A Unified Data and Service Integration Approach for Dynamic Business Collaboration

Chen Liu[1], Jianwu Wang[2], Yan Wen[3,4], Yanbo Han[1]
[1] *Cloud Computing Research Center, North China University of Technology, Beijing, China*
[2] *San Diego Supercomputer Center, UCSD, U.S.A.*
[3]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*
[4]*Graduate University of Chinese Academy of Sciences, Beijing, China*
{liuchen, hanyanbo}@ncut.edu.cn, jianwu@sdsc.edu, wenyan84@hotmail.com

***Abstract*— In many collaborations across multiple organizations, both data integration and service integration are equally important. Most existing information system integration approaches focus only on one aspect, resulting incomplete intergration results. In this paper, we propose a businss object model where data and its services are correlated, and a corresponding unified approach in which the modeling, composition and interaction of both data and service can be achieved coherently. This approach can help dynamic businss collaboration by on demand and automatic updates of both data integration results and service integration results. The feasiblity and advantages of the approach is validated via a use case and a preliminary implementation.**

Keywords- Data Integration; Service Integration; Information System Integration; Dynamic Business Collaboration

I. INTRODUCTION

Business collaboration is cooperation between multiple enterprises or organizations working together to achieve a business goal [1]. In recent years, as a new emerging paradigm for distributed computing, Service Oriented Architecture (SOA) has been widely adopted, which can provide more flexible and dynamic collaboration pattern by on demand integration of existing services cross organizational boundaries.

However, most of current service integration approaches, such as workflow modeling or service compositions [2][3][4], mainly focus on the integration of service interfaces. A complete collaboration process needs to be clearly predefined before its execution. In these approaches, the business data are hidden behind the control logic of processes. Users cannot understand the data on the whole. They can only observe the data from the inputs and outputs of a service. It is hard for users to quickly capture data changes and timely adjust processes, especially when handling emergencies.

The process of a natural disaster relief, such as earthquake rescue, is a typical example, which involves the collaboration of multi-departments, e.g., government, transportation and hospital, and their information systems. Along with the updates of the disaster situations, decision makers should constantly adjust their relief plans. For example, new material shortage reports have to be responded promptly by updating the original relief plan.

In our opinions, both data and service are equally important and should be coherently correlated in dynamic collaborations. The collaboration should promptly respond to both data and service changes. Therefore, in this paper, we explore how to achieve both data integration and service integration and coherent interaction between them. The main contributions are twofold. First, a business object model is proposed to give equal attentions to data as well as its handling interfaces (services), which provides a possibility to easily share and integrate them in a coherent way. Secondly, based on the lifecycle of the business object model, a unified data and service integration approach, called UDSI, is proposed for dynamic collaboration. The approach includes business object modeling and composition, key information monitoring and dynamic event response. Via the approach, we can achieve on demand and automatic updates of both data integration results and service integration results.

The rest of the paper is organized as follows. In the Section 2, a simplified earthquake rescue case is presented to illustrate the problems we are facing. To alleviate the problem, our proposed UDSI approach is explained in Section 3. Then we apply our approach in the rescue case to demonstrate its feasibility and advantages. A prototype for the approach is illustrated in the Section 5. Section 6 compares our work with related works. In Section 7, we discuss our conclusions and plans for future work.

II. CASE STUDY AND PROBLEM ANALYSES

Delivery of Earthquake Relief Materials. When an earthquake happens in a remote area, the Emergency Office (EO) of the local government needs to take actions immediately to deliver relief materials to the disaster area in time. Near the disaster area, there are several Disaster Preparedness Centers (DPCs), which store different kinds of relief materials. The EO needs to make proper delivery plans for these materials. The collaboration process among EO and DPCs can be roughly divided into the following steps: 1) the EO will get the required relief material lists; 2) the EO will rank available DPCs according to the distances between their locations and the disaster area; 3) the EO will query the nearest DPC about its available material and dispatch trucks to deliver them; 4) the EO will loop step 3 until all required material are in delivery.

However, such collaboration process might not simply proceed by following the steps. Disaster situations are dynamic and evolving. The EO usually needs to make

978-0-7695-4757-2/12 $26.00 © 2012 IEEE
DOI 10.1109/SE.2012.16

decisions by analyzing related business data. For example, after an earthquake happens, the EO may launch a material delivering process to ship 1000 tents to the disaster area. The number of the required tents is calculated by collecting data about the disaster survivors at the beginning of the process. However, in the initial stage of an earthquake, the number of survivors cannot be completely counted. During the shipment, the number of survivors may greatly increase. Therefore, the EO needs to first check the amount of tents already sent and the amount of tents in delivery, and then deliver more materials to disaster area according to the new survivor number.

Data integration is very important to effectively handle the above-mentioned situations. If decision makers can timely receive the notifications about tents shortage and be presented the related information, they can quickly make new decisions to dispatch more tents. Further, after these new decisions take effects, users want to know their impact back on key data information. However, in current researches, there still lack a mature approach on how to help users understand the data in a service integration environment, and how to present business data changes when there are service changes and vice versa.

In order to deal with the above challenges, we will explain the rationales of our proposed UDSI approach and demonstrate how it works.

III. THE UDSI APPROACH

The main idea of the UDSI approach is to mix up the border of data model and service model. Data and service should both be regarded as the first class elements for business collaboration. So we propose a ***business object*** model where business data, modeled as *data object,* and its handling operations, modeled as *data services*, are two equally important elements. Data objects are used to model the business data to be shared across the organizations. They are represented with the nested relational model, which provide an intuitive and visualized way for business users to handling complex data. A data service related to a data object encapsulates a meaningful operation to handle the data object. By combining data and its services, business object model provides a coherent way to share, use and integrate data objects and their services.

The Fig. 1 shows the rationales of the UDSI approach. First, organizations in collaboration will encapsulate their information systems, and share their business data and services as business objects. For each business object, its metadata will be put into a registry center, which is called *business object community*. Different from service registry centers where services are the only elements, business object community provides a way to organize/view/search resources based on services, data or their combinations. We note that the business objects might be heterogeneous since each organization itself can decide the content and structure of each business object to be shared.

Business objects from individual information systems can be integrated to get composite ones. The composition process contains both data integration and service integration. We will explain in detail in the following sub-sections how to facilitate business object composition by utilizing and extending our previously developed "Mashroom" tools [5].

To help data object representation and monitoring, ***key performance indicators*** (KPIs) are proposed on the top of atomic and composite business objects to get their key real time information that is interested in the business collaboration. KPI information of a composite business object will be updated if any of its constituent business objects is updated. This update can be done automatically via the service integration made during the business object composition.

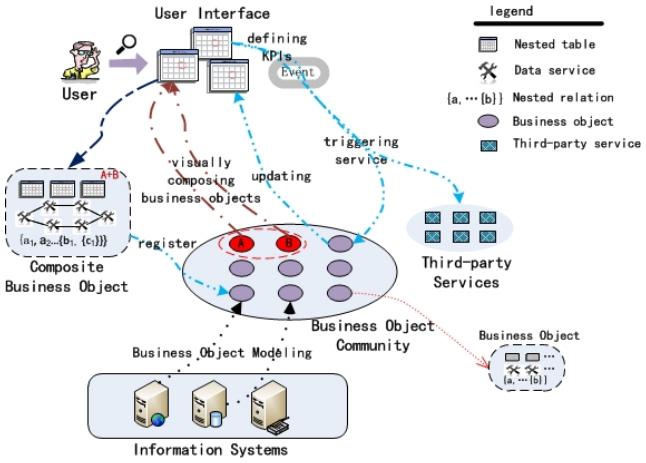

Figure 1. Rationales of the UDSI approach.

Besides key information notification, KPI can also be used to automatically trigger events once it changes. With these events, we can pre-define **event condition action** (ECA) rules to get recommended actions to deal with the event. Each action here is defined as an atomic or composite service, so that data evolutions/changes can be dynamically and automatically responded.

Based on the business object model and composition, data behind newly instantiated services by the above trigger actions can be easily gotten and combined into existing composite business objects. In this way, KPI can quickly reflect the impacts of the new services.

The UDSI approach mainly includes three phases. First, the business objects need to be modeled for the information systems of each organization in possible business collaborations. Secondly, the above-modeled individual business objects are composed to get collective data and services for certain specific business collaboration. Thirdly, the atomic and composite business objects are used to easily present key data information to users, and work together to respond to events like earthquake. We will explain these three phases in detail in the following sub-sections, respectively.

A. *Business objects modeling for individual information systems*

Organizations in collaboration first need to agree on how to interact with each other. In our approach, business object is used to model the information systems of each organization for their interaction.

Each business object is modeled by first modeling sharable data objects using nested relational model and then modeling data services affiliated with each data object using general operators of the nested relational model. We call these business objects that are modeled directly from individual information systems as *atomic business objects*.

Nested relational model is used to represent the internal structure and external view of a data object. This is because nested relational model is simple, intuitive, and expressive enough to represent the semi-structured or structured data [5]. For the definition of nested relational model, please refer to paper [6].

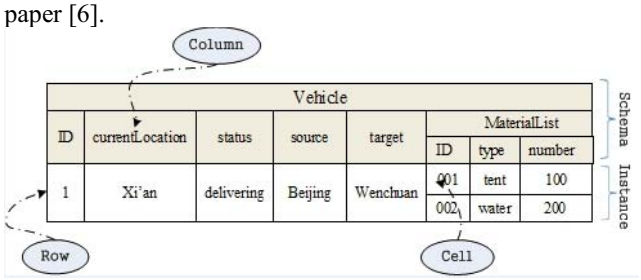

Figure 2. A sample of nested relation model.

With the nested relational model, business data can be visually presented as nested tables. Figure 2 shows a sample nested table. It presents the delivering information of a vehicle. The "Vehicle" relation has several atomic attributes (namely ID, currentLocation, status, source, target) and a sub-relation one named "MaterialList". The sub-relation describes the relief materials that are delivering or delivered by this vehicle.

As our previous work in [5], namely "Mashroom", we take "column" and "row" as first class objects of a nested table. A column represents an atomic attribute or a sub-relation, while each row represents a tuple. An atomic worksheet attribute can be one of six types: text, textlink, img, imglink, video, and videolink. Furthermore, the syntax of Mashroom formula is built from column references, row references, operators, and constant values. The syntax of column references and row references are built from path expression in order to express the nested relational model correctly. For example, a formula Vehicle/MaterialList/type specifies the "type" column of the relation "MaterialList" which is a sub-relation of the relation "Vehicle", and Vehicle/MaterialList specifies the first row of the sub-relation "MaterialList". When a row and a column of atomic attribute are given, then an array of cells is also defined. In the above example, when we choose the row 1and column "Vehicle/MaterialList/type", then an array of cells, namely [tent, water], is found.

As a data object is represented as a nested table, a data service can be built based on operators for handling nested tables. A set of operators for handling nested tables has been proposed in our previous work. Table I shows some commonly used operators. Complete operators could be found in [5]. With these operators, users can easily define required data operations and assign them with proper business semantics. To be compatible with third-party services, data services are also encapsulated as Web services.

TABLE I. UNARY OPERATORS FOR HANDLING THE NESTED TABLES

Operand	Operator	Semantics
Table	*CreateTable*	to create a new table by copying a sub-relation or an atomic attribute from one of the current tables
Column	*CreateColumn*	to create a new column and add it to a table
	DeleteColumn	to delete a column from a table
	RenameColumn	to rename a column from a table
Row	*Filter*	to filter a table by some condition, conditions is a group of condition expressions defined on the atomic attributes of this sub-relation
	Sort	to sort tuples in a table according to values of a certain attribute
Cell	*CellEdit*	to edit a cell in a table

Another challenge here is how to keep consistency when multiple services handle the same data object simultaneously. In our current design, the changes made by services should be transactional. An operation should have the effect of having exclusive control over the involved data object when making its changes.

B. Business objects composition from multiple information systems

For certain business collaborations among several organizations, such as delivery of disaster relief materials, the information of each organization needs to be shared and integrated on demand. So we need a simple way to compose the business objects of each organization in order to get an overall collective data and service.

Users can get composite business object by selecting needed business objects and composing them via the following steps: 1) choose business objects to be composed; 2) select or transform the data objects in these business objects as needed using unary operators in Table I; 3) combine multiple different nested tables into a new one using multivariate operators (see below for details); 4) modify the combined table again if needed using the unary operators; 5) define the combined table as a data object; 6) define the operator process used in step 1-4 as one special data service of the data object, called "update", and optionally define more data service for the data object; 7) publish a new business object based on the data object and its data services.

To facilitate operations on multiple tables, we defined two multivariate operators, namely *Merge*, and *Fuse*. More operators could be defined for other functionalities.

The merge operator merges schema of several tables and create a new head of a table. The formula of this operator for two tables is merge(a, b, <a.X, b.Y>), where both a and b can only be a relation or sub-relation, X and Y are atomic attributes of relation/sub-relation a and b. This operator is equivalent to the recursive union operation of the nested relational model. To note that only attributes with same names from two sub-relations will be merged. If the names of the attributes are not the same, users have to map the attributes first.

The fuse operator is designed to fuse values of attributes in several tables. The formula of this operator for two tables

is fuse(a, b, <a.X, b.Y>, function), where a, b can only be a relation or sub-relation and X, Y have the same semantics as they are in Merge operator. The function defines how to fuse different values. For example, for numeric values, we can do calculations including addition, subtraction, multiplication and others. For string values, we can concatenate or reverse them. The common functions can be pre-defined. Users can choose needed ones when fusing tables.

The above business object composition steps show both data integration and service integration can be achieved during the same process by including data and service in business object models. After the steps, users can choose to use this business object immediately or publish it for other users to use. We note that the same steps can be used to support nested business object composition.

By saving the composition process for each composite business object as a special data service, namely "update", the status and data of composite business objects can be automatically updated based on their constituent business objects changes. When a business object changes, it can trigger the changes of all composite business objects made from it by calling their "update" data services. Then the operators used for each composite business object will be calculated again to get its newest status and data.

C. Information Monitoring and Event Response

In many dynamic collaborations, existing systems need to promptly adapt themselves to fit new scenarios. In our approach, ECA rules are defined to dynamically respond to new events.

In the UDSI approach, we first define a set of values used to measure the status of a data object as KPIs. A KPI can be quantified as a value or a vector containing multiple values. The range of values can be flexible, such as number, character or string. The real time information of each KPI is gotten through periodically invoking the data services. KPI helps business users easily analyze the large-scale business data and quickly grasp their business semantics.

Since KPI represents key information in collaboration, changes of each KPI are also important. So we model the events and conditions of each ECA rule based on selected KPI information.

We model the action of each ECA rule as a service to interact with other services. An action might need to operate multiple services with dependencies, and these complex business logics can be modeled in services through service composition. In the service composition, both data services and third-party services could be employed.

After services are instantiated based on certain ECA rules, the data behind the services can be integrated into existing information monitoring via business object composition. New service instances could cause changes of existing business objects or creating new ones. As explained in Section III.B, business object composition processes are only based on the metadata of constituent business objects. So if changed business objects or the metadata of new business objects has been used in a business object composition, the composite business object can be automatically updated. The automatic update is done by including the new business objects and re-calculating its composition operators by calling its "update" data service. Otherwise, the new business object has to be manually added into composite ones, which will result in new composition process and "update" service. For instance, new materials in delivery, triggered by one ECA rule, need to be shown as parts of overall material information at the EO. If a composite business object already has vehicles as its constituent objects, its value can be automatically updated. This example will be illustrated in detail in Section IV.

In summary, dynamic interaction between data integration and service integration is achieved here. First existing service integration can be changed based on data updates by instantiating new services via ECA rules. Meanwhile, data integration results in composite business objects can also be updated based on the changes of data objects related to new services by calling or updating their "update" services.

IV. APPLICATION

In this section, the case for the delivery of earthquake relief materials in Section II is refined to illustrate how to use the UDSI approach. We will still use the three phases in Section III to demonstrate the application.

A. Atomic Objects Modeling

In this phase, each organization in the collaboration will model its information systems as a set of atomic business object, namely data objects as well as their data services. Main business objects in the case are showed in Fig. 3.

(1) DPC: Storage

The DPCs will model the Storage business object to represent their storage information about materials. Different DPC may have heterogeneous internal data structures and corresponding data services. In this case, we will use two different Storage business objects.

(1.a) DPC StorageA

The DPC StorageA mainly stores and manages the following materials: tents, stretcher, water and clothes. Therefore, its data object is StorageA (tents, stretcher, water, clothes). Several data services are provided for this object, such as getAvailableTentsNumber(), and getAvailableStretcherNumber().

(1.b) DPC StorageB

The DPC StorageB mainly stores and manages the following materials: stretcher, medicine and war. So its data object is StorageB (stretcher, medicine and water). It data services include: getStretcherQuantity(), getMedicineQuantity(), etc.

RequiredMaterial	
type	number
tent	1000
water	2000

Vehicle							
ID	currentLocation	status	source	target	MaterialList		
					ID	type	number
1	Xi'an	delivering	Beijing	Wenchuan	001	tent	100
					002	water	200

Storage A			
tents	stretcher	water	clothes
2000	3000	4000	1000

Storage B		
stretcher	medicine	water
1000	2000	1000

DestinationDepot		
ID	type	number
003	tent	500
004	water	300

Figure 3. Data objects and visual representation with nested-tables.

(2) EO: RequiredMaterial

The EO will model the RequiredMaterial business object to represent the required materials in the disaster area. The corresponding data object is RequiredMaterial (type, number). In that, the type attribute is for the type of required material, such as tent, water and so on. The number attribute is for the required quantity of a material.

Based on this data object, several data services can be provided, such as getRequiredMaterialList() and getRequiredMaterialNumber(type).

(3) EO: Vehicle

The EO will model the Vehicle business object to represent the information of a vehicle that is in charge of transporting materials to the disaster area. The corresponding data object is Vehicle (ID, currentLocation, status, source, target, MaterialList(ID, type, number)). For this data object, several data services are provided, such as getCurrentLocation (), getStatus(), deliver(), etc.

(4) EO: DestinationDepot

The EO will model the DestinationDepot business object to represent received material information in a temporary depot at disaster area. Its corresponding data object is (ID, type, number). An example of its data service is getAvailabelNumberOfMaterial(type).

B. Business Objects Composition

Furthermore, the above business objects can be composed into new business objects on demand. For example, when an earthquake happens, the materials like tents, water and medicine consume very fast. The EO needs to monitor the information about how many materials have been in delivery and how many materials are still available. The expected data object is CollectiveMaterial (type, numberInDelivery, availableNumber).

However, this CollectiveMaterial information is distributed among different kinds of data objects: Vehicle, StorageA, StorageB. With the operators defined in Section III, we can easily create a composite business object for this purpose and get its real time value. The combination process is illustrated in Fig. 4.

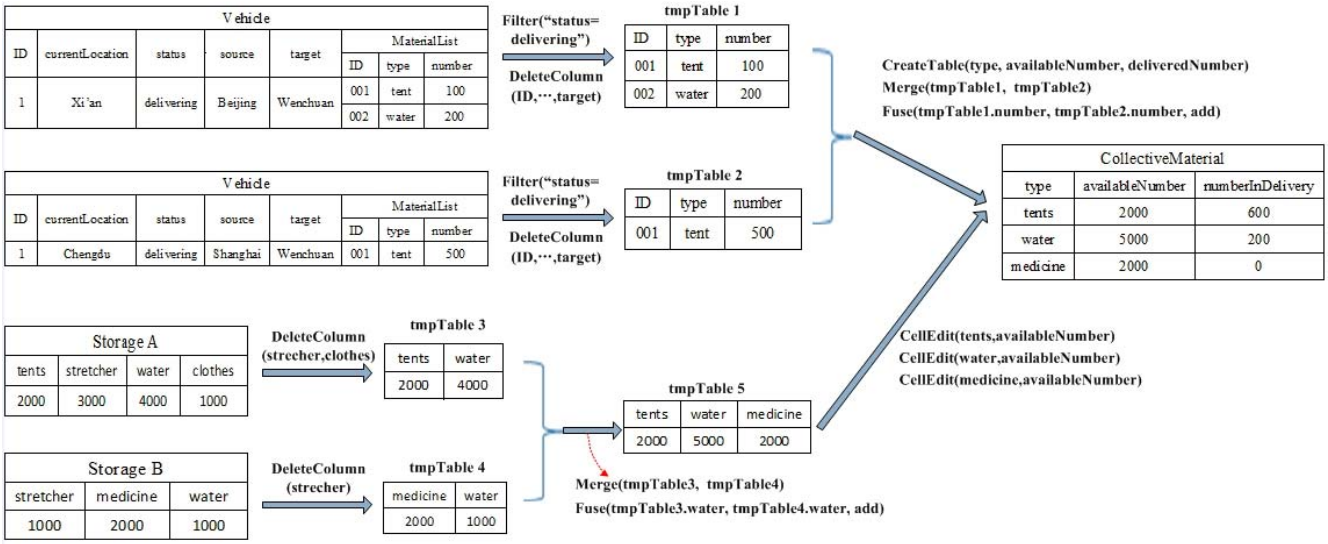

Figure 4. Composition of different data objects.

C. Information Monitoring and Event Response

First, real time value monitoring can be realized via KPI on business objects. Suppose the required tent number is one key information during an earthquake rescue process. When an earthquake happens, the required tent number needs to reported timely from the earthquake location. We can create a KPI based on the RequiredMaterial business object and its data service getRequiredTentNumber(). Its real time value is gotten through periodically invoking the data services, and notified to related users during the earthquake response period.

Data changes in KPI can trigger existing service integration changes. Once a new required tent number is reported, an event will be automatically created based on the KPI information and trigger its action defined by ECA rules. The action will trigger service invocation to get a recommended tent dispatch plan.

TABLE II. SAMPLE ECA RULES

1 If RequiredMaterial.getAvailableTent() > 0, then invoke createDispatchPlan(RequiredMaterial.getRequiredTentNumber());
2. If Vehicle.getStatus() = 'broken', then invoke createDispatchPlan(Vehicle.getMaterial());

Sample ECA rules are listed in Table II. The service logic of createDispatchPlan action is as follows. It will first rank the DPCs based on their distances to the disaster location (the nearer, the higher rank), then it dispatch all available tents in each repository until the summary tent number reach requested number. Detailed service logic can described using service composition language like BPEL [7].

Another KPI example is vehicle status. If a vehicle reports its status as "broken", the EO has to respond quickly. The same createDispatchPlan service in Table II can be used here to get a new plan based on the material amount in the vehicle.

Existing data integration result will reflect changes affected by new service instantiation reversely. For example, StorageA and StorageB need to decrease their material amount if more materials are sent out. And new Vehicle business objects are created for delivering the additional materials. CollectiveMaterial business object should reflect these changes. As shown in Fig. 4, business object CollectiveMaterial is composed from StorageA, StorageB and Vehicle. Existing changes for business object StorageA and StorageB and new Vehicle business objects can be easily incorporated into changes of the CollectiveMaterial object.

V. IMPLEMENTATION

We developed a prototype to support and validate the UDSI approach, which is shown in Fig. 5. Fig. 5(a) shows the tool to support the creation of an atomic data object. With this tool, users can first create the head of a nested table to describe the metadata of a data object. Then required data will be extracted from the data sources and transformed into a set of nested relations. Our current implementation can support several common data sources, including the relational database, XML, json and HTML. This process has two key issues. The first one is how to transform a given data source (such as HTML, XML, etc.) to the nested relations. The second one is how to let users decide what data should be transformed and how to realize it. The corresponding techniques can be found in our previous papers [8] [9].

Based on the nested relation model, the created atomic data objects can be easily visualized in a nested table. As Fig.5(b) shows, users may create data services by wrapping columns of a data object and choose required operators on them. The operators can be provided as services. The input parameters of an operator will be the input parameters of the service. For example, in Fig.5(b), when we select the city column, then we can choose the filter operators. The operator needs users to input the filter value. Here, we input the "北京(Beijing)". This causes to establish a query service. It will query all rows in the table where the value of city attribute equals to "北京(Beijing)".

Fig. 5(c) shows our tool to combine multiple business objects. The user interface is implemented on Flex [10]. The workspace is divided into several regions. Region ① displays the data service list. Once a data service is clicked, its example data will be displayed in region ② as a nested table. Region ③ is the current worksheet where the intermediate composition results will be displayed. Users can compose the services by simple drag the target columns from region ② to region ③ and use appropriate operators accordingly.

The KPI and corresponding ECA rules are realized with the EQL (Esper Query Language) [11]. The service for each action is either an existing Web service or a composite Web service expressed by BPEL. With the predefined ECA rules, the collaboration process can be realized in a more flexible way. When an ECA rule is triggered, the corresponding service will be invoked and the users who subscribe the captured event will be notified.

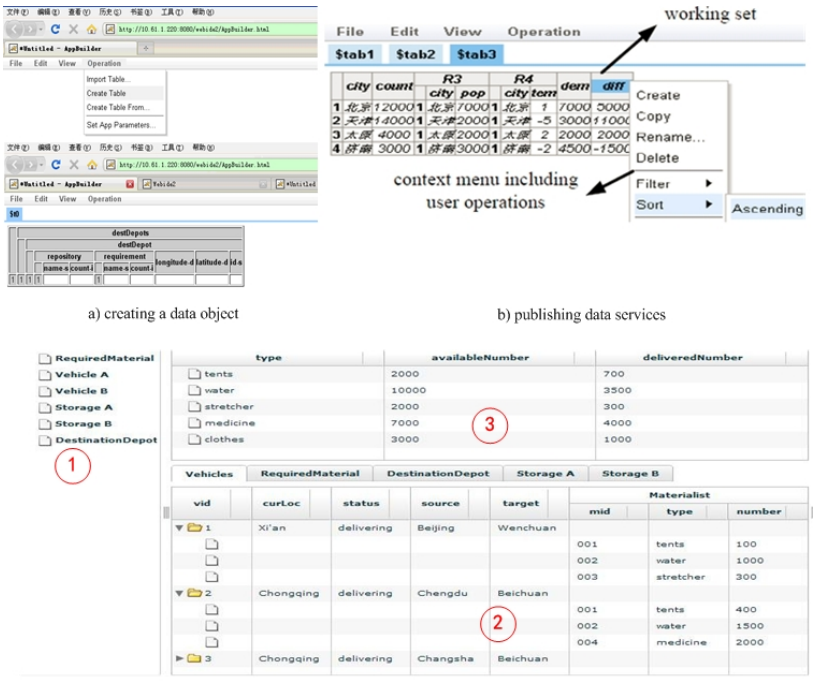

Figure 5. The implementation of the prototype.

VI. Related Work

In current research works, data integration and service integration usually belong to two different research areas. Although they have underlying relations, there still lack a powerful abstraction and practicable approach to coherently connect them.

Most data integration work aims to combining data residing at different sources, and providing the user with a unified view of these data [12]. In current works, a lot of attentions have been paid on how to establish a global schema, which can provide a reconciled and virtual view of the underlying sources [13][14][15]. Yet these works do not pay attention on how to enable data integration through services, and how to update it when services on the data changes.

Service integration study mainly focuses on combining the functionalities of Web services [2][3][4]. However, business data behind these services are hard to be combined. Only service inputs and outputs, and the mappings among them can be discovered and established.

Recently, some researchers have notified the importance of modeling mutual relations between data and services. Aalst et al. propose a new flexible paradigm for business process support - Case handling [16]. Different cases can establish a business collaboration via ECA rules, and the rules can be triggered by data changes. Cohn et al. recognize the importance of combining data with business process, and propose artifact-centric business process models [17]. A business artifact can be tracked as it progresses through a business process, and services in a business process can make changes to business artifacts. Müller et al. propose their COREPRO approach, which supports dynamic business process adaptation driven by its data [18]. However, these studies mainly focus on how data can help business process modeling and/or adaptation, and do not consider much on how to get a collective data results during business process execution and how the results are affected by business process changes. In contrast, our UDSI approach treats data and services equally in our business object model, explores how to model and compose data and services in a unified way, and specially investigates how data and service composition results can influence each other dynamically.

VII. Conlusions

In real application scenarios, data and services are often coherent ingredients for information systems. Many services are designed to handle business data and can be used to realize data integration. Further, data changes will influence the service invocations, and vice versa. This paper explores a unified data and service integration approach for dynamic collaboration, where the modeling, composition and interaction of both data and service can be achieved coherently.

In this paper, a business object model is proposed to model business data as data object and its handling services as two equally important elements. Based on the visual nested table operators, on demand composition of multiple business objects can be easily realized. We also propose KPI on the top of business objects to monitor data changes. KPI can trigger service invocations through pre-defined ECA rules so that the events of data changes can be dynamically responded. Further, data changes caused by the service invocations can be reflected in existing business object and

KPIs. Our preliminary prototype and application demonstrate the feasibility and advantages of our approach.

For future work, we plan to validate and refine the approach through more applications. Several aspects of the approach can be improved. First, we will study how to facilitate the composition of heterogeneous business objects via (semi-)automatic semantic matching. Secondly, we will investigate how to realize the role-based business object sharing and presentation since different business object might be shared based on user roles.

ACKNOWLEDGMENTS

The research work is partially supported by the National Natural Science Foundation of China under Grant No. 60970131 and No.60903048.

REFERENCES

[1] B. Orriens, and J. Yang, "Bridging the gap between business and IT in service oriented business collaboration," Proc. the IEEE International Conference on Services Computing (SCC05), IEEE, 2005, pp.315-318.

[2] S. Dustdar, and W. Schreiner, "A survey on web services composition," International Journal of Web and Grid Services, vol. 1(1), pp. 1-30, 2005.

[3] B. Benatallah, M. Dumas, and Z. Maamar, "Definition and Execution of Composite Web Services: The SELF−SERV Project," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 25(4), pp. 47-52, 2002

[4] F. Casati, M. Sayal, and M.C Shan, "Developing E-Services for Composing E—Services," Proc. the 13th Conference on Advanced Information Systems Engineering (CAISE 01), Springer, 2001, pp: 171-186.

[5] G. Wang, S. Yang, and Y. Han, "Mashroom: end-user mashup programming using nested tables," Proc. the 18th international conference on World Wide Web, 2009, pp. 861-870.

[6] L. S., Colby, "A recursive algebra and query optimization for nested relations," ACM SIGMOD Record, vol. 18 (2), pp. 273-283, 1989.

[7] BPEL, "Business process execution language for web services version 1.1," http://www.ibm.com/developerworks/library/specification/ws-bpel/, 2007.

[8] S. Yang, G. Wang, and Y. Han, "Grubber: allowing end-users to develop XML-Based wrappers for Web data sources," Proc. the Advances in Data and Web Management (APWeb/WAIM 2009), Springer ,2009, pp:647- 652

[9] G. Ji, G. Wang, and Y. Han, "Creating customized data services from web pages, " The journal of High Technology Letters, Accepted.

[10]Flex, http://www.adobe.com/products/flex.html, last retrieved at 2012.5.

[11] Esper, http://esper.codehaus.org/, latest retrieved at 2011-05-20.

[12] M. Lenzerini, "Data integration: A theoretical perspective," Proc. the 21th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2002, pp. 233-246.

[13] A. Levy, A. Rajaraman, and J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions," Proc. the 22th International Conference on Very Large Data Bases (VLDB96), Morgan Kaufmann Publishers Inc, 1996, pp.251-262.

[14] N. F. Noy, and M. A. Musen, "The PROMPT suite: interactive tools for ontology merging and mapping," International Journal of Human-Computer Studies, vol.59(6), pp. 983-1024 , 2003.

[15] A. Halevy, A. Rajaraman, and J. Ordille, "Data Integration: The Teenage Years," Proc. the 32th International Conference on Very Large Data Bases (VLDB06), VLDB Endowment, 2006, pp. 9-16.

[16] WMP. V. Aalst, M. Weske and D. Grünbauer, "Case handling: a new paradigm for business process support," Data & Knowledge Engineering, vol. 53(2), pp. 129-162, 2005.

[17] D. Cohn, and R. Hull, "Business artifacts: A data-centric approach to modeling business operations and processes," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 32(3), 2009.

[18] D. Müller, M. Reichert, and J. Herbst, "A new paradigm for the enactment and dynamic adaptation of data-driven process structures," Advanced Information Systems Engineering, vol. 5074/2008, pp. 48-63, Springer, 2008, DOI: 10.1007/978-3-540-69534-9_4.