A service-based framework for pharmacogenomics data integration
Kun Wang a, Xiaoying Bai a, Jing Li a & Cong Ding a
a Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
Available online: 09 Aug 2010

To cite this article: Kun Wang, Xiaoying Bai, Jing Li & Cong Ding (2010): A service-based framework for pharmacogenomics data integration, Enterprise Information Systems, 4:3, 225-245
To link to this article: http://dx.doi.org/10.1080/17517575.2010.498525

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
A service-based framework for pharmacogenomics data integration

Kun Wang, Xiaoying Bai*, Jing Li and Cong Ding

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

(Received 28 June 2009; final version received 1 June 2010)

Data are central to scientific research and practices. The advance of experiment methods and information retrieval technologies leads to explosive growth of scientific data and databases. However, due to the heterogeneous problems in data formats, structures and semantics, it is hard to integrate the diversified data that grow explosively and analyse them comprehensively. As more and more public databases are accessible through standard protocols like programmable interfaces and Web portals, Web-based data integration becomes a major trend to manage and synthesise data that are stored in distributed locations. Mashup, a Web 2.0 technique, presents a new way to compose content and software from multiple resources. The paper proposes a layered framework for integrating pharmacogenomics data in a service-oriented approach using the mashup technology. The framework separates the integration concerns from three perspectives including data, process and Web-based user interface. Each layer encapsulates the heterogeneous issues of one aspect. To facilitate the mapping and convergence of data, the ontology mechanism is introduced to provide consistent conceptual models across different databases and experiment platforms. To support user-interactive and iterative service orchestration, a context model is defined to capture information of users, tasks and services, which can be used for service selection and recommendation during a dynamic service composition process. A prototype system is implemented and cases studies are presented to illustrate the promising capabilities of the proposed approach.

Keywords: Web services; data as a service; mashup; context-awareness; ontology

1. Introduction

With the development of experiment methods and information retrieval technologies, the scale of scientific data and databases grows explosively day by day. For example, TERAPIX (TERAPIX, 2003) (Traitement Elementaire, Reduction et Analyse des PIXels), an astronomical data reduction centre, contains over 200 TB observation data and its size increases 320 GB per night (GTI 2004). Pharmacogenomics is a new interdisciplinary research area which combines the research of functional genomics and molecular pharmacology, focusing on the relationships between genetic mutations and diseases. It compares the drugs effects with different gene sequences to find new therapeutics for specified genes or even specified patients. The experiments generate multiple kinds of high-throughput data.

*Corresponding author. Email: baixy@tsinghua.edu.cn
Many databases have been created to store and share data from different perspectives. A research (Galperin and Cochrane 2009) from NAR (Nucleic Acids Research) shows that the number of databases published in annual database issue increases about 50 times from 1993 to 2009. According to the research, about 100 new databases were published in 2008, which resulted in a total of 1170 online molecular biology databases (Galperin 2008).

Data have become an essential resource for scientific research and practices. For pharmacogenomics research, data of different experiment results enable the scientists to better understand the cellular processes, the organism behaviours and the relations between diseases and genes. Hence, the integration over diversified data set is necessary for scientists’ daily research. Most of current databases focus on a specific aspect of biomedical. For example, NAR lists a set of databases (NAR 2009), including 3D-Genomics for structural annotations of complete proteomes (3D-Genomics 2004), KEGG for metabolic and regulatory pathways (KEGG 1999) and PubMed for citations and abstracts of biomedical literature (PubMed 2009). The integration of massive information and multiple data types is highly demanded to facilitate comprehensive data analysis practice. In addition, the value of the data can be raised to a higher-level while integrating with others (Smith et al. 2007).

However, data integration for pharmacogenomics suffers a lot from the heterogeneous problems. Data are maintained by independent owners in diversified formats and ad-hoc structures. For example, the raw data of Microarray Comparative Genomic Hybridisation (Microarray CGH) experiments are stored in different data formats and structures. The comparison experiments data are stored in text files with complex structure, which includes information of experiment equipments, potency of dyes, contract of scanned image and results values. The data of gene expression experiments are stored in structured files that contain potency of drugs and result values. Furthermore, most of current databases use inconsistent terminologies and semantics. In an NAR research (Galperin and Cochrane 2009), it classifies the databases into 14 types, each focuses on a specific aspect of biomedical research and experiments. Research shows that across the databases, the same element may have different terminologies with different semantics. The gene ‘Leukemia Inhibitory Factor’, for example, is named ‘LIF’ in Human Genome Organization (HUGO) (HUGO 1989) and ‘Hs.2250’ in UniGene (UniGene 2003) database. In PubMed literature database (PubMed 2009), there are about 70 identifiers that include ‘1692837’ and ‘1714745’ for the referral of specified gene. The heterogeneous problems have become major hindrances to seamless data understanding. Methods and tools are necessary to provide a unified approach in accessing and integrating the diversified data.

The widespread of web platform brings a new way for data sharing. Web-based databases and data integration are gaining attentions in recent years. Data as a Service (DaaS) emerges as a major trend to address the heterogeneous data sharing and integration problems in the Service-Oriented Architecture (SOA) (Hacigümüs et al. 2002). More and more databases are accessible through standard protocols and programmable interfaces over the Internet. Lots of online data services are now available for customers to access databases, transfer messages and structures in a new application. In biomedical research, database system KEGG (KEGG 1999) is accessible through Web Services protocols for searching and computing biochemical
pathway data in cellular processes or analysing the universe of gene data in completely sequenced genomes.

Traditional integration techniques like link-based, view-based and data warehousing (Stein 2003) are difficult to meet the requirements of cross-querying and correlating the Web-based databases. The article proposes a service-based framework for integrating diversified data in pharmacogenomics in various business scenarios. Services are autonomous and loose-coupled software components with exposed standardised interfaces (Papazoglou 2008). By encapsulating the database software into services, it enables integration at a higher abstraction level and hides the complexity and heterogeneity of internal structures and formats. Web services, an implementation of the SOA, introduces a set of standard protocols, including interface specification like WSDL, workflow specification like BPEL and semantic specification like OWL-S. They serve as contracts among the servicetised components and enable their dynamic interoperation and collaboration. In SOA, an application is defined as a collection of workflows with reusable services. To meet the business requirements, these interoperable service units can be dynamically composed or re-composed at runtime. It allows the system to switch functions or data providers in a flexible way. The loosely-coupled integration environment provides suitable methods to reduce the cost in system change and evolution during integration process.

In pharmacogenomics research, the scientists need to flexibly select data sources, types and to decide the workflow of data processing and analysis. The process depends heavily on the data from public databases and tools implementing standard algorithms. Taking the service-based approach, the users can benefit from following aspects: (1) to be transparent to the underlining programming details and heterogeneity problems, and focus on biomedical research objectives and experiments design; (2) to reuse, in a standard approach, the resources available from the Internet. The article introduces an extended OWL-S model to support semantic-based composition and dynamic binding of services in an interactive and iterative approach.

The article investigates the service-based framework in general and two key enabling techniques in particular: ontology system and context-aware service composition. Ontology is used to define a unified conceptual model in service orchestration. The ontology model can be used to enhance query translation and result filtering. A generic query can be mapped to a set of related terms with equivalent semantics, resulting in a larger query scope and better recall ratio. The query results can be analysed to filter duplicated, inconsistent and conflicted records. Context information is used to facilitate service recommendation and selection based on usage history and runtime system behaviour (Han et al. 2005; Schilit et al. 1999).

The rest of the article is structured as follows. Section 2 presents the overall design of the service-based mashup framework for data integration in pharmacogenomics. Section 3 briefly introduces the ontology-aided approach for data and functional service mashup. Section 4 discusses the context model and mechanism of context-aware interactive and iterative service composition. Section 5 depicts the system implementation and experiments to illustrate the proposed approach. Section 6 surveys some projects and research in related fields. Section 7 presents the conclusions and future work.
2. Service-based mashup framework

Mashup provides a light-weight technology for dynamic integration of service-based software. The article introduces the mashup concepts and technologies at different integration layers to establish a service-based environment for pharmacogenomics research. Figure 1 gives an overview of the three-layer mashup framework. The Data Service Mashup enables cross-database querying by dispatching the user query to multiple data sources in specified formats and unifying the gathered results from different data sources. It hides the differences in data formats, structures, storages, and access and query methods. The Functional Service Mashup helps the user to operate and analyse data by linking to the functional services over the Internet. A service pool of business functions is maintained for binding the service abstraction in the process specification to its real instances. The Process Engine interprets the abstract service process specifications and executes the process by dynamic binding to and invoking of the services instances indexed in the service pool. The Presentation Mashup enables the user to integrate necessary data views and tools in his/her customised layout.

2.1. Data service mashup

The Data Service Mashup provides a unified Web Service interface for accessing and querying data of various types and experiment levels from different Web-based data sources. The data can be in different fields and formats. For example, in this research, we collected data of gene profiles, annotation of biochips and experiment results from public, contracted and confidential databases.

As shown in Figure 2, the Data Service Mashup layer is composed of three main parts: the Query Translator, the Query Bridge and the Result Assembler. The original data query is defined by three attributes: the data type, the keyword and the expected database. The query conditions can be strengthened by filtering mechanisms. For example, it may query on the gene (data type) with a specific name (keyword) on a set

![Figure 1. The framework of service-based data integration.](image-url)
of platforms (filters). The Query Translator (QT) converts the original query in two ways: (1) it finds the set of services with equivalent functionalities based on its internal knowledge base and fulfils query with database service information; (2) it transforms the query into an XML-based internal representation in a unified format.

An example of an internal query is illustrated below. In this example, ‘PTGS2’ is the name of gene to be queried. However, it is marked as different attributes in different systems. For Affymetrix experiment data, it is identified as ‘Symbol Name’ while in KEGG system as ‘Gene Name’. The unified query hides the differences.

```xml
<query idType = 'Symbol'>
  <keyword>PTGS2</keyword>
  <service>
    <type>Web_Service</type>
    <servID>U95</servID>
    <URI>LOCAL</URI>
    <idType>Symbol</idType>
  </service>
  <service>
    <type>Crawler</type>
    <servID>Pathway</servID>
    <URI>166.111.134.69</URI>
    <idType>KEGG</idType>
  </service>
</query>
```

The Query Bridge (QB) is composed of Query Adaptor (QA), Query Invoker (QI) and Result Collector (RC). It converts the unified query to the specific formats required by various data sources (QA), binds and invokes the specific data service
(QI), collects the results (RC) and unifies the heterogeneous results into an internal representation (QA). The QI dispatches each statement which is generated by QA to its specific data source. It retrieves the data in different ways. For data and information that keep stable, a local repository maintains a copy of the data and synchronises periodically with the remote server. For frequently updated information, the system dynamic gathers the data from the web either through crawler or binding to the services through provided access points. QI uses different components for data retrieving, such as a download client, a crawler service and a service invoker. The RC gathers original results from multiple data sources and forwards them to QA for integration. The results may be presented in different forms below:

1. Plain text, which often comes from download client and page crawler.
2. URI from databases, redirection information and URLs on the pages which represents a chart or a data file.
3. Web page clips like data forms.
4. Structured results from data services.
5. Service invocation status which includes error messages from the service.

The results returned from different data sources differ in formats and semantics. The RA analyses the consistency of the cross query results, identifies the controversy or contradictions, filters the duplicated records and produces a cleaned results set. In general, the RA prepares the results in three steps:

1. Unification of data structures, which transforms diversified data structures into the same format for further comparison and analysis.
2. Terminology mapping, which converts the different terms into the same concepts following a unified conceptual model.
3. Filtering, which removes the duplicated records.

2.2. Functional service mashup

Given a set of data, scientists may go through various analysis methods to get their individual conclusions based on the scientists’ insights and experiences. A process is made up of data processing and analysis operations. For example, functional annotations are frequently-required operations, such as gene ontology, biology pathway and detection of transcriptional factors binding site. Many of these functional annotations are also published online in knowledge bases, which are accessible through the Web-based applications. The existing functional annotation tools and services are isolated from each other. But in a data processing and analysis process in pharmacogenomics, the functions need to interoperate with each other. Hence, an important requirement from scientists is to dynamically compose the analysis operations. A layer above the cross domain data query is necessary for process modeling and functional service integration. The goal of Functional Service Mashup is to integrate the data with functional annotations into a process, and the integration should be able to be customised and personalised.

As shown in Figure 3, the Functional Service Mashup layer consists of three components: (1) the Functional Service Pool where the service descriptions are stored; (2) the Process Definition to define the operation process using service abstractions; and (3) the Process Engine which interprets the process definition, binds each service
abstract to its instances and invokes specified services for execution. When a process is activated, the Process Engine loads the Process Definition and finds the services indexed in the Functional Service Pool.

A process is encoded using XML-based specification language. It is defined as a composition of a set of atomic tasks that can be dynamically bound to service instances and invoked. The Process Engine supports process execution in two modes:

1. **Automatic mode**: The process is fully defined and each service in the composite process has been bound to a real implementation in the service pool. In this case, the process engine works as traditional OWL-S process engine to perform the execution automatically.
2. **Interactive mode**: The process is not fully defined or the task is not instantiated to a dedicated service implementation. For example, a scientist may prefer to choose the activities or service implementations on-the-fly based on the results accumulated during the process. In this case, the process engine interprets the process step by step. At each interactive point, it allows the user to decide which activities to follow up and which service to choose from the available services in the service pool.

### 3. The ontology mechanism

To cope with the heterogeneous problem, a unified data model is needed to ensure consistent data operations and functional analysis. In the proposed framework, a data ontology system is introduced to provide the unified conceptual model for mashing up data and functional services.

#### 3.1. The data ontology

An experiment-centric ontology model is defined to identify the necessary concepts and information for data understanding and mashup. Based on the Open Biomedical Ontologies (OBO) (OBO 2007), we define our data ontology with four main concepts, as shown in Figure 4:

- **Experiment**: The concept of ‘Experiment’ describes basic information of biology experiment. The semantic information has three parts:
Experiment information that covers different levels of genetic engineering. For example, DNA level, mRNA level, protein level, etc.

(2) Platform (bio-chips) information that covers the manufacturer, experiment method and data throughput. Sometimes, the platform information also contains details of experiment materials.

(3) Normalisation method (e.g. MAS5, LOG2, GCRMA) which is needed to compare the raw data from experiment with numerical differences in different environments.

- **Exp_Data**: It defines the ontology for data from different experiment platform. In Figure 4, ‘GE_Data’ provides the data model for gene expression experiment. This kind of experiments uses specified GE chips to carry different celllines in ‘Cells’. Thus, the model of ‘GE_Data’ describes the position and cellline information of each data record. The ‘SNP_Data’ for single nucleotide polymorphism experiment provides chromosome location and sequence of each gene. Each type of experiments has its specified model.

- **Annotation**: The ‘Annotation’ includes environment status (e.g. temperature, humidity, drug concentration, etc.), platform information (e.g. manufacturer, etc.) and chip annotations (e.g. drug in cells, gene, cellline, etc.).

- **Profile**: The ‘Profile’ describes gene and celllines instances used in experiments. It includes information about gene name, official symbol name (name in HUGO (HUGO 1989)), gene description, chromosome location, alias, etc.

### 3.2. Ontology-aided service mashup

The ontology model enables the scientists to augment an initial set of keywords with data derived from domain ontologies. The enriched keyword sets are then used for querying across multi-sources and integrating the results. The ontology is used in the...
whole process of data query and service orchestration. A typical query scenario is given as follow:

(1) The user inputs a keyword for query;
(2) The system searches the ontology model for relevant terms and database query formats;
(3) The relevant terms are used as the keywords to further search across multiple sources;
(4) The query results are collected and mashed up together with the aid of ontology.

By enlarging the query scope to include the data related to the user-specified input, it may produce more results that potentially meet the user’s needs. In addition, it can also use ontology techniques such as semantic distance analysis to further filter the initial query set and the result set to find the most relevant data.

OWL-S, the semantic language for Web Services, is used to describe the semantic information of service process. OWL-S is an OWL-based semantic markup language. It specifies the intended system behaviour in terms of inputs, outputs, process, pre-/post- conditions and constraints using ontologies. The composite service is modeled as a workflow of processes including atomic, simple and composite processes. Two components are used to define the OWL-S process model: the Process ontology describes the service IOPE (inputs, outputs, preconditions and effects) properties; and the Control Construct Ontology describes the process control constructs such as sequence, iterate, if-then-else, and split. Following the OWL-S standard, composite processes are decomposable into other (non-composite or composite) processes. The execution of the process is governed by the control constructs such as Sequence and If-Then-Else (Figure 5).

In this research, the process is specified using extended OWL-S to support dynamic service binding. The ‘AtomicProcess’ in OWL-S is replaced by an object of ‘AtomicServiceSet’. The ‘AtomicServiceSet’ defines the atomic activity as a service type which can be implemented by a set of alternative services supporting the same functionalities. For example, Google and Yahoo both provide ‘search’ services on the Web. Hence, they can be categorised in the same ‘AtomicServiceSet’. The Functional Service Pool contains descriptions of functional services and tools. Tools in the pool are also wrapped as web services to support dynamic binding methods. Each functional service, either from the third party or local host, is described using OWL-S (Martin et al. 2004).

Figure 5. Extended OWL-S process.
4. Context-aware service mashup

Context-aware mechanism is introduced to facilitate the on-the-fly decision making for service selection and integration. Figure 6 illustrates the process of the context-aware service composition. The *Task Analysis Layer* is designed to analyse the user requirements and to identify a task that may meet the user’s preferences. The *Service Binding Layer* instantiates the services and binds them to the service implementations. The *Execution and Monitoring Layer* exercises the *Executable Task* to invoke the remote services and monitors events during the invocation. Context provides the necessary information of business requirements, service behaviours and user interactions through the interactive and iterative composition process. A feedback mechanism is provided through the monitoring layer to enable user awareness of the dynamic changes during the process.

4.1. Context model

The contextual information is modeled from three categories: task, service and user. Each type of context is further analysed from two perspectives: *Static Context* and *Dynamic Context*. The *Static Context* contains information and properties which are seldom changed, for example, the alternative tasks and the user profiles. The *Dynamic Context* is updated online during service composition and execution by monitoring system runtime status and learning from historic data, for example, the availability and response time of service implementations.

![Figure 6. Context-aware service composition.](image-url)
The contexts are specified using XML markup language. An example of dynamic task context specification is shown as follows. In this example, it records the pre- and post- tasks of a specific task in its execution history and counts the execution times to evaluate their relationships. The information will be used for task recommendation.

```xml
<TC>
  <DynamicTC>
  </TaskHistory>
    <PRE taskName = 'TaskB' count = 10 />
    <PRE taskName = 'TaskC' count = 4 />
    <POST taskName = 'TaskB' count = 10 />
  </TaskHistory>
</DynamicTC>
</TC>
```

**Definition 1: Task Context.** Given a task \( t_i \in T \), its context \( TC(t_i) \) is defined as a tuple of static context \( STC \) and dynamic context \( DTC \), that is, \( TC(t_i): = < STC(t_i), DTC(t_i) > \). Where

1. \( STC(t_i) := ALT(t_i) \) is the set of alternative tasks to \( t_i \). That is, \( ALT(t_i) \subseteq T \), and \( \forall t_j \in ALT(t_i) \), \( t_i \) and \( t_j \) achieve the same goal through different approaches.
2. \( DTC(t_i) := < p, t, n > \) records the history of the execution, in which, \( t \in T \) is a task that is once executed together with \( t_i \); \( p = PRE|POST \) identifies the sequence of \( t \) and \( t_i \). \( PRE \) means that \( t \) is exercised before \( t_i \), while \( POST \) means that \( t \) is exercised after \( t_i \); \( n \) is the count of the processes in the history that \( t \) and \( t_i \) are exercised in sequence.

Suppose that \( T = \{ t_i \} \) is the set of tasks. For task \( t \in T \), the Task Context captures the following information:

1. **What are the alternative tasks to \( t \)?** There may be alternative approaches to achieve the same business goal. For example, to process data, the user may choose different data normalisation methods, or different sequences of operations. Such information is defined as static context.
2. **What is the execution history of \( t \)?** The history records show how the tasks work together with others. Mining on such information can provide useful hints for recommendation. For example, the tasks that are most frequently exercised after \( t \) have a high rank for recommendation. Such information is defined by dynamic context.

The Service Context describes service’s pre-defined execution environment and runtime properties. From the static perspective, it identifies the set of alternative services providing similar functions, the necessary preparations for service invocation such as environment set up, and the necessary cleanups after service execution. From the dynamic perspective, it identifies the service operational profiles and execution status, such as accessibility, throughput, response time and current connections.
Definition 2: Service Context. Given a service \( s_i \in S \), its context \( SC(s_i) \) is defined as a tuple of static context \( SSC \) and dynamic context \( DSC \), that is, \( SC(s_i) := < SSC(s_i), DSC(s_i) > \). Where

1. \( SSC(s_i) := < ALT(s_i), PREC(s_i), POSTC(s_i) > \), in which, \( ALT(s_i) \) is the set of alternative services to \( s_i \). That is, \( ALT(s_i) \subseteq S \), and \( \forall s_j \in ALT(s_i), s_i \) and \( s_j \) are functional equivalent. \( PREC(s_i) \) is the set of preconditions while \( POSTC(s_i) \) is of post-conditions.
2. \( DSC(t_i) := < prop, value > \) records the runtime properties of a service using the key-value definition.

The User Context records user profile and interactions. User profile identifies static user characteristics such as identity and research background. Interactions capture the user preferences and selections in the history and in the specific process.

Definition 3: User Context. Given a user \( u_i \in U \), its context \( UC(u_i) \) is defined as a tuple of static context \( SUC \) and dynamic context \( DUC \), that is, \( UC(u_i) := < SUC(u_i), DUC(u_i) > \). Where

1. \( SUC(u_i) := < prop, value > \) is the set of user personal information defined by key-values.
2. \( DSC(t_i) := < cookie, < choice, count >> \) identifies user interactions in current session using key-value records of the cookie mechanism, and the selection history by counting the number of each choice.

4.2. Interactive and iterative context-aware composition

As shown in Figure 7, the context-aware composition is an interactive and iterative process as follows:

1. The user identifies the work to be done, such as data analysis across multi-dimensional data sets from different experiments. The system recommends the business process based on some pre-defined templates and history profiles and confirms with the user to define the process.
2. The system analyses the information recorded in the context database such as previous results, users’ preferences, task execution history and so on. It selects the tasks to perform and recommends them to the user.
3. The user confirms the task selection.
4. The system identifies the set of alternative services that are associated to the task. It analyses the monitoring log information and obtains the current accessibility and QoS properties of each candidate service. Based on the context information, it selects the services and recommends them to the user.
5. The user confirms the service selection.
6. The system binds to the selected instance of service implementation.
7. The user inputs the data as necessary.
8. The system invokes the service remote execution and gets the execution results.
9. The system returns the results to the user.
(10) Repeat 2) through 9) until all the tasks have been exercised and the work done.

An important mechanism to build dynamic context is to track the runtime information of task and process definition, service execution and user interaction. In this research, monitor function is instrumented into the execution engine of process interpretation and service invocation.

5. Experiments

A prototype system is implemented to illustrate the proposed approach. Figure 8 shows a snapshot of the prototype system - FunGene. The four-panel Web page shows the data and information from experiment database, a pathway service (KEGG (KEGG 1999)) and the visualised results using line charts. The data types and sources for mashup are selected to meet specific user requirements and personal profiles.

The FunGene system now integrates more than 10 GB annotation data and 17 million experiment data from 25 experiment sets on 15 different platforms, as listed in Table 1. The data are obtained online from public databases such as NCBI (NCBI...
The data covers experiment results of three levels: gene expression level, protein level and RNA level. Experiments are exercised on the FunGene prototype system which mashes up 13 services in 6 categories as listed in Table 2.

In the first experiment, we simulate a random failure rate of service connections. The system monitors the availability of the services and dynamically adjusts the recommendation and selection of services. In case a service becomes unavailable, the system will switch to another one with similar functionalities.

Table 3 compares the success rate service invocation between the approach with dynamic context information and that using pre-defined static process specification. Compared with the static approach, such context-aware approach can greatly enhance the success rate (about 13% in the example) and improve the system fault tolerant capability.

The second experiment shows how the context information can improve service recommendation. Traditionally, users are only provided with a service index for selection. With the help of context information, the system can rank the available services based on the services or users historic profiles so that it can recommend the services that have the highest possibility to meet the users needs and preferences. In

---

Table 1. Data sources and types.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Data type</th>
<th>Format</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCBI</td>
<td>SNP Exp.</td>
<td>CEL</td>
<td>8.1 GB</td>
</tr>
<tr>
<td>NCI (CellMiner)</td>
<td>GE/CGH Exp.</td>
<td>CEL/XLS/GPR/TXT</td>
<td>2.2 GB</td>
</tr>
<tr>
<td>Stanford U.</td>
<td>CGH Exp.</td>
<td>TXT</td>
<td>1.83 GB</td>
</tr>
<tr>
<td>BioInfo</td>
<td>Gene Info.</td>
<td>XLS</td>
<td>21 MB</td>
</tr>
<tr>
<td>Affyrmatrix</td>
<td>Annotation</td>
<td>CSV</td>
<td>540 GB</td>
</tr>
<tr>
<td>Previous research</td>
<td>ID systems</td>
<td>MDB</td>
<td>128 MB</td>
</tr>
</tbody>
</table>

---
the example scenario, if context information is not considered, to recommend randomly from the list after data query, it has only 1/13 (about 7.69%) possibility to choose each service. However, according to the context of history information, 47.21% user records choose ‘DataTable’ and 31.85% ‘LineChart’. Hence, the system may recommend the two services ranked highest. In general, with the context-aware approach, the system is able to (1) select the most often used service based on usage history profile; (2) select the most probable follow-up service based on the profile of current service; (3) select the most preferred service based on the profile of user preferences; and (4) select the most robust service based on the service QoS profile.

The third experiment investigates the impact of ontology in service composition. We exercise six experiments on two platforms (Affymetrix HG_U95A and Affymetrix HG_U133A) from NCI60 database (NCI60 1999). A probe with HUGO symbol name ‘PTGS2’ has two different probe IDs at the two platforms. At the HG_U95A platform, the probe is identified as ‘1069_at’ while at the HG_U133A platform, it is ‘204748_at’. A query is submitted to find the experiment data of probe ‘1069_at’. Without the ontology support, it can only find the records in the HG_U95A experiment database. With the ontology support, the initial query on ‘1069_at’ is then transferred to the query for all the experiment data that either ‘probe ID = 1069_at’ or ‘symbol name = PTGS2’. In this way, it finds all the records in both of the HG_U95A and HG_U133A experiment databases. However, there exist duplication in the result for the records in the HG_U95A database which satisfy both of the ‘probe ID’ and ‘symbol name’ criteria. The system then removes the

Table 2. Integrated online services.

<table>
<thead>
<tr>
<th>Data-centered Service</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEDataQuery</td>
<td>Data query</td>
<td>Query values from Affymetrix) experiments</td>
</tr>
<tr>
<td>DataTable</td>
<td>Data view</td>
<td>Use a data table to show all data results</td>
</tr>
<tr>
<td>LineChart</td>
<td>Data view</td>
<td>Use a line chart to illustrate data values</td>
</tr>
<tr>
<td>Variance</td>
<td>Data analysis</td>
<td>Calculate variance value</td>
</tr>
<tr>
<td>ConfidentialInterval</td>
<td>Data analysis</td>
<td>Calculate confidential</td>
</tr>
<tr>
<td>Mean value</td>
<td>Data analysis</td>
<td>Calculate mean value</td>
</tr>
<tr>
<td>DavidFullInfo</td>
<td>Gene information</td>
<td>Get full information of a gene</td>
</tr>
<tr>
<td>DavidBasicInfo</td>
<td>Gene information</td>
<td>Get basic information of a gene</td>
</tr>
<tr>
<td>DavidMinimalInfo</td>
<td>Gene information</td>
<td>Get minimal information of a gene</td>
</tr>
<tr>
<td>KEGGAllPathway</td>
<td>Pathway</td>
<td>Get biology all available pathways</td>
</tr>
<tr>
<td>KEGGRandomPathway</td>
<td>Pathway</td>
<td>Get a random biology pathway</td>
</tr>
<tr>
<td>KEGGBasicInfo</td>
<td>Gene information</td>
<td>Get basic information of a gene</td>
</tr>
<tr>
<td>KEGGStructure</td>
<td>Structure</td>
<td>Get 3D structure of a protein</td>
</tr>
</tbody>
</table>

Table 3. Success rate with and without context.

<table>
<thead>
<tr>
<th>Task</th>
<th>With context (%)</th>
<th>Without context (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1 (Query)</td>
<td>95%</td>
<td>95%</td>
</tr>
<tr>
<td>Task 2 (View)</td>
<td>99.75%</td>
<td>95%</td>
</tr>
<tr>
<td>Task 3 (Statistics)</td>
<td>99.9875%</td>
<td>95%</td>
</tr>
<tr>
<td>Task 4 (More info.)</td>
<td>≈ 99.99997%</td>
<td>95%</td>
</tr>
<tr>
<td>In Total:</td>
<td>≈ 94.7506%</td>
<td>≈ 81.4506%</td>
</tr>
</tbody>
</table>
duplicated results. As a result, the query without ontology support gets 180 records; while the query with ontology support gets 540 records in total and removes 180 duplicated ones, resulting in final 360 records. A higher recall ratio is obtained for query across experiments.

6. Related work

Many research projects have been devoted to biomedical data integration from both biology and computer-science perspectives. The Biomedical Informatics Research Network (BIRN) (Astakhov et al. 2005), the Shared Pathology Informatics Network (SPIN) (SPIN 2000) and the BioGRID (Stark et al. 2006) have funded many projects on the research of biomedical data sharing. The INBIOMED (López-Alonso et al. 2004) is a biomedical informatics network that coordinates biomedical informatics and biomedical research groups. It creates Virtual Organisations (VO) from each individual project. Each VO can control its own data and utilities and share them to specified users. The researchers from GSPH (Department of Human Genetics at the University of Pittsburgh’s Graduate School of Public Health) cooperate with MAYA Design and develop a Peer-to-Peer Universal Genetics Database (UGD) (Widdows and Barmada 2006). It can automatically integrate background information in genetics databases and experiment results together and provide the integrated data and tools to researchers. The system divides all information and data into U-forms that can be moved and copied within the network. New variables and properties can be added into any U-forms so that the information and data can change its structure on demand for different usage. The MMIM (Molecular Medicine Informatics Model) (Hibber et al. 2007) project provides a virtual research repository of clinical, laboratory and genetic data sets. It uses a federated approach to integrate data from distributed and independent organisations. A global glossary is defined to establish the mapping and connection among the different data sets.

L. D. Stein (Stein 2003) analyses the difficulties of bioinformatics database integration and compares the three integration approaches, as summarised below.

- **Link integration**: The link integration allows the researchers to begin their query with one data source, and then follow hypertext links to related information in other data sources. It is by far the most successful technique. However, it is still problematic on several fronts. First, the links represents a leap of faith that the page at the other end of the link is still valid. Second, the link integration is vulnerable to naming clashes and ambiguities.

- **View integration**: The view integration also leaves data in its source databases as link integration. But the view integration builds an environment that all data from other source seem to be part of one large system by using a cross-database query language (e.g. Kleisli and K2 (Davidson et al. 2001)). The view integration suffers from the complexity of add/remove sources.

- **Data warehousing**: Different from the former two methods, data warehousing duplicates data from all integrated sources under a unified data model. It is difficult to maintain a data warehouse. The biggest issue is to keep the data up-to-date. New information are being added to data sources, which means that the data warehouse has to continually duplicate the data, otherwise it will go
out-of-date. This situation makes the cost of data duplication, transformation and maintenance very high. In recent years, active data warehousing techniques are introduced to enable the data warehouse to be refreshed as frequently as possible and to react to data updates in a new real-time manner. However, it is still a centralised approach and not suitable yet for unstructured or semi-structured data.

These traditional integration methods are widely used but can not satisfy the integration requirements of bio data which are highly distributed, heterogeneous and frequently updated. Mashup provides a light-weight technology for dynamic multi-source integration. Google Maps (Google 2005) is a typical online mashup service which mashes up data from several data providers such as maps from Tele Atlas (Atlas 2008) and photos from Panoramio (Panoramio 2007). Moreover, Google Maps provides services as standard APIs that are accessible and reusable by other Internet applications. In particular, Belleau et al. develop a mashup system to integrate bioinformatics knowledge by normalising them to RDF documents from many sources (Belleau et al. 2008). In total, Altogether 140 GB XML and text data are converted into 46 million RDF documents in the Bio2RDF repository. The Bio2RDF project is also a successful illustration of semantic-aided mashup, which creates a knowledge space of RDF documents linked together with normalised URIs and common ontologies. Asuncion defines the ontology with modeling primitives (Asuncion et al. 2004): Classes, Concepts, Relations, Functions, Axioms and Instances. In general, there are three architectures of the ontology model for integrated services and data (Wache et al. 2001):

- **Single ontology**: Single ontology approaches use one global ontology system that provides a shared vocabulary for the specification of the semantics. All information sources are related to the global ontology.
- **Multiple ontologies**: Multiple ontology approaches integrate each information source with its own ontology. In principle, the ‘source ontology’ can be a combination of several other ontology systems but it can not be assumed that the different ‘source ontologies’ share the same vocabulary. Each source ontology could be developed without respect to other sources or their ontologies.
- **Hybrid approaches**: Similar to multiple ontology approaches, the semantics of each source is described by its own ontology. But in order to make the source ontologies comparable to each other, they are built upon one global shared vocabulary (Goh 1997; Wache et al. 1999). The advantage of a hybrid approach is that new sources can be added easily without modification in the mappings or in the shared vocabulary. It also supports the acquisition and evolution of ontologies.

The OBO Foundry (Smith et al. 2007) is a collaborative experiment carried by developers of science-based ontologies. They establish a set of principles for ontology development with the goal to create a suite of orthogonal interoperable reference ontologies in the biomedical domain. Lots of sub-domain ontologies have been defined. For example, the GO (Gene Ontology) (Ontology 1999) defines the ontology of cellular component, biological process and molecular function. The OBO ontology can be stored in ‘obo’ or ‘obo-xml’ format, and is able to be transformed to other formats easily.
In addition to the data model, a process model is also an important issue for data integration. To support scientific research, data need to be integrated in certain usage context, that is, the scientific process. The process is usually defined as a workflow of tasks which take the data as inputs, perform analysis on the data and produce the outputs. The abstract description of tasks may be bound to different implementations of functional services. Traditionally, the composition is modeled as a static process that is per-defined manually and parsed and executed by a process engine. The ability to flexible service selection and dynamic binding is necessary for process modeling. In recent years, many approaches have been attempted to level-up the intelligence and effect of data-in-process integration. Context-awareness is one of the most effective ways. Research on context-aware applications has gained significant attention since the early 1990s (Lamming and Flynn 1994, Dey and Abowd 1999, Ogata et al. 1999, Szyperski 2000, Held et al. 2002, Ceri et al. 2003, Klyne et al. 2004, Ardissono et al. 2007). Most of current works focus on two aspects. One is context modeling. Many of the existing systems define the context using a set of well-defined uniform models and protocols, such as key-value model, logic-based model, object-oriented model, graphical model, ontology-based model and markup schema model (Strang and Linnhoff-Popien 2004, Vuković 2007). For example, Schilit and his colleagues (Schilit et al. 1999) model situations and environment of the user as key-value pairs and use the pairs in a context-triggered system. N. Ryan (Ryan 1999) develops a simple XML-based Context Markup Language (ConteXtML) for exchanging contextual information, FieldNotes and map data between a mobile client and a server. Another research focus is the application of context models to the service recommendation and selection process with the purpose to improve the intelligence of service composition. Project Aura (Garlan et al. 2002) introduces a new layer between the user layer and the application service layer, called Prism. The Prism layer is designed to observe context changes, manage user tasks and resources adaptation. Han and colleagues (Han et al. 2005) have introduced their workflow language uWDL to specify the context information on the transition constraints of a workflow to support adaptive services. They also implement the tool of uWDL scenario editor and sample.

Driven by the needs of biomedical research, this article proposes a hierarchical mashup framework to integrate data from three layers, and introduces the context and ontology technologies into the mashup activities. The article defines the context model to facility composition activities in each layer, and discusses the mechanism for a context-facilitated interactive and iterative composition process. Besides, the article models the unified ontology for pharmacogenomics experiment data. In addition, the service-based framework provides an open platform for future extension with other open services that are accessible through Internet standard protocols.

7. Conclusion

This research is motivated by biomedical industry needs of integrating multi-dimensional pharmacogenomics data from Web-based databases. This article reports our initial attempts on a novel biomedical data integration framework based on the SOA. Particularly, it addresses three issues and needs in the data-centric service framework: (1) there is a lack of scalable architecture that allows
for rapid growth, frequent changes, and diversified implementations; (2) there is a need of a unified conceptual model that connects the different local definitions; (3) it requires a flexible process that integrates data with functions within various usage scenarios. To address the first issue, the service mashup technology is introduced to support light-weight integration at interface level and to hide the heterogeneous implementations. To deal with the second issue, the ontology system is used with case study in a limited scope in the experiment data integration. To cope with the third issue, service composition and workflow technology is discussed with context modelling capabilities to enhance service recommendation and selections. The preliminary implementation and experiments illustrate the feasibility and potential benefits of the proposed approach. It provides a convenient platform for biology scientists to access heterogeneous data and to organise the analysis workflows, and provides an integrated view of the separated databases distributed over the Internet.

However, many open issues exist in the generic framework. The first one is how to ensure the scalability of the ontology model? It could be quite expensive and error prone to establish and maintain a comprehensive ontology model with full domain knowledge. Next, in addition to the fixed-process and interactive-defined approaches, service may be also discovered and composed at runtime, as new data sources or data analysis functions are published online. How to compose and execute the dynamic process? Since services in a process may depend on each other, the dependent services need to be integrated, invoked or replaced as a whole. The dependence relationship may also relate to the data types and usage context. Hence, how to define the relationship and to ensure their consistent usage? Our future research will further extend current work to address these needs and problems.

Acknowledgement

The project is funded by Basic Research Foundation of School of Information Science and Technology of Tsinghua. The authors first thank their research partner, Dr Quick Que from TGen for all guidance and advises. They also thank Mr Jilong Liu from Xiamen University for his help and collaboration in this research. The authors thank all the people for their supports and encouragement during this research project.

References


