL service description would be the same (i.e. the action performed by the service), the semantic service description of these different implementations may differ, for example, use of different ontology terms. An instance of an atomic service is provided by an instance of a component implementation deployed and running on a container platform.

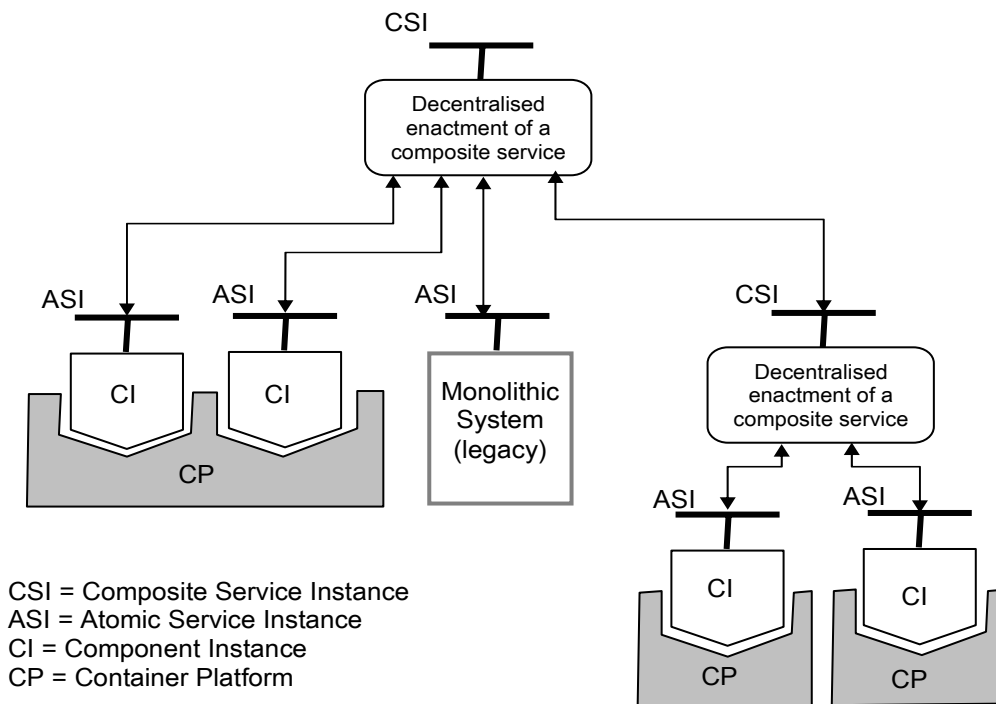


Figure 4.1. Composition of a Composite Service

The automatic service composition provides a way to implementing dynamic service adaptivity, especially in situations where a wide range of possible constituent services is available from which new composite service requirements can be met. However, in the following we will not pursue service composition as an adaptive technique, but examine the use of policy-based management as an adaptive mechanism *within* a given service composition. We assume that policy-based management is better suited to the adaptation of non-functional behavior involved in QoS assurance, and that the context provided by a service composition aids in the refinement of policies from the composite service level to the constituent service level.

5 A Policy Based Architecture for Composite Service Assurance

This section proposes a policy-based approach to supporting highly configurable adaptive behavior of composite services. The approach taken involves binding a policy management service to every composite and atomic service, which in this context are referred to as operational services. Each policy management service will expose a set of policy events, conditions and actions that relate to the behavior of the operational service and of the underlying component. This is accompanied by rules (meta-policies) that constrain which combinations of events, conditions and actions can be combined to form valid policy rules that influence the behavior of the operational service and the component that implements it. These constraint rules can also specify who is able to set which types of policy rules. The set of events, conditions, actions and policy-constraint rules form a policy language for that operational service. A distinction is made between policy rules that impact on the behavior of the service as observed through its operational interface and those that impact on the behavior of the underlying implementation of the operational service without impacting on the observed behavior of the service. An example of the latter is a policy rule that controls the logging of operational service events where that service offers no control or visibility of logging activities.

Thus a policy can be executed on a service, for example, to either change the level of monitoring of that service, or suggest changes in the service's behavior based on context or environmental considerations. For atomic services, policies submitted to the policy management service are enacted within the underlying component. This may involve transforming the policy to declarative rules native to the component's implementation, including rules used to control a component's interaction with its container, e.g. controlling thread allocation, transactional behavior etc.

For composite services, policies applied to the policy management service are transformed either into changes to conditional parts of the service composition's control and data flow, or into policies that are applied to the constituent services. The approach to executing policies passed to the constituent service will then depend on whether that service is itself atomic or composite. Note at this stage we make no assumption as to the way in which the composite service is implemented i.e. as an agent, process/workflow manager, script or application component. However, the architecture assumes that any *management* interaction between the component which implements the composite service and the services, which make up that composite service are via Policies executed on those constituent services. Thus the composite service component executes management policies on the constituent services, which are lower in the composition hierarchy.

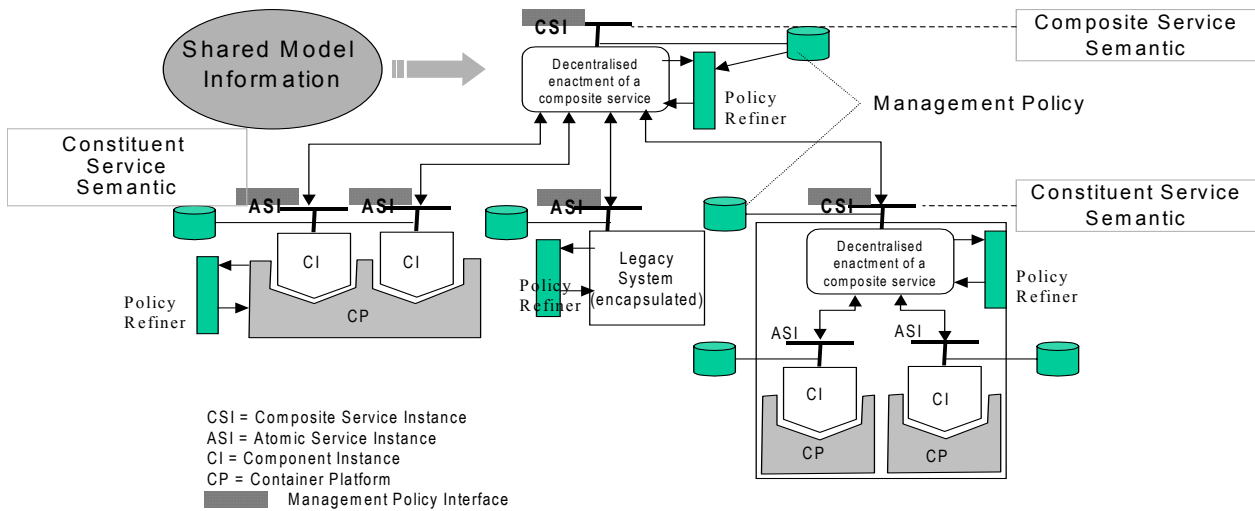


Figure 5.1. Required Knowledge and Information For Policy Refinement

In order to provide this policy refinement from Composite Services of their elemental atomic services, certain shared information and knowledge needs to be made available and commonly understood. For a specific semantic description of a semantic service this can be obtained from the aggregation of the ontology terms used in the definitions of the inputs, outputs, preconditions and effects of the composite and constituent services. Further links between these terms, additional to ones existing in their source ontologies, may potentially be derived from the process model for the composite service and the links it represents between the composite and constituent service descriptions. However, as the constituent services may be sources from a number of disparate sources, they may use ontologies in their service descriptions that are themselves from very different sources. This presents a barrier both to the use of these service semantics in the initial service composition and in the emergence of a coherent shared information model from the composite service model. We have performed some orthogonal research that indicates that automated mapping between ontologies used in separately sourced service description is possible given the use of a certain set of core semantics. The generation of shared information models to support reasoning about policy refinement therefore needs some level of commonality in the ontologies used for service descriptions. The level of this commonality is an open issues and beyond the scope of this paper. As the developers of constituent services also provide the policy vocabulary for that service, it is likely that the two specifications will share some semantics. However, as the policy vocabulary bound to a service needs to support service assurance tasks many of which will be similar regardless of the service provided, e.g. logging, security audits. Therefore, there is a higher chance that common semantics could be used in the definition of policy semantics, for instance based on the DMTF's CIM ontology, the TeleManagement Forum's Shared Information Model or generic management functions from OSI management. In the long term, the utility of common concepts in a service management policy may be reflected by increasing commonality in the descriptions of the services themselves, as service design and management design become better integrated.

6 Case Study

In this section, a case study of how the proposed system might work is presented. This is done to describe the context in which such a system could function. Additionally it provides a means by which the opportunities, challenges and benefits provided by producing the proposed system can be identified and expanded upon. A final benefit of producing this case study at this point is that it provides the criteria with which an evaluation of the system could be conducted at a later stage.

6.1 Case Study Overview

The scenario for our case study occurs in a university environment. A lecturer, Mark Power, is going to teach a one hour tutorial to a class of first year students in a lecture theatre capable of delivering smart space services. The tutorial presentation has already been prepared prior to the class to be delivered via the smart space services. At the end of the class, Mark always has a 10 minute question and answer session where he also hands out a summary of the material presented in the tutorial.

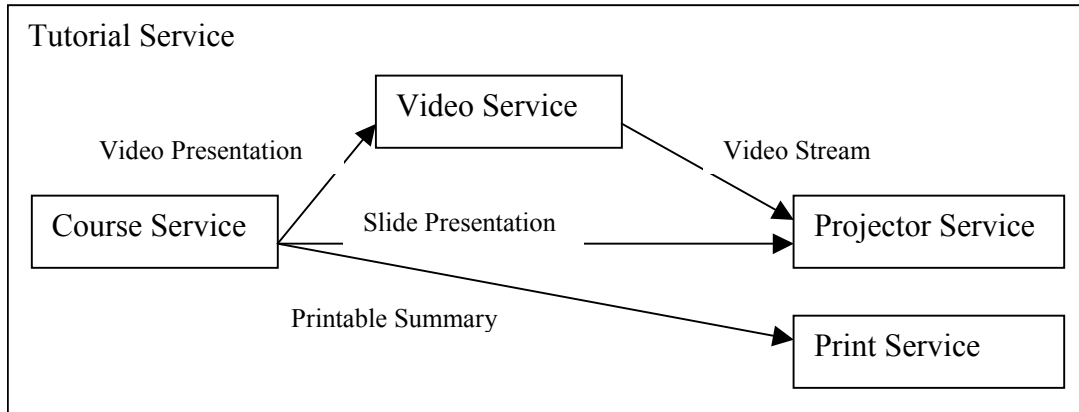
Mark starts the tutorial by accessing the list of smart space services available in the theatre and selecting the Tutorial Service, giving it the name of the pre-prepared material. Ordinarily this service presents the tutorial material for the entire duration of the tutorial as well as printing a copy of the presentation for each student in the course. Today, however, Mark wants to show a 20 minutes video as well as the slide presentation, yet still maintaining the 10 minutes session of question and answer at the end. He also wants the service to print a summary of the material presented for the students during this question and answer session as the printer in the room can be too noisy for use while lecturing.

The Tutorial Service is an instantiation of a more general service that provides access to the various resources (physical and virtual) associated with the lecture theatre. The basic service offered in the theatre is generic with a simple user interface so that it can accommodate the lecturer's requests with ease. However the basic service has a management interface that accepts management policies allowing the service's behavior to be modified, the service tasks to be monitored and assured with the use of these management policies. This allows the user to customize the service to his preference and have a guarantee that the service will meet his requirements and requests. So, when Mark opens his set of management policies and applies the following policies to this basic service:

- Presentation material is located in my office personal computer;
- Present course material based on course rules;
- Slide presentation must be shown in good quality;
- Give 10 minutes at the end for question and answer;
- Print a summary of the course material presented to the students present;
- Printing must not interfere with the slide or video presentation but must be completed no later than the end of the lecture;
- Show a video in this tutorial session preferably at the end;
- Video presentation must be shown in presentable quality;

6.2 Case Study Analysis

The Tutorial Service is in fact a composite service and it is composed of a Course Service that retrieves and formats the lecture's presentation material to the necessary format upon request; a Projector Service that presents the visual and audio lecture's presentation material, such as slides or video; a Video Service that streams the lecture's material in case of a video presentation, allocating bandwidth on the network for such stream; and a Print Service that prints in the lecture theatre a printable version of the lecture's presentation material.



The Figure 6.1. The Composition of the Tutorial Composite Service

The Composite Service has a *Policy Refinement Module* (PRM) attached to it and it is this that accepts the user’s management policies, which would be a set of high-level policies. The Composite Service PRM processes these high level abstract policies and retrieves the relevant policies from a policy repository for each of the constituent services and it transmits to their PRM. These PRMs attached to the constituent services have the same duty of refining the policies they receive. In the case of an atomic service, the PRM receives the policies and translates to parameters understood by the atomic service.

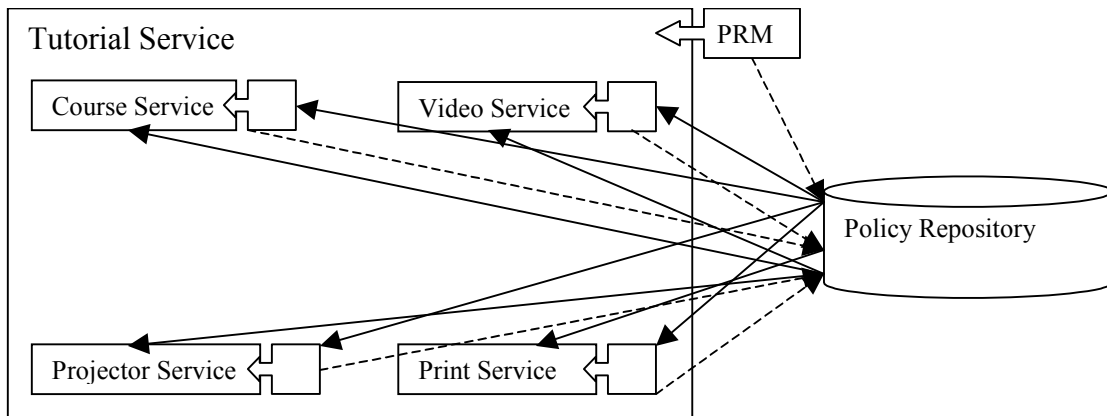


Figure 6.2. Flow of Policy refinement in the Composite Service.

Hence, the lecturer’s management policies are sent to the PRM of the composite Tutorial Service and after being processed, appropriate policies are retrieved from a policy repository and transmitted to the PRM of the constituent services. The Course Services PRM receives the following policies:

- Retrieve presentation material from lecturer’s office pc;
- Tutorial presentation duration is 50 minutes;
- Show video after the slide presentation;
- The entire video must be shown in one tutorial;
- Show slide presentation from where it was interrupted;
- Slide presentation can be interrupted;
- Provide printable summary of the slide presentation at the end of the tutorial;
- If slide presentation delivered then generate printable summary;

The Course Services PRM processes these policies and retrieves extra information such as the video duration is 20 minutes. Using this information, PRM calculates that it only has 30 minutes to show the slide presentation before the video presentation. It also composes a summary of the presentation given and formats it into a printable document, which is then sent to the Print Service

after the video presentation. The PRM retrieves the following policies from the repository for the Video Service to realize this task:

- Set course material location to \\MarkPowerOfficePC\Presentations;
- Set course material index to chapter 3, page 4 (last slide presented in the previous tutorial);
- Set course material format to slide presentation;
- Create an event to trigger after 29 minutes to indicate last slide and stop slide presentation;
- Create an event to trigger after 30 minutes to send video presentation to Video Service;
- When slide presentation is finished, generate a summary of the slide presentation shown;
- When video presentation is finished, send summary as printable document to Print Service;

The Video Services PRM receives the following policies:

- Stream video content with maximum quality of videoCD quality;
- Economize on the use of bandwidth;
- Allocate enough guaranteed bandwidth for video stream;

After checking the resolution of the video source (352x288 @ 29.97fps), the PRM decides to stream this video presentation at this resolution since it is less than videoCD but at 20 fps in order to economize on the bandwidth without deteriorating the quality too much. PRM retrieves the following policies from the repository for the Video Service to accomplish this task:

- Setup IntServ RSVP path between Video Service and Projector Service (in the theatre);
- Allocate a IntServ guaranteed class bandwidth for the stream;
- Stream video content at 20 fps and keep the video resolution at 352x288;

The Projector Service's PRM receives these policies:

- Modify projector to display the content received;
- Provide lecturer with navigation control;
- Notify lecturer when end of content;
- Present content as received but it must be in presentable quality;

The PRM refines these policies into parameters understandable by the Projector Service:

- If slide content, set mode to slide mode;
- If slide mode, set control with navigation buttons: 'next', 'previous' and 'end';
- If video content, set mode to video mode and lecture theatre light to low;
- If video mode, set projector resolution to 800x600;
- If video mode, set control with navigation buttons: 'play', 'pause' and 'stop';
- If end of video or last slide, warn lecturer with a beep;

The Print Services PRM receives the following policies:

- Print start when receives document and must finish at the end of the tutorial;
- Print speed must be set to maximum where quality can be less than normal if necessary;
- Number of copies must be the same as the students in the lecture theatre;

When the print service receives the printable document, the Print Service's PRM, using these policies, calculates that there is only 10 minutes to accomplish the job, 80 students present based on the lecture theatre sensor, it is a 8-page summary document and the printer can print at the maximum speed of 32 sheets per second in black and white at 100 dpi. Using this information the PRM deduces that it will take 20 minutes, so in order to complete this task in the time allowed, the Print Service must in addition be modified to print 2 pages per sheet. Hence it retrieves the following parameters from the repository and sends them to the Print Service:

- Set print mode to black and white;
- Set print speed to 32 pages per minute at resolution of 100 dpi;
- Set print copies to 80 and scale to 2 pages per sheet;

As we have seen in this case study that the use of management policies together with Policy Refinement Modules, gives the user an ability to modify the behavior of the composite service and providing an assurance that his criteria will be met when performing his request. Using policy refinement, it allows the user to specify its criteria in a high-level policy language, which is

translated to low-level policies that can be understood by the atomic service giving the user a QoS guarantee for the service received.

7 Results

Work has already been conducted in this area and a prototype management system capable of refining high-level Service Level Agreements (SLA) into low-level policies. This management system concerned itself with providing assurance support for composite services. Similar to the architecture proposed in this paper the prototype system supported the concept of distributed policy refinement modules (PRM), attached to each component of the composite service.

The prototype system functioned by taking an XML SLA, which defined in composite service specific terms the desired level of service, and deducing the policies necessary to enforce the SLA. Such deduction is possible through the SLA language terms being bound in advance by the composite service provider to monitorable parameters at the constituent services. The PRM is, therefore, able to generate the policies that controls how each constituent service gathers and reports the parameters relevant to the SLA being assured. From this work it was possible to identify a number of issues that must be addressed in the production of the architecture outlined in this paper. Of particular importance is the information necessary to perform this policy refinement. Within the prototype system the information identified as necessary is presented below.

- What is the composition of the composite service?
 - i. What services constitute the composite service?
 - ii. Where are there services located?
- What policies/management interfaces are supported by each constituent service?
- What policies should the composite service support?
- How do the overall system policies relate to the policies at each constituent service?

This information provides a basis from which management of a composite service can begin. Within the prototype system this information is modeled using extension to the DMTF CIM and has been shown to be sufficient for QoS monitoring of a composite service. However, for the case study outline in this paper it will be necessary to capture additional information and, in particular, information about how the composite service functions. This is necessary to allow the management to fully model the affect that managing each constituent service will have on the composite service.

Another important concept that was validated by the prototype was the way in which policies, which were not explicitly specified, could be inferred from the SLA. In particular due to the distributed nature of composite services the prototype has to support the creation and deployment of supporting policies. These policies were necessary due to support the evaluation of other policies and the algorithms used in their identification provides a basis from which further policy refinement algorithms can be developed.

8 Conclusions and Further Work

From the assumption that functionality available to users in a smart space may be modeled and offered as a service, we observe that the problem of integrating functionality from within and external to a smart space becomes one of ad hoc service composition. Research into the application of web services in the Semantic Web points to the use of ontology-based semantics as a promising approach to such dynamic service composition. Here we examine how quality of service assurance can be managed for such composite services, assuming their semantic service descriptions are available.

We propose that policy-based management techniques be used to provide such service assurance, by mapping the user's quality of service policies at the level of the composite service to policies applied to the constituent services. We presented an outline architecture showing how policy evaluation can be integrated into the infrastructure implementing services and how policy languages can be related to service descriptions. This approach raises the need for some form of policy refinement, a problem that remains largely unresolved by the policy-based management

community. Our initial results show that policies for constituent services can be generated automatically from SLA parameters for the enclosing composite service, but only when there is a high degree of shared semantics between the composite and constituent services, in this case provided by the DMTF's CIM ontology. Our future work is to examine how the expression of the relationship between the semantics of a composite service and its constituent service present in a DAML-S composite service description can be exploited in automated policy refinement between composite service assurance policies and constituent service assurance policies. This will involve extracting ontological information present in service inputs, outputs, pre-conditions and effects of composite and constituent services and the relationships between them as captured in the expression of the composition's process model. This will provide a form of shared information model, the efficacy of which in refining assurance policies will be examined.

Overall, this approach has the potential to support user and provider assurance policies for composite services as well as providing a framework for integrating the policy-based assurance capabilities of services from diverse sources – both important requirements for smart space service assurance.

9 Acknowledgements

This work was partially funded by the EU funded project FORM (IST-1999-10357) and the Irish Higher Education Authority under the M-Zones programme. The authors would like to thank Owen Conlan and other members of the M-Zones team for their useful comments.

10 References

- [1] Oodan, A. P., Ward, K. E., and Mullee, A. W., *Quality of Service in Telecommunications* Institution of Electrical Engineers, 1997.
- [2] Xiao, X. and Ni, L. M., "Internet QoS: A Big Picture," *IEEE Network*, vol. 13, no. 2, pp. 8-18, Mar.1999.
- [3] Wroclawski, J. The Use of RSVP with IETF Integrated Services. 1997.
- [4] Bray, T., Paoli, J., and Sperberg-McQueen, C. M. Extensible Markup Language (XML) 1.0. 1998.
- [5] Campbell, A., Coulson, G., and Hutchison, D., "A Quality of Service Architecture," *ACM Computer Communications Review*, vol. 24, no. 2, pp. 6-27, 1994.
- [6] Brickley, D. and Guha, R., V. Resource Description Framework (RDF) Schema Specification. 1999.
- [7] Nua, I. S. How Many Online Worldwide.
- [8] QoS Protocols & Architectures. 1999.
- [9] Siqueira, Frank, "Quartz: A QoS Architecture for Open Systems." Trinity College Dublin, 1999.
- [10] Coffman, K. G. and Odlyzko, A. M., "The Size and Growth Rate of the Internet," Oct.1998.
- [11] Landfeldt, B., Seneviratne, A., Diot, C. User Service Assistant: An End-to-End Reactive QoS Architecture 1998.
- [12] Bhoj, P., Singhal, S., and Chutani, S. SLA Management in Federated Environments. Sloman, Morris and Mazmdar, Subrata. 293-308. 1999. IEEE Publishing. Integrated Network Management VI.
- [13] Fauvet, M. C., Dumas, M., Benatallah, B., Paik, H. Peer-to-Peer Traced Execution of Composite Services 2001.
- [14] Baldwin, A. and Casassa Mont, M. Policy Based Monitoring of a Web-Based Service. HPL-98-76. 1998.
- [15] Lutfiyya, H. L., Bauer, M. A., Marshall, A. D., and Stokes, D. K. A Policy-Driven Approach to Availability and Performance Management in Distributed Systems. 27-8-1997.
- [16] Mont, M. C., Bladwin, A., and Goh, C. POWER Prototype: Towards Integrated Policy-Based Management. HPL-1999-126. 18-10-1999.
- [17] Wies, R., "Policies in Network and Systems Management - Formal Definition and Architecture," *Journal of Networks and Systems Management*, vol. 2, no. 1, pp. 63-83, Mar.1994.
- [18] Alcantara, O. D. and Sloman, M. QoS Policy Specification A mapping from Ponder to the IETF. 2003.
- [19] Damianou, N., Bandara, A. K., Sloman, M., and Lupu, E. C. A Survey of Policy Specification Approaches 2002.
- [20] Dulay, N., Lupu, E., Sloman, M., and Damianou, N. A policy deployment model for the Ponder language. 2001.
- [21] Damianou, N., Dulay, N., Lupu, E. C., and Sloman, M. The Ponder Policy Specification Language. 29-1-2001. Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks.
- [22] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and Terpstra, M. Routing Policy Specification Language (RPSL). 1999.
- [23] DMTF. CIM Policy Model White Paper. DSP0108. 14-3-2002.
- [24] Lobo, J., Bhatia, R., and Naqvi, S. A Policy Description Language. 2003. Proc. of AAAI. 1999.
- [25] Gai, S., Strassner, J., Durham, D., Herzog, S., Mahon, H., and Reichmeyer, F. QoS Policy Framework Architecture. 1999.

A Policy-based Approach to Composite Service Assurance for Ubiquitous Computing

- [26] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. Terminology for Policy-Based Management. 2001.
- [27] Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., and Sastry, A. The COPS Protocol. 2002.
- [28] Fernandez, M. P., Pedroza, A. d. C. P., and Ferrerira de Rezende, J. QoS Provisioning across a DiffServ Domain using Policy-Based Management. 2001. Globecom 2001. 2001.
- [29] Flegkas, P., Trimintzios, P., Pavlou, G., and Liotta, A. Design and Implementation of a Policy-based Resource Management Architecture. 2003. Proceedings of IEEE/IFIP Integrated Management Symposium (IM'2003).
- [30] Rajan, R., Verma, D., Kamat, S., Felstaine, E., and Herzog, S., "A Policy Framework for Integrated and Differentiated Services in the Internet," *IEEE Network*, vol. 13, no. 5, pp. 36-41, 1999.
- [31] Lymberopoulos, L., Lupu, E. C., and Sloman, M. An Adaptive Policy Based Management Framework for Differentiated Services Networks . 2003. Policy 2002.
- [32] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. Web Services Description Language (WSDL) 1.1. 15-3-2001. W3C.
- [33] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Frystyk Nielsen, H., Thatte, S., and Winer, D. Simple Object Access Protocol (SOAP) 1.1. 8-5-2000. W3C.
- [34] UDDI. UDDI Technical White Paper. 6-9-2000.
- [35] Aalst, V. d. Don't go with the flow: Web service composition standards exposed 2003 IEEE Intelligent Systems.
- [36] DAML-S. DAML-S: Semantic Markup for Web Services. 2002.
- [37] McIlraith, S. A., Son, T. C., and Honglei Zeng, H. Semantic Web Services. 2001. IEEE Intelligent Systems.
- [38] Chakraborty, D., Perich, F., Joshi, A., Finin, T., and Yesha, Y. A Reactive Service Composition Architecture for Pervasive Computing Environments. 2002. Computer Science and Electrical Engineering, University of Maryland Baltimore County.
- [39] Benatallah, B., Dumas, M., and Sheng, Q. Z. N. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. 2002 (ICDE'02).