Service Oriented Computing and Applications manuscript No.
(will be inserted by the editor)

Web Service Management System for Bioinformatics Research: A Case Study

Kai Xu · Qi Yu · Qing Liu · Ji Zhang · Athman Bouguettaya

Received: date / Accepted: date

Abstract In this paper, we present a case study of the design and development of a Web Service management system for bioinformatics research. The described system is a prototype that provides a complete solution to manage the entire life cycle of Web services in bioinformatics domain, which include semantic service description, service discovery, service selection, service composition, service execution, and service result presentation. A challenging issue we encountered is to provide the system capability to assist users to select the "right" service based on not only functionality but also properties such as reliability, performance, and analysis quality. As a solution, we used both bioinformatics and service ontology to provide these two types of service descriptions. A service selection algorithm based on skyline query algorithm is proposed to provide users with a short list of candidates of the "best" service. The evaluation results demonstrate the efficiency and scalability of the service selection algorithm. Finally, the important lessons we learned are summarized and remaining challenging issues are discussed as possible future research directions.

Keywords Web Service · Bioinformatics · Ontology · Workflow

Kai Xu was at CSIRO, Australia and is currently at the Middlesex University, UK.
E-mail: k.xu@mdx.ac.uk
Qi Yu is at the Rochester Institute of Technology, USA.
E-mail: qi.yu@rit.edu
Qing Liu and Athman Bouguettaya are at CSIRO, Australia.
E-mail: {Q.Liu, Athman.Bouguettaya}@csrio.au
Ji Zhang was at CSIRO, Australia and is currently at the University of Southern Queensland, Australia.
E-mail: ji.zhang@usq.edu.au

1 Introduction

The recent advancements of molecular biology experimental instruments, such as microarray technology [32], have led to rapid increase in the size and variety of available genomic data. Analysis of these data using computational methods—the so called *in silico* experiments—is becoming an integral part of modern biological studies. According to a recent survey, there are 1078 biological databases [17] and over 1200 bioinformatics tools [12] publicly available online.

Many of these online resources provide Web Service interface, which allows easy access and integration of a number of services. Significant progress has been made towards building integration platforms that utilize these resources to support bioinformatics analyses. A common feature of these platforms is providing searching facilities to help users identify required data and analysis services, and then compose them into workflows to perform complex analysis. *Taverna* [26] is such a platfrom widely used in the bioinformatics community, whereas *Kepler* [2] and Triana [24] are two popular platforms in the wider scientific community.

Recently there is a trend to extend such platforms to support semantic Web Services, as more semantically linked data repositories become available (with the Linked Data [8] being a prominent example). Semantic annotation provides richer information than Web Service description alone and can be used for automatic reasoning when they conform to certain ontology. The semantic support can be added through plug-in (such as the Feta plug-in [23] for Taverna) or expanding the existing system (such as the semantic "Tagging" function in Kepler [1]). Considerable work has also been done to develop bioinformatics-related ontology, which provides a language of describing bioinformatics services.

Examples include the ontology from *myGrid* [43] and *BioMoby* [42]. Finally, there are Web Service registries for listing available bioinformatics services. Such registries usually provide semantic description and thus more powerful searching functionality. Registries like Moby Central [42] and the more recent BioCatalogue [7] also monitor service availability and automatically remove unresponsive ones.

A biological research group within our organization is working on the genetic causes of colorectal cancer, and they conduct complex analysis procedures as described before on a daily basis. Currently, such analysis is done manually, which is time consuming and error prone. The existing systems for bioinformatics research are mainly designed for computer literate users. Our collaborators find it overwhelming given the complexity of the user interface and large number of resources listed. We propose to adopt a different development strategy as the existing systems. Instead of building a large and complex system that can accommodate a wide range of bioinformatics research requirements, our system is designed specifically for colorectal cancer research at this stage. The flexibility of our system architecture allows it to easily adapt to other disease studies later on if required. While there are some efforts in the existing systems to help users to find the "right" services (such as adding semantic annotation), they are still in very early stage and can not meet the requirements of our users. Also, we believe it is essential to support semantic Web Service at a system level to fully realize its potential. In other words, all the core system components should provide native support for semantic Web Service, which is not how the existing systems are designed.

This paper focuses on the design and development of a service management system prototype using bioinformatics as a target application. In this respect, we used experiments used in gene identification of colorectal cancer as a specific application. The goal of this work is not to provide ready-to-use tool but rather a proof-of-concept for the approach we are proposing. The proposed Bioinformatics Service Management System (BSMS) provides a complete system solution to manage the entire life cycle of Web services in bioinformatics domain, which include semantic service description, service discovery, service selection, service composition, service execution, and service result presentation. Therefore, the objective of this work is not just to present yet another bioinformatics platform. Instead, we aim to use BSMS as a proof of concept to demonstrate how the life cycle of bioinformatics Web services can be managed in a systematic manner. Several key design rationales of BSMS are summarized as follows:

– BSMS has an *all-service architecture*, i.e., all system components are services. This allows easy change of individual components and addition of new ones to adapt to other disease studies if required. Although the current system only has the components required for colorectal cancer research, it can be easily extended.
– The whole system is based on a Semantic Web Service framework (WSMO [30]). As a result, all system components provide native semantic support. Both bioinformatics and service ontology are included to provide functionality based service discovery.
– Considering the potentially large number of services, there may be functionality overlaps between different service providers. We believe that the non-functional properties of services, especially those related to Quality of Service (QoS), such as *reliability*, *performance*, and *analysis quality* should be considered when several services providing similar function or information. A service selection algorithm based on skyline query is proposed to help users identify the "best" service. Evaluation confirms the efficiency and scalability of our algorithm.
– An interactive workflow environment is incorporated to provide better usability for biologists. Only essential services are visible and automatically service filtering and selection are done wherever possible to avoid overwhelming users with large number of choices.

What reported here is our attempt to address these challenging issues (especially the third one), and by no means we have solved them. It is our hope that by sharing the experience others can learn from our lessons and become aware of the research problems that require further attention.

2 Related Work

2.1 Bioinformatics Web Services

There are a large number of Web Service-based workflow environments have been built to support scientific research [31,40]. Among them, *BioMoby* and *myGrid* are two of the most widely used bioinformatics platforms. BioMoby provides a registry of bioinformatics services and searching functionality. It started with what now is known as *Moby-Services* [42] project that realizes a subset of the functions specified in the Web Service standards. A later branched project called *Semantic Moby* [33] adopts the REST architectural style [16] and makes extensive use of Semantic Web technologies. BioMoby defines three ontologies: *Namespace*

Ontology, *Object Ontology*, and *Analysis Ontology*. The Namespace Ontology provides a list of abbreviations for the different types of identifiers that are used in bioinformatics. The Object Ontology defines bioinformatics data formats and the relationships between them. The Analysis Ontology provides a description of various bioinformatics analyses. The BioMoby service registry is called *MOBY Central*. All the registered services are described using Namespace and Object Ontology in Moby-Services, and additionally Analysis Ontology in Semantic-Moby. The annotations are used in service query to match services with the data users have (Moby-Services) and identify the service based on required functionality (Semantic-Moby). The focus of the Moby-services is to facility data exchange using the name convention and data format mapping defined in the Namespace and Object Ontology respectively. The Semantic-Moby adds the Analysis Ontology to describe the service functions. However, neither of these addresses the non-functional properties such as reliability, performance, and analysis quality.

The myGrid project is part of the UK government's e-Science programme [37]. Among a wide range of subprojects, the one of particular relevance is Taverna [26], which is a workflow construction environment and execution engine designed to support *in silico* biological study. It provides access to a large collection of data sources and analysis tools, many of which are accessed through Web Service interface. myGrid has its own ontology [43], which contains both *Domain Ontology* and *Service Ontology*. The Domain Ontology acts as an annotation vocabulary including descriptions of core bioinformatics data types and their relationships to one another, and the service ontology describes the physical and operational features of web services, such as, inputs and outputs. The aim of the myGrid ontology is to support service discovery. Users can perform semantic query in Taverna using the *Feta* plug-in [23] when searching for a service. Taverna is designed as a do-it-all environment, which can be overwhelming for biologists with limited computing background. While Taverna has a plug-in architecture that allows addition of new functions, any changes to the core system components are not trivial. Finally, semantic support is provided through plug-ins, not at the system level.

Another system worth mentioning is *Kepler* [2], which is designed for generic scientific workflows. Similar to myGrid, Kepler is a comprehensive environment that provides all the system layers from the back end computing infrastructure to the front end workflow compose bench. Kepler also provides certain semantic functionality, such as checking of semantic compatibility of two connected services and searching for semantically compatible services [6]. Other interesting features include automatic data structure transformation using semantic annotation between semantically compatible services in certain cases [10,11]. Kepler shares some of the issues of myGrid such usability, complexity, and semantic support.

2.2 Service Selection and Optimization

There is considerable work available on service selection and optimization [41,44]. In [36], an optimization algorithm is proposed to efficiently access Web Services. The optimization algorithm takes as input the classical database SPJ like queries over Web Services. It uses a cost model to arrange Web Services in a query and computes a pipelined execution plan with minimum total running time of the query. Quality-aware service optimization techniques have been studied in [45,47,48, 15,27]. These approaches rely on the computation of a predefined objective function and the users need to assign numeric weight to specify their preferences if multiple quality parameters are involved. This is a rather demanding task and an imprecise specification of the weights could miss user desired services. We propose a skyline computation approach to tackle the service selection issue. The skyline approach goes beyond the existing service optimization approaches by automatically selecting a set of best services. Skyline or similar concepts have been applied in the area of service computing. A service discovery framework was developed in [34] that integrates the similarity matching scores of multiple service operation parameters obtained from various matchmaking algorithms. The framework relies on the service dominance relationships to determine the relevance between services and users' requests. Instead of using a weighting mechanism, the dominance relationship adopts a skyline-like strategy that simultaneously considers the matching scores of all the parameters for ranking the relevant services. A concept, called p-dominant skyline, was proposed in [46] that integrates the inherent uncertainty of QoWS in the service selection process. A p-R-tree indexing structure and a dual-pruning scheme were also developed to efficiently compute the p-dominant skyline.

3 Scenario

Our system is designed to support the study of genetic cause of colorectal cancer, i.e., identify the genetic variation in human DNA that makes people susceptible to colorectal cancer. Identifying the related genes and studying their functions can lead to early detection and new treatment. The study is currently at mouse trial stage, i.e., using mouse as a disease model to study the cancer, as mice share more than 90% DNA with human. Once the related genes are identified and their functions are better understood, the study will move onto the human trial stage.

At the current stage, one of the critical tasks is to identify the genes related to the cancer. This is usually achieved by comparing the DNA of health mice (*control group*) with that of mice with cancer (*cancer group*) with *microarray* [32], which can measure the *expression level*—how active a gene is—of tens of thousands of genes in mouse DNA. By contrasting the results from the control and cancer groups, biologists can identify *candidate genes* through statistical analysis. In many cases, large portion of the candidate genes are not the ones that cause the cancer; they can be experiment noise, the artefact of the statistical method, or the product of the cancer. Further analyses are commonly performed to carefully examine each candidate gene to identify the cancer causing ones. Such analyses include searching for the functions known to these genes and the metabolic pathways these genes are involved in.

As discussed earlier, there are mainly two types of analyses involved in our study: *statistical analysis* and *function analysis*. There are a wide range of statistical analyses involved in our study, and we use the following ones as examples to illustrate our system:

- *Quality Control,* which is designed to identify significant errors in the experiment, such as those caused by contaminated tissue samples. If any anomaly is detected, the results are discarded and no further analyses are performed.
- *Normalization.* Microarray results from different mice need to be normalized before comparisons can be made. There are many normalization methods available, and they require parameter tuning to achieve the best results.
- *Differential Expression* is used to identify candidate genes by contrasting the results from control and cancer group. Again, there are a number of statistical methods available for this analysis.

There are vast amount of online databases and tools available for function Analysis. We include the following to illustrate our system:

- Searching for known gene functions and locations in the Gene Ontology (GO) [4] database.
- Identify the group of genes act in concert (being active or inactive together) using Gene-Set Enrichment analysis [38].

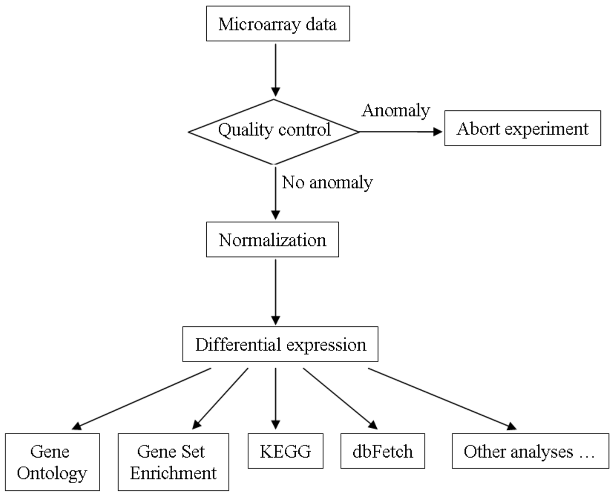


Fig. 1 Example: workflow of genetic study of colorectal cancer.

- Searching for the biochemical pathways that genes are involved in from the Kyoto Encyclopedia of Genes and Genomes (KEGG) [20] database.
- Services that aggregate analysis results from several other sources, such as the *dbFetch* [22] that searches several online databases for related information.

While there are common routines in statistical analysis, function analysis is often more of exploratory nature: biologists will try different analyses they have access to, and decide what step to take next based on the results. Quite often, they go back a few steps in the analysis chain and change the settings there to see how it affects the following analyses. Fig. 1 shows an actual workflow to be used as an example for discussion. It consists of two stages: the statistical analysis (from "Microarray data" to "Differential expression") and the functional analysis (the bottom layer).

4 Requirements

Based on consultation with the biologists, we summarize the requirements of the system as follows:

R1. The system should provide easy access to database and analysis services required for colorectal cancer research. Users find it difficult to identify the right service when the interface shows all available services (can be thousands when including irrelevant ones), even when they are organized into categories.

R2. When there exist multiple services providing similar information or function, the system should provide assistance to identify the "best" service based on user-defined criteria such as reliability, performance, analysis quality, or combination of them.

R3. The execution of database query and analysis should be done through the Graphical User Interface (GUI), and the results are captured and displayed automatically without user intervention.
R4. For complex analyses requires multiple services, users prefer specifying them as workflows through GUI.
R5. The analyses required for colorectal cancer study should complete within acceptable time.
R6. The GUI should provide easy access to all system functionalities to two types of users: biologists with basic computing knowledge and bioinformaticians who are experts in statistical analysis.
R7. The system should allow easy change of existing services and addition of new ones.
R8. While the study currently focuses on colorectal cancer, it is likely that this may change in the future. The system should be flexible enough to allow easy adaption to new disease studies.

Many of these requirements are similar to those of existing bioinformatics workflow environment, but with specifics to our use case. For instance, the system needs to orient towards biologists with limited computing knowledge. Among the requirements, we found R2 particularly challenging, because it requires the capability to select service based on non-functional properties, which is still not a well studied research problem. The users also expressed interested in provenance related functions such as storing all the information required to reproduce the instance of a workflow execution. However, after discussion we decided not to include such functions in this development iteration because it is currently not critical to the colorectal cancer study and many provenance related research issues are still unsolved.

5 System Design and Architecture

Based on the user requirements, we made the following system design decisions:

1. **Service Oriented Architecture (SOA).** With large number of data repositories and online tools providing Web Service interface and more being added every day, it seems a natural choice to adopt a service-oriented architecture, which avoids the complexity in ad hoc integration and provides a uniform way to access various resources: database or tools, online or local. This also provides the capability to automate database query and analysis execution (R3) and facilitates the addition and change of resources (R7).
2. **System components as services.** Given the project scope, we are only required to develop a system that can support the colorectal cancer study (R1). Therefore it is not our aim to develop a comprehensive system such as Taverna. However, the requirement to adapt to other disease study in the future (R8) requires a flexible system architecture. As a result, we decided to implement all system components as Web Services as well, so they can be easily updated or replaced for other disease studies.
3. **Semantic description of non-functional service properties.** The requirement to assist users selecting services based on non-functional properties (R2) entails the collection and storage of such information. We chose to store it as semantic description. This allows reasoning with such information, which is not possible otherwise and is important to service selection and optimization.
4. **Bioinformatics and service ontology.** The need to provide semantic description of non-function properties requires a formal service ontology. We also decided to provide semantic description of service function, which needs a bioinformatics ontology to describe them. An example ontology is given in Figure 2. Many concepts are not shown to keep the total number at a manageable level. The ontology captures a set of important concepts that are used to describe both the functional and non-functional properties of services. The functional concepts include precondition, postcondition, and input/output parameter of a service. The non-functional concepts include reliability, performance, and analysis quality. The ontology can be easily extended to include other important concepts. This is important for helping users to find required services through function search (R1), and it also helps compose the services together into a workflow (R4). As a result, all the services—including system components—have both descriptions.
5. **Native semantic support.** This decision is based on the fact our system heavily relies on the semantic description provided by the bioinformatics and service ontology. Besides, all the system components are semantic services, too. Therefore, we decided to base the system upon a semantic service framework (provides native support), rather than realizing through plug-in.
6. **Interface usability.** One of the main complaints we got about the existing systems is that users find them generally difficult to use. Part of the reason is that they are not designed for biologists. Therefore, we decided to give interface usability a very high priority, even it means reduction in functionality in some cases. For instance, we decided to only show services that we know are relevant (R1), which may

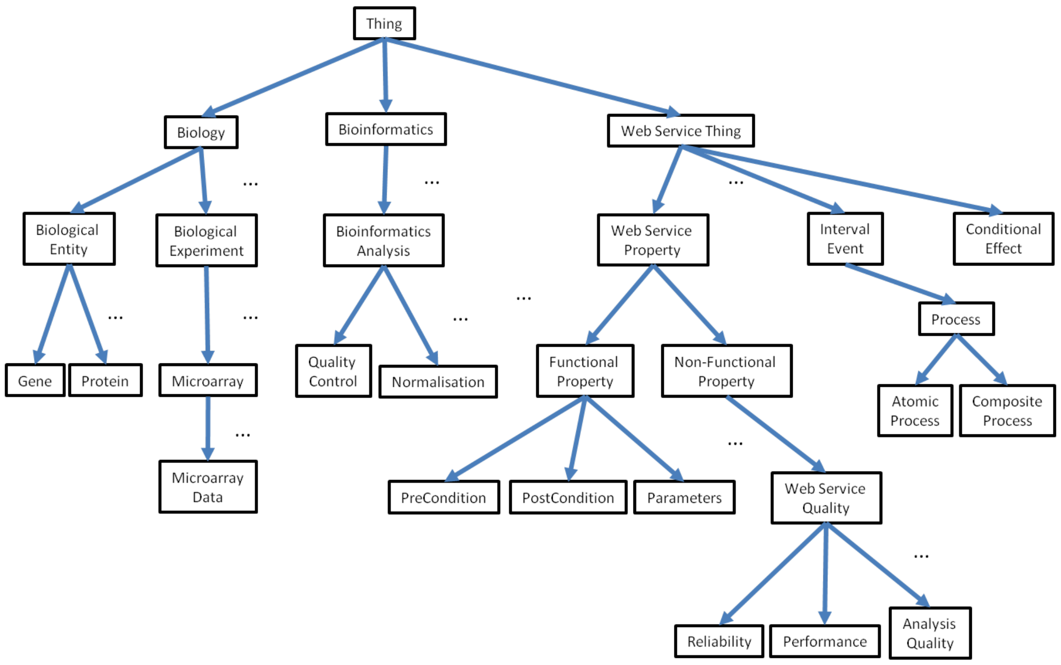


Fig. 2 An Example Ontology

exclude some potentially useful services. As part of the usability efforts, the system has an interactive workflow construction environment (R4) and all system functions can be easily accessed through GUI (R6).

The system architecture is based on these design decisions and has the following four layers (Fig. 3). From bottom up:

– **Web Services.** This layer contains all the data sources and analysis tools, which are all exposed as Web Services. The data sources include both local ones (such as the microarray experiment results described in Section 3) and remote ones (such as the Gene Ontology and KEGG pathway database). The analysis tools include various statistical methods that can be performed locally or remotely. Note that this is not a exhaustive list of all the databases, and the same apply to the analysis tools.
– **Ontologies.** In this layer, semantic description is added to the Web Services. The domain ontology provides a biological description of the services, while the service ontology provides the property information among others. The details of both ontologies are discussed in the next section.
– **Service Manager.** This layer consists of the core components for service management. The service selection is achieved through two system modules: "Service Discovery" and "Service Optimizer". The former retrieves available services according to their biological function (using domain ontology description) or service properties (using service ontology description). The latter optimizes the workflow execution by selecting the "best" for a given criterion (such as reliability) and balancing among multiple criteria according to user preference. The "Ser-

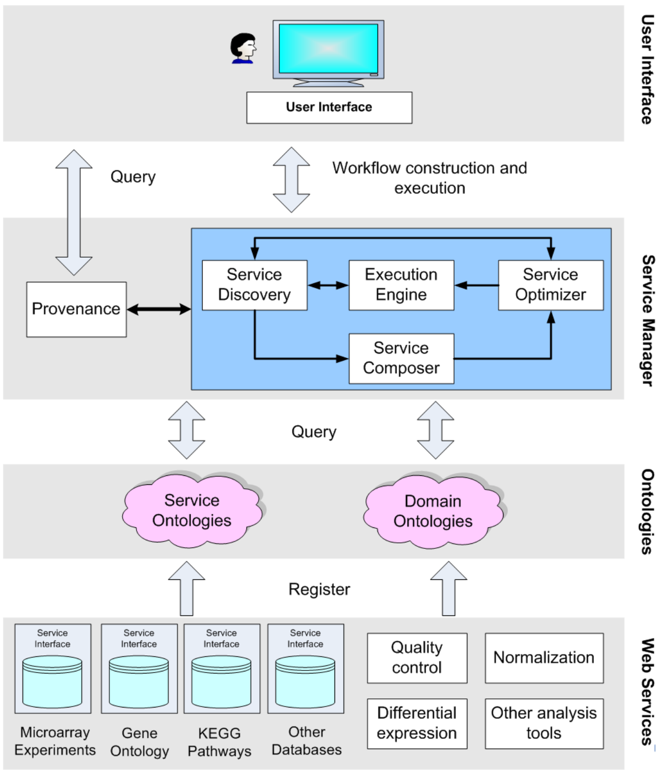


Fig. 3 System architecture

vice Composer" combines services with simple functions to perform complex tasks, and the optimized workflow is invoked by the "Execution Engine" before the results are returned to user. The "Provenance" component is currently mainly served as a place holder. While we are aware its importance and planed for further research, its full functionality is not included in the current version.

– **User Interface.** This layer is where users interact with the system. It is responsible for input gathering and result presentation. The interface provides graphical access to all the available services and system functions including an interactive environment for workflow construction. The details of the interface are covered in Section 6.4.

6 Implementation

In this section, we discuss, layer by layer, the methods and their implementation details in our system, again in a bottom-up order.

6.1 Web Service Layer

In this layer, the colorectal cancer microarray data are stored in a PostgresSQL database. There are a few experiments, each produces multiple microarray data files. All the information relates to one experiment (both the microarray data files and experiment metadata) is exposed as one service. Currently we are in the process of converting them to the microarray data standard MIAME [13], which represents a quite complex format. External databases, such as Gene Ontology and KEGG, are accessed through their Web Service interface. The statistical analyses are realized using the R language [29] and the BioConductor [18] packages. The statistical methods are invoked in Java using `Rserve`, which communicates between Java and R. `Axis2` is used to provide a Web Service interface for the Java code.

6.2 Ontology Layer and Service Property Modelling

Regarding domain ontology, we compared BioMoby ontology [42,33], myGrid ontology [43], and the collection of ontologies included in the Open Biomedical Ontologies (OBO) Foundry [35]. We found that the ontologies in OBO have a strong focus on the sub-fields within biological and medical research, which is different from the bioinformatics focus of our system. BioMoby and myGrid ontology are more bioinformatics oriented, but both are still evolving and neither is adopted as the standard or widely used within the research community. We see the myGrid ontology as the most promising one, but it can be too big and complex for our purpose. Therefore, we decided to use myGrid ontology as a reference and build our own domain ontology only covering the data and analysis required. Our bioinformatics ontology is essentially a small subset of the myGrid ontology, which should allow easy migration to the full myGrid ontology if we decide to do so. Every service available in our system is registered with the bioinformatics ontology term(s) that describe its function. Searching for service with specific function becomes identifying that function term in the ontology and returning all the services registered with it and its descendant terms.

For service ontology, we considered both OWL-S [25] and WSMO [30]. While both are capable of describing the service properties our system requires, we chose WSMO because its model of describing non-functional properties matches better with our requirements. Additionally, the availability of an integrated development environment WSMX [19] facilitates system implementation. As a result, the ontology is written in the WSML language [14], conforming to the WSMO model.

While there are many non-functional properties can be used for service description, we decided to focus on *reliability*, *performance*, and *analysis quality* after consultation with the users:

– The *reliability* measures the availability and stability of Web Services. In the bioinformatics context, this can be quantified as the percentage of the up time of the data or analysis services.
– The *performance* measures the time a Web Service takes to complete a specific task. Many of the bioinformatics analyses are computationally expensive, such as BLASTing against a large data collection and multiple sequence alignment. Slow response is a common experience (sometime up to hours) when a service is requested by a large number of users. Average performance over a long period can be used as an indicator of the service capability.
– The *analysis quality* is of particular importance in bioinformatics research. It includes both the quality of the source data and the accuracy of the analysis methods. Some of the bioinformatics data are of prohibitive size and being updated frequently. For instance, the GenBank currently has over 85 billion bases [5]) and its exponential growth in the past decades means new data are being added all the time. Querying and analysis against such data repositories are often done on a replication to reduce the load on the main server or improve processing time when the copy is located much closer in terms

of network distance. In such cases, the users need to be aware that they might not be using the latest data.

In terms of analysis methods, there are two common cases where accuracy is traded for performance. Using sequence matching as an example, sometimes it makes sense to not searching against the entire GenBank if user has a mammal sequence. However, it is important that the user is aware of this. As a second case, approximation algorithms are commonly used instead of exact counterparts, due to the computationally expensive nature of many bioinformatics analyses. The arguably most popular bioinformatics algorithm *BLAST* [3] is not an exact sequence alignment algorithms. Also, there can be many variations of the initial algorithm. For instance, there are *blastn*, *blastp*, *PSI-blast*, etc. The users need to be aware of the approximation nature of these analysis algorithms and the error bounds the results have.

All these properties are functions of a complex system that includes service providers, network environments, and local hardware/software setup. It is important to provide a quantified measurement of these properties, so the service optimizer can utilize the information during service selection. While the service reliability can be measured using the percentage of up time, performance and analysis quality are less straightforward to quantify.

– **Comparability.** Both properties are function dependent, i.e, only the performance and analysis quality of the services with same functions are comparable. The comparability can be derived from the bioinformatics ontology annotation the service has. For instance, two services are comparable if they have the same function annotation.
– **Relativity.** For performance and analysis quality, it is the relative order among the comparable services that is important for service selection. A numerical value on its own does not convey much useful information.

While completion time can be used as a indicator of performance, it also depends on the input data and execution parameters. It is possible to test all comparable services with the same input and parameter setting, but each test only represents one sample point in a very large high-dimension space of all possible combinations. In the end, we decided to record the execution time together with input data description (not the actual data) and parameter setting, which are used in the manual ordering of the performance of comparable services.

The quantification of analysis quality has similar issue. To make matter worse, there is no direct indication of the analysis quality (i.e., no counterpart of "completion time" for analysis quality) and the information such as the error bound of the approximation algorithm is not usually obtainable through service interface. As a result, we decided to manually assign the ordering of service analysis quality.

Formally, for a service s, its reliability, performance, and analysis quality to a client c are:

– **Reliability:** $f_r(s, n) \rightarrow [0, 1]$,
– **Performance:** $f_p(s, n, c, u) \rightarrow [0, 1]$,
– **Analysis quality:** $f_q(s, u) \rightarrow [0, 1]$.

The reliability f_r is affected by that of the service s and network environment n between the two. For performance f_p, the hardware and software configuration of the client c also has an impact and service comparability depends on its function. The analysis quality f_q is mainly a function of the service s (the quality of the source data and nature of the deployed algorithm) and again comparability is decided by function. As discussed earlier, f_r is measured as the percentage of total up time:

$$f_r = \frac{T_{up}}{T_{total}} * 100\%$$

where T_{total} is total number of attempts trying to use the service and T_{up} is the number of times the service can be successfully invoked. The value of f_p is assigned semi-automatically: the system keeps a record of the completion time and input settings of every service execution, and user can use this information to assign the order among the comparable services. The value of f_q is assigned manually. For f_p and f_q, it is the relative value of that is important and we limited both within the 0 to 1 range for implementation convenience.

6.3 Service Manager Layer and Service Selection

The "Service Manager" layer uses a customized version of WSMX, which handles conversion and grounding to the SOAP/WSDL services and the actual invocation.

Service selection through function description is achieved using the "Service Discovery" component, which searches the domain ontology for user query terms and then find any service registered with that term. In our system, this can be done implicitly: when user selects a data service, the system automatically searches for services that can be performed on that type of data. Service selection with non-functional properties and execution optimization are achieved with the "Service Optimizer" component. Each service is described by a *quality vector* $\mathbf{f}(f_1, ..., f_k)$ (some representative quality properties are given in Section 6.2), which the "Service Optimizer" uses to optimize workflow execution. In most cases, there does not exist one service that has the best value in all service properties.

We propose to use a skyline computation approach to tackle the service selection problem. Computing a skyline guarantees to include the best user desired services without any user intervention. Skyline computation has recently received considerable attention in database community [9,39,21,28]. For a d-dimensional data set, the skyline consists of a set of points which are not dominated by any other points. A point $\mathbf{p}$ $(p_1, ..., p_d)$ dominates another point $\mathbf{r}$ $(r_1, ..., r_d)$ if $\forall\ i \in [1, d], p_i \succeq r_i$ and $\exists\ j \in [1, d], p_j \succ r_j$. We use $\succeq$ to generally represent *better than or equal to* and $\succ$ to represent *better than.* In the context of Web Services, a service skyline can be regarded as a set of service providers or their compositions that are not dominated by others in terms of all user interested quality properties.

The skyline approach goes beyond the current service selection approaches, which require users to transform personal preferences into numeric weights [15,27, 47,48]. The objective function assigns a scalar value to each service based on the quality values and the weights given by the service user. The service gaining the highest value from the objective function will be selected and returned to the user. However, users may not know enough to make tradeoffs between different quality aspects using numbers. Furthermore, most existing approaches work like a "black box", where users submit their weights over quality parameters and the system selected provider is returned. Users thus lose the flexibility to select their desired services by themselves. Computing skylines brings two key benefits for service selection that can overcome these issues:

– The skylines are computed automatically based on the inherent quality properties of service providers. Thus, it completely frees service users from the challenging weight assignment task.
– Computing skylines won't lose any merit of using the objective function. This is due to a major property of the skyline. For a set $\mathcal{S}$ and any monotone objective function $\mathcal{S} \to \mathbb{R}$, if $\mathbf{r} \in \mathcal{S}$ maximizes the objective function, then $\mathbf{r}$ is in the skyline [9]. Thus, no matter how the weights are assigned, the skyline guarantees that the user desired service providers are included so that users can make flexible selection from them. In addition, the users can always choose to use any monotone objective function they prefer after the skyline is computed. The optimal solution will always be the same as computed from the original service space but with a much efficient manner because of the much smaller skyline size.

We adopt an approach which is similar to the BBS (Branch and Bound Skyline [28]) approach to compute the skyline from services. All the services with the same function are indexed by using a R-tree. The leaf nodes of the R-tree correspond to the actual services. An intermediate node represents a minimum bounding rectangle (MBR) of each node at its lower level. The algorithm also leverages a priority queue (or a heap) to make sure the services are enumerated in an ascending of their *mindist.* The heap is constructed to efficiently output the node (intermediate or leaf node) that has the least *mindist.* The *mindist* of a leaf node is the summation of all its coordinate values (i.e., all quality properties) whereas the *mindist* of an intermediate node is the *mindist* of its lower-left corner point.

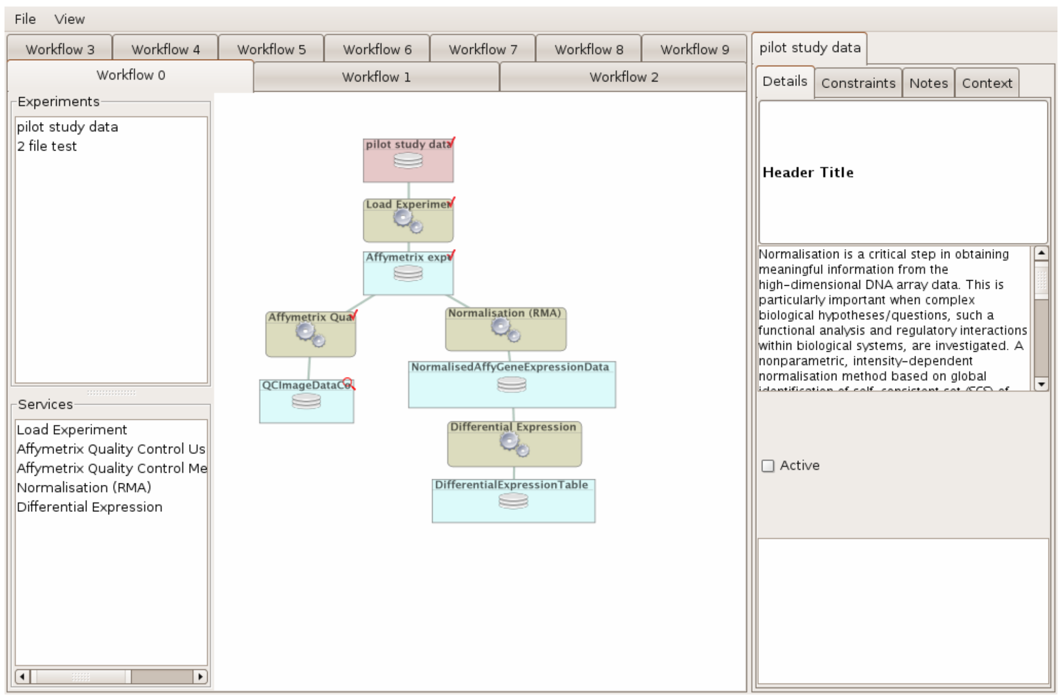
The detailed algorithm is given in Algorithm 1. It initially inserts all the entries in the root of the R-tree into the heap H. It then iteratively expands these entries based on their *mindist.* The expanded entry will be removed from the heap whereas its child entries will be inserted. When the first leaf node is returned, it will be inserted into the resultant skyline list L. A service R-tree, referred to as, $\mathcal{R}^S$, will then be initialized using the first skyline service. After $\mathcal{R}^S$ is constructed, the entries output from the heap will be checked against it for dominance. Specifically, if a top entry in the heap is dominated by some service in $\mathcal{R}^S$, it can be directly pruned. Otherwise, we have two situations:

1. If the entry is an intermediate node, it will be expanded into its child entries and these child entries will also be checked for dominance against $\mathcal{R}^S$ before inserting into the heap. The dominated entries can also be directly pruned.
2. If the entry is a leaf node, it will be inserted into both L and $\mathcal{R}^S$.

6.4 User Interface Layer

The user interface is a lightweight client written in Java, which allows deployment using *Java Web Start* without any pre-installation. It uses the `wsmo4j` implementation of the WSMO API to represent both available Web Services and domain concepts. The client communicates with the Service Manager layer using its SOAP entry points exposed by WSMX for service query and passing data.

A screen shot of the interface is shown in Fig. 4. The interface consists of three panels: on the left are the lists of available data sources and analysis methods, in the middle is the panel for workflow construction, and on the right is the panel that provides the description of the selected data source or analysis method. Workflows



Normalisation is a critical step in obtaining meaningful information from the high–dimensional DNA array data. This is particularly important when complex biological hypotheses/questions, such a functional analysis and regulatory interactions within biological systems, are investigated. A nonparametric, intensity–dependent normalisation method based on global

H

Fig. 4 User Interface.

Algorithm 1 Service Skyline Computation

```
Require: A R-tree $RT$
Ensure: A list of the SEP skyline points $L$
 1: $L = \phi$, $\mathcal{R}^S = \phi$;
 2: $H$ is initialized by the root entries of $RT$;
 3: while $H \neq \phi$ do
 4:   $e = H.extractmin()$;
 5:   if $\mathcal{R}^S \neq \phi$ then
 6:     check $e$ against $\mathcal{R}^S$ for dominance;
 7:     if $e$ is dominated then
 8:       prune $e$;
 9:     else
10:       if $e$ is an intermediate node then
11:         for each child entry $e.c_i$ of $e$ do
12:           if $e.c_i$ is not dominated by $\mathcal{R}^S$ then
13:             $H.insert(e.c_i)$;
14:           end if
15:         end for
16:       else
17:         $L.insert(e)$;
18:         $\mathcal{R}^S.insert(e)$;
19:       end if
20:     end if
21:   else
22:     if $e$ is an intermediate node then
23:       for each child entry $e.c_i$ of $e$ do
24:         $H.insert(e.c_i)$;
25:       end for
26:     else
27:       $L.insert(e)$;
28:       initialize $\mathcal{R}^S$ using $e$;
29:     end if
30:   end if
31: end while
```

can be constructed interactively. User can add service to the workflow by draging it from the left panel and drop it in the middle workspace. Service can be connected by linking the input of one service to the output of another using mouse. User can also right click on the output of a service and then choose from the list of available services that can be connected. This is achieved by performing a semantic query retrieving all services that can use the results from current service as input.

7 Performance Evaluation

7.1 Service Selection Algorithm

We conducted a set of experiments to assess the effectiveness of the skyline computation algorithm. Since there is not any sizable Web Service test case that is in the public domain and that can be used for experimentation purposes, we focus on evaluating the proposed skyline algorithm using synthetic quality properties. The quality properties are generated in three different ways. The quality properties of syntactic services are generated in three different ways following the approach described in [9]: 1) *Independent quality* where all the quality attributes of a service are uniformly distributed, 2) *Anti-correlated quality* where a service is good at one of the quality attributes but bad in one or all of the other quality attributes, and 3) *Corre-*

lated quality where a service which is good at one of the quality attributes is also good at the other quality attributes.

We setup a set of experiment parameters to evaluate and compare the performance the algorithm. These include the number of quality attributes in the range of 4 to 10 and the total number of services in the range of 10000 to 50000. By performance, we report both the total number of nodes accessed by the algorithm (which reflects the I/O cost and is independent of hardware settings) and the actual running time. Finally, we also present the sizes of the obtained skylines.

Figure 5 shows how the number of node accessed by the algorithm and the actual running time vary with the number of quality attributes on all three different quality distributions. The R-tree plus the priority queue strategy offers the optimal I/O performance [28], which has been demonstrated by the small number of nodes accessed by the skyline algorithm. Since I/O processing is the dominating factor in the overall performance of the algorithm, the skyline can be computed in a very efficient manner (as can be seen from the right-hand-side chart of Figure 5).

We show the effect of the number of services on the performance of the algorithm in Figure 6. We keep the number of quality attributes as 6 and vary the number of services from 10000 to 50000. The results are consistent with Figure 5 and further justify the efficiency the skyline algorithm.

In Figure 7, we investigate how the sizes of skylines change with number of quality attributes and the number of services. First of all, the skylines generated from anti-correlated quality have larger sizes than those generated from independent and correlated quality, which is just as expected. This explains why computing a skyline from anti-correlated quality is much slower than other quality distributions. The reason is that due to the large skyline size, a large amount of time will be spent on dominance checking, which slows down the overall performance. Second, the sizes of skylines clearly increase with the number of quality attributes. However, it is noteworthy that in most practical usage scenarios where the number of quality attributes is less than three, the sizes of the skylines are still within a practical range for user selection.

7.2 System Performance

One of the system requirements is acceptable running time for colorectal cancer experiment analysis (R5). In this section, we present a simple model of system performance and the results from empirical study.

The running time of an analysis service consists of three parts: service processing time, network transmission time, and local process time. This can be expressed as:

$$T(\alpha) = T_s(\alpha) + T_n(\alpha) + T_c(\alpha)$$

where $T(\alpha)$ is the total running time of an analysis service α, $T_s(\alpha)$ is the service processing time, $T_n(\alpha)$ is the network transmission time, and $T_c(\alpha)$ is the local processing time.

There are mainly two types of services in our system, database query and statistical analysis. The evaluation focuses on the latter because it is usually more data and computation intensive than the former. We tested two statistical analyses: Normalization and Differential Expression, as described in Section 3.

The data used were the microarray experiment data described in Section 3. It contains 20 microarray samples. Each sample is about 7 MB in size and the total size is 135MB. We recorded different running time (T_s, T_n, and T_c) with increasing sample number to evaluate system scalability. Our testing setup consists of four computers:

- **Computer 1 (C1)** hosts our user interface. This is where the statistical analysis requests are issued.
- **Computer 2 (C2)** hosts our system except the user interface.
- **Computer 3 (C3)** hosts the `R` statistical service.
- **Computer 4 (C4)** hosts the microarray database.

All computers are connected through a 100Mb/s Ethernet. The tests start with a statistical analysis request from $C1$ to $C2$, which then identifies that the data is available at $C4$ and statistical service is available at $C3$. Our system then requests $C4$ to send the data to $C3$, which returns the results to $C1$ once the analysis is finished. T_s includes all the processing time spent at $C2$, $C3$, and $C4$. T_n is the sum of the network transmission time between any pair of computers. T_c is the processing time at $C1$.

Figure 8 and 9 shows the time of Normalization and Differential Expression analysis respectively. The x axis is the input size, measured in number of samples. The differential expression analysis requires even number of CEL files and minimal 4 of them, so only these sample size are shown here. The y axis is the running time, measured in seconds.

The trends in the running time of the two analyses are similar: both T_s and T_n increase almost linearly and T_c is neglectable, as a result the total time T is close to a linear curve. While T_n increase linearly with the data size is expected, the change of T_s depends on the nature of the analysis algorithm. Both statistical methods tested happen to have a linear behavior. A

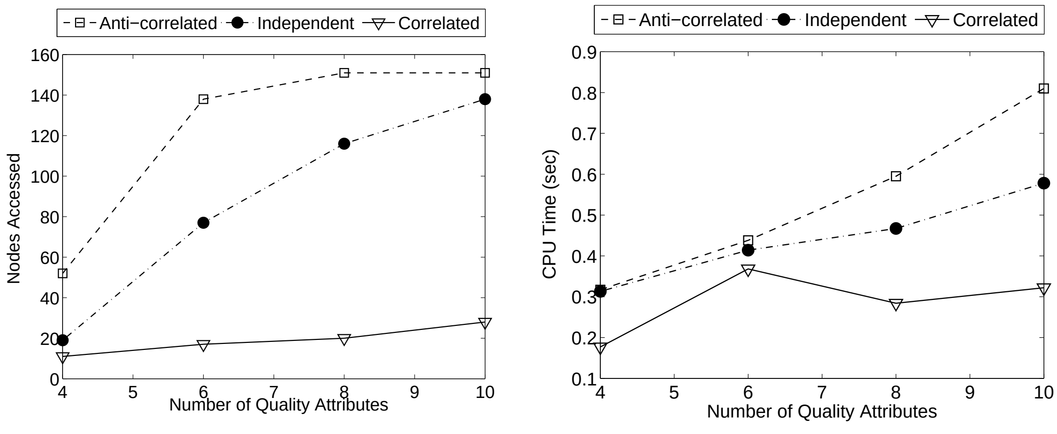


Fig. 5 I/O Accesses and CPU Time Vs. Number of Quality Attributes

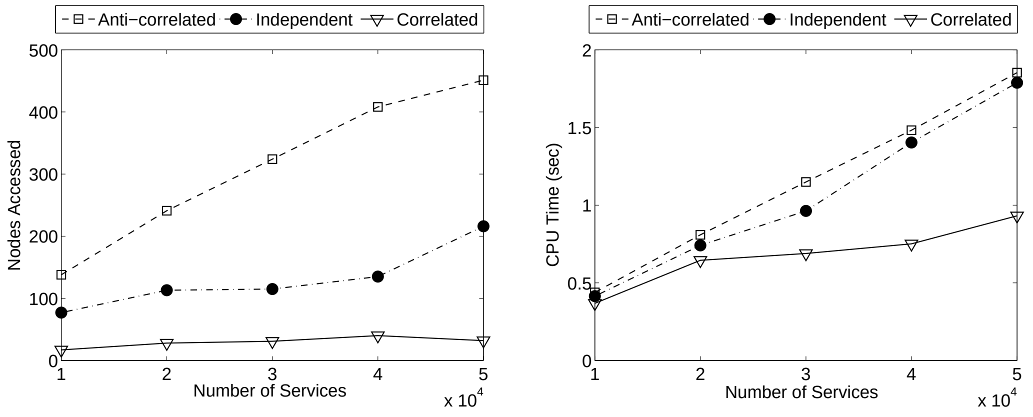


Fig. 6 I/O Accesses and CPU Time Vs. Number of Services

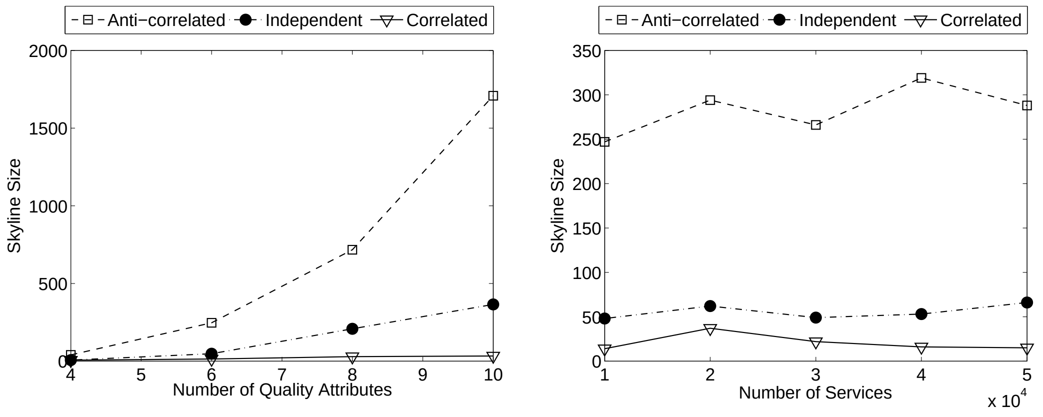


Fig. 7 Skyline Size Vs. Number of Quality Attributes and Number of Services

more computationally intensive algorithm could have a much steeper curve and quickly become the dominating factor in the T. In both tests T_n accounts for a large portion of the total time, which indicates network speed can a system bottleneck when the data size increases. This is especially the case in the Normalization analysis, because it returns the normalized dataset (the same size as the input dataset), whereas Differential Expression analysis only returns a list of gene names (small text string).

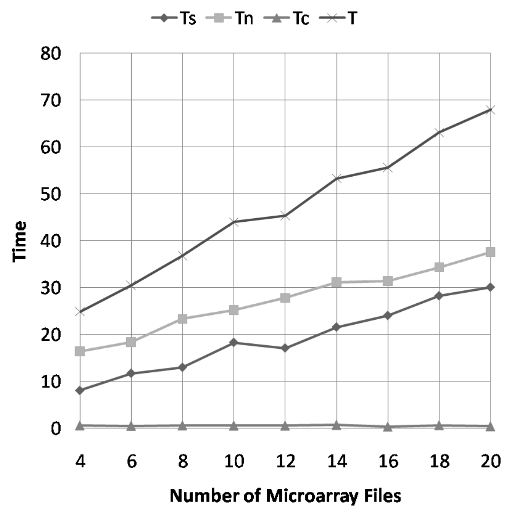


Fig. 8 Time of Normalization Analysis.

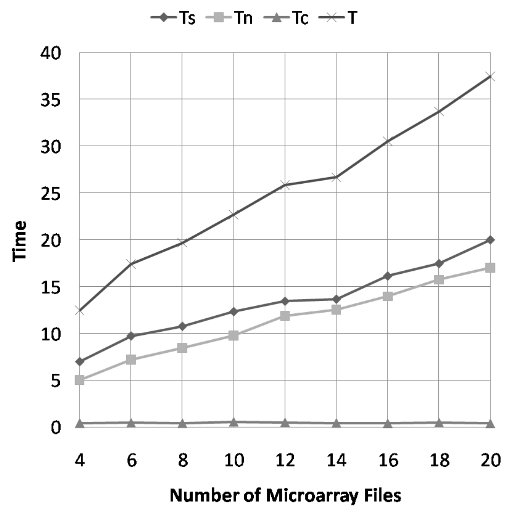


Fig. 9 Time of Differential Expression Analysis.

8 Lessons

In this section, we discuss the issues we encountered during the design and implementation of our system and how we try to address them.

- **Ontological annotation.** A difficulty we encountered is adding domain annotation to data and analysis services, which we had to rely on biologists for this. The lack of a commonly accepted bioinformatics ontology makes this process even more difficult. While it is unlikely that this process can be fully automated, a commonly accepted bioinformatics domain ontology that can be easily understood by biologists would certainly help. In our case, we designed a small domain ontology with the terms similar to our users, which appeared to alleviate the problem.
- **Non-functional property modelling and collection.** While we are aware that this is a research problem that has not been fully addressed, our experiences show that the difficulty varies among the properties. Reliability is the easiest to model and collect, and our model is well received by the users. Performance is less straightforward to model but there is still information available from the system that indicates the service performance. Analysis Quality is the hardest one and it requires a deep understanding of the service, such as the algorithm it deploys, and makes it time consuming process. A possibility is to use external information such as the feedback from other users to assist the assessment of analysis quality.
- **Service listing.** Currently there are many similar systems that list all the data and analyses services available. Our experiences show more is not always better in this case. In our system, only a small collection of services that are relevant to users are visible, and users actually prefer such interface to ones with large number of services. A better usability design is necessary if a system does need to present a great number of services.
- **Service execution optimization.** One of the important lessons we learned is that the current Web Service technologies are not fully ready to process the large amount of data involved in the bioinformatics research yet. We found that WSMX does not handle data transmission efficiently. We had to modify WSMX so data reference is passed between the services instead of the data themselves. As the evaluation results indicate, network transmission still requires considerable time even with the previously mentioned improvement and moderate data size involved in our project. With current technology, it is unlikely that performing similar analysis over internet can have acceptable response performance.
- **Global optimization.** The current service selection and execution optimization are limited at the service level, and do not guarantee the best performance at the workflow level. We don't see this as a pressing issue though, because similar issue exists in many other domains and has been relatively well studied.

9 Conclusions and Future Work

In this paper we presented a case study of the design and development of a semantic Web Service based system that aims to facilitate the colorectal cancer study. The system is designed to be small and agile. It includes limited services that are important to the colorectal cancer study and all system components are implemented as services to allow easy adaption to new requirements in future disease study. With biologists as the targeted user, usability is given high priority in the system. This results in a simplified user interface, easy access to all system functions through GUI, and an interactive workflow construction space. The system is built upon WSMO and thus provides native semantic support. A bioinformatics ontology is created to describe service function and a service ontology for non-functional properties. One of the challenges we faced is service selection based on non-functional properties. This requires modelling of properties such as reliability, performance, and analysis quality. Collecting service property information is of varying difficulty with manual intervention still required for Performance and Data Analysis. An skyline algorithm is proposed to select a list of "best" services that satisfy different combinations of non-functional property criteria. The evaluation results demonstrated that the service selection algorithm scales well with number of properties. Empirical study also shows the system performance behavior with increasing input data size. The issues we encountered and lessons we learned are discussed and summarized.

This is our first iteration of the system design and implementation, and there are a few features that we would like to develop further. Provenance data collection and management will be our focus in the next stage. This will enable the reproduction of an analysis instance if needed and the sharing of workflows among users. The availability of provenance data will also provide new information that can be used in non-functional property modelling and allow better property value estimation. We are also interested in the possibility of using user feedbacks to assist estimating service analysis quality, which currently requires a deep understanding of the service implementation.

Acknowledgements The Tasmanian ICT Centre is jointly funded by the Australian Government through the Intelligent Island Program and CSIRO. The Intelligent Island Program is administered by the Tasmanian Department of Economic Development, Tourism and the Arts. We would like to thank Mark Hepburn for his contribution to the implementation of the system.

References

1. *Getting Started with the Kepler Tagging Module*, Last accessed on 14 October 2010. https://code.kepler-project.org/code/kepler/releases/release-branches/tagging-2.1/docs/tagging.pdf.
2. Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. Kepler: An extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, page 423, Washington, DC, USA, 2004. IEEE Computer Society.
3. Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.
4. M Ashburner, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, M A Harris, D P Hill, L Issel-Tarver, A Kasarskis, S Lewis, J C Matese, J E Richardson, M Ringwald, G M Rubin, and G Sherlock. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.
5. Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank. *Nucleic Acids Research*, 36:D25–D30, 2008.
6. Chad Berkley, Shawn Bowers, Matthew Jones, Bertram Ludäscher, Mark Schildhauer, and Jing Tao. Incorporating semantics in scientific workflow authoring. In *SSDBM'2005: Proceedings of the 17th international conference on Scientific and statistical database management*, pages 75–78, Berkeley, CA, US, 2005. Lawrence Berkeley Laboratory.
7. Jiten Bhagat, Franck Tanoh, Eric Nzuobontane, Thomas Laurent, Jerzy Orlowski, Marco Roos, Katy Wolstencroft, Sergejs Aleksejevs, Robert Stevens, Steve Pettifer, Rodrigo Lopez, and Carole A. Goble. Biocatalogue: a universal catalogue of web services for the life sciences. *Nucleic acids research*, 38 Suppl(suppl_2):W689–694, July 2010.
8. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5:1–22, 2009.
9. S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, 2001.
10. Shawn Bowers and Bertram Ludascher. An ontology-driven framework for data transformation in scientific workflows. In *Proceedings of the First International Workshop on Data Integration in the Life Sciences*, pages 1–16, 2004.
11. Shawn Bowers, David Thau, Rich Williams, and Bertram Ludascher. Data procurement for enabling scientific workflows: On exploring inter-ant parasitism. In *Proceedings of the Second International Workshop on Semantic Web and Databases*, pages 57–63, 2004.
12. Michelle D. Brazas, Joanne A. Fox, Timothy Brown, Scott McMillan, and B. F. Francis Ouellette. Keeping pace with the data: 2008 update on the bioinformatics links directory. *Nucleic acids research*, 36, 2008.
13. Alvis Brazma, Pascal Hingamp, John Quackenbush, Gavin Sherlock, Paul Spellman, Chris Stoeckert, John Aach, Wilhelm Ansorge, Catherine A. Ball, Helen C. Causton, Terry Gaasterland, Patrick Glenisson, Frank C.P. Holstege, Irene F. Kim, Victor Markowitz, John C. Matese, Helen Parkinson, Alan Robinson, Ugis Sarkans, Steffen Schulze-Kremer, Jason Stewart, Ronald Taylor, Jaak Vilo, and Martin Vingron. Minimum information about a microarray experiment (MIAME) toward standards for microarray data. *Nature Genetics*, 29(4):365–371, 2001.
14. Jos De Bruijn, Lausen Holger, Axel Polleres, and Dieter Fensel. The web service modeling language WSML: An overview. In *Proceedings of the 3rd European Semantic Web Conference*, pages 590–604, 2006.
15. Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qos-aware service composition based on genetic algorithms. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1069–1075, New York, NY, USA, 2005. ACM.

16. Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. Chair-Richard N. Taylor.
17. Michael Y. Galperin. The molecular biology database collection: 2008 update. *Nucleic Acids Research*, November 2007.
18. Robert C Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.
19. Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Christoph Bussler. WSMX - a semantic service-oriented architecture. In *Proceedings of the IEEE International Conference on Web Services*, pages 321–328, 2005.
20. Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28:27–30, 2000.
21. D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, 2002.
22. Alberto Labarga, Franck Valentin, Mikael Anderson, and Rodrigo Lopez. Web Services at the European Bioinformatics Institute. *Nucleic Acids Research*, 35(Web-Server-Issue):6–11, July 2007.
23. Phillip W. Lord, Pinar Alper, Chris Wroe, and Carole A. Goble. Feta: A light-weight architecture for user oriented semantic service discovery. In *Proceedings of the Second European Semantic Web Conference*, pages 17–31, 2005.
24. Shalil Majithia, Matthew S. Shields, Ian J. Taylor, and Ian Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In *Proceedings of the IEEE International Conference on Web Services*, pages 514–524. IEEE Computer Society, 2004.
25. David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic markup for web services. Technical report, http://www.w3.org/Submission/OWL-S/, 2004.
26. Thomas Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
27. M. Ouzzani and B. Bouguettaya. Efficient Access to Web Services. *IEEE Internet Computing*, 37(3), March 2004.
28. D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 2003.
29. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.
30. Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubn Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.
31. Paolo Romano, Domenico Marra, and Luciano Milanesi. Web services and workflow management for biological resources. *BMC Bioinformatics*, 6(Suppl 4):S24, 2005.
32. 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, 1995.
33. Gary Schiltz, Damian Gessler, and Lincoln Stein. Semantic moby. In *W3C Workshop on Semantic Web for Life Sciences*, 2004.
34. Dimitrios Skoutas, Dimitris Sacharidis, Alkis Simitsis, and Timos Sellis. Ranking and clustering web services using multicriteria dominance relationships. *IEEE Transactions on Services Computing*, 3:163–177, 2010.
35. Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H Scheuermann, Nigam Shah, Patricia L Whetzel, and Suzanna Lewis. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007.
36. U. Srivastava, J. Widom, K. Munagala, and R. Motwani. Query Optimization over Web Services. In *VLDB*, 2006.
37. Robert D. Stevens, Alan J. Robinson, and Carole A. Goble. myGrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19:302–304, 2003.
38. Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545–15550, 2005.
39. K. Tan, P. Eng, and B. Ooi. Efficient progressive skyline computation. In *VLDB*, 2001.
40. Ian J. Taylor, Ewa Deelman, and Dennis B. Gannon. *Workflows for e-Science: Scientific Workflows for Grids*. Springer, December 2006.
41. Yao Wang and Julita Vassileva. A review on trust and reputation for web service selection. In *Proceedings of 27th International Conference on Distributed Computing Systems Workshops*, page 25, 2007.
42. Mark D. Wilkinson and Matthew Links. BioMOBY: An open source biological web services proposal. *Briefings in Bioinformatics*, 3(4):331–341, 2002.
43. Katherine Wolstencroft, Pinar Alper, Duncan Hull, Christopher Wroe, Phillip Lord, Robert Stevens, and Carole Goble. The mygrid ontology: bioinformatics service discovery. *International Journal of Bioinformatics Resesearch and Applications*, 3(3):303–325, 2007.
44. Hong Qing Yu and Stephan Reiff-Marganiec. Non-functional property based service selection: A survey and classification of approaches. In *Proceedings of the 2nd Workshop on Non Functional Properties and Service Level Agreements in Service Oriented Computing*, 2008.
45. Q. Yu and A. Bouguettaya. Framework for Web Service Query Algebra and Optimization. *ACM Trans. Web*, 2(1), 2008.
46. Qi Yu and Athman Bouguettaya. Computing service skyline from uncertain qows. *IEEE Transactions on Services Computing*, 3(1):16–29, 2010.
47. Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1):6, 2007.
48. L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.