WEB SERVICES ORIENTED TRANSACTIONS USING PARTIAL DEPENDENCIES

Rena Logothetis rlogothetis@iinet.net.au
Jinli Cao j.cao@latrobe.edu.au
Department of Computer Science & Computer Engineering
Latrobe University, Melbourne

Abstract: A Web Service is the integration of homogenous services in order to complete a specific service. It is important to develop compositions in order to create a simple environment for business transactions where a single web service can be used to complete many services. A web service that is used for business transactions must be organized where the user does not need to repeatedly enter the same data in order to complete a transaction. The composition must be structured so that data, derived from user input and committed transactions, is passed to the related services that assist in the customizing of the Web Service for the individual user.

This research is to improve the transaction processing for web services by focusing on the composition of web services to provide the automation of web services. The user can complete many services from a single web site with minimal interaction to obtain the optimized service. A structured composition approach is proposed. This allows the composition to function appropriately with transaction scheduling Web Net was created to resolve the issue of compositions at the application level where all the single services are accessed and integrated to form one complete service.

1. INTRODUCTION

The development of Web services [1], [2], [3], [4], [5] has been greatly beneficial to businesses, enabling their applications to be integrated into the processes of their suppliers, partners, and customers. This has improved the
functioning capabilities of businesses, reducing costs and gaining a competitive advantage.

Albornoz J. and Ryman A. [6], [7] define a Web Service as a set of related application functions that can be programmatically invoked over the internet via internet protocols using XML for messaging, description, and discovery in a platform independent way. It is a standardized way of integrating Web-based applications.

The main objective of a Web Service is to allow the user to achieve a service without the need of having to browse to other URLs and enter the same data repeatedly in each URL. This allows the user to complete any services in a shorter period of time and gives him the same freedom to search other URLs through UDDI registries.

Web Service compositions and transactions are the focus of this paper. Composition of web services can be identified as two levels, the low-level that houses many web services directly from the web service providers, or high-level utilizing the UDDI registry to achieve data collection from the service providers.

The composition of web services is yet ideal and is an open area for extensive research. Due to the large number of modular Web services, the integration poses some difficulty, particularly with transactions. We study compositions and transactions of Web services with the application level of web services.

Research completed in web services with respect to Compositions and Transactions are Pires’ WebTransact [8], and Kaltenmorgen N. and Klueber R.’s eService to integrate eBusiness with ERP (Enterprise Resource Planning) Systems [9]. Pires’ work identifies web services used in everyday life by any user. WebTransact provides mechanisms for describing the dissimilar transaction behavior of web services and for resolving their heterogeneities, for specifying reliable interaction patterns of Web Services, and for coordinating such interaction patterns in a transactional way [8]. Kaltenmorgen’s and Klueber’s work is suited for business to business scenarios.

Research illustrates web services of today apply a linear composition where all services are independent of each other. This results in the problem of repetitive data entry from the user. A specific web service studied, exhibited no association between the individual services and the user was presented with the same options for each variation of input.

WebJet is a web service currently available on the World Wide Web [http://www.webjet.com.au/], its behavior was observed in terms of the service it provided. It was noted that a lot of data were lost within the transition of services, leading to the creation of WebNet.

This paper introduces a framework, WebNet, to standardize compositions and assist in the services, and relates to external UDDI registries for information required by users. The proposed WebNet details the interrelation of many services within the compositions and the passing of data
through service-nodes, enabling the application to create customized UDDI registry searches for individual users. WebNet is designed to customize the service for individual users and shorten the period in which the user completes a transaction reducing the total time the user spends on the web service.

This research is to improve the transaction processing for web services by focusing on the composition of web services to provide the automation of web services. The user can complete many services from a single web site with minimal interaction to obtain the optimized service. A structured composition approach is proposed. This allows the composition to function appropriately with transaction scheduling.

The remainder of the paper is organized as follows. In section 2 we present a general picture of the Web service architecture and WebNet's contribution. Section 3 is the preliminary and related definitions of WebNet. Section 4 is the service-node scheduling in WebNet and an example of the application of WebNet concluding with section 5.

2. THE WEB SERVICE ARCHITECTURE

In this section we describe the architecture of the Web services. A Web service system is an integration of a number of existing and new business systems to provide seamless processes that produce values for customers, business partners, and organizations on the World Wide Web.

Web applications can dynamically register to the Web services system. The web service architecture is built on the three roles; Service Provider, Service Requestor and Service Registry. The interactions of these roles are depicted by the Publish, Find, Bind operations. WebNet interacts with the Service Requestor through message exchange operations.

The service provider implements and hosts the web service, produces a WSDL document describing the service and publishes it to a service requestor or a UDDI service registry. The publish operation is the publishing of the WSDL document. The service requestor is an application that searches and initiates communication with a service using SOAP standards. WebNet employs the service requestor as it's connectivity to the Web Service Architecture. The service requestor retrieves the service description, locally or from the service registry, using the find operation to query and retrieve. The service description, WSDL document, is used by the service requestor in order to bind with the service provider. The service requestor uses the bind operation to invoke communication or interact with the web service. The role of the service registry is for service providers to list their services with a WSDL service description and a URL to the web service.

WebNet is the end user application. The user interacts with WebNet to complete a web service, built on many services. WebNet collates the data from user inputs and committed transactions submitting it to the service.
requestor where it executes a customized search for a service from the Service Registry. WebNet, through the Service Requestor, can access the service provided by the Service Provider to complete transactions required by its own service.

![Diagram of WebNet's Relation with the Web Service Architecture](image)

*Figure 1. WebNet’s Relation with the Web Service Architecture [7]*

3. **PRELIMINARY AND RELATED DEFINITIONS**

WebNet’s foundation is built on predefined ontology for a selection of services. The relevant homogenous services available from the Service Registry allow for the implementation of the predefined compositions. The predefined composition identifies the homogenous services with its dependencies related to the specified WebNet web service including the homogenous services characteristics within the web service.

To understand the functionality of WebNet, its semantics must be defined. WebNet is a composition based on a tree structure. The ability to transfer data between services is based on two decision attributes, the degree of a relational dependency and the criticality of the service in the composition or is requirement in the web service.

Firstly, the system deals with three dependencies, partial dependency total dependency and independency. A combination of these dependencies leads to the X-factor relationship. The critical and requirement feature of the service is determined within the dependency services via look up tables. This allows the user to complete the process with minimal user interaction and when undoing a transaction only the dependent nodes to the transaction that has been undone must be redone.

There three types of dependencies between service nodes:
- Totally Dependent.
- Partially Dependent.
- Totally Independent.
These dependencies are represented by the dependencies of the service-nodes in question. A total dependency is represented by two full lined circles, a partial dependency is represented by two broken lined circle circles, an independent node is represented by a single full lined circle and an X-factor service-node is a broken lined circle. A service node can be one of three dependencies. Totally dependent service nodes are the strongest link within the composition whilst on the opposite end of the scale independent service nodes are only linked to the form. The four service nodes can take on either a critical or required attribute or both.

- **Total Dependency**: A total dependant service node requires data from the transaction of the service node it is dependant upon, including data from the transactions of the dependent service node of that service node, to create an appropriate search on the registry.

  For example, referring to figure 2, all service nodes in question are totally dependent. Service node C is totally dependent upon service node B and service node B is totally dependant upon service node A. Assuming service node A and B are completed, the data from service node B’s transaction derived from service node A’s transaction, along with the data from service node A’s transaction is passed to service node C in order to create a customized registry search.

![Figure 2. Linear Composition for a Totally Dependent Service-Node](image)

- **Partial Dependency**: Partial dependency is less complicated and offers added flexibility to a service. Partial dependency characterizes a direct relationship between two service-nodes. The partially dependent service node requires the data from the previous service nodes completed transaction.

  Referring to figure 3 we have added partially dependent service node D. to the composition depicted in figure 2. Figure 3 indicates service node D is partially dependent upon service node C, hence service node D is only interested in the data that is produced by the transaction of service node C to create the registry search. Service node D is not interested in the transactions of service nodes A and/or B.

![Figure 3. Linear Composition for a Partially Dependent Service-Node](image)
Independent: Independent service nodes create difficulty in the flow of the web service they are only dependent on the form refer to figure 4. they can have a relation with an x-factor service node.

![Figure 4. Totally Independent Service-Node](image)

X-Factor: An X factor service-node is a service-node that is dependent on two or more service-nodes \((n)\). An X factor service-node can occur for either a collection of all totally dependent \(n\) service-nodes or a combination of totally and partially dependent \(n\) service-nodes and very rarely some input from an independent node. The X factor can take the either the characteristics of a totally or partially dependent service-node from its previous service-nodes. The X outcome will refer to its look up table to decide what data it needs and from which service-node.

Expanding on figure 2, figure 5 depicts the addition of a totally dependent service-node, Service-Node E, a partial dependent service, another service available in the composition and Service-Node F. The registry search for Service-Node F is dependent on the transactions of Service-Node D and Service-Node E, due to the difference in dependencies of these service-nodes, Service-Node F identifies the dependencies by referring to its X-factor table, refer to table 1. The dependency of the related service-nodes is not relevant to the dependency of Service-Node F. Referring to figure 5, Service-Node E is depicted as totally dependent and Service-Node D as a partially dependent service-node. Service-Node F can have a relationship with Service-Node E as a partial dependency or a total dependency.

![Figure 5. Composition Depicting All Relationships](image)

The X Factor table associated with Service-Node F, refer to figure 5, is shown below in table 1. Service-Node F has a partial dependency with Service-Node D and a totally dependency with Service-Node E. The registry search for Service-Node F requires data from Service-Node A through to Service-Node E’s transactions (in the linear path) and the data from Service-Node D’s transaction.
4. SERVICE-NODE SCHEDULING IN WEBNET

To improve service orientation in Web services, we propose a Web serve framework: WebNet to utilize transactions in a composition to customize the service for individual users. WebNet consists of service-nodes that have a unique task and inter-related dependencies either to other service-nodes or the form. The look up table is introduced to facilitate the composition, storage and schedule of the inter-related activities and service-node.

4.1 Identification Name and Binary Dependency Value

Service-nodes need to be identified by other service-nodes within the composition. Each service-node must also know its dependency and the dependency of the service-nodes it is related to. An identification name is assigned to each service-node and with each service-node there is a dependency value associated with it. The identification name is set by the designer and is consistent across all service-nodes. The dependency value is made up of three binary bits. The position of a bit “1” identifies the service-nodes dependency. Total dependency is represented as “100”, partial dependency “010” and independent as “001”. “000” belongs to the form.

The service-node is built on three major sections. The first section determines the service-node dependency and deals with the request receiving of data. The second stage applies the data from the previous stage to execute a registry (UDDI) search, and commits a transaction. The data from the first and second stage are passed to the third stage for storage.

- **Stage One – Identification and Receiving**: The first process within this section identifies the current service-node and its dependency. The service-node verifies it has the required data, in order to execute a registry search, from the data received flag in the look up table. For any data not received, a request is sent to the service-node or the form containing that data.
  A partially dependent service-node forwards its data to the second section of the service-node. A totally dependent service-node aggregates its data prior to forwarding it to the second section. The data received is sent to section three for storage.

- **Stage Two – Search and Transaction**: The second process utilizes the data from the first process to create a customized registry (UDDI) search.

<table>
<thead>
<tr>
<th>Service-Node D</th>
<th>Service-Node E</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTIAL</td>
<td>TOTAL</td>
</tr>
<tr>
<td>DEPENDENCY</td>
<td>DEPENDENCY</td>
</tr>
</tbody>
</table>

Table 1. X – Factor Table
The registry search returns appropriate results that are suited to each user's request. Provided with a list of options, the user makes his selection and commits the transaction setting the status flag, indicating a completed transaction. The status flag is broadcast to the composition and any related service-node updates the status flag of that service-node in the look up table. The transaction produces the data payload for the current service-node and is passed onto section three for storage.

- **Stage Three – Data Storage and Transmission**: Section 3 focuses on the transmission of payloads, identifies the payloads required by service-nodes. The service-node stores the payload from stage two along with the payload(s) received from stage one into a data bank.

This section can foresee where the next service-node lies in the composition, if the next service-node is in a linear path, hence there are no user options and selections, the service-node will access the required data from the data bank and send it to the next service-node. If the next service-node within the linear path is a totally dependent service-node, section three will send all the data it has stored in its data bank. If it is a partially dependent service-node that follows, only the data achieved in section two will be passed on. This feature allows service-nodes in a linear composition to have a fast availability time, reducing the execution time of the web service. The service-nodes that receive the data do not need to reference look up tables, or send requests for data and wait to receive data. For service-nodes that are not in a linear path, the user selects the option he wants to complete first and the service-node will refer to the look up table in order to determine the required data and dependencies of the service-nodes.

### 4.2 Look Up Table

The look up table is referenced by all sections in a service-node. It lists service-nodes by name and dependency value with critical, required, status and data received flags. The first row in the table stores the information of the current service-node. The following rows in the table refer to the service-nodes that the current service-node depends on and the service-nodes that depend on the current service-node (refer to table 2).

#### Table 2. Look-Up Table

<table>
<thead>
<tr>
<th>Service-node</th>
<th>Binary Dependency value</th>
<th>Critical</th>
<th>Required</th>
<th>Status</th>
<th>Data Received</th>
<th>X Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service-node D</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Service-nodes this service-node depends on</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service-node</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### Flags

Each service-node has three flags that characterize the service-node; critical, required and X factor flags. The other two flags, status and date received, assist in the service-nodes functionality and with the error control.

- **Status Flag**: On completion of the transaction in a service-node, the status flag is set to binary “1” and broadcast to the related service-nodes which update their look-up table with that service-nodes status. If a service-node has not been completed the status flag is “0”.

- **Required Flag**: The required flag indicates the service-node is an important factor in completing the service, for a required service-node the flag is statically set to “1”. The web service is dependent upon the completion of this service-node. It is a flag that assist in the error control of WebNet. **ALL REQUIRED SERVICE-NODES MUST BE COMPLETED**.

- **Critical Flag**: A critical service-node is a service-node (Service-node N) that has a service-node depending on it (Service-node (N + 1)). For a critical service-node the flag is set to “1”. The critical flag is statically assigned by the designer and does not change throughout the execution of the web service.

  In a linear path of totally dependent nodes Service-Node (N) and Service Node (N+M), where M > 1, if the current node, Service Node (N+M) is critical all nodes prior to that node are critical. A partially dependent node can be critical alone.

- **Data Flag**: In the first section of a service-node, the service-node identifies it has the data needed in order to execute a registry search. When a node receives data from its related service-nodes the data received flag is set to “1”, indicating that the current service-node has received the data from that service-node and does not need to send any request to that service-node. If it does not have the data from a service-node, the data received flag associated with that service-node is set to “0” and a request is sent to it for its data.

- **X factor Flag**: The X factor flag, when set to binary “1” indicates that the service-node is an X factor service-node and has an X factor table related to it that must be referenced to.

<table>
<thead>
<tr>
<th>Service-node</th>
<th>Binary Dependency value</th>
<th>Critical</th>
<th>Required</th>
<th>Status</th>
<th>Data Received</th>
<th>X Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>1</td>
<td>1</td>
<td>000</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Service-nodes that depend on this service-node

<table>
<thead>
<tr>
<th>Service-node</th>
<th>010</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

**Form Data**

<table>
<thead>
<tr>
<th>Form</th>
<th>1</th>
<th>1</th>
<th>000</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
4.2.2 The Form: The form is the commencing point of the web service. The user enters the appropriate information for the web service to begin its process. Data from the form is carried out through the composition where the composition utilizes the entered data to create its intelligence. The form must be completed prior to commencing WebNet.

Any service-node can request data from the form at any time in the composition. The Form List exists in each service-node that is related to the form. The service-node identifies a relationship to the form from its look up table. The form list simply lists the data that the service-node needs from the form. For example, Figure 6 is a list that is included in a service-node that needs two sets of data from the form. The service-node can identify that it needs data on the CPU and the HDD from the form, these values are key values. It can then send a request to the form passing the two key values on the list. The form will identify the names and send the data that was entered in that name field.

<table>
<thead>
<tr>
<th>Form List</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>HDD</td>
</tr>
</tbody>
</table>

Figure 6. The Form List

4.3 Error Checking & Regulations

WebNet utilizes the flags in the look up table in order to complete error checking. Previously we discussed the service-node and the stages within. Each stage completes an error check in order for the section to be completed.

Stage one of the service-node, is required to refer to its look up table and verify it has the required data in order to create a registry search. The service-nodes that this service-node depends on must be completed and the status flag must be set to “1”. The service-node then checks to determine if it has all the data it needs from those service-nodes. The data received is set to “1” indicating it has the data from that specific service-node. If the service-node does not have the data it requires it sends a request. If the service-nodes it depends on are incomplete, status flag set to “0”, an internal error has occurred. Service-nodes that have service-nodes dependent on them must be completed prior to the commencing of the dependent service-node.

Committing a transaction in stage two of the service-node, the status flag is set to “1” and broadcast, before the broadcast occurs, a check is applied to verify the status is set to “1”, if the status flag is “0” an error has occurred in committing the transaction. The transaction must be done again.

This status flag is compared to the required and critical flags of the look up table. On comparison if the required flag is set to “1” the status flag should be “1”. If the service-node is critical and the critical flag is “1”, the status flag should also be set to “1”. If the status flag is set to “0” and an error
Stage three of the service node determines the service-nodes that depend on it, it also has the responsibility of ensuring that the dependent service-nodes have not been completed prior to the completion of that service-node and hence do no have the required data. All service-nodes that depend on it should have status flag and data received flag set to “0”, if this isn’t true an error has occurred within the system. All service-nodes that depend on the incomplete service-node must be undone and only upon completion of that service-node the dependent service-nodes can be completed.

This stage of the service-node also ensures that no logical transition is made within the composition. A totally dependent service-node cannot depend on a partially dependent service-node. It identifies the dependency of itself, from the look up table, and compares itself with the dependencies of the service-nodes that depend on this service-node. If a partially dependent service-node has a totally dependent service-node in its list an error has occurred with the design of the composition.

On completion of the web service, a final check is passed through ensuring all service-nodes that are required are complete hence a check is done on the status flag ensuring it is set to “1”. The user must complete any required service-node that hasn’t been completed. If that service-node is also critical, and its dependent service-nodes have been completed, an error has occurred in the composition. A check is done to ensure that any service-node dependent on an incomplete service-node has its transaction undone, if already completed. A web service looses its customized feature if the composition doesn’t check its rules.

4.4 Data Bank

A data bank is a section in the service-node where all the data related to that service-node is stored. The data bank will store the data from the current service-nodes transaction and the data from the other service-nodes that made that transaction possible.

The data bank stores the payloads in a LIFO (Last In First Out) stack. The last packet placed in the stack, derived from section two, is sent to all partial dependant service-nodes that follow the current service-node. If the service-node that follows is totally dependant, all the payloads in the stack are sent to that service-node.

When a service-node requires data, identified in section one, it sends a request to the service-node that has the data it needs. The request is controlled by the service-nodes identification name. The request is received by section three of the service-node where it processes the request. It determines what data that service-node needs by referring to that service-nodes dependency value in its look up table, depending on the dependency of the service-node, that service-node will get the required data from its data bank and send it to the service-node in need of the data.
It is important to note that the data bank does not communicate to any other service-nodes; it is simply a storing mechanism that is used by section three of the service-node.

4.5 A User Backtracking and Invalidating a Transaction

To backtrack and invalidate a transaction, the status of that service-node is reset to “0” and broadcast. The service-nodes within the composition that are dependent on that service-node will update their look up table. A service-node that was completed based on a payload(s) from the service-node that was undone has its transaction undone and broadcast as well. This allows the service-nodes that have data depending on certain service-nodes to be reset. The user must go through the composition again and complete the service-nodes affected by that single undoing of a transaction.

4.6 Application of WebNet

An application for WebNet is in the IT industry for building Personal Computers. In the IT industry there are many businesses where clients provide specifications of what they require in a PC and from the specs the technician builds the PC. A web service can be provided for the technician to purchase parts from suppliers online for the PC.

The form will require the technician to enter basic details on what he requires. An example of the type of information the form will request is depicted in table 3.

<table>
<thead>
<tr>
<th>Table 3. Form Request List</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>Motherboard</td>
</tr>
<tr>
<td>HDD</td>
</tr>
<tr>
<td>RAM</td>
</tr>
<tr>
<td>CD</td>
</tr>
<tr>
<td>DVD</td>
</tr>
<tr>
<td>LCD</td>
</tr>
</tbody>
</table>

The following figure 7 depicts the composition of the example. The first process is the completion of the form.
The first item the user selects is the Motherboard. The user can optionally complete the brand of the motherboard he wants, the service will search the Service registry narrowing the search for the motherboard with the information given about the CPU. The selection for an AMD processor and a Gigabyte motherboard customizes the options available for the user, from the search string sent to the Service Registry. The user selects his preferred Motherboard, and is placed in his items. The next item to be chosen is the CPU. This is totally dependent on the motherboard and the motherboard selected will narrow the search of the Service Registry for the compatible CPU’s available. The RAM is also totally dependent on the motherboard the search string sent to Service Registry is based on the selected motherboard.

The HDD has a total dependency on the motherboard and the required information from the motherboard relevant to the selection of HDD is collated with the HDD Size specified in the Form to query the Service Registry. The Service Registry returns the appropriate HDD’s listed and the user makes his selection.

The VGA is totally dependent on the motherboard the required information queries the Service Registry for an appropriate VGA. The user’s selection impacts the LCD selection. The user has selected a motherboard that supports a digital VGA. On selection of a VGA an appropriate LCD must be selected. The LCD is partially dependent on VGA, therefore identifying the VGA a query is sent to the Service Registry to find appropriate LCD’s that supports digital VGA.

The drives, such as floppy and CD or DVD, are independent and can be selected at any time through the service, most likely after the completion of the main features of the PC.

The power unit is the final feature for selection. It is totally dependent on the CPU that in turn depends on the motherboard. It is interested in the speed of the CPU and any other features on the motherboard that will affect the power required. It is partially dependent on the HDD, interested in the size, speed and number of HDD’s in the system, also accounting for the number of optical drives.
5. CONCLUSION

Web services offer: easier development, service, and upgrade of solutions, reuse of existing, proven assets and reduced dependence on implementation specifics. Web services allow businesses to better improve the business, concentrating on development efforts to computing assets that drive revenue. Businesses can out-source services that do not provide any value adding to the business and evolve business models and relationships. The cost of internal integration and of testing emerging market opportunities is greatly reduced and efficient interactions with marketplaces are established.

WebNet is created by service-nodes that each have a unique task and are inter-related either to other service-nodes or the form. These service-nodes make reference to tables in order to identify certain attribute of the service-node and its related service-nodes or form. The focal point of this paper is to improve service orientation in Web services by utilizing transactions in a composition to customize the service for individual users. WebNet allows a user to complete a service with minimal user interaction and without having to physically access external URLs.

6. REFERENCES


632