

A Framework for Automated Creation and Deployment of Consolidated Charging Schemes for Service Compositions

Lei Xu, Brendan Jennings
TSSG, Waterford Institute of Technology
Cork Rd., Waterford, Ireland
{lxu, bjennings}@tssg.org

Abstract—If environments offering facilities for service composition are to be commercially successful then it will be important that service providers can avail of efficient, automated processes by which composed services can be metered, charged and billed for. In many industries, for example telecommunications, sophisticated systems are employed to provide usage and content based charging for services. In these systems charging schemes are typically manually configured and verified prior to services being made available to customers. For composed services we require a more dynamic charging approach, in which charging schemes for composed services can be automatically generated and deployed. In this paper we describe a framework that meets this need. In it, charging schemes for services making up a composed services are collected, consolidated and deployed onto multiple rating engines, which then coordinate to calculate a charge for an composed service invocation.

I. INTRODUCTION

Where customers are charged for use of a service, key considerations are how usage of the service will be metered, how charges are calculated based on metering data and customers' profiles, and how customers are billed for service usage. Over the years many industries have evolved complex business models based as much around how a services is charged, as around what functionality the service provides. A prime example is telecommunications, where a large number of charging approaches (time-of-day based, day-of-week based, geographical location based, zero charge for a "circle-of-friends," etc.) developed around what was essentially the same service (two-way voice communication).

To flexibly support a range of charging approaches, services providers typically deploy complex support systems to provide metering, charging, and billing functionality. Unfortunately this flexibility comes with a price – these systems must be manually configured every time a new service (or a new approach to charging for an existing service) is deployed. This is a time-consuming and expensive process that increases the time-to-market of services, which results in the undesired long-term inhibition of the level of service innovation in the marketplace. Clearly, in environments where services can be composed from other services the manual configuration approach will no longer be tenable. Instead, accounting and charging logic should

be automatically configured when service compositions are initially constructed, or subsequently modified.

When charging for composed services a straightforward approach would be to calculate individual charges for the constituent services as if they had been invoked in a standalone manner and add these together to arrive at an overall charge. Whilst simple and straightforward to implement, this approach ignores the commercial reality that services constituting a composed service may be offered by different providers and that these providers will want the commercial relationships between them to be reflected in the way services are charged for. For example, if two providers have a partnership agreement they may want to incentivize a service composer to use their services together by offering a discount – this common business practice is known as price bundling of service offerings [1].

In this paper we describe a framework that allows for charging for composed services in a manner consistent with business relationships between different service providers. In the framework, service charging schemes are assumed to contain rules governing how services are to be charged for when invoked in isolation and rules governing how discounts (for partnering providers) or penalties (for aggressively competing providers) are to be applied when the service is invoked in conjunction with other services. Schemes for individual services are consolidated to give an interim charging scheme for the composed service, which is then broken up into separate segments for deployment onto multiple Rating Engines (REs) – components that receive metering data and calculate charges. REs can communicate to calculate a charge for a composed service invocation, which can then be forwarded to downstream billing applications.

This paper builds on our previous work [2], which initially introduced our framework, by targeting a more realistic hierarchical model of composed services, which introduces additional complexity to the charging scheme consolidation process and inter-RE communications. It assumes that charging schemes are specified using a domain specific language such as that described in [3]. The paper is structured as follows: in §2 we describe the composed service model the framework is designed to support; §3 provides an overview of the framework, with a focus on the Accounting Logic

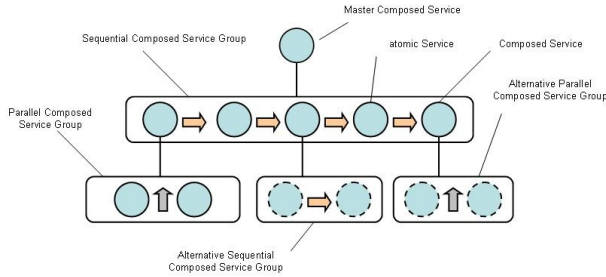


Figure 1. Composed Service Model.

Generator component; §4 specifies our charging scheme consolidation algorithm; §5 describes our charging scheme deployment approach; §6 briefly describes related work; finally, §8 draws conclusions and outlines areas for further work.

II. COMPOSED SERVICE MODEL

In this section we outline our model of multi-provider composed services, which attempts to capture all the aspects of composed services that are relevant for metering, charging and billing purposes. We model composed services as hierarchical structures of “atomic” and composed services, as depicted in Fig. 1.

In our model we consider an atomic service to be a service that appears to the metering, charging and billing system as a single service, such that metering data identifies it directly and that there exists a specific charging scheme for it. Such services may actually be realized as composed services, but this fact is masked by the provider for commercial reasons. In some cases the provider will want to generate the charge information for the service itself, either by maintaining its own RE, or by using a RE of a trusted third party. Thus, when the service is invoked as part of a composed service invocation its RE must be queried by the RE responsible for calculating the overall composed service usage charge.

In the more general case, the structure of a composed service will be transparent to the metering, charging and billing systems. This would typically be the case where an individual or company builds a service composition from services offered by other providers and offers this composed service within the service environment. In such cases, the atomic services constituting the composed service will themselves generate metering data every time they are invoked, so the charging and billing systems will need to be able to correlate metering data relating to given composed service invocations. We assume this is done via a unique `transactionID` value, which is shared across all individual service invocations and is passed with all metering data.

The framework presented in this paper supports four modes of execution of composed services:

- *Sequential*
Services are executed in a pre-defined sequence, where the outputs from one service are used as part of the inputs for the next service in sequence. From a charging perspective all metering data relating to individual service invocations are assumed to arrive in sequence at the relevant REs. If an error occurs, only those services whose execution has already completed will be an issue. There are two possible options: either the customer is charged for those services, on the basis that they complete successfully, or the user is not charged for anything, given that the composed service as a whole did not complete successfully. Both options will be appropriate in different circumstances, thus, whichever option is to be used should be specified by the service composer, agreed with the customer, and indicated in the charging scheme generated for the composed service;
- *Parallel*
Services have no dependence on outputs from other services, so can be executed in parallel. From a charging perspective metering data relating to individual services can arrive in any order at the relevant REs. If an error occurs all other services currently executing are likely to be aborted (of course some may have already completed). Once again, the decision as to whether to be charged for completed or partially completed services should be specified at service composition time and reflected in the composed service charging scheme;
- *Alternative Sequential*
Multiple services are available offering the same or similar functionality and these are invoked in a defined order, such that if the first invoked service fails, the next in sequence is invoked. This continues until one service in the sequence completes successfully. From a charging perspective it may be possible that a failed service has generated partial metering data (often long-lived service sessions result in the generation of interim metering data records), which the relevant REs should then ignore.
- *Alternative Parallel*
Multiple services are available offering the same or similar functionality and these are executed simultaneously, such that when one service returns successfully all other services are immediately aborted. From a charging perspective only metering data relating to the successfully returned service invocation should be acted upon and (interim) metering data for other services should be ignored.

We assume that all atomic services have charging schemes made of two parts: part 1 containing rules governing how the service is to be charged for when invoked in isolation and part 2 containing rules governing how discounts or penalties

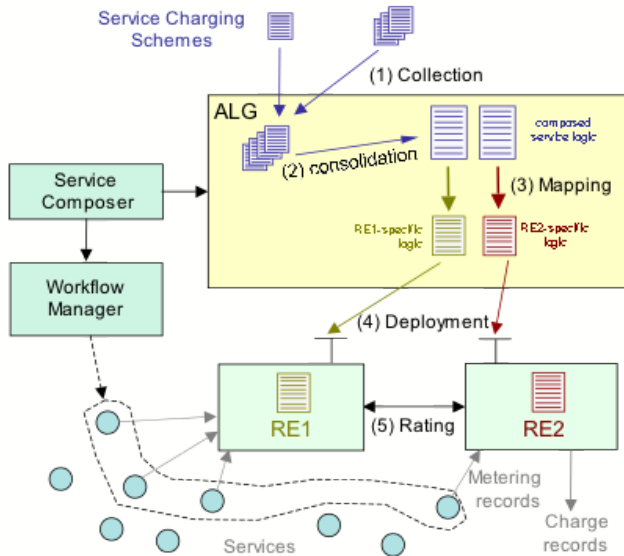


Figure 2. Framework Overview.

are to be applied when the service is invoked in conjunction with other services. We assume that all the part 2s are expressed using the same representation (such as the DSL described in [3]).

We assume all service instances have associated with them a `providerID`, which uniquely identifies, across the entire environment, the business entity (provider) offering that service. They have a `serviceID`, which uniquely identifies the service type within the set of service types provided by the provider 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 charging purposes). Therefore, the tuple of (`providerID`, `serviceID`, `instanceID`) uniquely identifies a service instance across the entire environment. The rules in the part 2s of the charging schemes can then use any combination of these three identifiers to specify discounts or penalties to be applied when the service in question is composed with other services.

III. FRAMEWORK OVERVIEW

The framework for flexible composed services charging is depicted in Figure 2. In the service environment a number of service providers offer their services, which can be used either on an individual basis or as part of service compositions. In this section we provide an overview of the functionality of the main components of the framework and the protocols for communication between them.

A. Service Composer

We take a broad view of the Service Composer (SC) as the entity that constructs a service composition and offers it to customers. In particular the contractual agreement for service usage will be between the customer and the SC, which in turn will have contractual agreements with third party service providers. For our framework the manner in which service compositions are constructed is not of relevance – it could be via a manual, semi-automated or fully automated process. The framework seeks to automate all configuration required for charging for a composed service once it has been constructed. If this can be achieved then it would be possible, in principle, to support real-time configuration of service charging logic when service compositions are themselves constructed on-the-fly.

We assume that all atomic services have associated with them a charging scheme, accessible via a URL. When the SC constructs a service composition it will provide information relating to the identification of the composed service, the structure of the composed service, the identification information for all the constituent services, and the URLs for their charging schemes to the ALG. Composed services are identified by a (`providerID`, `serviceID`) pair, where `providerID` uniquely identifies the SC and `serviceID` uniquely identifies that composed service amongst the set of composed services offered by that SC. Once the ALG has gathered all the charging schemes, consolidated them, and deployed charging scheme segments on the REs it responds to the SC with a success notification. In cases where more than one RE is to be used, the SC must indicate which will act as a "master," having responsibility to gather charge records from the other "slave" REs and forward a collated charge record either back to the SC or to a specified billing system. Having the SC specify this information means that REs will not need to be pre-configured with details relating to given service compositions.

Typically, a SC will charge the customer some commission for provision of the composed service. If the SC itself is handling the billing interaction with the customer then it can receive charge records from the REs following service invocations and add its commission to arrive at a final price. Alternatively, if the SC delegates the billing process to a third party it can include itself as one of the services constituting the composed service, passing its identification information and charging scheme URL to the ALG, which will treat it in the same manner as all other services constituting the composed service.

B. Workflow Manager

The Workflow Manager (WFM) is the entity responsible for coordinating the execution of individual composed service invocations. Our framework is independent of the manner in which this coordination is achieved – it could be via a workflow management component based on BPEL4WS

or WS-Choreography, or via a bespoke software component. The manner in which the SC provides service composition information to a WFM and the manner in which an WFM communicates with individual atomic service instances is outside the scope of our framework. However, we do assume that for each composed service invocation the WFM can generate a unique `transactionID` value, which is passed, along with the `(providerID, serviceID)` pair identifying the composed service, to the invoked atomic services, who subsequently include them in all metering data relating to this invocation. The `transactionID` value serves as a unique key which REs use to correlate metering data from all services relating to a given composed service invocation.

C. Accounting Logic Generator

The central component in our framework is the Accounting Logic Generator (ALG), which is responsible for generating charging logic to be deployed on a collection of REs to support a given composed service, whilst taking into account charge modifications introduced to incentivize or dis-incentivize particular service combinations. The ALG performs the following tasks:

1) *Collection*

Once the SC provides the ALG with details of the structure of the composed service the ALG retrieves charging schemes for all the atomic services comprising the composed service from the URLs indicated by the SC. If any charging schemes are not available an error is returned and the process is aborted;

2) *Consolidation*

The ALG analyses the set of collected charging schemes and generates an interim consolidated charging scheme for the composed service, which is split into segments for deployment on the REs involved in the charging process. Further details of the consolidation process are provided in section 4 below;

3) *Mapping*

Once a charging scheme segment for deployment on a given RE is generated it must be mapped from the common DSL representation to a representation that can be deployed directly to on that RE. Such representations will typically be scripting language such as TCL, or table-driven representations such as spreadsheets;

4) *Deployment*

When RE-specific charging scheme representations are generated the ALG will deploy them on the REs (this may be as simple as placing the scheme in a particular location in a file system). In addition to the information required to calculate charges, this scheme will contain information such as the `(providerID, serviceID)` pair identifying the composed service and a flag indicating whether the RE will act as a master or a slave.

We modify the charging schemes for services within a composition group based on the presence of other services, specified in the charging scheme, within the same group. This allows service providers to offer discounts or impose penalties when their services are used together, or in conjunction with services provided by business partners or competitors. To facilitate this our charging schemes for composed services have two parts: part 1 indicates how charges are to be calculated if that services is used in isolation, whilst part 2 indicates how the charging scheme is to be modified if that service is composed together with other specified services.

D. Rating Engine(s)

Rating Engines are used in a wide variety of application domains to map metered service usage data to monetary units based on various criteria, including time and duration of usage. For many (post-paid) session-based service types more than one "metering record" can generated during a session – one or more interim records, followed by a final record. In such cases the RE has the additional responsibility of correlating and collating this data before calculating a charge. Once this calculation is complete the RE forwards a "charge record" to a downstream billing system (which may be maintained by the SC).

In our framework we assume REs can be associated with individual services, individual service providers, or can be provided as third party services themselves. Service providers may dictate that their services can only be rated using specific REs, hence there may need to be more than one RE involved in the calculation of the charge for a given composed service invocation. In such cases one RE will need to act as a "master" RE, having the responsibility to collate the charge records generated by itself and the slave REs and forward the collated charge record to the billing system. Details of the operation of the REs are provided in section 5 below.

E. Inter-Component Protocols

In this section we specify two protocols involved in the operation of our framework. The first refers to the deployment by the ALG of charging schemes for a composed service specified by the SC onto one or REs, whilst the second refers to the rating process between multiple REs. The abbreviations we use for message payload items are listed in Table 1.

For the charging scheme deployment protocol the SC must first pass the structure of the composed service to the ALG, which then retrieves the atomic service charging schemes, deploys consolidated charging scheme segments on the REs, and acknowledges successful deployment to the SC. The interaction is as follows:

Table I
MESSAGE NOTATION

Notation	Description
P	providerID
S	serviceID
I	instanceID
T	transactionID
RE_x	identifier of rating engine x
M	identifier of master RE
f_M	flag indicating master RE
RE_M	master rating engine
RE_S	slave rating engine
$AS_x = \{P, S, I, RE_x\}$	atomic service x information
URL_{AS}	URL for charging scheme
cs_x	service charging scheme
E	execution mode
$CS_x = \{E, AS_1, \dots, AS_n, CS_1, \dots, CS_m\}$	info for 1 composed service
$CS_{info} = \{CS_0, \dots, CS_y\}$	full hierarchy info
$CS_{id} = \{P, S\}$	composed service identifier
f_{compl}	completion status flag
cs_{RE}	charging scheme for RE
CR_{RE}	RE specific charge record
CR_{CS}	consolidated charge record
B	identifier of billing system

1. $SC \longrightarrow ALG$: CS_{id}, CS_{info}, M, B
2. $ALG \longrightarrow URL_{AS}$: AS_x
3. $ALG \longleftarrow URL_{AS}$: cs_x
4. $ALG \longrightarrow RE_x$: $CS_{id}, CS_{info}, cs_{RE}, f_M, B$
5. $SC \longleftarrow ALG$: f_{compl}

For the rating process we assume that the WFM will inform the master RE of the completion of a composed service invocation. If the completion was successful the master RE will retrieve charge records from slave REs, generate a consolidated charge record and forward it to the billing system. This interaction is as follows:

1. $WFM \longrightarrow RE_M$: CS_{id}, T, f_{compl}
2. $RE_M \longrightarrow RE_S$: T, f_{compl}
3. $RE_M \longleftarrow RE_S$: T, CR_{RE}
4. $RE_M \longrightarrow B$: CS_{id}, T, CR_{CS}
5. $WFM \longleftarrow RE_M$: f_{compl}

If the composed service invocation was unsuccessful rating engines will immediately initiate the calculation of charge records in accordance with the rules specified in the charging scheme they use. These rules may indicate that completed atomic service invocations invoked as part of the service are to be charged fully, charged partially, or not charged at all.

IV. CHARGING SCHEME CONSOLIDATION

As noted in section 1 the most intuitive approach to calculating a charge for the invocation of a composed service is to sum the charges associated with the invocation of the individual services constituting the composed service.

However, service providers are like to use discounting to incentivize service composers and customers to use more than once service provided by them or their partners. Conversely, they may apply penalties as a means of disincentivising the use of their services together with services provided by their competitors. This will certainly be the case where many functionally equivalent services can be chose between.

To facilitate this form of incentivization in our framework we introduce the concept of two-part charging schemes. Part 1 of a scheme indicates how charges are to be calculated if the service is used in isolation (ie. it is equivalent to a standard charging scheme), whilst part 2 indicates how the charging scheme is to be modified if the service is composed together with other named services. Part 2 contains rules allowing two types of charging scheme modification. In the first, the rates in part 1 of the charging scheme can be changed (for example they can be reduced by a specified percentage). This is termed "rate_change." In the second, the charges generated after the part 1 rates are applied can be incremented or decremented by a specified absolute value (this is equivalent to adding a final step to the charge calculation process). This is termed "value_change."

Charge modifications can be applied more than once for a given service in a composition group (for example, if a service is composed with two partner services the charging scheme can be modified so that the sum of two absolute increments is applied to a charge). However it is important to note that all rate change modifications are expected to applied before any value change modifications. In this paper we assume that all atomic service charging schemes collected by the ALG are specified in a common representation – in our work we use the domain specific language described in Jennings et al.[3]. However, in practice, it may be possible to use model transformation techniques to map different charging scheme representations to a common representation that could be processed by the ALG.

The charging scheme consolidation process has two steps. In step 1 sets of charging schemes relating to the services constituting individual composed service groups within the overall composed service hierarchy are analysed on a pairwise basis. If, for a given service pair, the charging scheme part 2 of one of the services contains a rule relating to the application of a discount or penalty based on the presence of the other service in the composed services, a new rule reflecting the modification is added to part 1 of that service's charging scheme. In addition, the composed service identification information is added to part 1 of the scheme – so that REs know that this version of the service charging scheme should only be applied when the service is being executed in the context of that composed service. Note that in the current version of the framework we assume that composed services in the service hierarchy do not have separate charging schemes associated with them – thus charge modification rule can not be associated with them.

Table II
ADDITIONAL NOTATION

Notation	Description
CS_0	identifier of a composed service (top of service hierarchy)
$\{cs_x\}$	set of all charging schemes for services in CS_0
$\{s_x\}$	set of all services (atomic and composed) constituting a composed service
$cs_x[Part1]$	part 1 of charging scheme cs_x
$cs_x[Part2]$	part 2 of charging scheme cs_x
r_{s_j}	charge modification rule relating to service s_j
n_{s_j}	new charge modification rule added to $cs'_{s_i}[Part1]$ as a result of presence of compositing with s_j
$\{cs_{RE}\}$	set of RE-specific charging schemes

In step 2 of the process, all charging schemes generated in step 2 are grouped on a per RE engine basis (we assume that each charging scheme indicates one and one only RE that is used to calculate charges for invocations of that service). These groupings are then passed to the Mapping sub-component, which maps charging schemes to the representations used by different rating engines. The consolidation algorithm is specified in Alg. 1, using notation described in Table I and Table II. The consolidation process is illustrated graphically in Fig. 3.

A. Consolidation Algorithm Complexity

If we let N represent the number of services constituting a composed service (including the service itself – at the top of the service hierarchy). Let M , where $1 \leq M \leq N$, represent the number of composed services contained within the group of services. Let n represent the mean number of services that constitute the M composed services. For each composed service the consolidation algorithm compares the charging scheme of each service with the charging scheme of every other service constituting that composed service, giving $n(n-1)$ comparisons. We have M composed services, so that the total number of comparisons is $M \cdot n(n-1)$. Thus, the algorithm is $O(M \cdot n^2)$, or $O(n^2)$ when $n \gg M$. We would expect that in most cases the algorithm would be run on a flat composition hierarchy, ie. a single composed service constructed using composed services, in which cases the algorithm is quadratic in the number of services making up the composed service. We believe this is acceptable since the algorithm will be typically be run offline and the number of services making up a composed service will be in the tens at the very most.

B. Consolidation Example

We now present a simple example of how the consolidation process acts to modify charging schemes of services that form part of a composed service. A simple online music store service composition is depicted in Fig. 4. The service is composed from two services, a search/listing service and a music sale service, both provided by provider “Xu Inc.”

Algorithm 1: Consolidation Algorithm.

Input: $CS_{info}, \{cs_x\}$
Output: $\{cs_{RE}\}$

```

1 Function consolidate( $\{s_x\}$ )
2 // recurse down service hierarchy
3 forall  $s_i \in \{s_x\}$  do
4   if  $s_i$  is not atomic then
5     consolidate( $\{s_i\}$ )
6 // initialise new charging schemes
7 forall  $s_i \in \{s_x\}$  do
8   Set  $cs'_{s_i}[Part1] = cs_{s_i}[Part1]$ 
9   Set  $cs'_{s_i}[Part2] = NULL$ 
10  Add  $CS_{info}$  to  $cs'_{s_i}[Part1]$ 
11 // apply modification rules
12 forall  $s_i \in \{s_x\}$  do
13   forall  $s_j \in \{s_x\}$ , where  $s_j \neq s_i$  do
14     if  $\exists r_{s_j} \in cs_{s_i}[Part2]$  then
15       if  $type(r_{s_j}) = \text{“rate\_change”}$  then
16         Create  $n_{s_j}$  and add to  $cs'_{s_i}[Part1]$ 
17       elseif  $type(r_{s_j}) = \text{“value\_change”}$  then
18         Create  $n_{s_j}$  and add to  $cs'_{s_i}[Part1]$ 
19 return
20 Function segment( $\{s_x\}, \{cs_{s_i}\}$ )
21 Set  $\{cs_{RE}\} = NULL$ 
22 forall  $s_i \in \{s_x\}$  do
23   Read  $RE_{s_i}$  from  $cs_{s_i}$ 
24   if  $\nexists cs_{RE_{s_i}}$  then
25     Create  $cs_{RE_{s_i}}$ 
26     Add  $CS_{info}$  and  $cs_{s_i}$  to  $cs_{RE_{s_i}}$ 
27     Add  $cs_{RE_{s_i}}$  to  $\{cs_{RE}\}$ 
28   else
29     Get  $cs_{RE_{s_i}}$  from  $\{cs_{RE}\}$ 
30     Add  $cs_{s_i}$  to  $cs_{RE_{s_i}}$ 
31 return  $\{cs_{RE}\}$ 
32 begin
33 forall  $s_i \in \{s_x\}$  do
34   Set  $cs'_{s_i} = NULL$ 
35   Retrieve  $CS_0$  from  $CS_{info}$ 
36   consolidate( $CS_0$ )
37    $\{cs_{RE}\} = segment(\{cs'_{s_i}\})$ 
38 end

```

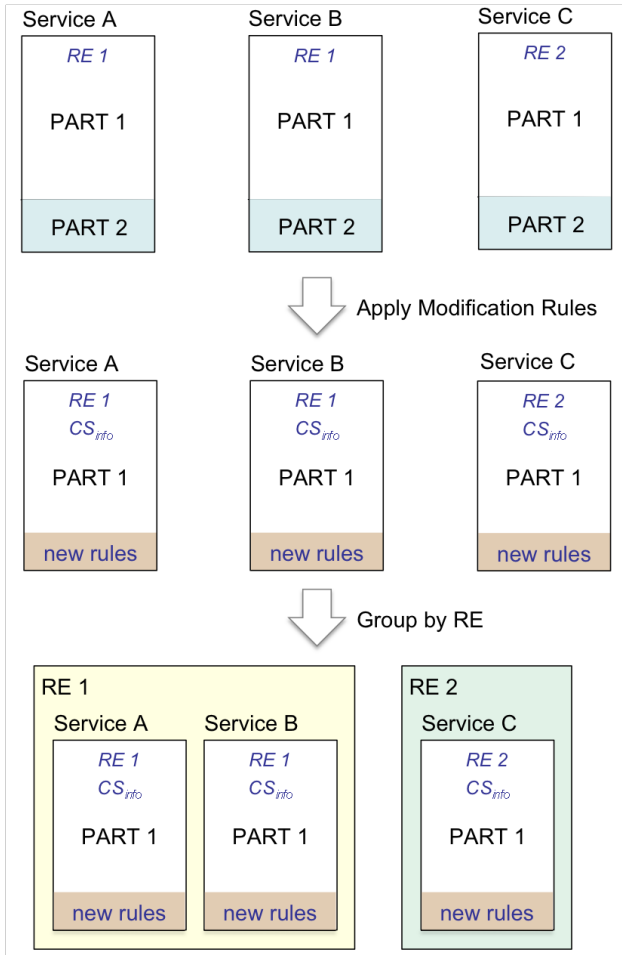


Figure 3. Consolidation Process Illustration.

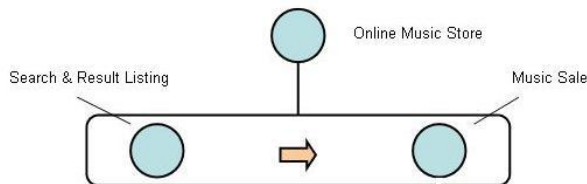


Figure 4. Example Composed Service.

The charging scheme for the music sale service (expressed using a Groovy based DSL), which states in Part 1 that every invocation (an album sale) carries a fixed charge of 8.99 currency units, is shown in Fig. 5.

The charging scheme for the search and listing service, shown in Fig. 6, states in Part 1 that every invocation carries a fixed charge of 1.00 currency unit, but, via Part 2, that this price is reduced by 0.50 currency units if the service is used in conjunction with the music sale service also provided by Xu Inc. The charging schemes for both services indicate that

```
// Music Sale Service
class Music_Sale extends Service{
  void scheme(){
    def builder=new ServiceBuilder()
    builder.Music_Sale(){
      SERVICEID_is "Music_Sale"
      INSTANCEID_is "Ireland"
      PROVIDERID_is "Xu_Inc"
      RATINGENGINEID_is "Jennings_RE"

      PART_1 =
      {
        CHARGE_is (8.99)
      }
    }
  }
}
```

Figure 5. Music Sale Service Scheme.

```
// Search List Service
class Search_Listing extends Service {
  void scheme(){
    def builder=new ServiceBuilder()
    builder.SR_Listing(){
      SERVICEID_is "Search_Listing"
      INSTANCEID_is "Ireland"
      PROVIDERID_is "Xu_Inc"
      RATINGENGINEID_is "Jennings_RE"

      PART_1 =
      {
        CHARGE_is (1.00)
      }

      PART_2 =
      {
        if (OTHERSERVICE_is("Xu_Inc"
          , "Mu_Sale", "Ireland"))
          DECREMENT_CHARGE_by 0.50
      }
    }
  }
}
```

Figure 6. Search and Listing Service Scheme.

the rating engine “Jennings_RE” is to be used for charging for invocations of these services.

The charging scheme shown in Fig. 7 is the result of the step 1 of the consolidation algorithm. We see that the composed service to which this charging scheme now relates is identified (for brevity the full listing of the structured is omitted here). Part 2 of the original search and listing charging scheme no longer appears. However, we see that a new rule is added to Part 1, instructing that the charge value that would have normally been generated is to be decremented by 0.50 currency units.

```

// Search List Service
class Search_Listing extends Service {
  void scheme() {
    def builder=new ServiceBuilder()
    builder.Search_Listing() {
      SERVICEID_is "Search_Listing"
      INSTANCEID_is "Ireland"
      PROVIDERID_is "Xu_Inc"
      RATINGENGINEID_is "Jennings_RE"

      COMPSERVINFO_is(
        "Online_Music_Store",
        .../* structure */)

      PART_1 =
      {
        // old rules
        CHARGE_is (1.00)

        // new rules
        CHARGE_is (CHARGE_is.decr
                   (0.50))
      }
    }
  }
}

```

Figure 7. Post-consolidation Search and Listing Service Scheme.

V. RELATED WORK

Given the critical importance of accounting and charging to any business, numerous standardization bodies have specified standards for charging systems, processes and protocols for a range of application domains. For example, in telecommunications, 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 [10]. 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.

Agarwal et al. [8] propose a method for metering and accounting for composite e-services, which is not dependent on prior 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) and the charge a composed service invocation is always the summation of the charges associated with standalone invocations of the constituent services. Taylor et al. [9] propose an event-driven architecture for billing in service-oriented architectures, focusing in particular on multi-organizational federated information services. Their

architecture incorporates a transaction manager within the customer-facing value-added service component, which coordinates the charging and billing process by interacting with the source information services (who calculate charges for their individual usage). Again, the total charge will be the summation of the charges calculated by the individual services, who will not be aware of the other services they are being used in conjunction with.

Günther et al. [4] address strategies for the pricing of composed web services, but without discussing the mechanics of the metering, charging and billing process. They conducted a series of online experiments involving over 200 participants in an effort to ascertain attitudes towards consumption and payment for pre-packaged composed services in comparison to self-composition options. Their overriding conclusion is that “most potential customers expect to pay less for a composite service than the total of the components’ prices.” They speculate that this is because of the belief that a third part service aggregator should be able to negotiate discounts with service providers that individual customers would not be able to access. This highlights the importance the need for a flexible composed service charging process such as the one proposed here.

van Hee et al. [5] propose a framework for service composition in SOA environments that addresses the proper termination of services and *a priori* calculation of the expected cost of an offered service. The expected cost calculation takes into account that different third party services will be able to meet different needs, but that each will have a different cost / success probability profile. Their algorithm for expected cost calculation is intended for use in assisting a service aggregator ascertain a suitable price point for the offered composed service. In comparison our approach facilitates the automated computation of the composed service charging scheme – no manual intervention is required. Many researchers have addressed the problem of building service compositions to satisfy non-functional constraints such as security, quality-of-service and cost; see for example Paoli et al. [7] and Zeng et al. [6], however all assume that the process of setting the price for the resulting service is a separate task requiring human intervention.

VI. CONCLUSIONS AND FUTURE WORK

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. 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 the framework could be applied in different environments.

Future work will initially concentrate on extending the functionality of the consolidation process. Firstly we will enhance the process to allow it support SC's specifying charge modification rules for the composed services they construct. Application of these rules when the composed service is itself composed will be more complex than in the current process, as multiple service charging scheme part 1s will need to have rules added to them.

Secondly, we will enhance the process to support temporal conditions in charging schemes. In the current framework charging scheme do not include temporal rules, for example, specifying that a particular rate should be applied with a specific time window, at a particular time of the day and/or on particular days of the week. Consolidation of multiple charging schemes with rules covering overlapping time periods will be more complex, involving the identification of a complete set of contiguous time intervals and the generation of charge modification rules for each of these regions. This will add significant complexity, however we believe it is required, given the prevalence in the marketplace of time-based charging models.

Finally, in the longer term we also plan to address the role of rating engines in providing service cost estimates that could be used in the service composition process. Such estimates would take into account the inter-provider discounts/penalties specified in the part 2s of individual service charging schemes.

ACKNOWLEDGEMENT

This work has been funded in part by Science Foundation Ireland, via 2005 Research Frontiers grant no. CMS006 and the 2008 "FAME" Strategic Research Cluster grant no. 08/SRC/I1403.

REFERENCES

- [1] H. Simon and G. Wuebker, "Bundling – A Powerful Method to Better Exploit Profit Potential," in R. Fuerderer, A. Herrmann and G. Wuebher, eds., *Optimal Bundling (Marketing Strategies for Improving Economic Performance)*, Springer, Heidelberg, 1999, pp. 7-30.
- [2] L. Xu and B. Jennings, "Automating the Generation, Deployment and Application of Charging Schemes for Composed IMS Services," in *Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, vol. 10, no. 1, 2007, pp. 856-859.
- [3] B. Jennings, L. Xu and E. de Leastar, "Specifying Flexible Charging Rules for Composable Services," in *Proc. 2008 IEEE Congress on Services - Part 1 (Services 2008)*, 2008, pp. 376-383.
- [4] O. Günther, G. Tamm and F. Leymann, "Pricing Web Services," *Int. J. Business Process Integration and Management*, vol. 2, no. 2, 2007, pp. 132-140.
- [5] K. M. van Hee, N. Sidorova, C. Stahl, and H. M. W. Verbeek, "A Price of Service in a Compositional SOA Framework," *Computer Science Report 07/16*, Technische Universiteit Eindhoven, 2007.
- [6] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition", *IEEE Trans. Software Eng.*, 2004, vol. 30, no. 5, pp. 311-327.
- [7] F. D. Paoli, G. Lulli, and A. Maurino, "Design of quality-based composite web services," in *Proc. 4th Int. Conf. on Service Oriented Computing (ICSOC 2006)*, LNCS 4294, Springer, 2006, pp. 153-164.
- [8] M. Agarwal, N. Karnik and A. Kumar, "Metering and accounting for composite e-services," in *Proc. 1st IEEE Int. Conf. on E-Commerce (CEC 2003)*, IEEE, 2003 pp. 35-39.
- [9] K. Taylor, T. Austin and M. Cameron, "Charging for information services in Service-Oriented Architectures, in *Proc. IEEE Int. Workshop on Business Services Networks (BSN 2005)*, IEEE, 2005, pp. 16-23.
- [10] 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.