Bioinformatics scientific workflows: combining databases, AI and Web services

Luciano Antonio Digiampietri[1], **João Carlos Setubal**[2], **Claudia Bauzer Medeiros**[1]

[1]Institute of Computing – University of Campinas (UNICAMP)
CP 6176, 13084-971 – Campinas – SP – Brazil

[2]Virginia Bioinformatics Institute, Virginia Tech, Bioinformatics 1,
Box 0477, Blacksburg, VA, 24060, USA

{luciano,cmbm}@ic.unicamp.br, setubal@vbi.vt.edu

Abstract. *Bioinformatics activities present new challenges, such as how to exchange and reuse successful experimental procedures, tools and data, and how to understand and provide interoperability among data and tools across different sites, for distinct user profiles. This thesis is an effort towards these directions. It is based on combining research on databases, AI and scientific workflows, on the Semantic Web, to design, reuse, annotate and document bioinformatics experiments. The resulting framework allows the integration of heterogeneous data and tools and the design of experiments as scientific workflows, which are stored in databases. Moreover, it takes advantage of the notion of planning in AI to support automatic or interactive composition of tasks. These ideas are being implemented in a prototype and validated on real bioinformatics data.*

1. Introduction

Scientific workflows [19] are being increasingly adopted as a means to specify and coordinate the execution of experiments that involve participants in distinct sites. Such workflows allow the representation and support of complex tasks that use heterogeneous data and software [2]. They differ from business workflows in several points. In particular, in bioinformatics they are characterized by a high degree of human intervention and variability in workflow design for the same task. Because bioinformatics is still a new area, there is not a well-defined consensus about how the tasks must be executed and how the results must be annotated [9].

Scientific workflows are usually designed manually. Manual composition is a hard work and susceptible to errors. Furthermore, in bioinformatics, due to the constant evolution of the area and the combinatorial explosion of alternatives, there are just too many alternatives for workflow construction. Thus, there is a pressing need for means to help scientists to design appropriate workflows.

The main idea behind this thesis is that the problem of automatic or semi- automatic composition of workflow tasks can be seen as an Artificial Intelligence planning problem. Moreover, our use of annotations based on ontologies forms the basis for tracking data provenance [6].

The thesis attacks the problem of constructing and annotating scientific workflows, under the assumption that they are the basis for specifying and executing tasks in a distributed laboratory environment. Each activity within such a workflow can be executed either by invocation of a Web service or of another (sub-)workflow.

Our main contributions are thus: (i) proposing a solution to the problem of composition of services, combining results from AI and database systems, thereby helping design scientific workflows, while at the same time documenting design alternatives; (ii) using ontology repositories to enhance the semantics in automatic workflow construction and facilitate tracking data and procedure provenance; (iii) validating the proposal by means of a prototype for genome assembly and annotation.

Our implementation takes advantage of WOODSS (WOrk-flOw-based spatial Decision Support System) [11], a scientific workflow infrastructure developed at the University of Campinas, Brazil. Originally conceived for decision support in environmental planning, it has evolved to an extensible database-centered environment that supports specification, reuse and annotation of scientific workflows and their components.

This paper is organized as follows. Section 2 describes related work. Section 3 presents our approach for workflow design and execution. Section 4 outlines our case study in bioinformatics. Sections 5 and 6 contain conclusions and future work.

2. Related work and concepts

2.1. Workflows and Web services in bioinformatics

Bioinformatics activities are typically complex, involving interactions among several basic tasks, human intervention and access to heterogeneous data sources. Bioinformatics laboratories often use pipelines or scripts to help automate this process. Each experiment can be seen as a workflow designed by scientists to help their daily activities [14]. However, this practice has little flexibility, hampering the edition and reuse of these workflows. One solution is to use Web services, invoked by workflow activities, to execute some bioinformatics task. Even then, there are several problems involved in the construction of such workflows. Among them, this thesis is concerned with: (1) data and tool provenance; (2) tool/task composition, translated into a problem of Web service composition; and (3) mechanisms for finding the appropriate tools to execute a task.

2.2. Planning and Composition of Web services

Automatic composition of Web services is a recent trend to meet some of the challenges and problems mentioned in the previous section. Users should be able to specify “what” they desire from the composition (high level goals and actions), and the system supplies the “how” - the Web services to be used, how to interact with those services, etc. The process of composing the services must be transparent to the users, and the detailed descriptions of the composed services must be generated automatically by the system from the users’ specifications.

The task of presenting a sequence of actions to achieve an objective is called in Artificial Intelligence *plan synthesis*, or *planning* [16]. Such techniques are currently used in mobile robots, manufacturing processes, satellite control, among others [1, 13].

Recent research efforts have investigated the use of planning to solve the problem of automatic composition of Web services [7]. According to [17], in order to use planning in the automatic composition of Web services, AI planning concepts must be extended to consider the following characteristics: plans need complex control structures with loops, non-determinism and conditionals; plans must support a complex object structure; and plans can produce new objects at execution time.

We highlight other important characteristics, not usually found in AI planning, such as: the use of non-functional attributes, like cost or quality, which can facilitate the choice of the plan most adequate to the user's needs; need to support semantic constructions such as hierarchies (abstractions), as well as compatibility with the different Semantic Web service description standards, such as OWL-S (www.daml.org/services); concurrency in service access and scalability.

Many planning systems and algorithms can be considered to compose Web services. For instance, there are solutions based on situation calculus, formal languages, rule based planning, symbolic model checking and hierarchical planning [10, 16, 18]

For a detailed comparison of these and other techniques, the reader is referred to [4], where we justify our choice of *hierarchical planning* to support (semi-)automatic specification of scientific workflows. Hierarchical planning is an AI planning methodology that creates plans by task decomposition. One well-known hierarchical planner is SHOP2 (Simple Hierarchical Ordered Planner 2) [12] which is based on Hierarchical Task Network(HTN) [16].

Our approach extends the current planners by treating complex objects and objects created dynamically, two very important characteristics within Web services. Moreover, planning algorithms do not consider the existence of relationships among objects which might result in plan improvement. We solve these issues by extending SHOP2 to take advantage of ontology repositories [4].

3. Our approach to workflow management

This section outlines our general architecture for composition of workflows and Web services. It relies heavily in the notion of repositories that store workflows, ontologies and an extended UDDI catalog. Part of this architecture has already been implemented.

3.1. Repositories

Our workflow design and execution process is based on combining AI planning techniques with information stored in three repositories: *Ontology Repository*, *Service Catalog* and *Workflow Repository*. While the Ontology Repository contains information about domains and service types, the Service Catalog stores information about service instances.

In more detail, the *Ontology Repository* contains two ontologies (Domain and Service) that will be used to support automatic composition and annotation of services and workflows – in our case study, information about genome assembly and annotation. The concepts in the Domain Ontology describe a given application domain. The concepts in the Service Ontology describe the different kinds of services and their relationships. It is used in automatic composition to help check compatibility among composed tasks (e.g., interface matching). The Service Ontology does not store descriptions of the services themselves. Rather, it contains what we choose to call "service type"- i.e., a generic description of each kind of service, its generic interface, parameters, etc. Thus, it will contain a description of a "printer" service, but no instantiation of printers. Service instantiation is left to the Service Catalog.

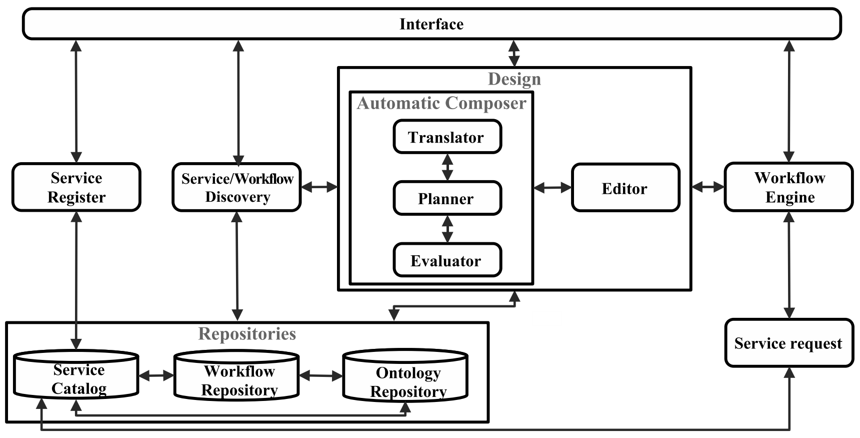

Figure 1. System Architecture

The *Service Catalog* plays the role of a UDDI (Universal Description Discovery & Integration) registry, enhanced with extended functionalities. Standard UDDI structures store information about service providers, the Web services they make available, and the technical interfaces which may be used to access those services. Our Service Catalog, besides, stores a service's non-functional attributes, such as execution time, quality, reliability and availability, used to rank composition solutions. Each service entry in the Catalog is annotated with the ontologic concepts of the Service Ontology.

The *Workflow Repository*, adopted from WOODSS, stores annotated (sub)workflows at different abstraction levels. The two main levels are abstract and concrete workflows [11]. The abstract level corresponds to a workflow specification (i.e., the result of a workflow design activity). Each specification can correspond to several concrete workflows. The concrete level corresponds to an instance that can be executed, and where each activity is instantiated with the indication of the appropriate Web service, and all of its input and output parameters. The Workflow Repository also stores all annotations and information on data needed to run a given concrete workflow. This includes pointers to files that store intermediate execution results and metadata associated with each execution (e.g., timestamps, actors involved).

The repositories are interrelated as follows. Workflows and data in the Workflow Repository are annotated with terms from both ontologies; the services invoked within concrete workflows come from the Service Catalog. Moreover, concepts of the Service Ontology are used to annotate terms in the Service Catalog; and concepts in the Domain Ontology annotate types stored in the Service Ontology. For an example of these interrelationships, the reader is referred to [4].

3.2. The composition architecture

Our architecture is able to deal with automatic composition of workflows based in Web services. Figure 1 shows this architecture, highlighting the main modules and their interactions. It combines basic features of a planning framework (e.g., such as that in [15]) with the scientific workflow framework of WOODSS [11], adding catalog and user interaction facilities. Whereas WOODSS is based on manual composition, our architecture supports automatic and semi-automatic composition via planning.

The Interface Layer allows the user to design, search, edit and execute a workflow. It also allows a user to register services and workflows, request the execution of a workflow and interact with this execution. It also supports syntax verification and suggestions of activities; automatic specification through AI planning; and iterative composition.

The Service/Workflow Discovery module is responsible for the search of services and workflows that meet user requests. Search can be based on functionality, context and syntax. Search for functionality is based on keywords, and can be local (Repositories) and global (on the Web). Whenever the global search returns a service that is not already registered in the Service Catalog, the user is required to annotate and register it in the local Repository, thus vouching for its quality. Search for context is based on ontologic annotations of services. Search on syntactic compatibility is based on the parameters of the services' interfaces. When no stored service or workflow meets the requests, this module will ask the Design module to create new workflows.

The Design module is responsible for constructing a workflow, which at any time can be edited by a scientist (the Editor box). The Automatic Composer encapsulates the Translator-Planner-Evaluator modules of AI planning. It receives a plan request R from the interface and generates workflows automatically or semi-automatically. To generate these workflows, the Translator needs first to convert the request to the planner's language. Next, the Planner interacts with the Workflow and the Ontology Repositories to obtain information for plan generation, and with the Service/Workflow Discovery facility to check for existing available services.

Rather than generating concrete executable workflows, our planner produces abstract workflow specifications. The reason is that plans refer to *service types* (defined in the Ontology Repository), rather than the services themselves (whose specification is stored in the Service Catalog). This choice was made mainly to improve efficiency and scalability in the planner. The Evaluator converts these abstract workflows into concrete (executable) workflows and chooses among them the workflow that best suits the request R. This selection is based on non-functional attributes (execution time, quality, reliability, etc) and can be guided by the user.

The Editor module has two main roles: workflow design and ontology update. It accesses the workflow repository and lets the user manually compose, reuse and annotate workflows. Annotations include free text and references to the ontology repository.

The user interacts with the Service Register module in order to define new services. These services are described in WSDL and OWL-S, and thus linked to the Ontology Repository. These services can be those developed and available locally, or those which are available elsewhere, but whose provenance has been certified by the user.

3.3. Execution environment

Execution is supported by the Workflow Engine, and allows user interaction, e.g., to validate or interrupt execution flow. The Engine module follows the specification of [3]. It is responsible for controlling the execution of all workflow activities. An activity can be a simple Web service or a complex workflow. The operations provided by the Workflow Engine are: interpretation of the complex process definitions; creation and management of the process instances; and supervisor and management functions [3]. The module sends the requests (and parameters) for service invocation to Service Request.

The Service Request module is responsible for the management of each Web service request, communicating with the Web server provider, sending input data and receiving the results. This module also detects service faults (e.g., non-availability). There are three alternatives to try to solve a fault. The first is try to re-execute the service that presented the fault. Since references to all data produced by the execution of a workflow are stored in the Workflow Repository, the system can try to re-execute the workflow starting from the point where the fault appeared. This is useful when a service is unavailable during a short period of time. The second alternative is to replace the faulty service by an equivalent service (of the same type), and to continue the execution of the workflow. In this case, the Workflow Engine module can ask the Automatic Composer module for the generation of an alternative plan that replaces the faulty service by an equivalent service or by a new workflow. If these alternatives do not work, the Workflow Engine can request new plans to solve the problem. These plans will be produced considering that all data already generated can be used to facilitate or optimize the creation of new workflows.

This architecture supports the three main kinds of composition: manual, iterative and automatic. In manual composition, the system will only let a user combine two activities if their inputs and outputs are ontologically compatible. Ontological compatibility is based on subsumption properties (see [5]). In iterative composition, for each iteration, the system suggests to the user activities or sub-workflows that have been previously stored in the repository and that can be used for an already defined task. In automatic composition, it designs a set of workflows (solutions) that satisfy the request provided by the user.

3.4. User interaction

Users can interact with the architecture to annotate and design workflows, monitor and change their execution, edit ontologies and register services. An important user interaction is the request for a plan (i.e., the construction of a new workflow). This process starts when a user (human or software agent) makes a request for a service. This request can be the description of a goal or task. Starting from it, the Planner generates alternative plans to meet the request. The planner accesses the domain and service ontologies to obtain the necessary information for the planning process. Once the plans are generated, they are passed on to the Evaluator, which chooses the best plan to meet user needs.

Domain and service ontologies, stored in the ontology bases, are key concepts to this process. Initially, they are used by the Translator to generate a request to the planner, disambiguating the user's demand for a service. Next, these bases are used by the Planner to generate the appropriate service compositions. The Planner accesses them to obtain the functionalities of the services and generates an abstract scientific workflow. The Planner uses the domain ontology to improve the efficiency of the planning process and to facilitate the modeling and the management of complex objects. The planner's output contains several workflows (the plans) with equivalent or similar functionalities.

4. Case study: bioinformatics problems

We implemented a prototype of our architecture to solve two important bioinformatics problems: genome assembly and annotation. Both problems can be solved by invoking a sequence of specialized tools, already available at several sites, in distinct flavors. The issue is to construct the appropriate workflow which, given a set of input files, e.g., containing DNA sequences, produces annotated alignments.

This requires expert knowledge (e.g., to choose tools among alternatives) and domain knowledge (captured by ontologies). We have specified such an ontology which extends TAMBIS (www.tambis.org). We have also created an annotated service catalog, containing information about the most common bioinformatics tools on the Web. Given all this information, the planner (constructed by extending SHOP2) supports automatic or manual workflow design. For more details on our case study the reader is referred to [4].

5. Concluding Remarks

This thesis presents a solution to the problem of specifying a scientific workflow to execute on the Web. Our main contributions lie in proposing and prototyping a framework that takes advantage of AI planning techniques, combined with ontologies and Semantic Web standards to support workflow design. The solution is based on repositories that store information on services and their characteristics, on service and domain ontologies, and on workflows. In particular, ontology repositories are extensively used in enhancing plan generation with semantics and in helping users design better scientific workflows.

Our architecture is generic, and can be instantiated for several domains. It helps the user in the three kinds of composition: manual, iterative and automatic. Manual composition is useful when the user knows exactly what activities he/she desires to compose. Iterative composition is advisable when the user has a general knowledge of the process that he/she wants to execute, but does not know what tools/services execute this process. In this case, the system suggests the activities. Automatic composition is advisable when the user knows pre- and post-conditions, but does not know (or is not interested in) how to design a workflow that satisfies these conditions.

We have built a first prototype to verify and validate our proposal, for bioinformatics problems, specifically for genome assembly and annotation. Several bioinformatics laboratories have reported the use of scientific workflows and of a workflow infrastructure to support their experiments (e.g. [8],[14]). Our work extends these approaches in three main directions: first, use of AI planning techniques to help find the "best" workflow for a task; second, the use of ontologies to semantically support workflow construction; third, the use of these ontologies in annotation and thus help traceability.

6. Present stage of the work

Architecture and workflow model have been specified. Present implementation supports manual and automatic workflow design, where the planner extends SHOP2. Ontologies were specified in OWL using Protégé. Ongoing work concerns mechanisms for improving provenance and traceability support. The next steps are related to workflow execution. For that, we intend to use some existing workflow engine.

Acknowledgments This thesis was partially financed by FAPESP, CAPES, CNPq and Microsoft Research. This work was developed in cooperation with J. J. Pérez-Alcázar from EACH, USP.

References

[1] H. M. Avila, D. W. Aha, D. Nau, R. Weber, L. Breslow, and F. Yaman. SiN: Integrating case-based reasoning with task decomposition. In *IJCAI 2001*, pages 999–1004, 2001.

[2] M. C. Cavalcanti, R. Targino, F. B. ao, S. C. Rössle, P. M. Bisch, P. F. Pires, M. L. M. Campos, and M. Mattoso. Managing structural genomic workflows using Web services. *Data & Knowledge Engineering*, 53(1):45–74, 2005.

[3] T. W. M. Coalition. Workflow management coalition terminology & glossary (issue 3.0) 1999. http://www.wfmc.org/standards/docs (as of 2005-09-20).

[4] L. Digiampietri, J. Pérez-Alcazar, C. Medeiros, and J. Setubal. A framework based on semantic Web services and AI planning for the management of bioinformatics scientific workflows. Technical Report IC-06-004.

[5] R. Fileto. *The POESIA Approach for Services and Data Integration On the Semantic Web*. PhD thesis, IC–UNICAMP, Campinas–SP, 2003.

[6] R. Fileto, C. B. Medeiros, L. Liu, C. Pu, and E. Assad. Using Domain Ontologies to help Track Data Provenance. In *Brazilian Database Conference*, pages 84–98, 2003.

[7] *Workshop on Planning and Scheduling for Web and Grid Services*, June 2004.

[8] Laboratory for Bioinformatics, Institute of Computing, University of Campinas. http://www.lbi.ic.unicamp.br (as of 2006-05-02).

[9] P. Lord, S. Bechhofer, M. D. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *3rd Intl Semantic Web Conference*, pages 350–364, 2004.

[10] S. A. McIlraith and T. C. Son. Adapting Golog for Composition of Semantic Web Services. In *KR2002*, pages 482–493, 2002.

[11] C. B. Medeiros, J. Perez-Alcazar, L. Digiampietri, G. Pastorello, A. Santanche, R. Torres, E. Madeira, and E. Bacarin. WOODSS and the Web: Annotating and Reusing Scientific Workflow. *ACM SIGMOD Record*, 34(3):18–23, 2005.

[12] D. Nau, T. C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[13] D. Nau, S. Gupta, and W. Regli. Artificial Intelligence Planning versus manufacturing-operation planning: a case study. In *IJCAI 1995*, pages 1670–1676, 1995.

[14] 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*, 20:3045–3054, 2004.

[15] J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. In *SWSWPC 2004*, volume 3387, pages 43–54, 2004.

[16] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[17] B. Srivastava and J. Koehler. Web Service Composition - Current Solutions and Open Problems. In *ICAPS 2003*, pages 28–35, June 2003.

[18] P. Traverso and M. Pistore. Automated Composition of Semantic Web Services into Executable Processes. *Lecture Notes in Computer Science*, 3298:380–394, 2004.

[19] J. Wainer, M. Weske, G. Vossen, and C. B. Medeiros. Scientific Workflow Systems. In *NSF Workshop on Workflow and Process Automation Information Systems*, 1996.