A Novel Approach for Bioinformatics Workflow Discovery

Walaa Nagy
Information Systems Dept.
Faculty of Computers and Information
Cairo University.
Egypt

Hoda M. O. Mokhtar
Information Systems Dept.
Faculty of Computers and Information
Cairo University.
Egypt

***Abstract*—Workflow systems are typical fit for in the explorative research of bioinformaticians. These systems can help bioinformaticians to design and run their experiments and to automatically capture and store the data generated at runtime. On the other hand, Web services are increasingly used as the preferred method for accessing and processing the information coming from the diverse life science sources. In this work we provide an efficient approach for creating bioinformatic workflow for all-service architecture systems (i.e., all system components are services). This architecture style simplifies the user interaction with workflow systems and facilitates both the change of individual components, and the addition of new components to adopt to other workflow tasks if required. We finally present a case study for the bioinformatics domain to elaborate the applicability of our proposed approach.**

***Index Terms*—Web services; In-silico Workflows; Quality of Services (QoS); Web services for bioinformatics; Bioinformatics services.**

I. Introduction

Due to the large number of available web services, the sheer complexity of data, and the frequent lack of documentation, discovering the most appropriate web service for a given task is a user challenge. In addition, in current practice there is a disparity between bioinformatics workflow conceptualization and specification. We believe that this disparity can be resolved using semantic Web. Semantic Web technologies can be used to translate from the specification of analytical functions or final state to executable workflows. Such workflow abstraction can consequently result in a unification of workflow conceptualization and specification. Nevertheless, such a level of abstraction will be useful for both novice and expert users, and will provide a means to easily and efficiently create workflows for other life sciences researches. Many life science tools are currently available as web services. The use of a workflow system to orchestrate these services and create in-silico experiments seems to be a logical approach. Thus, the methodology for creating bioinformatics workflow depends heavily on the knoweledge that the researcher has about the tools that he will use. In our research the user was enabled to emphasize the desired final state instead of providing the details of the process that he wants to perform or the relationships between his/her tools, which corresponds more closely to the objective of the research, provides further abstraction and reduces the workflow conceptually to a single functional operation as follows:

construct phylogenetic tree to a reference sequence.

This statement "Construct phylogenetic tree" subsume the steps required to generate phylogenetic tree [1] which are:
Step 1: Choosing an appropriate markers for the phylogenetic analysis.
Step 2:Performing multiple sequence alignments.
Step 3:Selection of an evolutionary model.
Step 4:Phylogenetic tree reconstruction.
Step 5:Evaluation of the phylogenetic tree.

Currently, to the best of our knowledge, there is no guide to assist users in the Web services discovery process. However, there is a significant progress that has been made towards building integration platforms that utilize online Web services interfaces to support bioinformatics analyses. Taverna [2] is such a platform that is widely used by the bioinformatics community, whereas Kepler [3] and Triana [4] 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 are now available. Semantic annotation provides richer information than Web service description alone and can be used for automatic reasoning when they conform to certain ontology.

Other approaches focus on the discovery of Web services that are annotated with a specific vocabulary. This is the case of the myGrid[1] project, whose aim is to provide a controlled vocabulary to make annotations. BioCatalogue [5] is a life science Web service registry with 1180 registered Web services that are meant to be annotated using life sciences ontology. Nevertheless, another issue to be taken into account is that, in many cases, multiple services provide very similar functionality (a particularly insidious example is the multitude of services providing variants of alignments of genes and proteins). In such cases, the user has to decide which service is the most appropriate based on diverse quality criteria (availability, coverage of the domain of interest, etc.). To

[1] http://www.mygrid.org.uk

address this problem, assessment techniques must be applied to provide the user with some information about the quality and the functionality of each service [6].

In this research a workflow representation was suggested that can solve the disparity between how life scientists conceptualize their workflows and how the workflows are specified in practice. The resolution is through a description of workflows through specification of primary analytical operations or desired analysis final state that enables the design of high-level workflow models. These high-level workflow descriptions are closer to the conceptual models that life scientists have about experiments. Consequently, the cognitive load on those scientists has been reduced [7]. In our work we use the semantic Web technology to translate from final state functions to an executable workflow.

The proposed idea depends on the idea of standardization of Web service interfaces. This standardization enables the language to be later used in life science workflow systems to describe and implement classes of tasks. For example, the proposed workflow representation can help to provide a standardized interface for the famous biological sequence alignment tasks in a workflow, then; the abstract alignment task encompasses all alignment resources, including (a) synchronous Blast services, and (b) Blast services. There are several key design rationales of our language that can be summarized as follows:

1) Our proposed services selection for in-silico workflow enables the life scientist to think in terms of the operation he wants to perform instead of the services he needs to use. Thus our solution reduces the gap that currently exists between the level at which a life scientist thinks about the life science problem and the actual workflow implementation.
2) The proposed workflow has 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 workflow tasks if required.
3) The whole system is based on a semantic Web service framework (WSMO [8]). As a result, all system components provide native semantic support.
4) In our proposed workflow services selection method only essential services are available and automatic service selection is done. .
5) the proposed approach select the appropriate Web services that can execute the task dynamically at run time instead of hard coding the services that can execute the task during the design of the workflow. This feature in turn overcomes the issue of dead or unavailable services.
6) Finally, the user provided with the best available Web services that performs his request based on both quality of services criteria, and his/her stored profile values. This feature considers the fact that currently there exist potentially large number of services from different service providers that can functionality overlap we believe that the non-functional properties of services, especially those related to Quality of Service (QoS), such as reliability, performance, and sensitivity, should be considered when several services provide similar function or information. A service selection algorithm is used to help users identify the "best" service [12].

The rest of the paper is organized as follows: Section II, presents background, related work, and motivates the problem. Section III, discussed the workflow discovery problem and solution.Section IV explain the worklet architecture. Section V, presents a use case that are used to evaluate the proposed solution. Finally, section VI concludes and outlines directions for future work.

II. RELATED WORKS

A. Current Workflow Design Systems.

There have been a large number of workflow design and execution engines that supports In silico biological study. Taverna [2] 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 a Web service interface. 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 non-trivial.

In [9] a prototype solution for the service selection problem in the life-sciences is proposed. The solution uses the Moby environment that uses lazy breadth-first search over an implicit graph of the service space. Also, the presented method is able to highly rank desirable services at interactive speeds. Although the solution improves the service selection process so that the user is presented only with a small set of the most salient services to the time and effort required to build workflows, the solution purposely uses a minimal amount of semantic information (data-type matching only), and does not consider the un-experienced user who is not familiar with workflow usage.

B. Semantics and quality-aware Service (QoS) in Current Workflow Design Systems.

Some workflow design systems have incorporated semantic component to support workflows execution. A bioinformatics semantic workflow is the workflow specification of a complex scientific task in terms of the biological analysis objectives and the semantic characteristics of the computational tools needed for the analysis [10] without delving into the computational details of the tools. The semantic workflow is at the level of biological concepts, that are much closer to the scientific research. Although Taverna has semantics to describe workflow activities, it is limited to supporting the discovery of proper services during the design of a workflow [9].

In [10], the author presented ***Sesame*** a semantic bioinformatics workflow design system with new ontology for bioinformatics tools/services. Sesame allows

the biologists to perform their analyses using terms that they are familiar with. After designing the semantic workflow, ***Sesame*** ask the user to instantiate it by associating each analyses entity with the instances of bioinformatics tools/services and data.
Although Sesame free the biologists from the necessity of learning the details of the computational aspects of the bioinformatics tools. yet, it still requires its users to have knowledge in scripting tools/ services, and algorithms that they will use.
Also, ***Sesame*** can only perform simple instantiation cases and for each analyses entity Sesame ask the user to select one instance of bioinformatics tools/services. Then, the user specifies the parameters and input data for the selected tool/service which overload the unexpirencied user with alot of work that he/she may not understand.

The work in [11] presents a new way for describing workflow templates and instances. All the used constituents are semantic objects that are described with properties and workflow level constraints. Once a workflow template is created and validated by an experienced user, it is easy for more junior scientists to create sophisticated analyses simply by specifying input data for pre-defined templates. The system ensures that the input data specified is appropriate given the definitions in the workflow template, and automatically generates a workflow instance that can be mapped to execution resources.
Despite all those efforts, in this work a better approach was presented that ensures the existence of all the resources that can perform the task rather than having a template that may contain a link to a dead service or an unavailable resource. In addition, the proposed approach also consider the QoS information in selecting the services that best execute and meet the user requirements.

The authors in [12] have utilized several semantic technologies to identify the scientists intent, and then to facilitate the control of workflow execution and enrichment of workflow provenance. While the integration of semantic components with the original system improved the usability of the workflow system, users still have to be familiar with those workflow design systems in order to accomplish a data analysis task.

Zhang et al. proposed a two-step approach to automatically transform geospatial procession conceptual workflow to Kepler workflow [3]. However, their transformation is limited to only one geographical information system package.

For a more complete selection of services in addition to the syntactic and the semantic dimension in building the workflow the quality of resources are also important factors for making adequate service choice.
In [13], an optimization algorithm is proposed to efficiently access Web services. The 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 minimal total query running time. In addition, in [14] quality-aware service optimization techniques have been studied. 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.

On the other hand, in our proposed approach we learn those weights dynamically from the user profile instead of asking the user about the weights of QoS criteria which could lead to missing the user desired services due to the user inability to precisely specify the weight of each QoS parameter if multiple quality parameters are involved. Nevertheless, our proposed approach follows the work presented in Adams et al. [15]. In this work the authors introduced a technique to design abstract workflows to describe the treatment processes of patients. The central component in their approach is worklets. A *worklet* is a small workflow that covers all the actions required to perform a higher level task (or step). The main advantage of worklets is that it black boxes implementation details from a workflow designer point of view. The approach supports flexibility and evolution in workflows through the support of flexible work practices, based not on proprietary frameworks, but on accepted ideas of how people actually work. Our approach although borrows the worklet concept, differs in the way we build and select the worklet assigned for each workflow task. In addition, we present a new application direction for workflows through employing worklets that target bio-informaticians. In addition, we employ worklets to achieve an efficient Web service selection interface. The remaining of the paper elaborates our approach in using worklets to provide a standard language for in-silico service selection.

III. Workflow Discovery

Workflow discovery in our work is responsible for finding available workflows for the specified functionality. Workflows are discovered using information on their specification, the desired functionality, and available services. For simplicity, we provide an example of workflow discovery using only the desired functionality (e.g., phylogenetic analysis).

Each task of a workflow is linked to an extensible repertoire of actions. In this work, we present those repertoire-member actions as “worklets”. In effect, we borrow the definition of a worklet from [15] where a worklet is defined as a workflow that handles one specific task in a larger, composite workflow. The use of worklets enables the design of abstract, reusable workflows that can be used for different cases with a similar high level structure. Finally, a sequence of worklets is chained to form an entire workflow process.

At the same time, a repertoire for each workflow task is dynamically constructed as different approaches and methods for completing those tasks are continuously developed. The input variables of the original task are mapped to the input variables of the selected service on the worklet, and then the worklet is launched. When the worklet is completed, its output variables are mapped back to the output variables of the original work process.

Our workflow execution use the same model of execution presented in [16]. In [16] the workflow is composed of a service that can execute the required task but our workflow is composed of worklets that enable the design of abstract, reusable workflows that can be used for different cases with a similar high level structure. The execution of a workflow depends on a utility service such as *satisfyPrecondition*, this service is called at the beginning of each process model for worklets with preconditions. As a result, the prerequisite worklet are executed first, in order to satisfy the preconditions; once all preconditions are met, the initial service is finally executed. The precondition dependency check is recursively applied until a service with no preconditions is reached.

In this way, each service prerequisite will help compose an executable workflow. Subsequently, the workflow will be composed backward and at runtime reversed for proper execution. Figure 1 shows this method.

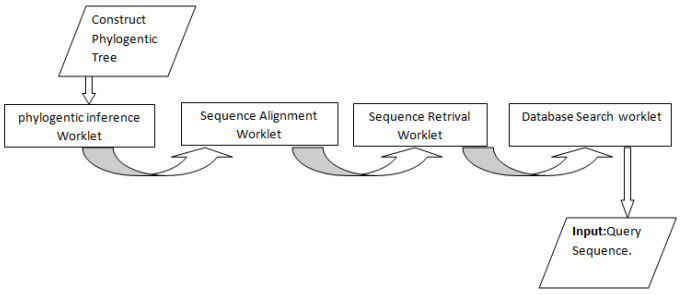

Figure 1: Workflow Backwards Composition

A worklet in our work is handled like a normal Web services. Semantic services are defined using OWL-S [17]. A semantic worklet service description is constructed with a precondition that reflects the prerequisites needed in order for the service to successfully complete; this precondition is sufficient to construct workflows for the desired functionality. We have considered a way to define worklets preconditions. For example, a phylogenetic analysis worklet named "Phylogenetic Inference Worklet, This service requires a multiple sequence alignment as an input and can be described as follows:
Worklet: Phylogenetic Inference.
Precondition: Multiple sequence alignment.
Input: Optional multiple sequence/analysis parameters.
Output: Phylogenetic Tree.

As we are searching for life sciences Web services, we apply the taxonomy of categories used by BioCatalogue [5] in order to classify the worklet. BioCatalogue is a registry of curated Life Science Web Services. The aim of BioCatalogue is to provide an easy way for scientists to discover Web services of interest. BioCatalogue has a shallow taxonomy of Web services categories, and most of the registered Web services have at least one category. So, we use this taxonomy to classify the worklets with the aim of using those categories in the discovery of the Web services that are suitable for a given worklet function.

Example 3.1: Consider a worklet "Sequence Alignment" building problem. In this worklet we collect all services that can perform sequence alignment like (a) synchronous Blast services, and (b) Blast services. Hence, the user simply specifies that a Sequence Alignment is desired. Then, we select the worklet named Sequence Alignment to execute his request. At the lowest level, expert users can specify all the analysis parameters for every step in each analysis that constitutes the workflow.

Web services in BioCatalogue have four main different types of annotations: descriptions, operations, tags, and categories. Each worklet will contain a set of comparable services in function, the comparability is derived from the bioinformatics ontology annotation that the service has. For instance, two services are comparable if they have the same function annotation, in the following section we will provide the details of how we discover those services.

IV. Worklet Architecture

A worklet is basically nothing more than a workflow specification that has been designed to perform one part of a larger parent specifications. However, a worklet differs from a decomposition or sub-net in that it is dynamically assigned to perform a particular task at runtime, while sub-nets are statically assigned at design time. So, rather than being forced to define all specification of the task such as algorithms to be used and the service during the design time of a workflow, the worklet service allows the definition of a much simpler specification that will evolve dynamically as more worklets are added to the repertoire for particular tasks.

In addition, as we aim to support flexible system architecture, we therefore implement the worklet components as Web services as well. Such implementation approach enables us to easily update and/or replace a worklet for other workflow system usage. Besides, we provide a semantic description for service function which requires bioinformatics ontology to describe them. In our work we choose to store functional and non-functional services descriptions as semantic description on the service disk entity. This allows reasoning with such information, which is not possible otherwise and is a vital component for service selection and optimization.

In general, worklets may be associated with either an atomic task, or a multiple-instance atomic task. Any number of worklets can form the repertoire of an individual task, and any number of tasks in a particular specification can be associated with the worklet service.

A. Worklet Services Selection

The selection of services inside the worklet depends on the categories, functionality, and ratings of previous interactions. In this work, we use categories as the criterion for the discovery process of those comparable services since they very well express the functionality of services. The discovery process consists basically of: querying BioCatalogue, and using its recent launched API [5]. Each query searches for a specific category and retrieves a set of web services that are annotated with this category. In most cases, the search retrieves more than one service, since there are many services annotated with the same category; however, the set of retrieved services may include services that do not provide exactly the required functionality. This instance is due to the fact that some categories are too general to describe a specific functionality. Among those services we select the services that are comparable based on the services ontology.

Besides, we find "myGrid" ontology [18] the most promising ontology to use, but it can be too big and complex for our purpose. Therefore, we use "myGrid" ontology as a reference and build our own domain ontology only covering the data and analysis required. Our bioinformatics ontology is hence 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 equivalent to identifying that function term in the ontology and returning all the services registered with it and its descendant terms.

Also, we use the service ontology for describing services capabilities. For this purpose, we considered both OWL-S [17] and WSMO [8]. While both are capable of describing the service properties that our system requires, we select WSMO because its model of describing non-functional properties matches better our requirements. On the other hand, as there exists many non-functional properties that can be used for service description [19], [20], we mainly focus on reliability, performance, and sensitivity. Those non-functional measures are good indicator of the QoS in our case. Those measures are defined briefly in the following discussion:

- **Reliability R:** 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.

$$Reliability = \frac{T_{up}}{T_{total}} \times 100 \quad (1)$$

  Where T_{total} is the total number of attempts trying to use the service, and T_{up} is the number of times the service can be successfully invoked.
- **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 indicators of the service capability. The value of performance is assigned semi-automatically: the system keeps a record of the completion time and input settings of every service execution, the evaluation of service performance P is given by the normalized utility function:

$$P = 0.5 \;\; rt \quad (2)$$

  Where rt is the relation between the service response time and the size of the input file, and is given by the following relation

$$rt = \frac{response \;\; time}{input \;\; size} \quad (3)$$

- **Sensitivity:** refers to the ability to identify all significant information that is related to the input data independent of its quality. Sensitivity is calculated by corroborating results of different services: this repeats the request sent to one service to a different service, and compares the matches given by both. The premise is that services that give top hits are similar enough to validate each other results and, thus, the comparison of different results can identify both true positives and false positives between the best hits.

After we retrieve the set of services that match a specific category and functionality, semantic description is added to those services. The domain ontology provides a biological description of the services, while the service ontology provides the property information among others. Then, we use the service selection algorithm proposed in [21]. The service selection algorithm presented in [21] can be used as tool to help in the selection of Web services based on the available providers and user requirements. In brief the algorithm proceeds as follows, after the selection of the available service providers that serve the user request; the algorithm selects the best set of services as required by the user. The authors rank those set of candidate services and only present those services that most likely solve the user request. A key feature of that approach is that, instead of asking the user for the non-functional properties, the algorithm uses the importance level for QoS parameters from his profile, which makes this algorithm easy to use even for someone not very familiar with the different QoS attributes.

Then, the system allows the user to rate any of the matched services, indicating how relevant or appropriate they are for his request. Besides, a key advantage of the work in [21] is the storing of meta-data about earlier successful services invocations even by other users. Finally, the algorithm keeps services updated by checking their status periodically and providing the user with a report about services usages. In this work, we employ the same algorithm to select the best service based on QoS values.

A workflow specification often contains the physical locations of the resources used. Resources can be (temporarily)

unavailable, can be replaced, or can be moved to a different location. This complicates workflow reuse. Delaying the choice of resources until instantiation time would be a solution [22], and is known as late binding [22]. Late binding is supported in our workflow execution since we select the appropriate Web services that can execute the task dynamically at run time instead of hard coding the services that can execute the task during the design of the workflow as described below.

B. Service Selection Movie

In this paper we propose to use a movie set to represent our idea of building the worklet and its selection. The Movie metaphor is introduced in the Dutch Driving Simulator [23] to create an easy-to-understand architecture that supports dynamic generation of traffic scenarios.

In the Movie Metaphor, the world is seen as a movie set in which actors play roles and executes tasks conforming to that role. In the following discussion we demonstrate how this Movie Metaphor approach (see Table I) can be applied to create an easy-to-understand architecture for a workflow task execution. In the movie set, the worklets are the actors and the workflow engine is the director. The role describes the function of the actor required to execute the task [24]. The director controls the set; he selects the tasks to be executed based on the script. The director does not work in isolation but gets help from the casting director to select the actors to execute tasks (known as actor assignment [25]). Like the script of a theatrical play, a workflow specification can be performed (instantiated) more than once and each performance can have different actors involved. Table 1 presents an overview of the terms we borrowed from the movie set and what they stand for and how they can be interpreted in a workflow context.

The actors, director, and casting director interact. Based on the script, the director determines the tasks to be executed. For each of those tasks, the director asks the casting director for an actor to execute it. The casting director searches for an actor in so-called actor repositories, which acts as casting agencies. Multiple actors can be available to play the same role. Based on the task and the role attached to it, the casting director selects a capable actor and delivers him to the director. In workflow terms (see Figure 2), a suitable worklet is selected based on the role (category of worklet) and the task (operation type). The director delegates the task to the selected actor, which then executes the task. This results in 3 possible scenarios:

1) Workflow designers can define a preferred actor for a task. If the casting director finds this preferred actor, it returns this actor.
2) If no preferred actor is set or available and only a single actor is suitable, the casting director selects this actor.
3) If no actors are available, this means that this task has not been considered before. Hence, a new worklet is to be created, and then we create a new actor that represents the new added worklet and store it inside the actor repositories.

Table I: Terms in the movie set and their usage in the workflow paradigm.

Term	Movie Set	In Workflow Context
Script	The story describing the movie	A (hierarchical) workflow specification. A workflow that specifies the tasks to be executed.
Scene	A unit of action, taken at a single location.	A (sub)workflow, consisting of tasks constituting a higher level task which we refer to as a Worklet in our workflow.
Actor	A person with the capabilities to play a certain role.	A resource, i.e. a Worklet that carry certain task.
Role	Specification of a character and its tasks.	A specification (category) of actor required to perform a task.
Director	Person who directs the movie.	The workflow engine; it selects and schedules tasks
Casting director	Person responsible for selecting actors based on the role descriptions.	The component that selects actors based on the role attached to the task.

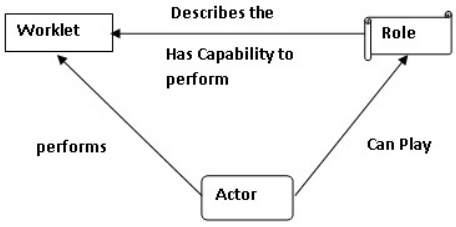

Figure 2: Worklet Role Assignment.

The worklet categories are translated to roles and are registered in our system. Only one role representing that worklet category is registered. To simplify the discovery of roles, an actor repository is created for each worklet (i.e. reference to the worklet).

Worklets are discovered using information about the desired functionality that the user needs. The user specifies the workflow and its script. The casting director then searches for actor(s) (i.e. worklet) using the role attached to the available actors. Note that even with the case that the user specifies the full specification of the workflow he needs, there is still a great deal of abstraction. For example, the algorithm that will be used, the accuracy of the service that provides the result, etc. This alone is a major step forward compared to the current state of in this research direction. In practice, most users will set their own personal preferences for the individual service choices and associated options, and so in most cases the interaction will return higher levels of abstraction in subsequent use. In addition, the worklet service creates a

Table II: Discovered Worklets

Worklet		
Name	Category	ID
Retrieve Protein sequence	Protein Sequence Retrieval	1
Gene prediction	Gene Prediction	2
Protein sequence alignment	Protein sequence alignment	3
Phylogentic tree	Phylogyn	4
Protein sequence analysis	Protein sequence analysis	5
Protein sequence repeats analysis	Protein sequence repeats analysis	6

Table III: Actors for Discovered Worklets and their Roles

Actor				
P	A	N	Role	ID
✓			Retrieves information from the database of protein	1
	✓		produces a list of predicted genes given a sequence of DNA.	2
	✓		Display basic information about a multiple sequence alignment	3
✓			Builds the most accurate phylogenetic tree.	4
✓			A protein sequence and annotation database.	5
		✓		6

process log to provide a complete operational history of each process that is then stored in the system.

V. Use Case

In this section, we develop a bioinformatics case study extracted from [26]. In this way, we can illustrate how to use our approach to guide the user in a real web service discovery tasks. The case study considers biological research that analyzes the presence of specific genes involved in the genesis of Parkinson Disease, called LRRK2 genes, in different organisms. The goal is to know more about the biochemical and cellular functions of these genes. The authors study the presence of the LRRK2 genes in organism "N. Vectensis", since previous studies have shown that this organism is a key organism to trace the origin of these genes. The authors describe the process step-by-step.

We have selected this case study because it describes with details the techniques used in every step, and it could be useful to validate our approach. However, our intention is not to model a concrete case study, but to offer a guide for more general cases. In the rest of this section we present a short description for each step of our approach in order to discover the web services that provide the functionality required by the scientist.

In this case study the user requirement is to build a workflow to obtain a comparison of the LRRK2 genes in different organisms. The steps the user will do manually by himself to build the workflow are [26]:

1) Retrieve the protein sequences of the different domains;
2) Predict the gene structure automatically for the sequences retrieved in Step (1);
3) Align protein sequences to build phylogenetic trees;
4) Build the phylogenetic trees;
5) Analyze the structure of the proteins;

In our model the execution process will begin from the last step in the case of "Analyze the structure of proteins", and continue until no more pre-conditions are needed in the process of "Retrieve protein sequences". The experiments we have made until now suggest that the task descriptions are short and simple sentences not complex.

We consider each of the tasks of the workflow as a scene; for each scene, the casting director searches for an actor who can perform this task. This actor can be: 1) Preferred, 2) Available, or 3) Not available, as shown in Table II, and Table III. For each of the user tasks a worklet is selected. For example, when the user wants to perform Worklet ID=1, the actor that represents the "protein sequence retrieval worklet"' is the matching actor, this actor is set as the *preferred* one, and we select it based on its role.

In case of worklet ID=6 whose actor is not available, in other words there is no worklet that matches user request, in such case a new worklet is created and its category is transformed to the role attached to the actor that will represent this worklet. For each of these worklets, the Web service discovery process is carried out by searching in the BioCatalogue registry services that are annotated with the same category as the user defined tasks; This search retrieves a set of web services per user-defined task, and there is no way to know in advance which one is the most appropriate for the user-defined task. Using service ontology and domain ontology we retrieve a number of comparable services for each worklet as shown in Table IV.

We apply the selection algorithm in [21] on those services and select the best matching services for each task based on a simple user profile values that assume all QoS criteria are of equal importance. Thus, all criteria have the same weighing factor. We then present them to the user to continue with his workflow.

Table IV: Number of Matched Services without Considering QoS

ID	Number of Services
1	73
2	151
3	219
4	21
5	1132
6	122

Table V: Final matched services considering QoS

ID	Number of Services
1	2
2	5
3	4
4	1
5	6
6	4

Consequently, the proposed conceptual abstraction insulates the user from the difficulty of data format conversions and tool compatibility. However, this abstraction does not compromise power, and therefore provides both the novice and bioinformatics expert with an appropriate tool with significant

advantages compared to existing alternatives. In addition, the user is certain that the provided services are working correctly during run time. Table V presents the final number of matching services for each of the selected worklets.

VI. Conclusions and Future Work

In this work we provide new interface for workflow execution that simplify the user interaction with workflow systems. The success, of the proposed approach depends on the degree the community standardizes worklet interfaces. For future work, there are many ways in which this research could be directly extended; results might be improved if the user is given the option to perform more advanced searches by explicitly specifying keywords that represent the service or type descriptions. Also, we would like to perform user studies to assess how well the intent-declaration mechanisms work, and to determine if it increases user productivity, which is the ultimate goal. It is also possible that rigorous evaluations would help to identify which of the proposed features should be re-examined. If positive results are achieved, the logical next step would be to incorporate the idea described here into the Taverna workflow client, as this software is used by a relatively large number of scientists to perform actual life-sciences research. Also, we need to generalize the repository used to incorporate other bioinformatics registries.

References

[1] N. R. BP, "Basics for the construction of phylogenetic trees," *Webmed Central BIOLOGY*, p. 12, 2011.

[2] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, "Taverna: a tool for the composition and enactment of bioinformatics workflows." *Bioinformatics*, vol. 20, no. 17, pp. 45–54, 2004.

[3] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. 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*. Washington, DC, USA: IEEE Computer Society, 2004.

[4] S. Majithia, M. Shields, I. Taylor, and I. Wang, "Triana: A graphical web service composition and execution toolkit," in *Proceedings of the IEEE International Conference on Web Services*, ser. ICWS '04. Washington, DC, USA: IEEE Computer Society, 2004.

[5] C. A. Goble, K. Belhajjame, F. Tanoh, J. Bhagat, K. Wolstencroft, R. Stevens, E. Nzuobontane, H. McWilliam, T. Laurent, and R. Lopez, "Biocatalogue: A curated web service registry for the life science community," *Nature Precedings*, pp. 2–3, 2009.

[6] J. Cardoso, "Quality of service for workflows and web service processes," *Web Semantics Science Services and Agents on the World Wide Web*, vol. 1, no. 3, pp. 281–308, 2004.

[7] K. K. Verdi, H. J. Ellis, and M. R. Gryk, "Conceptual-level workflow modeling of scientific experiments using nmr as a case study." *BMC Bioinformatics*, vol. 8, p. 31, 2007.

[8] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web service modeling ontology," *Appl. Ontol.*, vol. 1, pp. 77–106, January 2005.

[9] M. DiBernardo, R. Pottinger, and M. Wilkinson, "Semi-automatic web service composition for the life sciences using the biomoby semantic web framework," *J. of Biomedical Informatics*, vol. 41, pp. 837–847, October 2008.

[10] L. Zhang, Y. Wang, P. Xuan, A. Duvall, J. Lowe, Y. Wang, A. Subramanian, P. Srimani, F. Luo, and Y. Duan, "Sesame: A new bioinformatics semantic workflow design system," in *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, Dec 2013, pp. 504–508.

[11] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim, "Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows," in *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, ser. IAAI'07, 2007, pp. 1767–1774.

[12] E. Pignotti, P. Edwards, A. Preece, N. Gotts, and G. Polhill, "Enhancing workflow with a semantic description of scientific intent," in *The Semantic Web: Research and Applications*, ser. Lecture Notes in Computer Science, S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds. Springer Berlin Heidelberg, 2008, vol. 5021, pp. 644–658.

[13] U. Srivastava, K. Munagala, J. Widom, and R. Motwani, "Query optimization over web services," in *Proceedings of the 32nd international conference on Very large data bases*, ser. VLDB '06. VLDB Endowment, 2006, pp. 355–366.

[14] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 1069–1075.

[15] M. J. Adams, A. H. ter Hofstede, D. Edmond, and W. M. van der Aalst, "Worklets: A service-oriented implementation of dynamic flexibility in workflows," in *the 14th International Conference on Cooperative Information Systems (CoopIS'06)*, R. Meersman and Z. Tari, Eds. Montpellier, France: Springer, 2006, pp. 291–308.

[16] N. Hashmi, S. Lee, and M. P. Cummings, "Abstracting workflows: Unifying bioinformatics task conceptualization and specification through semantic web services," *W3C Workshop on Semantic Web for Life Sciences*, 2004.

[17] P. Romano, D. Marra, and L. Milanesi, "Web services and workflow management for biological resources." *BMC Bioinformatics*, vol. 6, p. 24, 2005.

[18] K. Wolstencroft, P. Alper, D. Hull, C. Wroe, P. W. Lord, R. D. Stevens, and C. A. Goble, "The mygrid ontology: bioinformatics service discovery," *Int. J. Bioinformatics Res. Appl.*, vol. 3, pp. 303–325, September 2007.

[19] M. R. Rodrigues and M. Luck, "Evaluating dynamic services in bioinformatics," *Cooperative Information Agents X*, vol. 4149, pp. 183–197, September 2006, lecture Notes in Artificial Intelligence.

[20] K. Xu, Q. Yu, Q. Liu, J. Zhang, and A. Bouguettaya, "Web service management system for bioinformatics research: a case study," *Service Oriented Computing and Applications*, vol. 5, pp. 1–15, 2011.

[21] W. Nagy, H. M. Mokhtar, and A. El-Bastawissy, "A flexible tool for web service selection," in *Proceedings of the Fifth International Conference on Intelligent Computing and Information Systems*, Ain Shames Univ, Cairo, Egypt, 2011.

[22] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the challenges of scientific workflows," *Computer*, vol. 40, pp. 24–32, December 2007.

[23] I. H. C. Wassink, E. M. A. G. van Dijk, J. Zwiers, A. Nijholt, J. Kuipers, and A. O. Brugman, "Bringing hollywood to the driving school: Dynamic scenario generation in simulations and games," in *INTETAIN*, 2005, pp. 288–292.

[24] I. Wassink, H. Rauwerda, P. E. van der Vet, T. M. Breit, and A. Nijholt, "E-bioflow: Different perspectives on scientific workflows," in *Bioinformatics Research and Development, BIRD 2008, Vienna, Austria*, ser. Communications in Computer and Information Science, M. Elloumi, J. Küng, M. Linial, R. F. Murphy, K. Schneider, and C. Toma, Eds., vol. 13. Berlin: Springer Verlag, 2008, p. 15.

[25] P. Barthelmess and J. Wainer, "Workflow systems: a few definitions and a few suggestions," in *Proceedings of conference on Organizational computing systems*, ser. COCS '95. New York, NY, USA: ACM, 1995, pp. 138–147.

[26] I. Marin, "Ancient origin of the parkinson disease gene," *Journal of Molecular Evolution*, vol. 67, pp. 41–50, 2008.