Specification of Transactional Requirements for Web Services Using Recoverability

Kanchana Rajaram, SSN College of Engineering, Anna University, Chennai, Tamil Nadu, India
Chitra Babu, SSN College of Engineering, Anna University, Chennai, Tamil Nadu, India
Arun Adiththan, City University of New York (CUNY), New York, NY USA

ABSTRACT

In Service-Oriented Computing (SOC), a business transaction comprises of several web services provided by multiple enterprises. The transactional behaviour of individual web services must be considered for service selection so that the composition of web services results in a reliable execution. It is difficult for a business analyst to envisage the desired business policies of a process in terms of transactional properties of the corresponding service. Hence, an abstract mechanism that enables the business analyst to specify the transactional properties in a simple manner must be introduced. Towards this objective, it is proposed to express the transactional properties in terms of the recoverability of services. The transactional web services are grouped into different levels of recoverability based on their recovery cost. The estimated recovery costs are empirically verified and validated.

Keywords: Composition, Recoverability, Recovery, Transactional Property, Web Services

INTRODUCTION

In Service-Oriented Computing (SOC) (Singh, 2005) environment, web services composition has gained momentum over the past decade since business services primarily focus on their core functionalities and there is a tendency to outsource sub-functionalities to third party providers. In this case, multiple web services offered by different business organizations and deployed over Internet must be dynamically composed (Chan, 2008). Failures are inevitable during the execution of such a composite service due to dynamic and volatile nature of the Internet. In the event of a service failure, it is important to restore the execution of composite service to a consistent state by adapting an appropriate recovery mechanism (Erl, 2005; Erradi, 2006). The recovery mechanisms are based on the behavioural dependencies (Bhiri, 2005) among the component services which determine their transactional capabilities. For instance, payment
and order processing are behaviorally dependent i.e. when the order is canceled, the payment for the order may need to be refunded completely or partially. This behavioral dependency determines the transactional capability that needs to be possessed by both the services i.e. payment service must be compensatable and order processing service must be cancellable. In order to achieve failure tolerant or reliable execution of a composite service, each of the component services must be selected with appropriate transactional capabilities so that the desired recovery mechanism can be applied in case of its failure during execution (Haddad, 2010). Such transaction aware selection of services requires transactional capabilities or properties of a service to be specified in its service description (Rajaram, 2011). In addition, the desired transactional properties of services must be obtained from business analyst and matched with the transactional capabilities of the advertised services in the service registry in order to select suitable ones. However, it is very difficult for business analysts to visualize or map the business policies as transactional properties of the services and hence it is essential to enable them with an abstraction that is simple and flexible.

EXISTING WORK

The Web Service Conversation Language (WSCL) (Banerji, 2002) and the Web service Choreography Interface (WSCI)(WSCI, 2002) offer conversational meta-models that describe the external behaviour of a service in terms of the acceptable sequence of web service invocations. In addition, WSCI supports message correlation, message choreography, and service operation compensation. Web service transaction protocols such as Web Service Coordination (WS-C) (Carera, 2005), Web Service Transaction (WS-T) (WST, 2005) and Business Transfer Protocol (BTP) (Ceponkus, 2002) propose specific conversations that can be used to coordinate interacting parties and provide transactional properties. However, they do not provide a set of relevant abstractions to model service behaviors.

Web Services Transaction Language (WSTL) (Pires, 2003), a language extension over BPEL (Jordan, 2009) that provides transactional description support. WSTL defines its root element, transactionDefinitions, as a direct child of the wsdl:definitions element. The transactionDefinitions element has two child transactionBehavior elements each containing transaction semantics information on the operations supported by that Web service. The transactionBehaviour element has a type attribute whose values can be compensable, virtualcompensable, retriable, and pivot. These values are not much meaningful to the business analyst and the pros and cons of a service with a specific type of behaviour are not visible. The transactional behaviour type or property does not provide any abstraction for the business analyst to choose a service with an appropriate behaviour.

Jiuxin et al. (2010) defined conception-constrained rules for expressing the business logic of a transaction. In the absence of a suitable abstraction, it is a tough task for the business analyst to express the business requirements in the form of constraint rules.

Wang et al. (2007) proposed Tx-QoS attributes to be specified in a transactional SLA that is mutually agreed by the consumer and the developer. TxQoS attributes include Fluency that defines how often a process fails, Alternation that states what options the consumer prefers in case of interruptions, Transparency that defines the visibility of a service, and Interferability that defines the level of influences a consumer has over a service. However, these attributes do not capture all the behavioural properties of transactional services and a mapping of these TxQoS attributes to behavioural properties is missing. Moreover, a methodology to validate the TxQoS approach is not suggested.

Mikalsen et al. (2002) proposed the notion of transactional attitudes in which a centralized middleware monitors and controls client transactions according to transactional capability advertised by the providers. Most of the
existing works on transaction aware selection of services require the business analyst to specify the transactional dependencies or properties which is a difficult task.

Gaaloul et al. (2010) specified the transactional dependencies among services using Event Calculus (EC) predicates to verify and validate the consistency of composition at design time and after runtime. Even though, a parser is proposed that transforms transaction dependencies into EC rules, specification of all kinds of dependencies between every pair of services is a tough task.

Chalin et al. (2008) presented an extension for use cases that enables accurate capturing of transactional requirements for individual business transactions. However, the extension is not suitable for long-running transactions and it does not support specification of transactional properties or transactional dependencies.

Benetallah et al. (2004) identified a framework that builds on current standards to help developers define extended service models and richer web service abstractions. The conversation meta-model provided by their framework can be analyzed only after the functional requirements are known. The abstractions do not relate to recoverability and recovery cost of services. In contrast, the proposed recoverability level concept in this paper abstracts the transactional properties of services based on their recovery cost.

The cost incurred while substituting a failed service with an alternate service along with or without compensation of completed services was derived by Yin et al. (2011). However, this work does not consider services composed in parallel. Moreover, costs for other types of recovery mechanisms such as forward and cancellation were not addressed.

**MOTIVATION**

Most of the existing works on specification of transactional properties require the business analyst to write rules or dependencies which is a difficult task. The absence of a suitable abstraction for transactional properties motivated us to propose a concept of recoverability for web services that abstracts the transactional requirements. The recovery costs of services with different transactional properties are also derived. It enables gradation of recoverability into levels and proper grouping of services under different recoverability levels.

The abstraction of transactional requirements provides the expressiveness to describe the behaviour of services. Since the abstraction is based on the cost of recovery, the business analyst is better aware of the recovery cost while specifying the transactional requirements. The abstraction of recoverability levels helps in finding alternate services that are functionally equivalent and behaviourally compatible when the required service is not available. In addition, abstractions are useful to derive a set of required transactional properties of component services which will in turn be used to derive the desired behaviour at the composite service level.

**BACKGROUND**

The behaviour of a web service is represented by its transactional property. Selection of component services with appropriate transactional properties results in reliable execution of the composite service. The extended transaction model (Elmagarmid, 1990; Mehrotra, 1992; Zhang, 1994) for Multi Database Systems (MDBS), proposed CPR model comprising of three transactional properties namely compensatable (cp), pivot (p), and retriable (r). A successfully completed service having compensatable property can be semantically undone upon failure of another service. A retriable service can be invoked again in case of its failure during execution. A retriable service is guaranteed to succeed, since it can be retried until successful completion. A pivot service is neither compensatable nor retriable. A pivot service is guaranteed to succeed if another service fails. A retriable service is not guaranteed to succeed but once it completes successfully, it cannot be semantically undone. A retriable service, after its successful completion, may be capable of semantically undoing its
effects upon failure of another service in which case, it is compensatable retrievable with properties $cpr$. Alternatively, a retrievable service which is incapable of performing such a rollback, is pivot retrievable with properties $pr$.

In general, B2B applications involve composition of several long-running business processes (Dayal, 1991) that span from minutes to days. The business requirements or business policies are volatile and change very frequently according to the market trends. Whenever business policies change, the corresponding long-running services need to be terminated in order to incorporate the changed policies since the execution of the service to completion with the older policies is no longer meaningful. Hence, in addition to the properties supported by the CPR model, $cancellable (cc)$ property has been considered to enable external interruption of a service. A cancellable service capable of semantically undoing its effects after its successful completion, upon failure of another service is compensatable cancellable with properties $cpcc$. Alternatively, a cancellable service which cannot be compensated is pivot cancellable with properties $pcc$. A pivot cancellable service which is guaranteed to succeed by retrying it for a finite number of times due to internal failures is cancellable retriable having properties $pccr$. When successful completion of a cancellable retrievable service can be undone, it exhibits properties $cpcr$. Therefore, the set of possible transaction properties for a service is $\{p, pr, cp, pcc, cpr, pcr, cpcc, cpcr\}$. A web service is termed as a transactional web service, if it exhibits one of these transactional properties.

Multiple services that are composed either in sequence or in parallel result in a composite service whose transactional property depends on the transactional properties of the component services. A composite service cannot have pivot property since it represents a composition of multiple component services. However, a composite service can be atomic ($a$) (Haddad, 2010) if it cannot be rolled back on successful completion. In case of a failure or cancellation in the middle, the completed component services are compensated to reach a consistent state. Similar to a non-composite service, composite service has a set of possible transaction properties $\{a, ar, cp, acc, cpr, accr, cpcc, cpcr\}$.

While executing a composite service, if one of the component services fails, it needs to be recovered to a consistent state by invoking it again or by undoing a previously completed component service. Forward recovery refers to retrying the same service or another functionally equivalent service until successful completion. A retrievable service enables forward recovery and thus is guaranteed to succeed. In a composite service, the process of undoing or compensating the results of all successfully completed component services due to the failure or compensation of a subsequent component service is termed as backward recovery. The completed services are compensated by executing their corresponding compensating logic.

Whenever a long-running cancellable component service is interrupted, termination of its execution may lead to an inconsistent state in the composition. This concern must be addressed by an appropriate recovery mechanism. In a composite service, as a result of cancellation, the process of undoing the completed portions of the canceled service and compensating the results of the previously completed component services is known as cancellation recovery. A cancellable service enables cancellation recovery by providing the cancellation logic. The cancellation logic for a service must provide a mechanism to terminate the service execution. A long-running service is not constrained by isolation property and it is likely that certain amount of work has already been completed at the time of interruption. If this completed work is not compensated, the interrupted long-running service may not terminate in a consistent state. Hence, the cancellation logic must also compensate the completed work of the long-running process at the time of cancellation. Further, all component services that have successfully completed earlier must also be compensated.
RECOVERABILITY OF TRANSACTIONAL WEB SERVICES

The recovery mechanism that can be applied for a failed service is determined by its ability to recover. The recoverability of the service is based on its behavioral property. For example, only when a service is capable of performing a compensating action i.e. it has compensatable property, it is possible to apply backward recovery. Table 1 describes the different kinds of recovery mechanisms enabled by the various types of transactional web services. A pivot service does not enable any kind of recovery mechanism as its failure does not leave execution in an inconsistent state. A service which is compensatable, retrievable, and cancelable enables backward, forward, and cancellation recovery mechanisms. A gradation for degree of recoverability, of transactional services can be obtained based on their recovery cost. The recovery cost depends on the incurred time to recover from a failure or cancellation of a service. The estimation of recovery time for transactional services is discussed in the following subsections.

Estimation of Recovery Cost

A composite service comprises of several component services composed in sequence or in parallel. Let us assume that a composite service has n components and consider the failure of i\(^{th}\) service. In general, the compensation logic of a compensatable service does not incur more time compared to its normal execution time. Similarly, the cancellation logic of a cancellable service consumes lesser time than its normal execution time. It is assumed that there is no failure during the execution of compensation or cancellation logic. The pattern in which the services are composed has an impact on the recovery time of the failed service. Hence, the recovery time of a failed service is computed to be the larger of the recovery time incurred when it is sequentially composed and the recovery time incurred when it is composed in parallel. The assumptions made for recovery cost estimation and the adapted notations are tabulated in Table 2.

Backward Recovery Cost

In the event of the failure of i\(^{th}\) service that is sequentially composed in a composite service, all the i-1 services that have been previously completed must be compensated for consistent termination. The compensation recovery time required to recover from the failure of a non-retrievable service i, RT\(_{\text{cp}}\)\(^i\), is computed to be the maximum of the recovery time incurred when it is sequentially composed RT\(_{\text{cp}}\)\(^{\text{seq}}\) and the recovery time incurred when it is composed in parallel RT\(_{\text{cp}}\)\(^{\text{par}}\)

Table 1. Recovery types enabled by transactional web services

<table>
<thead>
<tr>
<th>Type of TWS</th>
<th>Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Forward</td>
</tr>
<tr>
<td>Pivot</td>
<td>×</td>
</tr>
<tr>
<td>PivotRetrievable</td>
<td>✓</td>
</tr>
<tr>
<td>Compensatable</td>
<td>×</td>
</tr>
<tr>
<td>PivotCancellable</td>
<td>×</td>
</tr>
<tr>
<td>CompensatableRetrievable</td>
<td>✓</td>
</tr>
<tr>
<td>PivotCancellableRetrievable</td>
<td>✓</td>
</tr>
<tr>
<td>CompensatableCancellable</td>
<td>×</td>
</tr>
<tr>
<td>CompensatableCancellableRetrievable</td>
<td>✓</td>
</tr>
</tbody>
</table>
The recovery time incurred by the failure of a non-retriable service that is sequentially composed, $RT^i_{cpSeq}$ is given below.

$$RT^i_{cpSeq} = \sum_{j=1}^{N_{sc}} T^j_{cp}$$

(2)

When a non-retriable service is composed in parallel with $P$ component services, its failure triggers compensating all the completed parallel services and cancelling all the active concurrent services. Then all the $i-1$ services that have been completed prior to the $i^{th}$ service that failed, are compensated. The recovery time incurred by the failure of a non-retriable service that is composed in parallel, $RT^i_{cpPar}$ is lesser when some of the concurrent services have completed while some of the other are active compared to when all of the concurrent services have completed. Hence, $RT^i_{cpPar}$ is computed to be the maximum of recovery time incurred when all the parallel processes have completed their execution and recovery time incurred when all of them are active.

The compensation or cancellation of concurrent processes occurs simultaneously and hence the compensation or cancellation time of all completed or active concurrent services is taken to be the largest of all.

The recovery of the failed non-retriable service, when the concurrently running services are active, involves compensating their completed portions. Whereas, it involves entirely compensating all the concurrently running services when all of them have completed their execution. Thus, from Equation (3) the recovery time of a non-retriable service that is concurrently composed is given by:

$$RT^i_{cpPar} = \max \left[ \left( \max_{1 \leq k \leq P} \left( \sum_{j=1}^{N_{sc}-P} T^j_{cp} \right) + \sum_{j=1}^{N_{sc}-P} T^j_{cp} \right), \right. \left. \left( \max_{1 \leq k \leq P} \left( \sum_{j=1}^{N_{sc}-P} T^j_{cp} \right) + \sum_{j=1}^{N_{sc}-P} T^j_{cp} \right) \right]$$

(3)

The recovery of the failed non-retriable service that is composed in parallel involves compensating the concurrent services in parallel, in addition to compensating the previously completed services. Alternatively, in case of the failure of a sequentially composed compensatable service, the recovery involves compensating all the previously completed services in sequence in the reverse order of their execution. Therefore, the recovery time to restore from the failure of non-retriable service $RT^i_{cp}$, given in

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T^i_e$</td>
<td>Execution time incurred by $i^{th}$ service</td>
</tr>
<tr>
<td>$N_{sc}$</td>
<td>Number of successfully completed services</td>
</tr>
<tr>
<td>$0 &lt; a_i &lt; 1$</td>
<td>Amount of work completed before a failure or cancellation of $i^{th}$ service occurs</td>
</tr>
<tr>
<td>$0 &lt; b_i \leq 1$</td>
<td>Effort required for compensating $i^{th}$ service expressed as a percentage of its execution time</td>
</tr>
<tr>
<td>$0 &lt; c_i &lt; 1$</td>
<td>Effort required for cancelling $i^{th}$ service expressed as a percentage of its compensation time</td>
</tr>
<tr>
<td>$T^i_{cp} = b_i \times T^i_e$</td>
<td>Compensation time for undoing $i^{th}$ service</td>
</tr>
<tr>
<td>$T^i_{cc} = c_i \times T^i_{cp}$</td>
<td>Cancellation time for interrupting $i^{th}$ service</td>
</tr>
<tr>
<td>rCount</td>
<td>Number of retries before a service succeeds</td>
</tr>
</tbody>
</table>

Table 2. Assumptions and notations
equation 1 is derived from Equations (2) and (3) as below.

$$RT^i_{cc} = \sum_{j=1}^{Nsc} T^j_{cc}$$  \hfill (5)

### Cancellation Recovery Cost

When an $i^{th}$ service that is long-running and sequentially composed is interrupted due to changes in input requirements, in addition to terminating its execution and compensating its completed portions, all the previously completed $i-1$ services are compensated. The recovery time of an interrupted cancellable service $i$, $RT^i_{cc}$, is computed to be the larger of the recovery time incurred when it is sequentially composed $RT^i_{ccSeq}$ and the recovery time incurred when it is composed in parallel $RT^i_{ccPar}$.

$$RT^i_{cc} = \max(RT^i_{ccSeq}, RT^i_{ccPar})$$ \hfill (6)

The recovery time incurred by the interruption of a cancellable service that is sequentially composed, $RT^i_{ccSeq}$ is given below.

$$RT^i_{ccSeq} = T^i_{cc} + \sum_{j=1}^{Nsc} T^j_{cc}$$  \hfill (7)

The cancellation of a cancellable service that is composed in parallel with $P$ component services triggers compensating its completed portions and compensating completed concurrent services or cancelling the active concurrent services. Then all the services that have completed prior to the interrupted service are compensated. The recovery time incurred by the cancellation of a cancellable service that is composed in parallel, $RT^i_{ccPar}$, is computed to be the maximum of recovery time incurred when all the parallel processes have completed their execution and recovery time incurred when all of them are active.

$$RT^i_{cc} = \max\{\max(T^i_{cc}, \max_{1 \leq k \leq P} (T^k_{cc} + \sum_{j=1}^{Nsc-P} T^j_{cc})), \max_{1 \leq k \leq P+1} (T^k_{cc} + \sum_{j=1}^{Nsc} T^j_{cc})\}$$ \hfill (8)

The recovery of the canceled service, when all the concurrently running services are active, involves compensating only the completed portions of concurrent services. Whereas, the recovery of a canceled service, when the concurrently running services have completed their execution, involves entirely compensating them. Thus, the recovery time of a cancellable service that is concurrently composed is given by:

$$RT^i_{ccPar} = \max\{T^i_{cc}, \max_{1 \leq k \leq P} (T^k_{cc} + \sum_{j=1}^{Nsc-P} T^j_{cc})\} + \sum_{j=1}^{Nsc} T^j_{cc}$$ \hfill (9)

The recovery of canceled service involves compensating its completed portions. In addition, the recovery of the canceled service that is composed in parallel involves compensating the concurrent services in parallel and compensating the previously completed services. Alternatively, in case of the failure of a sequentially composed cancellable service, the recovery involves compensating all the previously completed cancellable service, the recovery involves compensating all the previously completed services in sequence in the reverse order of their execution. Therefore, the recovery time of a canceled service failure $RT^i_{cc}$ given in equation 6 is derived from Equations (7) and (9) as below.

$$RT^i_{cc} = T^i_{cc} + \sum_{j=1}^{Nsc} T^j_{cc}$$  \hfill (10)

### Forward Recovery Cost

When an $i^{th}$ service that is retriable fails, it is invoked multiple times until it succeeds. For every failure, the time incurred is lesser than its actual execution time. The failure is restored to
consistent state by successfully executing the service at the last retrial. The recovery cost is the same in cases of the retriable service composed in sequence as well as in parallel. The recovery time, RT, of i-th service that is retriable, is computed as below.

\[ RT_i = \left( \sum_{j=1}^{rCount-1} a_j \cdot T_j^i \right) + T_e^i \]  \hspace{1cm} (11)

**RECOVERABILITY LEVELS OF TRANSACTIONAL WEB SERVICES**

The proposed concept of recoverability levels pertaining to transactional web services abstracts their transactional requirements. The eight kinds of transactional web services discussed in Background Section must be ordered based on their recovery cost representing its recovery time so that they can be chosen with an optimal cost and required recoverability.

In a composite service, the failure of a pivot component service never leaves any trace which means that it does not leave the execution of composition in inconsistent state. The execution of the composite service can proceed with subsequent component services, ignoring the failure of the pivot service. Therefore, failure of a pivot service does not require any type of recovery. In other words, a pivot service does not enable any kind of recovery mechanism. Its recoverability level is low and is considered to be one.

In comparison with compensation recovery cost, the cancellation recovery of i-th service involves additionally the cost of compensating completed portions of the canceled service i (see Equations (5) and (10)). The cancellation recovery cost is found to be higher than backward recovery cost. Hence, the service with compensatable property is grouped in recoverability level two and the service with cancellable property is grouped in level three. The recovery cost of a service with transactional properties compensatable and cancellable (cpcc) is the larger of its compensation cost and cancellation cost which means it incurs same cost as a service with property cc. Hence, it is also grouped in recoverability level three.

In the event of failure of i-th service that is not retriable, all the previously completed services need to be compensated in order to recover to a consistent state. If the failed service is retriable, then the number of times the failed service or an alternate service is reattempted before it succeeds, determines the execution effort in recovering the i-th service. It succeeds in the last reattempt. Since the service may fail any time during its execution, the average execution effort in n reattempts will be \( \frac{1}{2} T_e^i \). The number of reattempts plays a vital role in comparing the backward or cancellation costs with the forward recovery costs when i-th service fails. The number of reattempts rCount_{ccMin} up to which cancellation cost is equal or more than the forward recovery cost is given below:

\[ rCount_{ccMin} = \left\lceil \frac{RT_i}{T_e^i} \right\rceil \left( \frac{1}{2} T_e^i \right) \]  \hspace{1cm} (12)

The number of reattempts rCount_{cpMin} up to which compensation cost is equal or more than the forward recovery cost is computed as below:

\[ rCount_{cpMin} = \left\lceil \frac{RT_i^c}{T_e^i} \left( \frac{1}{2} T_e^i \right) \right\rceil - 1 \]  \hspace{1cm} (13)

The forward recovery cost exceeds backward (cancellation) recovery cost if the failed service is reattempted beyond rCount_{cpMin} (rCount_{ccMin}) times. In general, it is desirable to retry a failed service until it succeeds to recover from transient failures, rather than undoing the previously completed services. In the worst case, the retriable failed service may be reattempted more than rCount_{cpMin} times i.e. rCount > rCount_{cpMin} (rCount > rCount_{ccMin}) until it succeeds, in which case forward recovery cost is higher than the backward
(cancellation) recovery cost. The cancellation cost was proved to be more than compensation cost. Hence, services with property $r$ are grouped in recoverability level four that is higher than that for non-retriable cancellable services.

A service with $cpr$ property may succeed after $rCount$ retries and may be compensated later. While comparing a service with transactional property $cpr$ and that with $ccr$, the recovery cost of $ccr$ service is higher as cancellation is more expensive than compensation. Hence, a $cpr$ service is grouped in recoverability level four same as that of a service with property $r$. Whereas, a $ccr$ service is grouped in a higher recoverability level of five. A service which has compensatable, retriable, and cancellable properties may get cancelled or compensated after $rCount$ retries and its recovery cost is the larger of that of a service with $ccr$ property and a service with $cpr$ property. Hence, its recovery cost is the same as that for $ccr$ service and its recoverability level is five.

The transactional web services are grouped into five levels and each level exhibits certain kind of recoverability. For example, in level 1, the pivot service which cannot be rolled back after its completion is placed. Level 2 holds a service which can be rolled back after its completion. Level 3 is meant for externally interruptible services. The services which are guaranteed to succeed are placed in level 4. Level 5 services are also guaranteed to succeed unless they are cancelled. In each of the levels 3, 4, and 5, two kinds of transactional services with same recovery cost, one which cannot be rolled back and the other one which can be rolled back on completion have been placed. Level 1 has the lowest recovery cost whereas level 5 offers the most expensive recovery. Thus the transactional properties are abstracted to the analyst in terms of recoverability levels as shown in Figure 1.

**EMPIRICAL VALIDATION OF RECOVERY COST**

In order to measure the recovery costs incurred by different recovery mechanisms, several compositions involving 10, 20, 30, 40, and 50 services have been considered. The services are deployed in Glassfish web server of different

---

**Figure 1. Recoverability levels of transactional web services**
machines each with 2.4 GHz Intel Core2 Duo processor and 4 GB RAM, connected in the Intranet. In each composition, a failure of some service $i$ is considered and its recovery cost is estimated. Since a composite service comprises of component services of different execution times, for a composition of $n$ services in which $i^{th}$ service is assumed to fail, the following scenarios have been considered. Basically, the services are of two types viz., normal services (short running) and long running services.

1. All services upto $(i-1)^{th}$ service are normal and $i^{th}$ service is also normal.
2. All services upto $(i-1)^{th}$ service are normal and $i^{th}$ service is long running
3. All services upto $(i-1)^{th}$ service are long running and $i^{th}$ service is normal
4. All services upto $(i-1)^{th}$ service are long running and $i^{th}$ service is also long running

All the services were created with compensation and cancellation operations and according to the scenario tested, necessary operation was invoked. Since the services are not having any specific functionality, sleep times were introduced. For a regular or normal service, sleep time was 500ms. The sleep times for compensation and cancellation operations were 450ms and 400ms respectively. For a long running service and its compensation and cancellation operations, the sleep times were 3000ms, 2700ms, and 2400ms respectively. The recovery time required to restore from the failure of a pivot service is negligible. As described in the previous section and from Figure 1, the recovery time of a service with property $cpcc$ is the same as that with property $cc$. The recovery time of a service with property $cpr$ is the same as that with property $r$. The recovery time of a service with property $cpccr$ is the same as that with property $ccr$. Hence, recovery times have been measured for the failure of services with properties $cp$, $cc$, $r$, and $ccr$. Suitable transactional properties have been assumed for the failed service as well as the services executed earlier, in order to apply the corresponding recovery mechanism. Based on the scenario needed to be tested, the sleep times of the services were changed. A composer service was designed to compose component services at runtime and deployed. The composer service composes the required number of component services on the fly and the elapsed time from the failure or cancellation of a service till the completion of recovery mechanism was measured.

**Experiment 1**

A composition of 30 services and the failure of 25th service are considered to observe the recovery cost incurred in four different scenarios for different recovery mechanisms. The recovery time for recovering from the failure of 25th service in a composition of 30 services, considering four scenarios listed above, are tabulated in Table 3. The comparison of recovery time for different recovery mechanisms in four scenarios is depicted using a graph in Figure 2. It is evident from the graph that the recovery time incurred in Scenario 1 to scenario 4 are in the increasing order. The comparison of recovery costs in four scenarios for four different recovery mechanisms is shown using a graph depicted in Figure 3. The compensation time for Scenario 1 and Scenario 2 are almost the same, since, both of them involve compensating the same set of normal services. The compensation time for Scenario 3 and Scenario 4 are almost the same, since, both of them involve compensating the same set of long running services. The cancellation time for Scenario 2 and Scenario 3 are more than that in Scenario 1 and Scenario 4 respectively, since cancellation time of long running service is more than that of a normal service. A similar observation is made with respect to the remaining recovery types as retrying the long running service incurs more time than that of a normal service.

**Experiment 2**

The second testing is performed to compare the recovery costs when the failed or cancelled service is composed in sequence or in parallel. A sequential composition of 30 services where 24 services have completed their execution
Table 3. Recovery costs of transactional web services

<table>
<thead>
<tr>
<th>Recovery Type</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward</td>
<td>2.426</td>
<td>2.567</td>
<td>17.751</td>
<td>17.731</td>
</tr>
<tr>
<td>Cancellation</td>
<td>2.843</td>
<td>3.917</td>
<td>23.901</td>
<td>24.602</td>
</tr>
<tr>
<td>Forward</td>
<td>3.154</td>
<td>4.437</td>
<td>25.776</td>
<td>34.237</td>
</tr>
<tr>
<td>Forward and Cancellation</td>
<td>4.873</td>
<td>6.286</td>
<td>42.567</td>
<td>51.549</td>
</tr>
</tbody>
</table>

Recovery Time in Seconds

Figure 2. Recovery time incurred for different recovery mechanisms

Figure 3. Recovery time in different scenarios
Figure 4. Comparison of recovery costs when the failed/cancelled service is composed in sequence or in parallel

Figure 5. Recovery costs of different recovery mechanisms
and 25th service has failed, is considered. Another composition where 20th to 25th services are running in parallel, is considered. In this composition, the recovery cost due to the failure of 20th service is considered, since the number of completed services (including concurrent services) is 24 at the time of failure and is comparable to the sequential composition. For both compositions, the average compensation and cancellation costs of the failed service for four different scenarios have been measured. The recovery costs by applying different recovery mechanisms are measured for both compositions in four scenarios and average values are plotted in Figure 4. It is observed that the forward recovery cost of the failed service is the same when it is composed in sequence or in parallel. The backward and cancellation recovery cost of the failed service is less expensive when it is composed in parallel since the compensation or cancellation logic is applied simultaneously for concurrent services.

### Experiment 3

As a third testing, the recovery costs incurred by different recovery mechanisms have been compared. The recovery times of recovering from the failure of n-5th service in compositions of n services where n = 10, 20, 30, 40, 50 are measured 10 times for each of the four different scenarios listed above and the average is considered. The average recovery time for each type of recovery in compositions of n different services are tabulated in Table 4 and plotted as a graph in Figure 5. While recovering from the failure of ith service, it is observed that backward recovery is the least expensive; forward recovery followed by cancellation recovery is the most expensive; cancellation recovery is more expensive than compensation; forward recovery is more expensive than cancellation. From the empirical analysis, it is evident that compensation, cancellation, forward, and forward followed by cancellation recovery techniques incur costs in the ascending order. Hence, their grouping into recoverability levels in the increasing order, as shown in Figure 1 is justified.

### CONCLUSION

In this paper, an abstract way of expressing the transactional properties of web services in terms of their recoverability has been introduced. The proposed recoverability levels are grouped in the increasing order of service recoverability and recovery costs. The recovery cost for different recovery mechanisms in the case of service failure in compositions of various sizes have been measured and compared with the estimated recovery time. The gradation of recoverability levels is clearly justified using the empirical experiments. The recoverability levels help the business analyst to choose the required level for the services, based on the expected behaviour of the services and the associated recovery cost.

### Table 4. Comparison of average recovery costs of transactional web services for different scenarios

<table>
<thead>
<tr>
<th># of Services</th>
<th>i - Service that fails</th>
<th>RT\text{\textsuperscript{cp}} \text{\textsuperscript{i}}</th>
<th>RT\text{\textsuperscript{cc}} \text{\textsuperscript{i}}</th>
<th>RT\text{\textsuperscript{r}} \text{\textsuperscript{i}}</th>
<th>RT\text{\textsuperscript{ccr}} \text{\textsuperscript{i}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>2.146</td>
<td>2.694</td>
<td>3.600</td>
<td>5.554</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>10.269</td>
<td>13.816</td>
<td>17.006</td>
<td>26.319</td>
</tr>
<tr>
<td>40</td>
<td>35</td>
<td>16.563</td>
<td>17.423</td>
<td>24.597</td>
<td>33.484</td>
</tr>
<tr>
<td>50</td>
<td>45</td>
<td>26.015</td>
<td>27.200</td>
<td>32.508</td>
<td>39.339</td>
</tr>
</tbody>
</table>

Recovery Time in Seconds

Copyright © 2013, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
REFERENCES


Kanchana Rajaram is an Assistant Professor at the Department of Computer Science, SSN College of Engineering, Chennai. She holds a M.E. in Computer Science from National Institute of Technology, Tiruchirapalli and currently pursuing her research in Service Oriented Architecture. She has more than 20 years of teaching experience.

Chitra Babu is a Professor and Head of the Department of Computer Science, SSN College of Engineering, Chennai. She obtained PhD in Computer Science from IIT, Madras, Chennai and M.S. in CIS from Ohio State University, USA. She is currently guiding 6 PhD research scholars.

Arun Adiththan is currently pursuing his PhD in Computer Science at City University of New York, USA. He finished his M.E. in Computer Science from SSN College of Engineering, Anna University - Chennai.