In: *Management Solutions for the New Communications World: NOMS 2002 Proceedings of the 2002 8th International IEEE/IFIP Network Operations and Management Symposium , Florence, Italy*
IEEE Publishing, April 2002

Using Policy–Based Concepts to Provide Service Oriented Accounting Management

I. Radisic
Munich Network Management Team
University of Munich
Oettingenstr. 67
D-80538 Munich, Germany
Phone: +49-89-2180-9167, Fax: -999167
radisic@informatik.uni-muenchen.de

Abstract

With service orientation gaining more importance over the last couple of years, new requirements have evolved which are posed on accounting management solutions. In brief, accounting management systems are required to support a wide–range customization of the provided accounting functionality according to customer needs. This includes among others the support of complex–structured, usage–based, customer–individual tariffs as well as the support of increasing dynamics due to change management activities. All in all, an accounting management system is needed which is able to handle the complete service life cycle regarding accounting in a customer– and service–oriented way. Additionally, from a provider's point of view a high automation of management tasks is demanded.
This paper analyzes accounting–relevant management processes and presents a management solution based on policy–based concepts to overcome the shortcomings of previously developed solutions.

Keywords

Accounting Management, Policies, Service Management, Process Orientation

1 Introduction

Over the last couple of years enterprises experience an increasing dependence on their network infrastructure and their networked systems: The economic success of a company is directly linked with the availability and performance of the services provided by the networked systems. As the complexity of services is also increasing, many companies decide to concentrate on their core business and hand over the operation of their IT infrastructure to third parties (so called service providers). Besides providing network connectivity, service providers also offer the operation of application services like WWW, Email, DNS, business applications, etc.

This fast–growing trend of *outsourcing of services* and service orientation in general has an immense impact on accounting management with new requirements coming up, e.g.: Usage–based and QoS–dependent charging is replacing flat–fee based approaches; service oriented accounting units reflecting service functionality are needed (e.g., transactions) to replace simple measurable accounting units (e.g., transmitted bytes); dynamic accounting systems which are adaptable during service operation should be used instead of static accounting systems. In consequence, a *flexible* accounting *management* system is required which is able to direct accounting systems with regard to changing needs evolving during service provisioning.

Additionally, todays accounting systems are implemented by various distributed components. Usually, charging and billing is carried out by so called billing systems that require information about service usage to calculate the total amount that has to be charged. Therefore, special metering software is used that delivers detailed data about service usage. Unfortunately, neither standard interfaces nor data formats have been adopted yet. Thus, communication between the various accounting components is still established by supporting proprietary interfaces or implementing gateways. Even worse, no standard management interfaces to control accounting systems have been established so far. As a consequence, introducing a new accounting system component into a provider's infrastructure today means to implement parts of the accounting management from scratch. Thus, accounting management becomes a time–consuming and error–prone task. Therefore, an *integrated* accounting management system is needed to be able to direct the various participating accounting components in a uniform way.

To be able to handle the aforementioned increasing complexity, the management tasks regarding accounting management have to be *automated* as far as possible. This allows to relieve administrators of repetitive management activities.

To sum up, a highly integrated, flexible and automated accounting management is required to accomplish the needs rising from the trend of service outsourcing. In this paper we propose the application of policy–based concepts on accounting management along with the complete service life cycle to realize the required management functionality. Using *policies* to direct the components involved in accounting a service allows to abstract from the distributed components realizing the accounting system in all life cycle phases. Hence, an accounting manager has a uniform view on accounting management and the requirement of integration is fulfilled. Additionally, the flexibility of the accounting management system is directly dependent on the flexibility of the policy language. Thus, the requirement of flexibility is fulfilled as long as the policy language is flexible in expressing the needed management actions. To achieve a higher degree of automation in change management regarding service accounting we additionally propose the application of *meta–policies* which control activation, deactivation, creation and deletion of accounting policies.

The remainder of the paper is organized as follows: In Section 2 a typical scenario is described to introduce the environment we focus on in this paper. In Section 3 we discuss related work. In order to investigate the required dynamic aspects of accounting management in outsourcing scenarios, we analyze in Section 4 interactions taking place and processes involved. Furthermore, this analysis is used to determine basic accounting entities. Afterwards, in Section 5 we present the architecture of our management solution which uses policy–based concepts to overcome the shortcomings of solutions developed so far. Finally, Section 6 concludes the paper by presenting open issues and further work.

2 Scenario

In this Section we are presenting a snapshot of an outsourcing scenario (see also Fig. 1) from one of our cooperation partners in order to demonstrate problems that typically arise regarding todays accounting management.

A large german service provider (SP) operates a so–called Intranet Extranet Service Area (IESA) for an international assurance company. The IESA provides a connectivity service to the assurance company's Intranet Service Area (ISA) on the one side to the Internet on the other. Additionally, DNS and email service are provided to the assurance company. Assurance agencies and their employees that get in contact with the end customer (assurance holder) are the actual users of the IESA: they download up–to–

date assurance forms, use specific assurance applications to calculate contracts from the ISA site, use an email account to get in contact with other agencies, and in fact use the Internet for non–business applications. On site of the assurance agencies ISDN dial–up routers are installed to establish the connection to the IESA. The assurance company demands a usage–based charging for the IESA, where charges for the ISA usage have to be separated from the charges for internet usage. The intention is that the assurance company is able to recover the costs for “private”, non–business usage of the IESA from their agencies resp. employees.

To be able to assign metered usage data to specific employees and agencies the SP needs up–to–date information about all possible IESA users. Therefore, a user data rollout between the assurance company, which is aware of all possible users, and the provider has to be done frequently. To separate business from non–business IESA usage, usage data is collected from various sources: from a WWW proxy server, from several Dial–In Authentication Servers and from routers which establish the connection to the assurance company’s ISA. WWW proxy usage data is always regarded as private usage whereas the remaining usage data is regarded as business usage. Afterwards, the usage data collector assigns the metered data to an accounting user ID and additionally decides whether usage data is cumulated to agencies or to single employees.

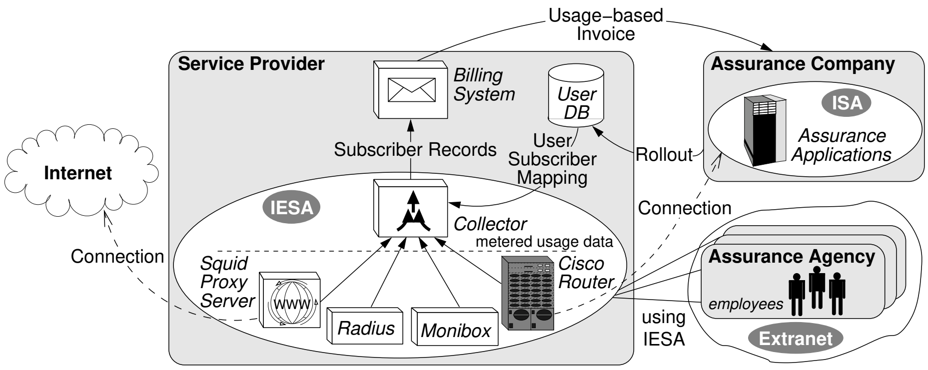

Figure 1: Scenario

As previously mentioned in Section 1, standard interfaces have not been adopted by accounting components so far. Thus, in this case the SP had to implement various gateways and data translators between the components to set up the required accounting system. As a matter of fact, the high dynamics inside the assurance company resulted into several challenges regarding accounting management the SP had to cope with. For instance, the assurance company had participated in an enterprise fusion with another company which resulted into a new company structure. The customer demanded a varying tariff for the newly added agencies, so that new user groups had to be configured. Furthermore, the company demanded a QoS–dependent tariff for which new meters on client site had to be installed (one per Dial–Up router). Consequently, the collector software had to be reconfigured and new gateways had to be implemented. Additionally, every time a new user was introduced the corresponding meter had to be reconfigured. This becomes especially a costly and time–consuming task when automated tool support is missing and one has to deal with over 10.000 agencies and about 40.000 potential users. Moreover, the assurance company also demands customer–individual billing as well as hot–billing and –reporting for which the billing system needs to be exchanged. Plans for the future are to add new services to the IESA for which varying tariffs are demanded, too. Overall, due to the high dynamics resulting from change management and the missing support of the management tools, the SP had to partly reimplement the accounting system every time something changed that was relevant for accounting management. Besides, as automated tool–support is missing, accounting management became a complex and time–consuming task. Therefore, the SP demands means for accounting management which are able to cope with the high dynamics in todays outsourcing scenarios by considering the complete service life cycle and providing an integrated, adaptable view on accounting management.

3 Related Work

All known work done by standardization organizations as well as within the research community concentrates on problems of one particular accounting sub–process or life cycle phase without taking the remaining sub–processes in other phases into account. Thus, the required integrated, uniform view on accounting management as a whole is still missing as well as the support of the dynamic aspects of accounting management.

Accounting Management is one of OSI's Systems Management Functional Areas and the appropriate standardization document [11] is specifying an accounting management architecture by introducing a usage metering, a charging and a billing component. The usage metering component is the only one which is investigated in more detail and for which managed objects are specified. The remaining parts are left for further study. TMN is adapting OSI's FCAPS for telecommunication services and mentions in [12] some relevant accounting *function sets*. For most of them specific management functions are not specified at all, while realization concerns are completely left out of consideration. In contrast, TINA-C presents in [6] a detailed description of required accounting management interactions. However, these are restricted to TINA–conform telecommunication services and management architectures and thus are not applicable to outsourcing scenarios in the focus of this paper. The IETF working group *Authentication, Authorization and Accounting (AAA)* has also adapted OSI's accounting architecture for IP (connectivity) services [1]. The main focus is centering around usage metering by specifying accounting protocols as well as accounting data formats. Furthermore, the non–profit organization IPDR is standardizing the so called *Internet Protocol Detailed Record (IPDR)* [10] and additionally a proper accounting protocol.

The research community has mainly been involved in developing new techniques and methods for pricing as well as in implementing an appropriate infrastructure to enable these pricing models. Intensive research has been made to accomplish dynamic pricing models like *smart market* and *effective bandwidth* for connection–oriented networks like ATM. [19] is based on the results of the ACTS project *"Charging and Accounting Schemes in Multiservice ATM Networks (CA$HMAN)"* and gives a good overview of the mathematical, theoretical foundations as well as current research in this field and additionally presents a prototypical implementation for automatic, intelligent agent based tariff negotiation. The application of dynamic pricing models in packet–switched networks like IP based on, e.g., DiffServ, is described in [13] and [20]. Furthermore, the EC funded project *M3I* (`www.m3i.org`) is focusing on work around the field of resource management on basis of differential charging for multiple levels of service.

Our work presented in this paper was inspired by [7], in which a policy–based architecture for enforcing tariffs based on various pricing models is presented. Basically, a layered architecture is described where every single accounting software component of the usage phase is managed by applying appropriate policies. Overall, beside the fact that this work is considering only one phase of the service life cycle, it is concentrating on using DiffServ to implement and enforce tariffing policies. Thus, a specification of a policy language and a policy transformation for a general accounting management is missing which is in the focus of this paper.

4 Accounting Process Model

As stated in the introduction one of the main requirements for service– and customer–oriented accounting management is that it has to support the dynamic aspects regarding accounting of services. Thus, our goal is to develop a management solution which supports every activity needed for service accounting in an appropriate way. For this purpose, we obviously need to identify and analyze these activities first. From a more abstract point of view, activities can be grouped to *processes* that accomplish service accounting as a whole. Therefore, we need to investigate the dimension of (management) processes which incorporate all these activities needed for service accounting. In order to structure and to identify all relevant processes, we are using the service life cycle dimension [9]. Combining the process and life cycle dimension reveals all relevant dynamical aspects of service accounting and expresses consequently the requirements regarding the flexibility of a service–oriented accounting management solution. In Section 5 we will use this process analysis to specify the basic building blocks of our policy–based management solution regarding the policy language as well as the management architecture.

In the following, we are presenting the afore mentioned process analysis that we carried out as a first major step towards a new solution For this purpose we give first a short description of service life cycle phases. Afterwards we describe our analysis of accounting sub–processes in combination with the service life cycle. Finally, in order to identify the most relevant accounting entities, we additionally analyze the information flows between the sub–processes.

4.1 Service Life Cycle

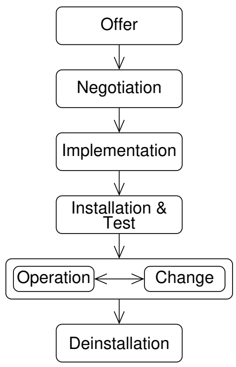

Figure 2: Service Life Cycle

In the following, we describe 7 life cycle phases (see Fig. 2) which represent an extension of the service life cycle proposed in [4]. In the *Offer* phase the provider submits a proposition consisting of a service description and a desired tariff for service usage. The customer is usually reviewing several propositions from various providers at the same time. The *Negotiation* phase deals with the process of concluding a contract between the provider and customer of a service. Usually, a contract contains details about service functionality, quality of service (QoS) and of course tariffs as well as penalties. Afterwards, the provider is implementing the agreed service functionality in the *Implementation* phase. In the *Installation & Test* phase all resources like, e.g., equipment, end systems, interfaces, applications, etc. that are needed to provide service functionality are installed and tested so that the service can start its operation. After the customer expresses the statement of acceptance regarding the service provisioning, the *Operation* phase of the service life cycle starts. Regarding accounting management, in case of usage–based tariffs the actual service usage has to be metered and an invoice has to be compiled and sent to the customer. The *Change* phase combines all interactions in regard to changing service functionality, service implementation and service management. Therefore, the change phase has to be seen in parallel to the Operation phase. The service life cycle ends with the *Deinstallation* phase within which the implementation's resources are released.

4.2 Analysis of Sub–processes

The accounting process as a whole can be separated into several sub–processes that consolidate some of the interactions and activities already mentioned above. As we have shown in Section 3 about related work, only the sub–processes in the operation phase have been considered so far when talking about accounting management. But, to be able to realize an overall integrated accounting management we need to take sub-processes of all life cycle phases into account. In the following, we present an overview of relevant accounting sub–processes that we have identified by analyzing the activities along the service life cycle applying the top down approach proposed in [4]. In the following, we will give a brief description of the accounting sub–processes which are also depicted in Fig. 3 using the UML activity diagram notation.

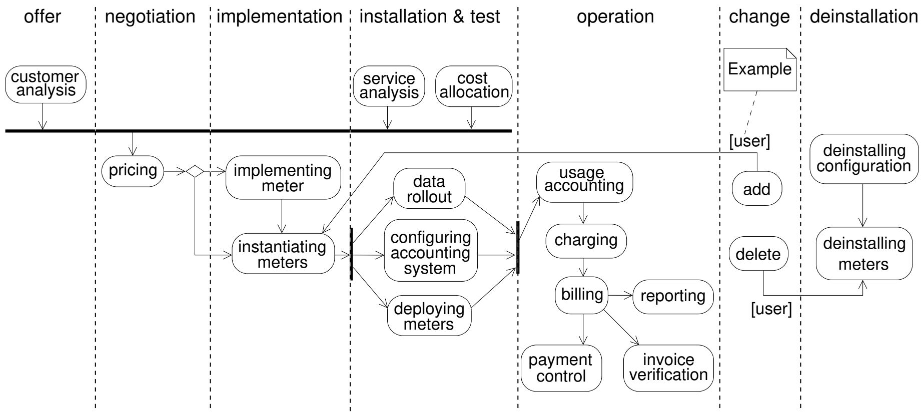

Figure 3: Accounting sub–processes along the service life cycle

customer analysis A provider will not be able to sell the provision of a certain service if its usage is too expensive. Therefore, to react on customer needs concerning the tariff of a service, a detailed analysis of customer requirements is done. This analysis serves as an input for the pricing sub–process.

service analysis When applying a usage–based tariff for charging, the actual usage of the service has to be metered by counting *accountable units*. Accountable units depend on the functionality and implementation of a service. Therefore, these aspects have to be identified by analyzing the service and its implementation. Result of this sub–process is a *tariff template* which consists of all possible accountable units as well as their weight within the tariff.

cost allocation A provider is typically interested in providing a service cost–covering. Therefore, he needs to identify these costs by analyzing the service implementation and its management. Ideally, this process results in a cost function which determines the provider's costs in relation to, e.g., service usage.

pricing This sub–process deals with the determination of the price/tariff for service usage. In general, one distinguishes the following pricing categories:
Static Pricing This means that the tariff for service usage is agreed between customer and provider in a service agreement and obtains as long as the agreement is not canceled. For this purpose the provider is also considering customer

requirements and a cost allocation of providing the demanded service. Currently, this pricing category is mostly applied in outsourcing scenarios.

Dynamic Pricing This means that the tariff for service usage is dynamically calculated every time *before* the service is actually used depending on, e.g., service demands from other users. Several pricing models have been developed in this area which are referred to as so called auction pricing and feedback pricing.

implementing/instantiating meters To be able to measure actual occurring service usage proper meters have to be implemented resp. instantiated. The meters have to count accountable units agreed upon in the service agreement.

deploying meters Depending on the agreed tariff and on the service, the usage can be metered on various locations, e.g., on server–side, on edges of the network, on client–side, etc. Various techniques for metering service usage and the resulting metering locations are classified and discussed in [8].

configuring accounting system All components are configured in a way that proper accounting can be done. This means that, e.g., all meters, charging and billing components are configured with information about users, service subscribers, invoice recipients, tariffs, etc.

data rollout In outsourcing scenarios service usage is often restricted to a certain group of users. Additionally, users are differently charged according to various criteria like employing department, office location, etc. To achieve this, the customer needs to deliver the required data (e.g., login accounts of all users) to the provider.

usage accounting To be able to do usage–based tariffing, the actual service usage has to be metered, e.g., according to the experienced QoS. The process of usage accounting combines all actions related to metering service usage and especially aggregating usage data. If a flat–rate is applied, this process contains no activities.

charging The charging process applies the agreed tariff on the metered data. For this purpose, the right tariff has to be selected according to criteria like user, customer, QoS etc. The result is a customer detailed record.

billing The billing process combines all relevant customer detailed records to an invoice according to the agreed format and sends the compiled invoice to the appropriate recipients.

payment control This process monitors the payment behaviour.

invoice verification For reporting and internal controlling, the invoices sent to the customers are reviewed regarding its correctness.

reporting Reports are compiled and sent to appropriate recipients on customer side that contain detailed data about the service provided in a given time interval.

change This sub–processes incorporates activities like adding/removing/altering user data, adding service functionality, etc. In most cases altering accounting data means that some of the components need reconfiguration that spans more than one life cycle phase. For example in Fig. 3 adding a new user might lead to activities like installing and deploying a new meter or at least assigning a present meter to the new user.

deinstalling configuration/meters When the validity of the agreed contract ends, the complete configuration of all components involved in the accounting system has to be removed. In case of dedicated accounting components (like installed meters within the customer domain), these have to be deinstalled additionally.

We have to emphasize that the transitions leading out of the change phase in Fig. 3 are only one possible instance; in real scenarios these transitions can vary depending on the service subscribed and on the agreed contract or tariff, respectively. One major challenge is to develop an accounting architecture which is capable of managing the accounting process with varying change management transitions.

4.3 Relevant Accounting Entities

In the previous subsection we already mentioned entities which are either source, target or result of activities and consequently are objects which should be considered by the management system. Hence, we need to specify these entities in more detail. Fig. 4 gives an example of a simplified analysis of the reporting process. Relevant Entities are visualized by specifying their name near an arrow. An arrow pointing to an activity of a process expresses the input and an arrow leading out of an process expresses the output of the process and activities, resp. The result of our analysis is illustrated in Fig. 5 by refining the MNM service model [4]. In the following, we will describe an brief overview of the model elements.

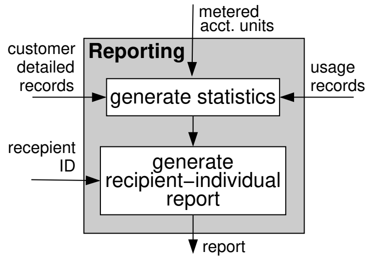

Figure 4: Example of identifying accounting entities

First a *customer* and a *provider* agree on a *service agreement* about the provisioning a certain service. Besides the description of service functionality and quality–of–service (QoS) a service agreement contains a *tariff* which is a refinement of a *tariff template*. A tariff template is the result of the service analysis sub–process done by the *accounting manager* and contains all possible accountable units and a cost function for a given service. Therefore, when specifying a tariff usually accountable units simply have to be selected from the tariff template and a price function has to be ascertained. When it comes to service operation the actual service usage of a *user* has to be charged. For this purpose, the user is using a *client* to access the service. Furthermore, the accounting manager is sending an *invoice* which contains the total amount to be paid by the *invoice recipient* on the customer side. Additionally, the accounting manager compiles *reports* sent to *report recipients* on customer side, which in contrast to invoices contain detailed data about service usage and the applied tariffs but no request for payment. The provider *directs* the accounting manager by, e.g., specifying a set of policies enforced by the accounting manager. The latter is realizing the accounting process and thus usually this role is taken by several accounting components like meters, collectors, billing systems, reporting tools etc. and the management system which controls these components.

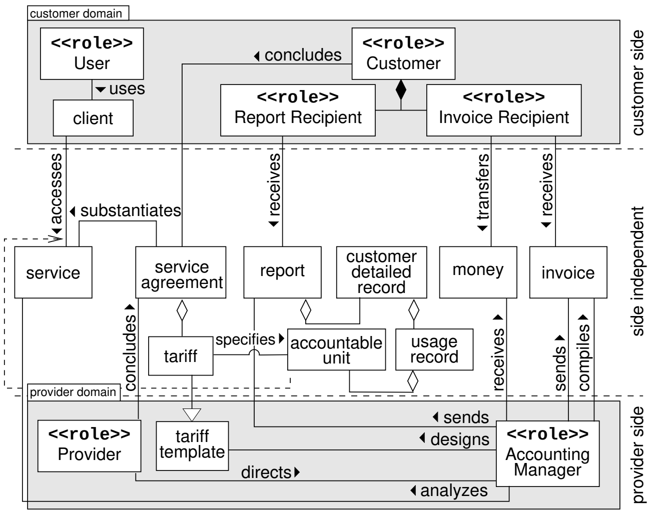

Figure 5: Excerpt of Accounting Service Model

5 Managing the Accounting Process

In general, todays accounting management solutions have to support and manage the accounting process described in Section 4.2 as a whole in order to provide the needed customer– and service–oriented management functionality. In consequence, the dynamics of the accounting process have to be taken into account as well as the fact that the process is implemented using heterogenous and distributed software products and components. On the one hand the dynamics result from varying sub–process transitions and on the other from change management processes that result into management actions spanning more than one service life cycle phase. To cope with these challenges, we propose the application of policy–based concepts to manage the accounting process.

The basic idea is that every accounting sub–process is managed by appropriate (accounting) management policies. Several advantages arise by using policies for this purpose: The needed abstraction of heterogenous accounting components is a "built–in" feature of policy–based concepts. The same is true for the needed integrated view on the managed objects. To automate management tasks that span more than one life cycle phase and thus span more than one sub–process we additionally need to express the sub–process transitions. Our idea is to express these transitions by concatenating the appropriate accounting policies. As we will show, some change management activities can not be expressed solely by enforcing a chain of existing accounting policies: Additionally the generation of new accounting policies that eventually replace existing ones is required. For this purpose we want to use the concept of meta–policies which basically provide the possibility to specify rules for creation, (de–)activation and deletion of policies.

In the following, we first present the developed policy–based management architecture and explain the design decisions made on basis of the analysis in Section 4. Finally, we describe the policy description language as well as the specification of accounting policies.

5.1 Designing a Policy–based Accounting Management

Management Architecture On the one hand we need to specify the elements of a policy description language (PDL) and on the other a management architecture that is capable of enforcing the specified policies. Hence, the specification of the policy–based management architecture determines the execution environment of policies. To fulfill the requirement of presenting one integrated view on accounting management, we introduce a *mediation layer* between the management system and the actual managed objects (i.e., the systems and applications implementing the accounting sub–processes). In this way, the management activities are executed on representatives of the managed objects and thus a uniform access on object interfaces during execution of policies is possible. The resulting policy–based accounting management architecture is depicted in Fig. 6. In order to increase the flexibility of the management environment we are modularizing the management application as well as the managed accounting system according to functionality classes. Regarding the management application we separated the functionality into operations which manipulate policies (like, e.g., creating and deleting policies) and into operations which enforce

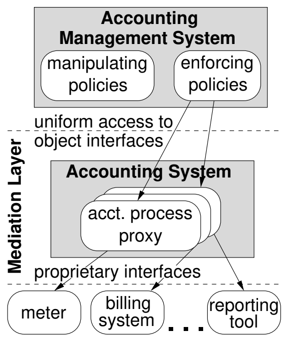

Figure 6: Policy–based Accounting Management Architecture

policies (i.e., interpreting the policy and executing the specified management actions). Regarding the accounting system we are using the sub–processes identified in 4.2 as a separation guideline: the functionality of every single sub–process is implemented by a separate module. When accessing the modules' interfaces within the mediation layer and thus calling a provided operation, the call is transformed to an appropriate call on the interface of the actual managed object. This could result into an API/SNMP/etc.–call as well as into altering a configuration file.

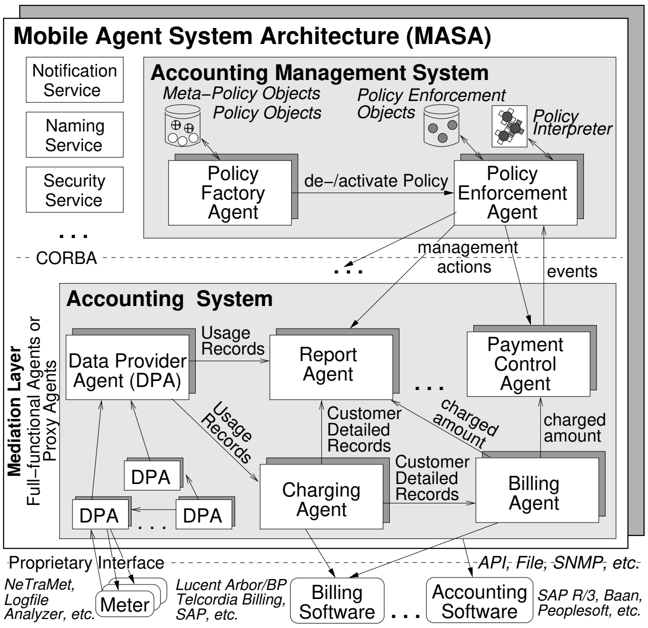

Figure 7: Implementation details

Implementation Details A reliable and secure accounting management system is the prerequisite for successful operation of an accounting system. Therefore, we decided to use the *Mobile Agent System Architecture (MASA)* [5] as our development and runtime environment. Basically, MASA can be separated into Agents providing a specific functionality and into the Agent System which provides the runtime environment for the Agents. MASA is completely written using Java/CORBA [3] and offers, as an MASIF [18] compliant agent system, several basic services like naming, notification and security. Additionally, to increase reliability of agents, a load balancing on basis of agent migration is possible. Following the design decisions described above, the *accounting management system* consists of two agents (see also Fig. 7): the *Policy Factory Agent (PFA)* and the *Policy Enforcement Agent (PEA)*. The PFA offers an interface for creating, deleting, activating and deactivating accounting management policies as well as meta–policies. Furthermore, the PFA stores and retrieves *policy objects* and *meta–policy objects* which are specified in the policy description language (PDL) using XML. XML is used for this purpose, as by using a standard XML parser we do not need to implement a policy parser from scratch. In contrast to the PFA, the PEA controls and manages so–called *policy enforcement objects (PEO)* which are created just when a policy object is activated. A PEO is a CORBA object and reacts on received *events* by enforcing the specified management actions. In this case a PEO is executing CORBA calls using the *CORBA Dynamic Invocation Interface (DII)*. Before executing management actions resp. invoking CORBA operations a PEO has to parse the policy. For this purpose a PEO uses a *Policy Interpreter* which is an adapted XML parser that evaluates the specified constraints. Basically, the Policy Interpreter returns "true" or "false". In case of "true" the PEO is firing the specified management activities. The mediation layer is implemented using various MASA (proxy) agents which represent the *accounting system*. These agents are either full–functional agents or—as in most cases—proxy agents for software products/components which offer the required functionality. Every agent implements an IDL interface which is corresponding to the sub–process' functionality. The IDL interface is developed by examining the sub–process interactions and activities from Section 4 in more detail. In case of a proxy agent a CORBA call at the agent's interface is transformed into appropriate instructions of the software component's interface. Thus, the underlying software components like billing systems can be exchanged without the need to neither reimplement the management system from scratch nor to respecify existing policies. All in all, from the manager's point of view an integrated, uniform view on accounting management is established.

5.2 Specifying Policies for Accounting Management

Determining the Policy Language Various different concepts for expressing a policy already exist which are discussed, e.g., in [17]. Our main decision objective for choosing a specific policy concept is that it already comes with a proven implementation. Thus, we decided to apply the concept of *policy objects with attributes* as proposed in [14, 15, 16]. A policy is seen as an object with several attributes like *subject*, *target*, *activity* and *constraints*. Herein, a subject is seen as a set of objects, for which the specified activities are performed on a set of targets (in case that the specified constraints evaluate to true). Our research revealed that these four elements are sufficient to express appropriate policies in order to manage exactly one single accounting sub–process, but not to manage the accounting process as a whole, as sub–process transitions cannot be expressed. For this purpose we are adding the attribute *event* to the policy language combined with the demand that the implementation of every sub–process (i.e., the appropriate proxy agent) at least generates one event like the start or end of an activity. Specifying these events within a policy can then be used to express sub–process transitions. To sum up, an accounting policy has at least the attributes contained in the following statement:

`POLICY` *id* `FOR` *subject* `ON` *target* `ON EVENT` *events* `DO` *activities* `CONSTRAINT` *constraints*`;`

The next step is to determine how the elements of a policy are specified. Therefore we need to take the possible element–"values" into account. First, solely accounting entities identified in Fig. 5 can occur as either the subject or target of a policy, as these are the only ones which are affected by the accounting process and thus are the only ones which are possibly affected by management activities resp. management policies. Therefore, the entities as well as the identified associations have to be expressed in an appropriate way. As the class diagram in Fig. 5 is instantiated when operating the accounting system, we use an OO notation style for specifying subjects and targets. In order to specify events as well as the activities we have to take a close look at the environment in which the activities are executed. As we have introduced the mediation layer in our management architecture, all activities and events are executed in a CORBA environment. Hence, we use CORBA IDL notation to specify the policy activities and the Notification Service notation for specifying events.

Specifying Policies As we want to manage every single accounting sub–process by applying appropriate policies, we have to basically find out what the subjects, targets, activities and constraints for every single sub–process are. In this paper we are concentrating on sub–processes which are mainly operated by IT resources and thus can be managed by an appropriate policy management system in an autonomous and automatic way. Therefore, the sub–process pricing as well as the sub–processes needed for pricing are left out in this step of policy specification as usually in outsourcing scenarios these sub–processes are mainly carried out by human interactions.

Table 1 summarizes the identified subjects and targets for the remaining sub-processes. Every table column represents one accounting entity identified in Fig. 5. The table rows represent policies managing specific sub–processes. For every single policy the created matrix contains exactly one circle representing the subject and exactly one cross representing the target. Additionally, several triangles were added representing that there is some kind of dependency between this specific policy and the marked accounting entity. These dependencies were identified during the analysis in Section 4.3 and are needed for specifying meta–policies, as we will show.

accounting entity / policy type	user	customer	report recepient	invoice recepient	accountable unit	charging formula tariff	customer detailed record	report	invoice	service	meter	collector	charging component	billing system	reporting tool	rollout component	runtime environment	development environment
implementation					▽					▽	○							✖
rollout	▽	○														✖		
installation					▽	▽					○						✖	
metering	○				▽						✖							
collecting		▽			▽	○	▽				▽	✖						
charging	○					▽						▽	✖					
billing				○					▽					✖				
reporting			○					▽							✖			
	customer side				side independent						provider side							

○ subject ✖ target ▽ dependency

Table 1: Accounting policies and their dependencies

Obviously, policies for the sub–processes in the change phase are missing. As shown in our change management activity in Fig. 3 (i.e., adding a user), change activities often deliver the impulse for executing activities that usually belong to other sub–processes and consequently to other life cycle phases (e.g., installing/configuring a new meter, etc.). Overall, usually a chain of existing sub–processes is executed due to change management activities. Therefore, we need to take the sub–process transitions into account for managing the accounting process as a whole. As in our case sub–process transitions are enforced by events, we actually need to identify proper events for every transition. Specifying these events within the appropriate policy reveals the feasibility to express the execution of policy chains and thus the management of sub–process chains as especially needed in case of change management activities. Generic events generated by every sub–process are the start and the end of an activity.

However, some change activities can not be expressed solely by enforcing sub–process chains. Instead new policies need to be created which in some cases replace existing ones. Our research revealed that only changes affecting entities which are specified in the service agreement (i.e., the side independent part of table 1) might lead to creation of new policies. E.g., if there is a new charging formula for a group of users, a corresponding new charging policy has to be specified. In this case, the specification of meta–policies can help to declare which specific change activities result in creation/deletion/activation/deactivation of “normal” accounting policies. For this purpose we can reuse table 1 to analyze which policies are potentially affected by meta–

policies: as circle/cross/triangle–symbols also express a dependency between an entity and the corresponding policy only those policies have to be checked where the column of a specific entity contains such a symbol. This immensely reduces the number of required tests.

Table 2 summarizes the elements of the policy description language for both the accounting policies as well as meta–policies in a semi–formal way.

Policy	Sub–proccess–*ID*	**Meta–Policy**	ID
For	{ *entity*{*.association.entity*}}	**For**	*entity*
On	{ *entity*{*.association.entity*}}	**On**	*Policy ID*
On Event	{*Activity.(started*$\mid$*ended)*}	**On Changes**	*newly added*$\mid$*existing* (*altered*$\mid$*deleted*)
Do	{*(target*$\mid$*object)*.IDL–Operation}	**Do**	*create*$\mid$*delete*$\mid$*(de)activate* Policy
Constraint	Boolean Expression	**Constraint**	Boolean Expression

Table 2: Semi–formal presentation of policy description language

6 Conclusion and Further Work

As shown in this paper, new requirements for accounting management evolve from the trend of service– and customer–orientation. In this paper, we have proposed the application of policy–based concepts to provide the required highly integrated, flexible and automatic accounting management which especially supports the dynamic aspects of change management. To achieve this, we have first analyzed the dynamic aspects of service accounting and its management from a process–oriented view. By identifying sub–processes and analyzing sub–process transitions we have derived relevant accounting entities which were used to compose an accounting service model. Afterwards, we have used the requirements to design a policy–based management architecture for accounting management. Finally, the process analysis as well as the accounting service model served as a source for design decisions regarding the policy language. As we use policies to manage every sub–process and process–transition involved in service accounting and additionally the concept of meta–policies, we achieved to meet the posed requirements.

We are about to finish a prototypical implementation of the described architecture. Several mediation agents like, e.g., for NeTraMet [2], Apache Log Files, etc. have already been realized. The policy interpreter on basis of an XML parser is on the way. We are planning to install an accounting management test bed based on the concepts presented in this paper on site of one of our cooperation partners to gain experience from real–life scenarios.

In the future, we will investigate the reuseability of policies and meta–policies with the goal of defining either policy schemes or a methodology for specifying accounting policies. Additionally, in order to reduce costs in accounting management we are also doing research on reusing accounting software components like usage collectors and billing systems for different customers, as up to now usually every component is exclusively dedicated to one customer. In this case "clientele processing" becomes one of the major requirements which have to be fulfilled.

Acknowledgment The author wishes to thank the members of the Munich Network Management (MNM) Team 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 researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its web–server is located at `http://wwwmnmteam.informatik.uni-muenchen.de`.

References

[1] B. Aboba, J. Arkko, and D. Harrington. RFC 2975: Introduction to Accounting Management. RFC, IETF, October 2000.

[2] N. Brownlee. RFC 2123: Traffic Flow Measurement: Experiences with NeTraMet. RFC, IETF, March 1997.

[3] The Common Object Request Broker: Architecture and Specification. OMG Specification v2.4.2, Object Management Group, February 2001.

[4] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, M. Langer, M. Nerb, I. Radisic, H. Roelle, and H. Schmidt. Towards generic Service Management Concepts – A Service Model Based Approach. In *Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001)*. IEEE, May 2001.

[5] B. Gruschke, S. Heilbronner, and H. Reiser. Mobile Agent System Architecture — A platform for flexible IT Management (in german). Technical Report 9902, Ludwig-Maximilians-University Munich, Dep. of CS, Munich, August 1999.

[6] T. Hamada. Accounting Management Architecture. Version 1.3 (draft), TINA Consortium, February 1996.

[7] F. Hartanto and G. Carle. Policy-based Billing Architecture for Differentiated Services. In *Proceedings of IFIP Fifth International Conference on Broadband Communications (BC'99)*, September 1999.

[8] R. Hauck and I. Radisic. Service Oriented Application Management — Do Current Techniques Meet the Requirements? In *Proceedings of the 3rd IFIP International Working Conference on Distributed Applications and Interoperable Systems (DAIS 2001)*. Kluwer, September 2001.

[9] H.-G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application*. Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999. 651 p.

[10] Network Data Management – Usage (NDM–U) For IP-Based Services. Version 3.0, IPDR, Inc., November 2001.

[11] Information Technology – Open Systems Interconnection – Systems Management – Part 10: Usage Metering Function for Accounting Purposes. IS 10164-10, International Organization for Standardization and International Electrotechnical Committee, 1995.

[12] TMN Management Functions. Recommendation M.3400, ITU, April 1997.

[13] M. Karsten, J. Schmitt, B. Stiller, and L.Wolf. Charging for packet-switched network communication – motivation and overview. *Computer Communications*, 23(3):290–302, February 2000.

[14] T. Koch. *Automated Management of Distributed Systems*. PhD thesis, Fern-Universität Hagen, Germany, 1997.

[15] T. Koch, C. Krell, and B Krämer. Policy Definition Language for Automated Management of Distributed Systems. In *IEEE Workshop on Systems Management*. IEEE, 1996.

[16] D. Marriott and M. Sloman. Implementation of a Management Agent for Interpreting Obligation Policy. In *7th International Workshop on Distributed Systems Operations and Management (DSOM 96)*. IFIP/IEEE, October 1996.

[17] D. A. Marriott. *Policy Service for Distributed Systems*. PhD thesis, Imperial College London, June 1997.

[18] Mobile Agent System Interoperability Facilities Specification. OMG TC Document orbos/98-03-09, Object Management Group, March 1998.

[19] D.J. Songhurst, editor. *Charging Communication Networks – From Theory to Practice*. Elsevier Science B.V., first edition, 1999.

[20] B. Stiller, P. Reichl, and S. Leinen. A Practical Review of Pricing and Cost Recovery for Internet Services. In *Netnomics – Economic Research and Electronic Networking*, volume 3. Baltzer, The Netherlands, March 2001.