

# Supporting Bioinformatic Experiments with A Service Query Engine

Xuan Zhou Shiping Chen Athman Bouguettaya Kai Xu  
CSIRO ICT Centre, Australia  
{xuan.zhou,shiping.chen,athman.bouguettaya,kai.xu}@csiro.au

## Abstract

*We describe a service-oriented approach to model, design, and implement biological processes and data. In particular, we summarize a novel service query framework to query bioinformatic services. Our approach treats biological data and computational tools as Web services. The query framework is part of an all-encompassing system that manages the end-to-end life-cycle of services, called Web Service Management System. The proposed service query framework allows users to efficiently conduct bioinformatic experiments through service queries. We describe a real life implementation of the proposed framework and our experience in its deployment.*

## 1 Introduction

Advances in bioinformatics technologies have led to a variety of biological datasets and analytical tools. A significant number of them have recently been published on the Web. However, using this wealth of information to perform biological experiment still remain as an open challenge [6]. A typical process of bioinformatic experiment contains several steps. First, datasets and tools needed in the experiments should be identified. Following that, these datasets and tools need to be compared, and the optimal ones are selected to construct an experiment process. Finally, inputs / outputs of the datasets and tools needed to be tuned to ensure smooth conduction of the experiments. In most cases, such a process requires intensive manual work [6], making it inefficient and error-prone.

In the WSMS-GT<sup>1</sup> project, we address this challenge through the Web Service Management System (WSMS) [9]. WSMS is an all-encompassing platform that manages the end-to-end life-cycle of services. It is composed of a set of tools for modeling, querying, optimizing and maintaining services and workflows. In particular, the query engine of

the WSMS is our key solution to facilitate bioinformatic experiments. By encapsulating biological data and analytical tools into Web services, the entire process of bioinformatic experiment, including experiment setup, configuration and execution, can be accomplished by simply performing service queries. This would relieve biologists of the tedious process mentioned above.

Our query engine is fundamentally different from that of UDDI [1], which performs service discovery through categorization and keyword search. In contrast, our approach models a number of domain specific services using a semantic service schema. The schema abstracts heterogeneous services into a tidy set of abstract functionalities, and annotates each functionality with semantics. With the schema, biologists can simply declare the wanted experiments using the abstract functionalities, without knowing the details of concrete services. The query engine will automatically setup and perform the experiments for them. In addition, optimization techniques have been integrated into the query process, to ensure that the experiments are performed using quality datasets and tools.

In this paper, we describe the overall design of the WSMS query engine and its deployment to the WSMS-GT scenario. We summarize our experience in the deploying and using the query engine. We also discuss the advantages and weakness of this current approach and propose directions for future research .

The rest of this paper is organized as follows. Section 2 presents the scenario of the Genome Tracker project and summarizes the basic requirement for bioinformatic experiments. Section 3 gives the architectural design of the WSMS query engine and its deployment in the Genome Tracker scenario. In Section 4, we report our experience and the lessons we have learned. Finally, Section 5 concludes the paper.

## 2 The Scenario

The recent development of the microarray technology [7] have led to rapid increase in the variety of available data and analytical tools. Some recent surveys published in *Nucleic*

---

<sup>1</sup>A collaborative project between the Web Service Management System Group and Genome Tracker Group

*Acids Research* describes 1037 databases [5] and over 1200 tools [3]. The analysis of microarray data commonly requires the biologist to query various online databases and perform a set of analysis using both local and online tools. The objective of the WSMS-GT project is to create a platform to facilitate the analysis of microarray data.

The challenges biologists facing when performing such analysis include:

- Not aware of available databases and tools. Due to the large number of databases and tools, it is difficult for individual or a small group to keep track of all of them. Additionally, new resources appear and old ones disappear, which adds to the difficulty of maintaining a knowledge of them.
- Not able to choose the “right” database and tool. Usually, there are multiple copies of a dataset or analysis tool available online. Often, there is a slight difference among these copies, which is difficult to detect unless the user is an expert in the domain. Also, due to the lack of computing background, a biologist usually can not make choice based on service technology. For instance, the performance of a query involves large amount of data can be significantly improved if a server with fast access is used.
- Manually assembly is time consuming and error prone. It requires a considerable amount of work to “link” all the data and analysis tools together. The large amount of work involved also increases the possibility of human mistakes during the process. Such analysis process is very difficult to maintain if any change is required.

Here, we present an example to illustrate these challenges. In this example, the biologists are studying the genetic cause of colorectal cancer, i.e., identify the genetic variation in human DNA that makes people susceptible to colorectal cancer. By studying the functions of the genes involved, biologists can have a better understanding of the cancer and find possible cure. The first step in this study is to perform experiments on mice, which share more than 90% DNA with human. The experiment involves two groups of mice (10 for each group), the first group—which is called *control group*—was treated with saline and thus does not have cancer, while the second group—the *cancer group*—was inject with azoxymethane, which is known to induce colorectal cancer in mice. The mice in both groups were killed six hours after injection and samples were taken from their colon for *microarray test* [7], which can measure the expression level—how active a gene is—of a large number of genes in one experiment. Microarray experiments are performed on both cancerous and healthy mouse colon tissues. By comparing the results from mice with and

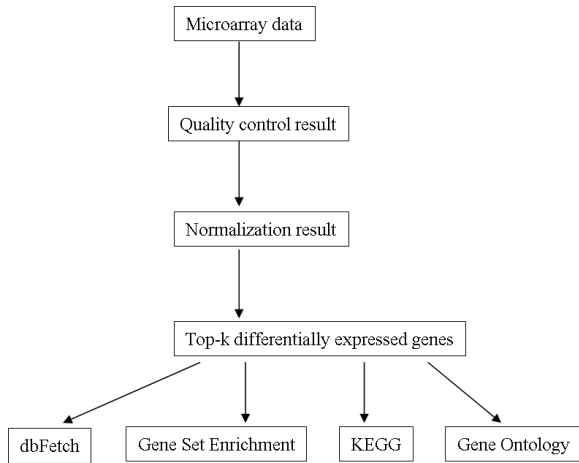
without colorectal cancer, biologists can identify *candidate genes* that may cause the cancer. Further analysis—such as searching for the functions known to these genes—are commonly performed to examine whether and how the candidate genes relate to the colorectal cancer. Specifically, the following analysis are required for the analysis of microarray experiment results:

1. **Quality Control.** This step is designed to identify significant errors in the experiment. The raw microarray result data are visualized and inspected by an expert, who can identify errors such as those caused by contaminated tissue samples. If any anomaly is detected, the results are discarded and no further analysis are performed.
2. **Normalization.** Microarray results from different samples need to be normalized before any meaningful comparison can be conducted. There are many normalization methods available, and they differs in terms of performance and result quality. Additionally, each method commonly requires the tuning of a set of parameters to achieve the best results.
3. **Gene Differentiation.** By contrasting the results from cancerous and healthy tissues, *differentially expressed genes*—genes that are active in cancer but not healthy tissue or vice versa—are identified. Again, there are a large number of statistical methods available for this step and the analysis is computationally expensive.
4. **Gene Study.** The list of most differentially expressed genes are commonly used as candidate genes to understand the biological foundation of the disease being studied. Many resources are available to study these genes, such as the gene sequence, pathway database, and gene function ontology. There are multiple related by different databases for every of these aspects, and a good decision among them requires knowledge of both the domain and the database functionalities.

The analysis process can be captured in a workflow as shown in Figure 1.

A system to support such bioinformatic experiments will require:

- The ability to identify data and analysis tools to perform the tasks specified by a biologist. For example, if a biologist declares that he wants to perform quality control over a set of microarrays, the system should present a service of quality control to him.
- The ability to select the optimal data sources and tools according to some quality and performance criteria. For example, if there are multiple services that can perform normalization of microarray data, the system should select one that is most efficient and reliable.



**Figure 1. The Process of Microarray Analysis**

- The ability to compose the process required to perform user specified task. For example, a biologist might not know that quality control and normalization are prerequisite of the differentiation analysis, and requires to perform the analysis directly on raw microarray data. In this case, the system will automatically compose a workflow which perform arrange quality control, normalization and differentiation analysis sequentially.

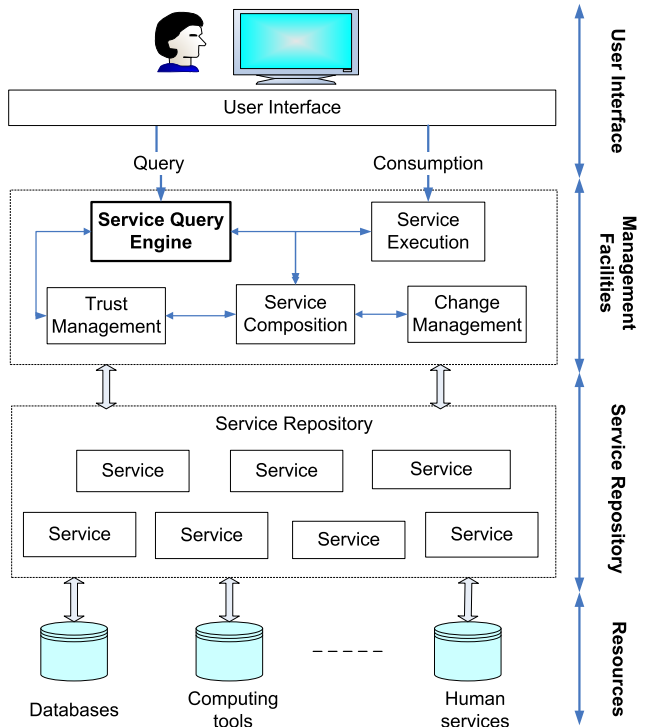
### 3 The Design

This section first gives a brief introduction of WSMS. It proceeds to presents the design of our query engine as well as its deployment in the WSMS-GT scenario.

#### 3.1 The WSMS Architecture

In WSMS, we treat data, tools and all other resources as services, and use a set of facilities to manage their life-cycles, including service query, service composition, service optimization, service execution and service maintenance. Figure 2 provides overview of the WSMS architecture, which consists of a user interface, as service repository and a set of management tools. Their functionalities are summarized as follows:

- **Service Repository** stores the access point of each Web service as well as its semantic description.
- **Service Query engine** receives queries from end-users and/or applications, interpret the queries, and find the best service or workflow that satisfy the functional and non-functional requirements specified in the query.



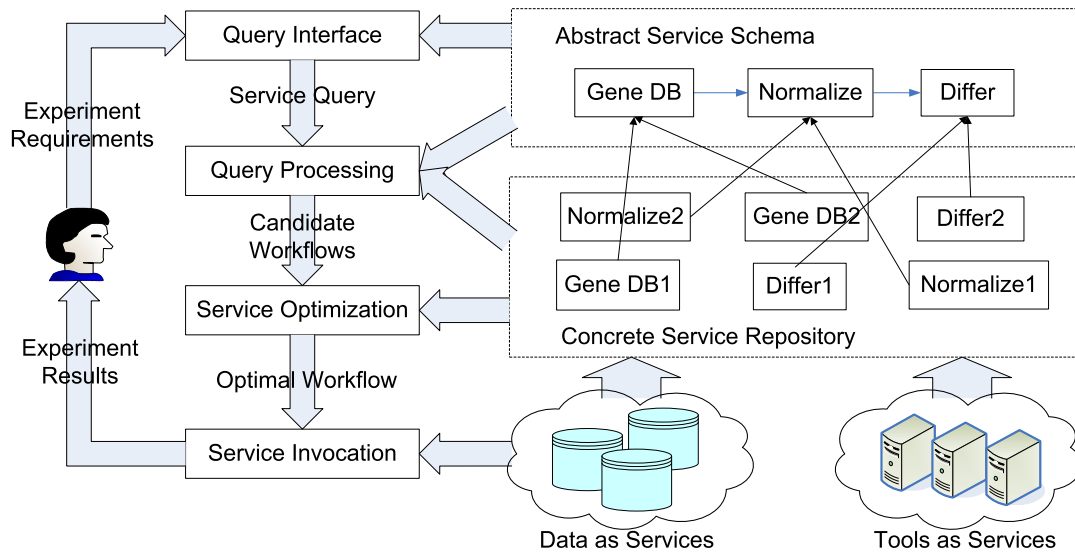
**Figure 2. Overview of the WSMS architecture**

- **Service Composition** is responsible of composing individual services into workflows to handle complex tasks.
- **Service Execution** conducts two jobs: (a) orchestrating workflows; (b) invoking Web services;
- **Trust Management** collects and disseminates trust information (e.g., reputation) of web services.
- **Change Management** maintains Web services and workflows, and adapts them to various environmental changes, such as changes of user requirements and changes of service availability.

The service query engine is the main component for helping users locate and consume services. It is thus the key solution we applied to handle the challenge in the WSMS-GT scenario.

#### 3.2 The Query Engine Architecture

The detailed design of the WSMS query engine in the WSMS-GT deployment is illustrated in Figure 3. In the deployment, all the bioinformatic data and manipulating tools are first encapsulated as Web services and imported to a service repository. Besides the service entries, the repository



**Figure 3. The Architecture of the WSMS Query Engine**

also records the various properties of each services, such as inputs / outputs, performance, reputations and etc. Following that, an abstract service schema is designed jointly by computer scientists and biologists to model the services in the repository. The schema contains a set of abstract services, each representing a typical functionality of the available services. For instance, there can be a number of services providing gene data and a number of services for performing gene differentiation. Thus, in the service schema, a single abstract service will be defined to represent all the gene data services, and another abstract service will be defined to present all the gene-differentiation services. Finally, mappings are established between the concrete services in the repository and the abstract services in the service schema.

When conducting bio-informatic experiments, a biologist operates on a query interface to declare the the experiment he wants to conduct. The result of this process is a service query described using the elements of the service schema. The service query is then passed to the query processor, which looks into the service repository to generate a set of candidate workflows for performing the experiment. While these workflows are composed of different concrete services, they possess the same functionality required by the biologist. These candidate workflows are then processed by the service optimizer, which selects the workflow with the best quality. Finally, the experiment is conducted by executing the selected workflow, and results are returned to the biologist. The whole process is fully automatic. A biologist only needs to specify his requirements through the query interface. The query engine will automatically select and perform the experiment for him.

In the following sections, we describe the detailed design of the service schema, service repository and the techniques for generating and optimizing services or workflows.

### 3.3 Service Schema & Service Description

The service schema we used in Genome Tracker is adapted from the one proposed in [8]. In a nutshell, a service schema is a directed acyclic graph (DAG), where each node represents an abstract service (or abstract service operation) and each edge represents a dependency between two abstract services. In the case of bioinformatic services, the dependency is the prerequisite relationship between services. For instance, Figure 3 shows a partial service schema composed of three abstract services, i.e., *gene database*, *normalization* and *gene differentiation*. The *gene database* is the prerequisite of the *normalization* and the *normalization* is the prerequisite of the *tgene differentiation*. With such dependencies, a user only needs to specify the main task he wants to accomplish, and the query engine will discover the prerequisite process and construct a workflow automatically. In addition, each abstract service is annotated with semantic data which describe its functionality, input, output and other information that can be used in query.

The concrete services in the service repository are described using the relational model proposed in [8]. This model describes each concrete service using a set of attributes, such as its availability, reputation and response time. The values of these attributes are then stored in a relational table. When conducting query processing and service optimization, efficient algorithm can be devised by exploiting the relational tables.

### 3.4 Query Interface

The query interface allow users to specify what they want (goals) at the level of service schema, and let the WSMS query engine to figure out how to achieve the goals. On the one hand, since WSMS is designed to be used for both applications and end-users, its interface is required to be simple for ease of use. On the other hand, the query interface should be rich and powerful to express various concepts and their relations. Based on these requirements, we develop a simple service query language in XML, called CSQL (CSIRO Service Query Language), as an interface to WSMS query engine, whose schema is shown in Figure 4:

As we can see, CSQL emulates database SQL in attempt to express any queries in a structured sentence, i.e. *I want to do something (select, insert, update, delete ...) somewhere (from, into...) if some conditions meets (where...)*. However, the scope and notations that CSQL aims to express would be much wider and richer than database SQL. Therefore, we extend SQL by allowing any action and any task. By composing the two variables (Action and Task) with support from the domain ontology, CSQL is capable of expressing goals that the users want to achieve across a large range of domain, e.g. “find the cheapest fare”; “book tickets”; “sell a car”; “do Gene Pathway Analysis”. In the scenario of WSMS-GT, the Action can refer to either “find” or “execute”, and the Task always refer to an abstract service.

In addition, CSQL must be able to provide enough information as inputs to invoke these web services. To provide such a uniform schema to host different amounts and types of input data, we allow each task to have a set of inputs elements. The input elements has a very generic data structure (DataType), consisting of three attributes (*name, type and domain*) and a single element (*Value*). The key reason why we do not define the value as an attribute is that the value may carry a large amount of data (such as a document and/or an image) for some applications, which would be improper and/or impossible to put it as an attribute. While directly using the pair of name and value meets a large range of requirements for web services input, in theory, CSQL is capable of specifying any more complex inputs. This can be achieved by binding a XML schema to the type attribute and serialise the input as an XML string.

In addition to the capability of expressing goals and a variety of data inputs, CSQL is also able to specify non-functional requirement using standard SQL prediction clauses. For example, we can express: *To find the best neighbours for GeneID=eco:b0002 if the service's reputation > 90%* in CSQL, as shown in Table 1.

The current design and implementation support only one logic operation between predictors, i.e. “AND”. We are planning to support more logic operations and functions with support from domain ontology. While CSQL can

```

<CSQL>
  <Action>Find</Action>
  <Task name="BestNeighbors">
    <Input name="GeneID" type="string"
      domain="Biology" >
      <Value>eco:b0002</Value>
    </Input>
  </Task>
  <Where>
    <Predictor name="reputation" relation=">="
      value="90%"/>
  </Where>
</CSQL>

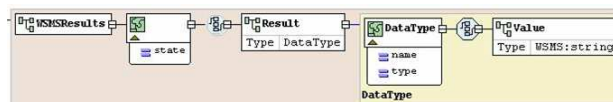
```

**Table 1. An example of CSQL query**

be presented in different forms (webpage, web services, JDBC-like driver, etc.) as an interface to WSMS Query engine, the current implementation is a web service, which provides a single operation as follows:

```
String WSMSResults = query( String csql )
```

where WSMSResults is a string in XML, whose schema is defined as follows:



Let us reuse the above the query example. Assume the selected service be KEGG (Kyoto Encyclopaedia of Genes and Genomes) service, the execution results by invoking the KEGG service are listed in Table 2:

As we can see, CSQL is a simple service query language with simple syntax and structure. Therefore, it can be used either directly by end-users or indirectly via applications. Since CSQL is domain-independent, it should be easy to apply CSQL in various domains with support of the domain ontology.

### 3.5 Query Processing

When receiving a service query, the query engine first converts it to an algebraic expression. In [8], the authors defined three basic algebraic operators. To handle bioinformatic services, we used two of the operators, that is, *F-map* and *Q-select*. *F-map* utilizes the dependence relationships defined in the service schema to retrieve all dependent abstract services of the services specified in the query. It then

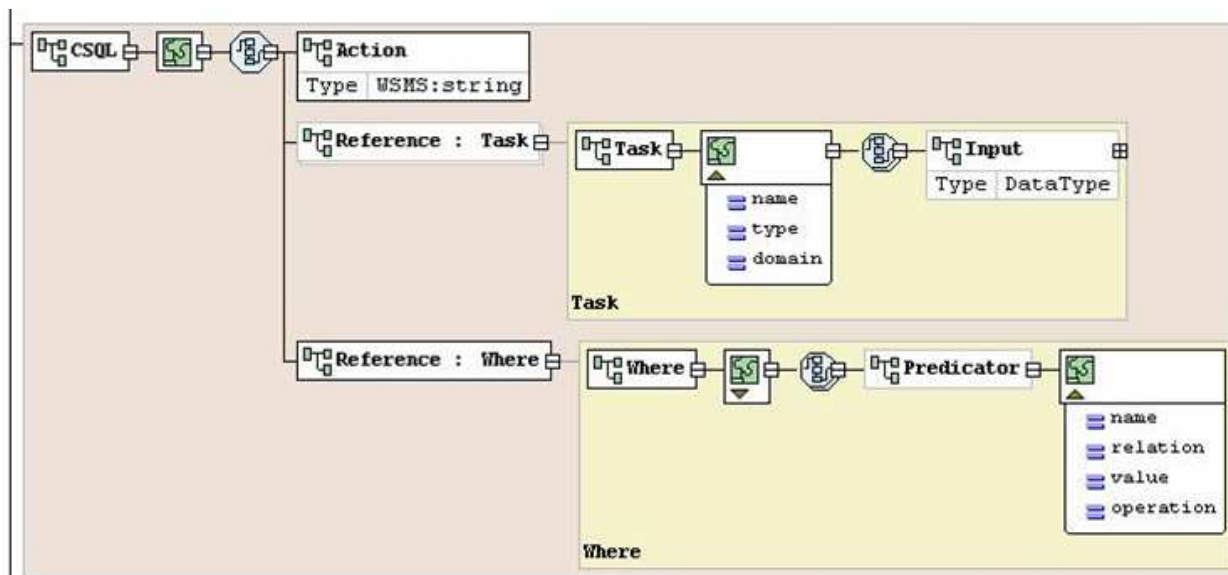


Figure 4. XML Schema for CSIRO Service Query Language (CSQL)

```

<WSMSResults state="OK">
  <Result name="BestNeighbor" type="string">eco:b0002;ecg:E2348_C.0002;5283</Result>
  <Result name="BestNeighbor" type="string">eco:b0002;ecg:E2348_C.0002;5283</Result>
  <Result name="BestNeighbor" type="string">eco:b0002;ecg:E2348_C.0002;5283</Result>
  .....
</WSMSResults>

```

Table 2. An example of query results in XML

generates an abstract workflow out of the retrieved abstract services. *Q-select* applies the comparison criteria in the service query and retrieve a set of candidate services from the service repository to instantiate the abstract workflow. After these two types of operators in service query are all finished, a set of candidate workflows are generated.

### 3.6 Service Optimization

Service optimization is responsible of finding the best workflows in terms of QoWS among those generated by query processing. As there are a number of QoWS attributes for each service or workflow, we need to aggregate their values into a single utility score to represent the overall QoWS of a service or workflow. In our deployment, we use a utility function defined by an experienced biologist, so that the computed utility is sensible to the scenario of bioinformatic experiments. With this utility function, we apply the DP-DAC algorithm in [8] to pick the optimal workflow with the maximum utility. This workflow is then executed to conduct the experiment required by the user.

## 4 Discussion

The implementation and deployment of the WSMS query engine has given us a number of insights of its performance and usability. We summarize these experiences as follows.

- Simplification of bioinformatic experiments: Using the WSMS query engine, a biologist can significantly reduce his workload in performing bioinformatic experiments. This benefit mainly comes from the abstraction of the service schema. First, the schema maps databases or tools with similar functionalities to a single abstract service. This hides a significant amount of heterogeneity from end users, i.e., the biologists. As a result, instead of working with a large number of concrete databases and tools, a biologist only needs to operate on a small number of abstract services. Second, the concrete services of each abstract service are annotated with a uniform set of attributes, which enables automatical selection of services that best satisfies user

requirements. Lastly, the dependencies among services allow biologists to focus on outcomes of experiments and ignore most preparation steps. This further reduces their workload.

- **Design complexity:** To achieve the simplification described above, a proper design of service schema and service repository is essential. Design was proven to be the most time consuming work in the deployment of our query engine. First, the design of a service schema is not easy. On the one hand, the schema should capture all functionalities of the available services. On the other hand, the schema needs to be as simple and natural as possible for biologists to use. During the deployment, the Genome Tracker service schema was revised repeatedly. Even after the deployment, changes were made frequently. Second, when mapping a set of heterogeneous concrete services to an abstract service, their input and output parameters need to be unified and transformed to that of the abstract service. Wrappers have to be created manually for this transformation.
- **Maintenance complexity:** We also observed that the service schema and the service repository can keep changing after the deployment. When new data types and analytical tools are discovered, new abstract services or dependencies need to be defined and added to the service schema. This sometimes may incur major revision of the entire schema. When new features of services are found to be useful in service optimization, and all service descriptions in the repository need to be modified to incorporate these features. Without proper tools, such changes may incur a lot of work.

Based on our experience, we believe that the following issues need to be addressed to further improve the usability of the WSMS query engine.

- A guided process for designing service schema and service repository. Analogous to database design, a number of principles need to be discovered to guide the design process, so as to enable fast and accurate creation of service schema and service repository.
- Tools for mapping concrete services to abstract services. This requires a language for users to easily define the mappings as well as techniques for automatic wrapper generation.
- Tools for changing and manipulating existing service schema and service repository. These tools should be easy to use. They should be able to populate changes and handle inconsistencies automatically.

## 5 Conclusion

In this paper, we presented our deployment of the WSMS query engine to support bioinformatic experiments in the scenario of the WSMS-GT project. We showed that service query engine is a novel and promising approach to reducing biologists' workload in bioinformatic experiments. We summarized our experience in the deployment, and proposed a number of future research directions.

## References

- [1] Uddi specification. 2009. <http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm>.
- [2] I. M. Author. Some related article I wrote. *Some Fine Journal*, 99(7):1–100, January 1999.
- [3] M. D. Brazas, J. A. Fox, T. Brown, S. McMillan, and B. F. F. Ouellette. Keeping pace with the data: 2008 update on the bioinformatics links directory. *Nucleic acids research*, 36, July 2008.
- [4] A. N. Expert. *A Book He Wrote*. His Publisher, Erewhon, NC, 1999.
- [5] M. Y. Galperin. The molecular biology database collection: 2008 update. *Nucleic Acids Research*, pages gkm1037+, November 2007.
- [6] M. Gertz and B. Ludaescher. Scientific data and workflow management tutorial. In *10th Intl. Conf. on Extending Database Technology (EDBT)*, 2006.
- [7] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, October 1995.
- [8] Q. Yu and A. Bouguettaya. Framework for web service query algebra and optimization. *ACM Trans. Web*, 2(1):1–35, 2008.
- [9] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. Deploying and managing web services: issues, solutions, and directions. *VLDB J.*, 17(3):537–572, 2008.