# Flexible Charging for Multi-provider Composed Services using a Federated, Two-phase Rating Process

Brendan Jennings and Paul Malone
Telecommunications Software & Systems Group,
Waterford Institute of Technology,
Waterford, Ireland
{bjennings, pmalone}@tssg.org

***Abstract*—We present a framework enabling charging for composed services comprised of services offered by multiple providers. In the framework rating engines may generate charge information for individual services and provide this information upon request to other rating engines when these services are used as part of a composed service. Rating engines additionally employ a two-phase rating process which allows potentially complex business agreements between providers to be reflected in composed service charges. Charges can vary depending on the context in which a service is executed, for example, as part of a composed service which includes services offered by a partner provider. Crucially, the process allows rating engines to calculate these varying charges without having to be manually pre-configured with details of the structure of individual composed services. In the paper we provide an overview of the framework, specifying in detail the rating process and inter-rating engine communications, and describe via an example its deployment in a distributed environment supporting the execution of composed services.**

***Keywords: Charging, Service Composition, Multi-provider Composed Services, Federated Rating.***

## I. INTRODUCTION

Driven by the requirement to generate a return on their recent investments in 3G licenses and network infrastructure, telecommunications network operators are beginning to promote the development of ecosystems of third party application/service providers who will offer their services to users via the operator's network. Already, in the Internet world, this vision is becoming a reality, with a vast array of providers, large and small, both offering online services directly to users and trading with each other in virtual ecosystems. Although the markets for networked services are maturing rapidly, there are a number of technical limitations that may prevent them reaching their full potential. Among these are the challenges associated with offering composed services – groupings of services that are executed in an orchestrated manner in order to deliver "added-value" functionality to a user.

Many in the research community see service composition, in particular dynamic service composition – in which services are composed on-the-fly to meet evolving needs – as essential to aid users effectively interact with a service-rich networking environment. However, the task of managing networks supporting composition of user-facing services will be complex. Existing management systems typically target management of user-facing services on an individual basis; they do not consider the possibility that such services can be collectively orchestrated in an almost arbitrary manner to fulfill changing requirements. It is unclear how these management systems can be evolved to provide, for example, performance assurance, service-level security, accounting or fault detection for composed services. We consider service usage accounting to be one of the most pressing issues, since the commercial success of systems facilitating service composition will be contingent on the ability of providers to charge and collect appropriate fees for service usage. Our work has focused on the development of an approach enabling flexible charging for composed services comprised of services offered by multiple providers. In this context, we believe the key challenge is to realize a model reflecting the potentially complex business agreements between multiple providers, one that can provide automated charging for composed services of which the accounting and charging system may have no *a priori* knowledge.

This paper presents a charging framework targeting composed, multi-provider networked services. The framework is designed to grant service providers the ability to design and deploy charging schemes for their services that govern both the charges generated for usage of these services on a standalone basis and the charges generated when the services are used in conjunction with other services. Furthermore, these charging schemes are generic, in the sense that they do not need to be specified and deployed each time a service becomes part of a composed service. The framework thus supports flexible charging for composed services and, importantly, offers a means for providers to incentivize the inclusion of their services in service compositions by offering appropriate discounts. This paper provides a full description of the charging framework, incorporating a federated version of the two phase rating process originally presented in [1], together with specifications of communications protocols and error handling functionality.

To motivate the need for the proposed charging framework we first briefly review the state of the art in accounting and

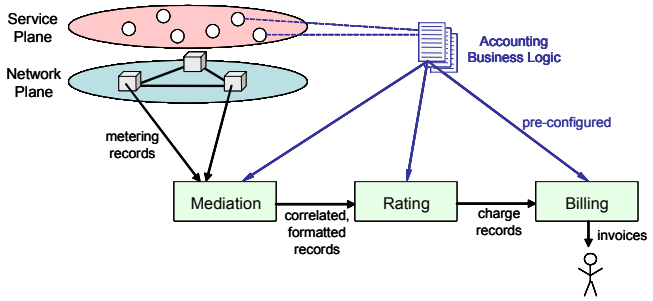


Figure 1. Accounting System for Networked Services.

charging for networked services. We then provide a generalized view of the service composition environments in which we envisage the framework being employed. The framework is then specified in detail; it is comprised of protocols for rating engine to workflow manager and rating engine to rating engine communications, a novel two-phase rating process, and associated error handling functionality. We describe how the framework has been implemented as an enhancement to a previously developed advanced rating engine prototype and discuss its deployment in the Digital Business Ecosystem business-to-business service execution environment. To illustrate the operation of the framework we examine an example rating scenario for a mobile email service. Finally, we draw conclusions and outline plans for further work.

## II. Accounting and Charging for Networked Services

As shown in Fig. 1, accounting systems for (post-paid) networked services typically incorporate facilities for metering, mediation, rating and billing. *Metering* takes place within the network infrastructure; it is concerned with the accurate recording of service usage data and exposing collected metering records to the mediation subsystem. *Mediation* involves reliable collection of "metering records" from metering devices; correlation of records relating to the same service usage sessions; transformation of disparate metering record formats into a common format suited to the needs of the rating subsystem; and reliable transfer of processed records to the rating subsystem. *Rating* involves the application of models ("charging schemes") for the mapping of usage data to monetary units based on various criteria: each record received from the mediation subsystem is examined and the appropriate charging scheme is applied, resulting in the generation of a "charge record" for the service session. Finally, the *Billing* subsystem collates charge records for individual customers, who are invoiced on the basis of these charge records, and any additional subscription fees and discounts.

Accounting and charging for services, including composite services whose structure are known in advance, is a relatively mature area. For example, in the telecommunications domain there is wide deployment of complex systems supporting sophisticated usage- and content-based charging schemes for pre-paid and post-paid, private and corporate, customers. Such systems are typically optimized for reliability and speed, with the components involved in the accounting process being manually configured to account for specific services at the time those services are initially deployed. However, even for relatively simple, standalone services, configuration of accounting operations is today a time consuming and costly activity. Large businesses, such as telephone network operators, invest considerably in complex systems to track usage of their services and charge customers accordingly.

Given the critical importance of accounting and charging to any business, numerous standardization bodies have specified standards for accounting systems, processes and protocols for a range of application domains. For example, the 3GPP SA5 working group has specified standards governing the GPRS/UMTS mobile network entities involved in accounting, whilst the IETF Authentication, Authorization and Accounting (AAA) working group has specified protocols for the transfer of accounting data in IP networks. A detailed survey of these and other efforts can be found in [2]. We note that the resulting standards universally assume that accounting and charging is done on an essentially per-service basis; they do not consider the complexities involved in charging for composed services in a manner that can flexibly reflect business agreements between multiple service providers.

The European Commission funded IST FP5 project FORM recognized that the emergence of the business-to-business e-Commerce market was influencing how telecommunications and Internet services were being provided, charged and billed for. One of the challenges addressed by the project was that of providing a service bill that integrated charges for individual service usage of those services constituting a composed service. As a solution the project introduced the concept of an *Inter-Enterprise Service Provider (IESP)*, a business entity that acts as a broker between a service consumer and the third party service providers, consolidating the charges relating to third party service providers and presenting the service consumer with a single bill. The IESP is responsible for the service composition and the management of relationships, contracts, usage data and financial transactions between the service providers and consumers. This approach has been termed *Federated Accounting* [3] in that it specifically supports the requirement for service providers to cooperate in the provision of composed services in a federated manner and share the generated revenue. However, the approach addresses only statically composed services – an IESP will need to have pre-configured the accounting process for a composed service well in advance of deployment of that service.

Agarwal et al. [4] propose a method for metering and accounting for composite e-services, which is not dependent on *a-priori* knowledge of the service composition. However, their approach supports only two specific service charging models (flat rate per amount of resource used and flat rate per transaction). More significantly, the charge for an invocation of a composed service will always be the summation of the charges associated with standalone invocations of the constituent services; as discussed previously, this will not always reflect the business relationships between the service providers. For example, providers may wish to offer discounts if their services are used in conjunction with services offered by a partner, or conversely charge a premium if they are used with services of a competitor. The intention of these measures would be to provide cost incentives for the use of particular

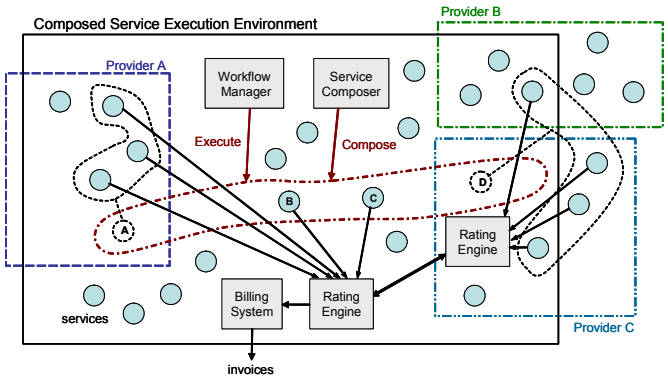


Figure 2. Generalized Composed Service Execution Environment.

service combinations over other combinations providing the same, or similar, functionality.

## III. Multi-Provider Service Composition Environment Model

Our model of a commercial multi-provider composed service execution environment is illustrated in Fig. 2. This environment contains services that can be composed together and various components that support the service composition, execution and accounting processes. The *Service Composer* is responsible for constructing service compositions to meet specific requirements, whilst the *Workflow Manager* is responsible for coordinating the execution of composed service invocations. Accounting and charging functionality is provided by a *Rating Engine* and a *Billing System*. For simplicity we omit other accounting related components, such as mediation systems, CRM systems, and fraud management systems. As individual services are consumed, *metering records*, detailing service utilization patterns, are generated and forwarded to a Rating Engine, which is responsible for applying charging schemes that map service usage data contained in those metering records to monetary charges. The Billing System accepts charge records from the Rating Engine and consolidates them to generate customer invoices. We assume that all services are offered on a post-paid basis.

From the accounting perspective, service providers can adopt a number of approaches in offering services, specifically:

- *Services are offered as atomic services (e.g. service B in Fig. 2):*
  In this approach services appear as individual (non-composed) services for which metering records identifying the service type in question are generated, and for which specific charging schemes are available. Note that these services may actually be realized by the provider as compositions of other services, but for operational or commercial reasons the provider wishes to mask this;
- *Services are registered as atomic services and the provider itself controls generation of charge information for the service (e.g. service D in Fig. 2):*
  In this approach services again appear as atomic services, even if they are actually realized as service compositions. However, the provider also takes responsibility for generating charge information for the service, either by maintaining its own rating engine, or by using a rating engine provided by a trusted third party. Thus, if the service is itself used as part of a "higher-level" service composition, its rating engine would be queried by the rating engine responsible for the higher level service;
- *Services are registered as composed services and the services comprising that composed service are identified (e.g. service A in Fig. 2):*
  In this approach the structure of the offered composed service is visible to the execution environment. Metering records are generated by the (atomic) services comprising the composed service, and the accounting components identify these records as relating to an invocation of the composed service. The composed service may or may not have a specific charging scheme associated with it; if not, the rating engine must be able to calculate charges on the basis of the collection of charging schemes associated with its individual constituent services.

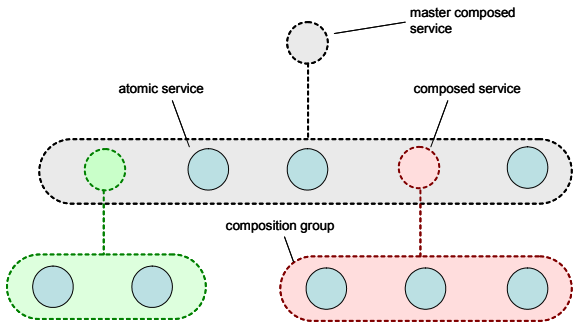


Figure 3. Composed Service Model for Accounting Purposes.

For the purpose of accounting we model composed services as hierarchical collections of atomic and composed services, as illustrated in Fig. 3. The service at the top level of the hierarchy is the *master composed service* – the service to be rated. Services at the bottom of the hierarchy are all *atomic services* (they cannot be further decomposed for accounting purposes). We call a group of service directly composed together a *composition group*.

All Service instances have associated with them a *providerID*, which uniquely identifies, across the entire environment, the business entity offering that service. They have a *serviceID*, which uniquely identifies the service type within the set of service types provided by the business entity with the service instance's *providerID*. They also have an *instanceID*, which uniquely identifies an instance of a service type within the set of instances of services with the same *serviceID* and *providerID*. This reflects the fact that a service provider may maintain multiple instances of the same service type, and the potential need to distinguish between these instances for rating purposes. Therefore, the tuple of (*providerID*, *serviceID*, *instanceID*) uniquely identify a service instance across the entire environment.

In order to accurately rate dynamically composed services it is of critical importance is able to detect the context in which an atomic service it is rating is being executed – as a

standalone atomic service, or as part of a particular composed service. Our approach is to have the workflow manager assign a unique *transactionID* to every invocation of a composed service. The *transactionID* uniquely identifies the invocation of the master composed service during the timeframe between initial invocation and the completion of all processing associated with that invocation. It is passed to every service comprising that composed service and is included in all metering records relating to those services. This approach has the advantage that services do not need to be aware of their full execution context (at least for accounting purposes). Furthermore, every service instance executed within the context of a composed service is passed an *invocationID* which uniquely defines that invocation of that service instance within the context of the master composed service. Thus, if a master composed service involves two or more independent invocations of the same instance of a given service these invocations can be distinguished by a rating engine.

We assume that when the workflow manager is about to execute a master composed service it knows the location of a rating engine capable of rating this service. A rating engine is capable of rating a master composed service if charging schemes associated with all the atomic services making up that composed service can be accessed, directly or indirectly, by that rating engine. Direct access to a charging scheme implies that the scheme is deployed on the rating engine, whereas indirect access implies that the scheme is deployed on another rating engine that can be queried by the rating engine. The rating engine that is responsible for generating a charge for the master composed service is termed the *master rating engine*; other rating engines it queries in the course of ascertaining this charge are termed *slave rating engines*.

We also assume that all rating engines (and indeed workflow managers / service composers) are themselves viewed as services in the composed service execution environment, hence they are assigned a unique (*providerID*, *serviceID*, *instanceID*) tuple which allows them to be identified and located within the environment. A potential advantage of this approach is that use of rating engines themselves could be charged for by rating engine providers – by having them treat themselves as a constituent services of the master composed service.

## IV. Charging Framework Description

In our model the key controlling entity is the Workflow Manager, which manages the execution of the composed services. In particular, it will be responsible for coordinating the charging process, selecting one or more rating engines to which services will forward metering records. The model facilitates *federated rating* – the ability for rating engines to communicate with each other in order to ascertain a charge for the execution of a (composed) service. We believe that support for federated rating will be crucial if the type of multi-provider composed services envisaged are to be made available commercially, since many providers would otherwise be very reluctant to have their charging schemes deployed on rating engines they don't control. Given the above, there is a clear requirement for our charging framework to support communications interfaces between both the workflow manager and rating engines, and between rating engines themselves. The protocols to facilitate these communications are described in §IV.A/B below.

A key design goal of our framework is to limit the amount of manual configuration of rating engines necessary to allow them generate charges for composed services. Once charging schemes are available for all the atomic services constituting a master composed service it should not be necessary to manually configure individual rating engines so that these charging schemes can be applied together. We therefore conclude that rating engines should not be pre-configured with knowledge of the structure of individual master composed services, rather that this information be supplied to them on-the-fly by a workflow manager. Note that this approach opens the way for supporting charging for dynamically composed services – services which are composed on-the-fly to fulfill some new requirement.

Generating a charge for a master composed service of which a rating engine will have no *a-priori* knowledge creates a significant challenge: how can we produce a charge that accurately reflects the potentially complex business agreements between different service providers? It would be relatively straightforward to generate charges for all the individual atomic services, and then simply sum these to produce a charge for the master composed service. However, providers are likely to wish that their services are charged at different rates depending on the context in which they are executed. For example, a provider may offer a percentage discount of two or more of its services are used together within the same master composed service. To facilitate this kind of flexible charging for composed services we propose a two-phase rating process, in which atomic services comprising a master composed services can be each initially rated as if they were executed standalone; and subsequently, the generated "interim" individual service charges can be modified based on rules embodied in the service's charging scheme. Alternatively, composed services can have charging schemes that supersede those associated with the atomic and composed services that constitute them, thereby allowing the provider to apply charging models for a composed services that are completely independent of the underlying services. This process is specified in detail in §IV.C.

Finally, in §IV.D, we discuss how errors in execution of composed services can be handled from the charging perspective.

### A. Workflow Manager – Rating Engine Communication

A workflow manager about to execute a master composed service must select the rating engine(s) responsible for rating that service execution instance. There will be one master rating engine and zero or more slave rating engines depending on the make up of the master composed service. The workflow manager passes a message to each rating engine embodying information regarding the service(s) they will be required to rate, and indicating the *transactionID* that will be assigned to the session. The rating engines ascertain if they have access to the required charging schemes and inform the workflow manager if they do.

The workflow manager then commences execution of the master composed service, resulting in metering records being passed to the appropriate rating engines. Once the master composed service execution session completes the workflow manager sends a message to the master rating engine indicating how the session completed (successfully or unsuccessfully); this acts as a trigger for the master rating engine to start phase 2 of the rating process, which, once completed, results in a final charge for the master composed service execution session. To support this behavior the following four message types are passed between the workflow manager and the master and slave rating engines: **rateRequest**, **rateResponse**, e**xecutionComplete**, and **ratingComplete**.

*1) **rateRequest** Message*

The **rateRequest** message contains the *transactionID* that will be contained in all metering records relating to all atomic service invocations for the master composed service invocation. It contains the *ratingEngineRole* which is set to "Master" or "Slave" depending on whether the workflow manager wishes the rating engine to act as the master or as a slave rating engine.

In the case where the workflow manager is requesting the rating engine to act as the master the message will include the *providerID*, *serviceID*, *instanceID* for the master composed service itself; and a description of the underlying service composition hierarchy in terms of each service's *providerID*, *serviceID*, *instanceID* and *invocationID*. For those atomic services that will be rated by a slave rating engine the message will also contain the *providerID*, *serviceID* and *instanceID* of that slave rating engine. In the case where the workflow manager is requesting the rating engine to act as a slave the message will include the *providerID*, *serviceID*, *instanceID* for each of the atomic services (there may be more then one) that the slave rating engine will be required to rate. It will also include the *providerID*, *serviceID* and *instanceID* of the master rating engine (the slave will only pass charge information to its master).

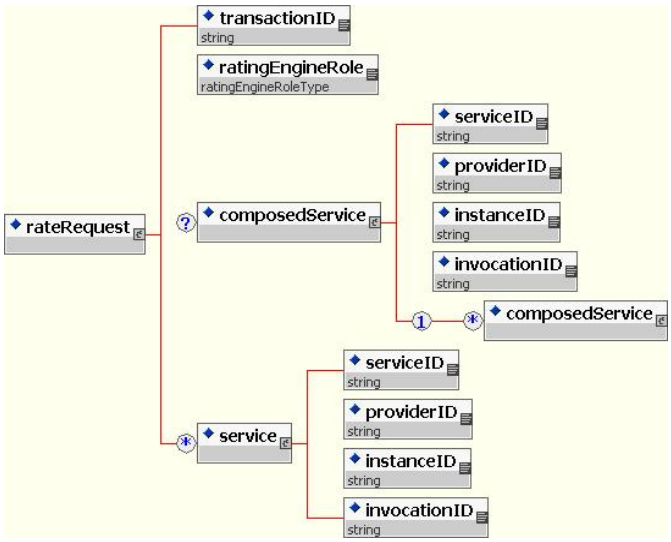


Figure 4. **rateRequest** XML Schema Representation.

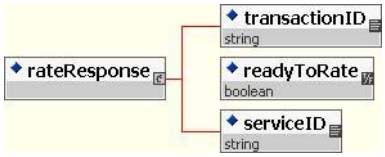


Figure 5. **rateResponse** XML Schema Representation.

The structure of the **rateRequest** message is illustrated in Fig. 4. The representation in this figure is generated from an XML schema for convenience and conciseness of presentation – our framework does not require the use of XML messaging – only that the specified information is reliably transferred between the entities in question.

*2) **rateResponse** Message*

Upon receipt of a **rateRequest** message, a rating engine ascertains whether it has access to charging schemes for the required services. In the case of the master rating engine this access may be indirect (via slave rating engines). It responds with a **rateResponse** message indicating the *transactionID* and containing a list of all the atomic services in question. For each service *readyToRate* is set to "Yes" if the rating engine has direct/indirect access to the required charging scheme, and "No" otherwise. Explicit knowledge of which services a rating engine may not be currently able to rate may allow the workflow manager to seek another rating engine for these services. Once the workflow manager receives positive **rateResponse** messages from the master and slave rating engines (indicating that all the required services can be rated) it will proceed with execution of the master composed service. Fig. 5 provides a representation of the **rateResponse** message format.

*3) **executionComplete** Message*

The workflow manager sends the master rating engine an **executionComplete** message once the master composed service execution session completes. This message contains the *transactionID* and *completionStatus*, the latter indicating that the session as a whole was "successful" or "unsuccessful". If the session was successful the master rating engine can proceed with phase 2 of its rating process as described in §IV.C. If not, the message also contains a list of all the atomic services in the master composed service, specifying their *providerID*, *serviceID*, *instanceID*, *invocationID*, and *executionStatus*. The *executionStatus* indicates whether that invocation of that service instance was "completedSuccessfully", "notStarted" or "completedPartially", before the error condition occurred. The

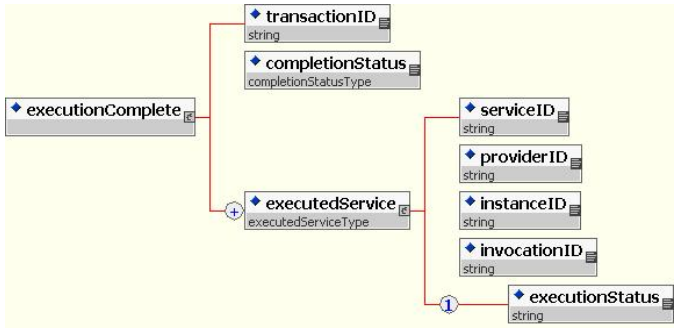


Figure 6. **executionComplete** XML Schema Representation.

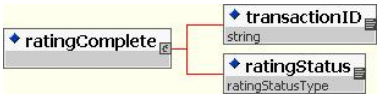


Figure 7. **ratingComplete** XML Schema Representation.

master rating engine then proceeds with the error handling version of phase 2 of its rating process, as described in §IV.D. Fig. 6 illustrates the **executionComplete** message format.

*4) **ratingComplete** Message*

The **ratingComplete** message is sent by the master rating engine to the workflow manager upon completion of phase 2 of the rating process. As shown in Fig. 7 it contains the *transactionID* and the *ratingStatus*, with the latter being "successful" or "unsuccessful".

## *B. Rating Engine – Rating Engine Communication*

As discussed above, master rating engines must be able to request charge information from slave rating engines who are responsible for rating one or more of the atomic services comprising the master composed service. To achieve this we introduce **chargeRequest** and **chargeResponse** messages, as follows:

*1) **chargeRequest** Message*

Upon receipt of an **executionComplete** message from the workflow manager the master rating sends a **chargeRequest** message to the relevant slave rating engine for each of the atomic services it has not itself rated. As shown in Fig. 8 these messages contain the *transactionID*; the *providerID*, *serviceID*, *instanceID* and *invocationID* for the service instance invocation for which the charge is being requested; and also a list of the other services ("*partnerServices"*) that constitute that service's composition group in the master composed service (this list specifies the *providerID*, *serviceID*, *instanceID* and *invocationID* for each such service). Inclusion of the composition group information allows the slave rating engine execute phase 2 of the rating process, in which the charge for the service instance invocation may be modified (see §IV.C). The receiving slave rating engine will confirm that the **chargeRequest** originates from the expected master rating engine (as indicated in the original **rateRequest** message it received from the workflow manager); if it does it runs phase 2 of the rating process for the service instance invocation in question.

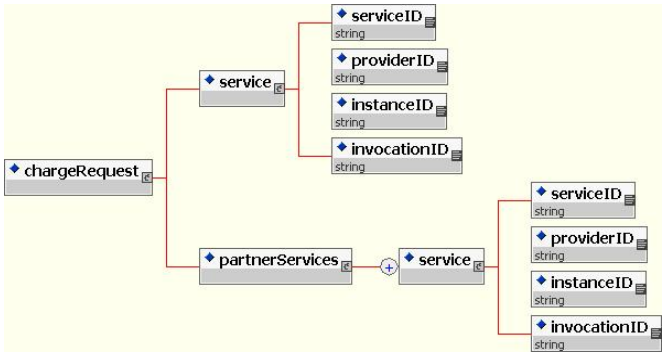


Figure 8. **chargeRequest** XML Schema Representation.

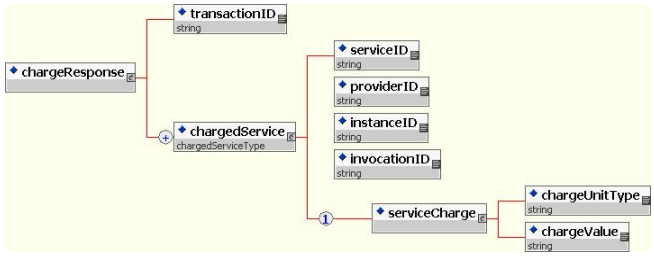


Figure 9. **chargeResponse** XML Schema Representation.

*2) **chargeResponse** Message*

Upon completion of phase 2 of the rating process the slave rating engine transfers the resulting charge in a **chargeResponse** message to the master rating engine. As illustrated in Fig. 9 the **chargeResponse** message contains the *transactionID*; the *providerID*, *serviceID*, *instanceID* and *invocationID* for the service instance invocation; the *chargeUnitType* and the *chargeValue*

## *C. Two-phase Rating Process Specification*

In this section we specify the two-phase rating process for composed services. During the process atomic services comprising a composed service are each initially rated as if they were executed standalone; then, in the second phase, a charge for the composed service reflecting modifications to atomic service charges is generated. To facilitate this process we use charging schemes consisting of two distinct parts: part 1 dictates how charges are to be calculated when the service is invoked in isolation (not as part of a composed service); whilst part 2 dictates any modifications to charges generated by application of part 1 in cases where the service is invoked in the context of a composed service.

Charging schemes can be associated with composed services as well as with atomic services. For composed services these schemes can consist of a part 1 only, a part 2 only, or a part 1 together with a part 2. Where a part 2 is present it dictates how charges associated with an invocation of that composed service are modified if that composed service is itself invoked as a constituent of a composed service. If a part 1 is present it is assumed to contain information necessary to process all of the metering records associated with all of the atomic services directly comprising that composed service, and will effectively replace the charging schemes associated with these services. In this way the charging model applied for the composed service can be made completely independent of the charging models associated with the services that constitute that composed service. Note however, that the charging schemes associated with the constituent services may still be applied in parallel with that of the composed service, so that providers of these services can be informed of the level of service usage undertaken – this information may be an important means of verifying that the payment they receive for this service invocation is correct.

We now outline the process in terms of both phases, assuming for simplicity that composed services do not have charging schemes containing a part 1 associated with them:

TABLE I. NOTATION FOR COMPOSED SERVICE CHARGING ALGORITHM.

| Notation | Constraint | Definition |
|---|---|---|
| $s_m$ | | The master composed service |
| $\{S_M\}$ | | The set of all services of which $s_m$ is comprised. |
| $M$ | $M \geq 2$ | The number of all services of which $s_m$ is comprised |
| $\{S_D\}$ | | The set of services of which $s_m$ is *directly* comprised (i.e. the top level set of services which are combined together to realise $s_m$) |
| $D$ | $2 \leq D \leq M$ | The number of services that directly comprise $s_m$ |
| $\{S_O\}$ | $\{S_O\} \subset \{S_M\}$ | The subset of services comprising $s_m$ that are rated by other rating engines |
| $s_i$ | $s_i \in \{S_M\}$ | an arbitrary service of which $s_m$ is comprised |
| $\{S_{s_i}\}$ | | The set of services of which $s_i$ is comprised |
| $n_i$ | $0 \leq n_i \leq M-2$ | The number of services that service $s_i$ is comprised of |
| $c_i$ | | The charge associated with service $s_i$ in the relevant interim charge record |
| $\Delta c_i$ | | The change in $c_i$ as a result of phase 2 of the rating process |
| $c_m$ | | The final charge associated with $s_m$ |

Inputs: $\{S_M\}$, $\{S_D\}$

Output: $c_m$

1: *Function:* **ratePhase2CompServ**( $\{S_X\}$ )
2: {
3: For all $s_i \in \{S_X\}$ such that $n_i > 0$ do:
4: **ratePhase2CompServ**( $\{S_{s_i}\}$ )
5: For all $s_i \in \{S_X\}$ such that $n_i = 0$ and $s_i \notin \{S_O\}$ do:
6: For all services $s_j \in \{S_X\}$, where $s_i \neq s_j$ do:
7: Calculate $\Delta c_{ij} = f(c_i, c_j)$, where $\Delta c_{ij}$ is the change in $c_i$ mandated by the presence of $s_j$ in $\{S_X\}$.
8: Set $\Delta c_i = \Delta c_i + \Delta c_{ij}$
9: Set $c_i = c_i + \Delta c_i$
10: Set $c_k = c_k + c_i$, where $c_k$ is the charge associated with the service $s_k$ for which $\{S_{s_k}\} = \{S_X\}$
11: }
12:
13:
14: *Start*
15:
16: Set $c_m = 0$
17: For all $s_i \in \{S_M\}$ do:
18: Set $c_i = 0$
19: Set $\Delta ci = 0$
20:
21: **ratePhase2CompServ**( $\{S_D\}$ )
22: return $c_m$

Figure 10. Composed service charging algorithm.

*1) Phase 1: rating atomic services as standalone services*

As the atomic service instances comprising the composed service are invoked and executed metering records relating to them are transferred to the rating engine (in the cases where the rating engine has direct access to the associated charging scheme). Every such metering record received by the rating engine is rated using part 1 of the charging scheme associated with that service, with the first one resulting in the generation of an *interim charge record* for the associated invocation of the atomic service instance. As well as the *chargeUnitType* and *chargeValue*, interim charge records include the *transactionID*, *providerID*, *serviceID, instanceID* and *invocationID.*

Subsequent metering records for this service instance invocation result in the generation of a charge delta and the *chargeValue* in the interim charge record is updated accordingly (note that in the vast majority of cases this delta will be an increment, not a decrement, but both are possible). The rating engine continues processing metering records in this manner until the service execution is complete. Note that many service instance invocations may result in the generation of only a single metering record.

*2) Phase 2: modification of interim charges and overall charge calculation*

In phase 2 the interim charges generated in phase 1 are modified in accordance with the part 2 of the services' charging schemes. Part 2 contains rules specifying how much a charge is to be incremented or decremented by if other services are present within the same composition group in the service composition hierarchy. (As discussed in §IV.D part 2 may actually be comprised of two separate rule sets: for successful and unsuccessful execution of the master composed service). The motivation for these rules is to allow providers specify discounts or penalties to incentivize or disincentivize composition of their services with other services. This is based on our expectation that service cost will be used as one of the primary criteria used by a service composer when choosing between services that provide similar functionality.

Discounts or penalties can be applied, for example, as fixed amounts, percentages, or as varying fixed amounts / percentages based on the absolute value of the interim charge. The rules can specify that they be applied based on the presence of services with specified combinations of *providerID*, *serviceID* and *instanceID*. Thus, a charging scheme part 2 could specify that a service is discounted by a percentage value if it is used with another service offered by the same provider, or that a fixed penalty be applied if the service is used with a service offered by a competitor.

The initial step for the master rating engine in phase 2 is to contact the relevant slave rating engines to ascertain the charges generated for those service instance invocations for which the rating engine did not itself perform phase 1 rating (see §IV.B). The slave rating engines then apply a slightly

adapted version of the algorithm discussed below to modify the charges for these service instance invocations (they treat the composition group as the overall composed service, but only modify the charge for the service instance invocation they themselves rated).

Once the master rating engine receives charge information for all the atomic services that were rated by slave rating engines it executes the recursive algorithm specified in Fig. 10 to calculate discounts/penalties and ascertain the overall charge for the invocation of the master composed service. The notation used to describe this algorithm is described in Table 1. The algorithm employs recursion, with the master composed service composition hierarchy being traversed so that charges related to composition groups lower in the hierarchy are processed before those at higher levels. Charges for these groups need to be calculated first, as they are in turn needed for calculation of charges at the higher levels.

When executed by the master rating engine, the algorithm produces a charge for the invocation of the master composed service. This charge is put into a charge record for the invocation, along with the *providerID* and *serviceID* of the master composed service, the *transactionID*, and any other information taken from the metering records that is required by the billing system, for example, the *consumerID* associated with the service consumer. At this point the master rating engine also sends a **ratingComplete** message to the workflow manager. The charge record is subsequently transferred to the billing system for further processing (depending on the implementation it may be streamed immediately, or send as part of a batch at a later time).

### *D. Error Handling*

In this section we address how errors in the execution of a composed service potentially impact on the charging process and describe how our framework handles critical errors. Clearly, during any service execution there is potential for both critical errors that force immediate halting of the service execution, and less serious errors that can be recovered from. The latter are less likely to impact directly on the charging process, since they can be expected to be reflected within the metering records relating to the service invocations in question. For example, an error that results in a lower level of delivered Quality-of-Service (QoS) to the user will be reflected in the metering records and handled accordingly by the service charging scheme, typically resulting in a reduced charge when that metering record is rated (in the normal manner).

On the other hand, errors that halt service execution severely effect how a composed service is to be charged for. Consider the case where a service halts due to error in the middle of a larger composed service execution: a number of (unrelated) services may have already completed fully or be partially completed, and a number of (unrelated) services may not have yet been invoked. The end user is unlikely to wish to be charged (even partially) for a composed service that, taken as a whole, failed; however, providers of services that were invoked and completed successfully will expect to receive payment as normal. To overcome this potential conflict of interest it can be considered essential that the providers and customer agree in advance a service level agreement (SLA) for the master composed service, specifying in detail how services will be charged for when critical errors occur.

When a critical error does occur during the execution of a particular service instance invocation other service instance invocations will be in one of the following states: (1) completed, (2) partially completed, or (3) not started. Clearly no charges will be applied for service instance invocations not started. However, for completed and partially completed invocations an number of options are possible:

1. charge for these services instance invocations as normal, i.e. based on the metering records received to date and the rules specified in the relevant charging schemes;
2. charge for these service instance invocations, but applying some form of discount (this may be realized by an alternative set of charging scheme rules to those normally applied);
3. do not charge for these service instance invocations.

Which of these options is to be applied to which service instances must be detailed in the master composed service SLA, and reflected in the corresponding charging schemes. Crucially, option 2 requires that the master rating engine is made aware that the error occurred, that is, that the master composed service did not complete execution successfully. As is evident from §IV.A.3) this is achieved by setting the *completionStatus* of the **executionComplete** message to "unsuccessful" and including a list of the *executionStatus* of each of the services.

Upon receipt of an **executionComplete** message indicating a critical error the master rating engine requests charge information from slave rating engines for services that were partially or fully completed. It then runs the phase 2 algorithm, flagging the error condition, so that the discount/penalty charging scheme part 2 rule set for error conditions are applied instead of the standard part 2 rule set. Note that these rules may allow providers specify discounts/penalties on the basis that their (successfully executed) service was composed together with one or more services that were not yet invoked when the error condition occurred.

## V. CHARGING FRAMEWORK IMPLEMENTATION

In this section we describe a prototypical implementation of the proposed charging framework. This implementation has involved the enhancement of a previously developed rating engine prototype called the Rating Bureau Service (RBS), and its deployment as part of the Digital Business Ecosystem (DBE) – a distributed, open-source business-to-business environment supporting advanced service recommendation and service composition.

### *A. Rating Bureau Service Enhancement*

The Rating Bureau Service was developed during the IST FP5 AlbatrOSS project [5] as rating engine exposed via a web services interface and adhering to the IPDR NDM-U protocol [6], a standard for describing and conveying rateable events for IP services in XML format (though it can be readily applied to

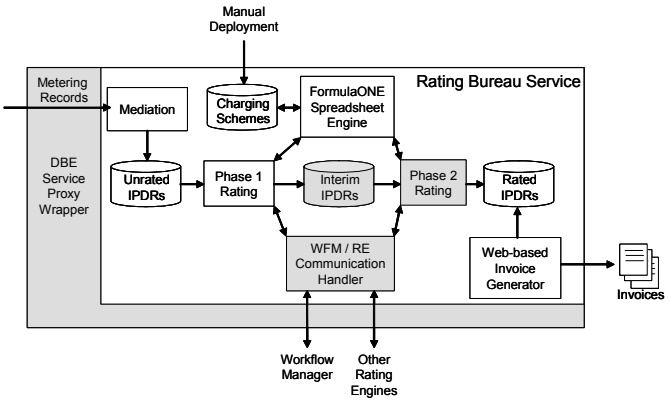


Figure 11. Enhanced RBS System Architecture.

arbitrary non-IP service types). The RBS also includes a web services implementation of the IPDR document transfer protocol for exchange of IPDRs between mediation and rating components. It performs mediation of metering records into IPDR records, rating of these records via a spreadsheet engine that employs charging schemes represented as Microsoft Excel compliant worksheets, and performs basic billing functionality through the generation of web-based customer invoices.

The RBS exposes an interface which allows for service elements or mediation systems to push IPDR usage data for subsequent charging. Several records, each detailing usage events of an atomic service instance, can be passed to the RBS per data transfer. For each of these records the RBS first stores the data record in a 'unrated' table in a database and then determines the appropriate charging scheme for the usage instance, extracts relevant values from the usage data and inserts these into the charging scheme. Charging schemes in the RBS are realized as worksheets containing named cells denoting which elements of the usage data should be inserted. The appropriate parameters are read from the record and inserted into the worksheet. The charge is calculated by a spreadsheet engine and then read from a 'charge' named cell. This resulting charge is inserted into the record in a charge element. Full details of the process are provided in [7].

The original RBS, as described above, provides the base functionality to realize phase 1 of the proposed rating process. To be compliant with our charging framework it was enhanced to support communication with the workflow manager and other rating engines (as either the master or slave), and, more significantly, the existing stateless rating process was enhanced to support stateful, two-phase rating. Finally, for deployment as a DBE service, the RBS was enhanced with a DBE service proxy wrapper to support communication with other DBE entities. The high-level system architecture of the enhanced RBS is shown in Fig. 11; shaded boxes represent the new sub-systems required for compliance with the proposed charging framework.

### *B. Deploying the enhanced RBS in the Digital Business Ecosystem*

The DBE project [8] is developing an open-source distributed environment ("the DBE") that can support the spontaneous creation of applications through the composition of (not necessarily open-source) software services and components. In doing so the project is adopting an approach based on business modeling techniques, complemented by evolutionary algorithms inspired by biological processes; the latter are intended to provide bottom-up incremental improvement of business models through run-time feedback on service performance. The DBE is being targeted primarily towards small businesses, which will be able to concatenate their offered services within service chains (compositions) formulated on a pan-European basis. By offering access to a large pool of service providers and consumers, and itself providing advanced recommendation systems and evolutionary algorithms, the DBE will support continued global optimization of service compositions, benefiting all of its participants [9].

From an architectural perspective the DBE can be viewed as consisting of a service factory environment and a service execution environment. Clients of the DBE will use the service factory environment to specify business models and generate associated software artifacts for subsequent implementation, composition and use. The service execution environments hosts implemented services, managing the process of registering, deploying, searching for, recommending, composing, retrieving and consuming services.

Fig. 12 illustrates the main component types constituting the DBE service execution environment. The Service Composer (SC) component is responsible for constructing service compositions to meet specific requirements captured in the DBE service factory environment. The Recommender component provides the service composer with ranked lists of services that could fulfill specific functions within a service composition. To rank services in this manner it uses historical data relating to service use and performance, which are stored in the Knowledge Base repository. This information is organized as a "fitness landscape," with fitness levels being assigned to services on the basis of a range of criteria; the fitness landscape is also used as the main input into the evolutionary algorithms that optimize service compositions throughout their lifetimes. Once service compositions are constructed, their actual execution is coordinated by the Transaction Workflow Manager (TWFM), which also coordinates the charging process.

During the execution of individual DBE services metering records detailing their utilization patterns are generated and forwarded to an accounting system (itself realized as a standard DBE service). The enhanced RBS, which realizes the charging framework presented in this paper, is incorporated into the DBE as the default accounting system service. At the time of

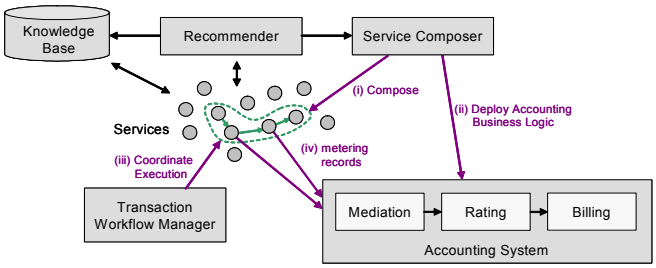


Figure 12. DBE service execution environment.

writing the DBE effort is ongoing, thus, full distributed trials of the operation of the environment have not yet occurred. However, in the next section we will discuss initial validation tests of the functional efficacy of the enhanced RBS in realizing our charging framework in the context of the DBE.

## VI. Example Use Scenario

In this section we show how an example multi-provider composed service is rated in our charging framework. The example service, depicted in Fig. 13, realizes an email client accessed via a GPRS-enabled mobile device when a user is roaming and cannot access their home email system. At the top level of the composition hierarchy the master composed service is comprised of a web-based email client GUI service, a functional email service, a virtual storage drive service (for storage of draft/received email attachments), and the GPRS data transfer service offered by the visited network operator. The functional email service is itself a composed service, comprised of a service offering access to the user's home IMAP server and a service offering access to an SMTP server in the visited domain.

As can be seen from Fig. 13 there are four separate providers involved in the composed service. The GPRS data transfer is provided by the visited network operator (provider D), which rates for this service using its own rating engine. The SMTP service is provided by a partner (provider C) of the user's home IMAP service provider (provider B); both offer a discount for using services of the other. The client GUI and virtual storage drive services are both provided by provider A and are assumed to have been selected for inclusion in the service composition based on a particular user preference. This is in spite of the fact that provider D imposes a penalty for the use of these two services (since it itself offers similar services). All provider A, B and C services are rated by a master rating engine, the use of which is not charged for. Fig. 14 summarizes the charging schemes associated with each of these services, specifying the discounts/penalties as outlined above.

To validate how the mobile email service is rated by our framework all the services except for GPRS were realized as prototype DBE services, for whom metering data is collected by DBE service proxy and forwarded to an instance of the RBS

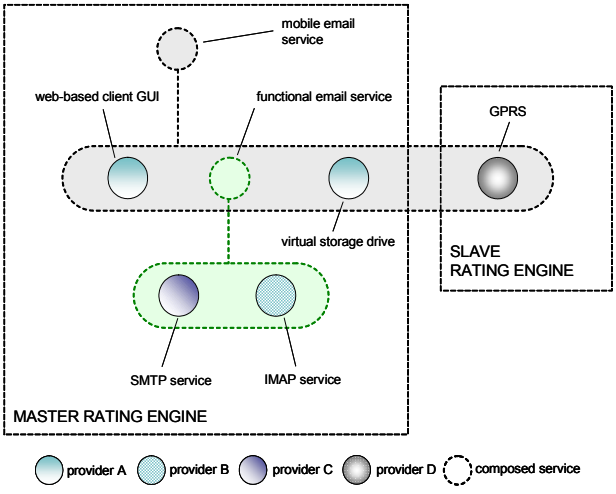


Figure 13. Example multi-provider composed mobile email service.

| *ProviderID: C* | *ServiceID:* SMTP |
|---|---|
| *Part 1:* | Charge = <NumberOfEmailsSent> * €0.06 |
| *Part 2:* | If *providerID*(<OtherService>) = B then:<br>Discount = Charge * 0.10 |

| *ProviderID: B* | *ServiceID:* IMAP |
|---|---|
| *Part 1:* | Charge = <NumberOfEmailsReceived> * €0.04 |
| *Part 2:* | If *providerID*(<OtherService>) = C then:<br>Discount = Charge * 0.10 |

| *ProviderID: A* | *ServiceID:* Web-based Client GUI |
|---|---|
| *Part 1:* | Charge = <HoursOfUsage> * €0.10 |
| *Part 2:* | If *providerID*(<OtherService>) = A<br>and *serviceID*(<OtherService) = VirtualStorageDrive<br>Discount = Charge * 0.20 |

| *ProviderID: A* | *ServiceID:* Virtual Storage Drive |
|---|---|
| *Part 1:* | Charge = <Num10MbBlocksPartiallyUsed> * €1.00 per week |
| *Part 2:* | |

| *ProviderID:* D | *ServiceID:* GPRS |
|---|---|
| *Part 1:* | Charge = <AmountOfDataTransferred> * €1.00 per Mb |
| *Part 2:* | If Provider(<OtherService>) = A then:<br>Penalty = Charge * 0.02 |

Figure 14. Charging scheme summaries for composed service example.

– the master rating engine. As shown in Fig. 15 a second RBS instance – the slave rating engine, was used to rate GPRS metering data, which was generated by a simple simulator to emulate the GRPS data transfer relating to a small set of usage patterns of the other four services. A prototype of a simplified DBE transaction workflow manager was used to ensure that the charging process was initiated and completed as expected.

Let us assume that in the course of a single day the user is logged into the mobile email service for 8 hours continuously, sending 25 emails and receiving 40 emails, corresponding to a total GPRS data transfer of 7.5 Mb, of which the user placed 3.5Mb of attachments on the virtual storage drive. The calculations during phase 1 and phase 2 of the rating process executed by the master and slave rating engines are:

Phase 1 – Master Rating Engine

| | | | |
|---|---|---|---|
| GUI Interim Charge | = | 8 * €0.10 | = €0.80 |
| SMTP Interim Charge | = | 25 * €0.06 | = €1.50 |
| IMAP Interim Charge | = | 40 * €0.04 | = €1.60 |
| Storage Interim Charge | = | 1 * €1.00 | = €1.00 |

Phase 1 – Slave Rating Engine

| | | | |
|---|---|---|---|
| GPRS Interim Charge | = | 7.5 * €1.00 | = €7.50 |

Phase 2 – Slave Rating Engine

| | | | |
|---|---|---|---|
| GPRS Delta | = | +(€7.50 * 0.02) | = €0.15 |
| GPRS Final Charge | = | €7.50 + €0.15 | = €7.65 |

Phase 2 – Master Rating Engine

Bottom Level (SMTP and IMAP)

| | | | |
|---|---|---|---|
| SMTP Delta | = | -(€1.50 * 0.10) | = -€0.15 |
| IMAP Delta | = | -(€1.60 * 0.10) | = -€0.16 |
| Func. Email Charge | = | €1.50 - €0.15 + €1.60 - €0.16 | = €2.79 |

Middle Level (GUI, storage, func. email, GPRS)

| | | | |
|---|---|---|---|
| GUI Delta | = | -(€0.80 * 0.20) | = -€0.16 |
| Storage Delta | = | | = €0 |

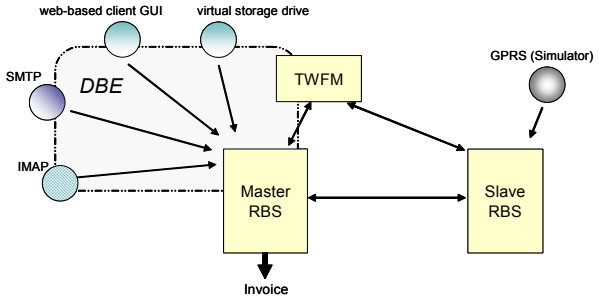


Figure 15. DBE based test-bed for charging framework validation.

Func. Email Delta = = €0

Master Composed Service Level:

Charge = €0.64 (GUI) + €2.79 (func. email) +
€1.00 (storage) + €7.65 (GPRS) = €12.08

For this case the final charge is €12.08, compared to an amount of €12.40 were the initial service charges simply summed (i.e. no application of discounts or penalties in phase 2 rating). The difference in final charge is relatively small in this case, however, we note that the main intention of the discounts/penalties would be to incentivize the composition of the services together in the first place.

## VII. Concluding Remarks

Traditional network accounting and charging systems are manually configured and rigorously tested prior to the activation of a service offering – a time consuming and expensive process. In many cases such service offerings are realized by compositions of a number of services, sometimes provided by different providers. However, since the nature and structure of these relatively small number of statically-defined service compositions is negotiated in advance, it is possible to configure accounting systems appropriately. In networks supporting ecosystems in which third party user-facing services can be arbitrarily composed together this will no longer be the case. Accounting and charging systems will have to be configured automatically and it will not be possible to predict the structure of service compositions. We argue that accounting logic, in particular charging schemes, relating to individual services should incorporate information relating to how these services are to be treated when composed with other services, and that accounting and charging systems must have the ability to automatically synthesize and apply this information as services are composed and consumed.

The charging framework proposed in this paper provides a model for how charging schemes can be dynamically combined to generate service usage charges that reflect business agreements between multiple providers in environments supporting service composition. However, support for the framework, requires that rating engine architectures are significantly redesigned in order to introduce support for communication with other entities and support for a stateful, two-phase rating process. The framework is generic in the sense that it does not depend on the presence of any specific underlying implementation technologies – if the services, workflow managers and rating engines realize the charging functionality described in the paper the framework could equally be applied in, for example, a web services environment or a 3G IP Multimedia Subsystem.

As described in the paper we have validated the proposed framework by enhancing the previously-developed RBS rating engine and deploying in the context of the Digital Business Ecosystem. Full trials with real DBE users are planned for 2006; these trials are expected to provide more concrete feedback on the efficacy, performance and scalability of the framework, in particular the two-phase rating process. Before these trials the RBS will be further enhanced, for example, to support automated deployment of charging schemes by the DBE service composer (as opposed to manual deployment in the current implementation). We also plan to address both the potential role of rating engines in providing service cost estimates that could be used in the DBE service composition process. Beyond that, we hope to expand our framework to incorporate facilities for flexible billing and payment for composed services.

## Acknowledgments

This work has been funded in part by: Science Foundation Ireland via the 2005 RFP grant no. CMS006; the Irish Higher Education Authority via the PRTLI M-Zones research program; and by the European Commission via the Digital Business Ecosystems integrated project (IST integrated project no. 507953). The authors wish to acknowledge the significant input of Jonathan Brazil, Jason Finnegan, Gary Gaughan and Frank Walsh into the implementation of the enhanced RBS.

## References

[1] B. Jennings, P. Malone and S. van der Meer, "A Two Phase Rating Process for Dynamically Composed Services," in Proc. 12th Workshop of the HP OpenView University Association (HPOVUA 2005), B. F. Marques, T. Nebe and B. F. Oliveira, Eds., pp. 155-171, 2005.

[2] M. Koutsopoulou, A. Kaloxylos, A. Alonistioti, L. Merakos, and K. Kawamura, "Charging, accounting and billing management schemes in mobile telecommunication networks and the Internet," IEEE Communications Surveys, vol. 6, no. 1, 2004, [Online], Available: http://www.comsoc.org/pubs/surveys [2006, January 9].

[3] B. Bhushan, M. Tschichholz, E. Leray and W. Donnelly, "Federated Accounting: Service Charging and Billing in a Business to Business Environment," in Proc. IFIP/IEEE Int'l Symp. on Integrated Network Management (IM'2001), pp. 107-121, 2001.

[4] V. Agarwal, N. Karnik and A. Kumar, "Metering and accounting for composite e-Services," in Proc. 1st IEEE Int'l Conf. on E-Commerce, pp. 35-39, 2003.

[5] AlbatrOSS IST project, "Architecture for Location Based Application of Third generation Operation Support System," [Online], Available: http://www.ist-albatross.org/ [2006, January 9].

[6] IPDR.org, "Business Solution Requirements - Network Data Management-Usage (NDM-U) For IP-Based Services," [Online], Available: http://www.ipdr.org/public/DocumentMap/ BSR-NDM-U3.5.0.1.pdf [2006, January 9].

[7] J. Brazil, E. de Leastar, C. Ryan and M. Ó Foghlú, "Workbook Approach to Algorithm Design and Service Accounting in a Component Orientated Environment," in Proc. IEEE Workshop on IP Operations and Management, 2002.

[8] Digital Business Ecosystems (DBE), EU IST Integrated Project No. 507953, [Online], Available: http://www.digital-ecosystem.org/ [2006, January 9].

[9] T. Heistracher, et al., "Pervasive Service Architecture for a Digital Business Ecosystem," in Proc. 1st Int'l Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT04), [Online], Available: http://wcat04.unex.es/wcat04/index.htm [2005, April 27]