E-Biology Workflows with Calvin

Markus Held, Wolfgang Blochinger and Moritz Werning

E-Biology Workflows with CALVIN

Markus Held[1], Wolfgang Blochinger[2], and Moritz Werning[3]

[1] Symbolic Computation Group, Eberhard Karls University
Sand 14, 72076 Tübingen, Germany
mheld@informatik.uni-tuebingen.de
[2] Institute of Parallel and Distributed Systems, University of Stuttgart
Universitätsstr. 38, 70569 Stuttgart, Germany
wolfgang.blochinger@ipvs.uni-stuttgart.de
[3] SUM IT AG
Täfernstr. 28, 5405 Baden-Dättwil, Switzerland
moritz.werning@sumit.ch

Abstract. Web portals enable sharing, execution and monitoring of scientific workflows, but usually depend on external development systems, with notations, which strive to support general workflows, but are still too complex for every-day use by biologists. The distinction between web-based and non-web based tools is likely to further irritate users. We extend our work on collaborative workflow design, by introducing a web-based scientific workflow system, that enables easy-to-use semantic service composition with a domain specific workflow notation.

1 Introduction

Biologists usually ”connect” web applications or services by cutting and pasting data. As this is error-prone and hard to retrace, many different bioinformatics workflow systems have emerged [1]. Increasingly web portals are used to access and to share scientific workflows. Some provide workflow construction facilities via Java Web Start, while true web-based workflow tools lack sophisticated user experience. Often, desktop-based scientific workflow systems with complex user interfaces are used.

Supporting a large set of different workflows and different types of services increases the complexity of a notation, thus reducing usability by domain experts. In contrast, high user-friendliness limits the possible set of workflow patterns and access to arbitrary services. Domain experts should be able to compose and execute workflows in a domain specific modeling system, and fall back to a collaboration with software engineers if a more complex workflow model is needed. We propose a hybrid approach of augmenting a collaborative workflow design tool with a biology-specific workflow system, where a reduced workflow notation is compiled to the Business Process Execution Language (BPEL) [2]. In this paper, we formalize e-biology processes, deriving requirements for workflow systems, and show, how low-level workflow languages can be combined with domain-specific workflow notations to provide maximum ease-of-use and flexibility. We present the workflow system CALVIN[4], which abstracts from many of the intricacies of workflow design, builds on similar technology as and augments our collaborative workflow design system HOBBES [3].

[4] See http://www-sr.informatik.uni-tuebingen.de/workflows for demos.

2 Requirements Analysis

We have held meetings and interviews with research staff from the Tübingen Center for Plant Molecular Biology (ZMBP[5]) to evaluate their requirements on a workflow system, and have given demonstrations of our evolving workflow system. At the ZMBP, ”in silico experiments” are frequently used to prepare ”wet” experiments in the laboratory. Existing workflow solutions have been evaluated but failed to satisfy the users, due to intricate user interfaces and lack of user interaction.

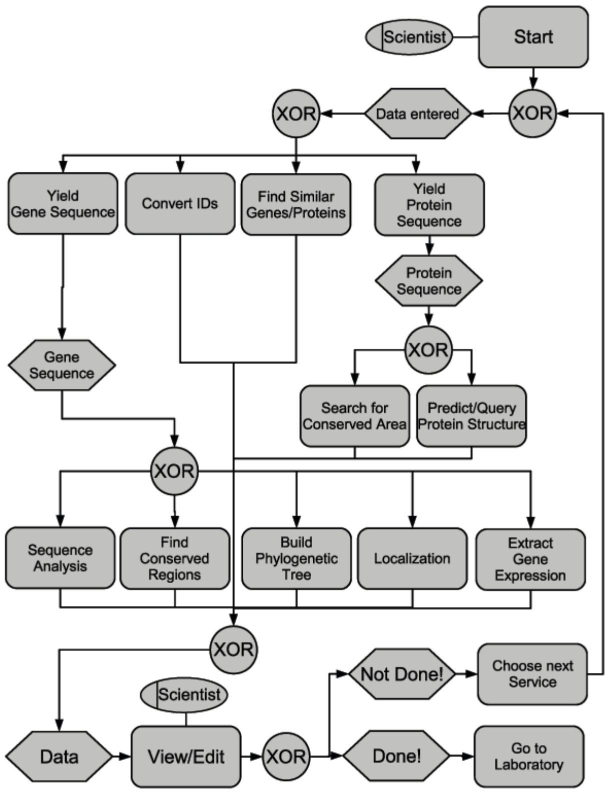

Fig. 1. The Meta-Process of in silico Microbiology in Plant Science.

Figure [1] represents a *Meta*-Process of computational plant science, given in the Event-Driven Process Chain notation. We have derived the Meta-Process from our interviews at the ZMBP, and informal drawings[6] from ZMBP staff members. The Figure does not represent a specific bioinformatics workflow, but outlines the daily usage of web applications by biologists. Every path between the events ”Data entered” and

[5] `http://www.zmbp.uni-tuebingen.de`, last accessed: 22/09/2008.
[6] `http://www-sr.informatik.uni-tuebingen.de/workflows/draw.htm`.

”Data” represents a pipeline of data processing. The ”view/edit” activity triggers either the execution of a new pipeline or the end of the in silico experimenting phase. In many cycles, the ”view/edit” activity can effectively be left out, and is only enforced by the ”copy & paste” mode of operation. In some cycles it is necessary to enable a biologist to edit the data or decide, whether to go on with the workflow. All input/output data has to be recorded for tracing the experiment. Effectively, the set of cycles performed by a biologist before going on to a ”wet” experiment, forms a tree, where the result of one in silico experiment is the input to a set of other in silico experiments.

2.1 Requirements

Collaboration `[R1]` *Gil* et al. see a need to ”orchestrate the steps of scientific discovery and bridge the differing expertise of collaboration members” [4]. E-biology collaborations consist of biologists, bioinformaticians, and software engineers. Biologists will prefer ”stripped down” workflow notations, that enable them to easily sketch *runnable* workflows, while software engineers need a complete workflow language.

Reproducibility `[R2]` Reproducibility of scientific analyses is needed as well [4]. In ”copy & paste” processes biologists see all results of all service invocations. There is no guarantee, that a service will always give the correct answers, as databases can be corrupted or services be subject to software errors. Hence, each workflow result will have to document the actual control-flow and data-flow of the workflow instance, i.e. the results of every activity in the workflow. Thus, biologists need an interface to easily navigate through intermediary and final results.

Intuitive Composition `[R3]` Biology curricula do not encompass programming, so biologists strongly differ in their computing skills. Workflow notations for biologists have to be kept very simple and may only provide necessary features, leaving out control structures where possible. [5] point outs, that ”wet laboratory biologists are uncomfortable using even the most abstract workflow tools currently available”.

Service Invocations at Build Time `[R4]` Users may want to invoke services at build time. First, this enables an exploratory way of workflow construction. As [6] points out, users sometimes will start composing a workflow with a given piece of information without knowing the goal. Second, biology web services are provided by research institutions, that cannot guarantee the correctness or the availability of a service. Thus, testing a service enhances the chance of yielding a correct workflow for a given task.

Semantic Service Discovery `[R5]` When constructing a workflow, a typical task is to find a service, that can serve as a successor for a given activity. The opposite scenario can also be the case, where a predecessor for an activity is needed, e.g. if a user is searching for a way to get to a specific result type.

User Interaction `[R6]` It can be necessary to interact with a workflow at predefined events in the workflow. This matches the ”view/edit” activity from the meta-process. In data-flow oriented models, these events can be identified as the invocation or return of activities. After viewing the input/output of an activity, the user may decide either to resume the workflow, to alter the data and resume or to quit the workflow execution.

Web Integration `[R7]` The web browser is the standard e-science tool for biologists. Users are most likely to accept a new system if it is integrated into the environment they are used to. Even if department policies allow installing new software, many biologists prefer using pre-configured and centrally managed systems.

Standards Conformance `[R8]` Scientific workflows lack a common standard. To ensure that a workflow can be understood and used by future users, it has to conform to a standard language, e.g. BPEL, as [5] recommends.

3 The CALVIN Life Science Workflow System

We now introduce the CALVIN System, which enables biologists to easily define workflows for common tasks in a simplified workflow notation. If more sophisticated workflows are needed, biologists can request the help of software engineers or bioinformaticians `[R1]`, to refine the workflow via a BPEL editor, e.g. HOBBES [3].

CALVIN can be accessed on the web `[R7]`. The main screen consists of an editor canvas, showing the activities and data-flow connections of the workflow. The workflow notation is kept very simple and effectively represents a tree of exploration steps so users do not have to deal with control structures `[R3]`. Services that process the output type or produce the input type of an activity can be found using a semantic search facility. By clicking on an activity, a query window is opened, that can be used to search the BioMOBY database for a list of adequate services `[R5]` [7][8]. Users can open a description of the service represented by an activity, or invoke the service directly from the activity properties menu. This enables testing the behavior of a service prior to running the workflow, thus enabling an explorative and data-oriented way of workflow composition `[R4]`. Activities can be marked as requiring user interaction, before or after their execution `[R6]`. Workflows are compiled to BPEL and deployed to an execution engine. We have chosen BPEL, as it represents a commonly accepted standard `[R8]`. Deployed CALVIN workflows can be re-used as new services. Outputs of a CALVIN service can serve as input of other activities.

The CALVIN Workflow Management Console presents a list of the workflows that are available on the server and can be loaded into the editor. The workflows are already deployed in the BPEL execution engine and can be executed from the console. Before beginning the actual execution, a window will show a graphical editor for workflow inputs. The input variables are sorted by the activities that they are provided to. Two different kinds of input parameters are available, primary inputs, which consist of biological data, and secondary parameters that influence the behavior of a service (e.g. its output format). Standard parameters are provided, that can be overwritten by the user. Every input variable is presented with a description provided by the BioMOBY database. During workflow interaction, the user can be prompted to verify if the workflow should go on, or to view and edit result data `[R6]`. External clients can be developed using the WSDL file provided by a workflow. Those WSDL-files are annotated with documentation data from the BioMoby database, that yield a complete description of the semantics of each service invocation.

Every workflow execution yields a result file containing the time of invocation, the inputs and service parameters, and all final and intermediate results. Together with the serialized workflow model, this enables retracing every event in the workflow and the origin of its results `[R2]`. The result files can either be downloaded from the server or be explored using an AJAX based provenance browser.

Compilation to BPEL enables the development of arbitrary workflows beyond the limits of any domain-specific workflow language. Leaving the WSDL interface of a BPEL document unchanged is a sufficient condition for the invocation of the workflow from CALVIN. A mapping from BPEL to the CALVIN notation is impossible, since BPEL is Turing-complete. It is not *necessecary*, on the other hand, since modified CALVIN workflows can be embedded in new CALVIN workflows. As sophisticated workflow development demands a close cooperation of domain experts and software engineers, HOBBES enables teams to collaboratively edit a BPEL model `[R1]`.

3.1 Example: Searching for Orthologous Gene Sequences

Figure 2 shows a workflow for searching for orthologous gene sequences, a common case in genomics. Genes from two species are called orthologous, if they are divergent copies of a single gene in a common ancestor species. The input is a list of sequence identifiers, which first are converted to the needed ID format in step ➊ and then yield the according sequences in step ➋. The sequences are subject to three different BLAST searches (➌, ➍, ➎) to query different databases with different parameters at the same time. Before that, two sequence conversions are necessary (➏, ➐). Afterward, sequence identifiers are extracted from the results, which can be used for further processing.

3.2 Representation of Biology Specific Workflow Models

The CALVIN workflow notation represents a tree of exploration steps, which has to be transformed to a complete workflow. This implies, that not all workflows can be expressed via CALVIN, and that workflows have to be automatically augmented by the system. During the transformation process, the data-flow links are compiled to control-flow links and variable assignments. Additional data-flows augment the tree to a directed acyclic graph, that collects all produced data elements. Workflow models as viewed by biologists are represented by a tree $T = (N, E), E \subset N \times N$ with root n_1, where N is a set of BioMoby activities. CALVIN compiles the tree model T to a complete workflow model W, by adding a start activity α and a final activity ω, a set of control flow edges C, a set of data flow edges D which connect activity inputs and outputs, and a set of parameter edges P:

$$\begin{aligned}
W &= (N \cup \{\alpha, \omega\}, C, D, P), \;\; \alpha \notin N \wedge \omega \notin N \\
C &= E \cup \{(n, \omega) | \forall k \in N : (n, k) \notin E\} \cup \{(\alpha, n_1)\} \\
D &= E \cup \{(\alpha, n_1)\} \cup N \times \{\omega\} \\
P &= \{(\alpha, n) \in \{\alpha\} \times N, \; where\, n\, has\, parameters\}
\end{aligned}$$

W consists of three graphs defined on the same node set $N \cup \{\alpha, \omega\}$. α takes the client's input and serves parameters to those activities, which have parameters. ω takes

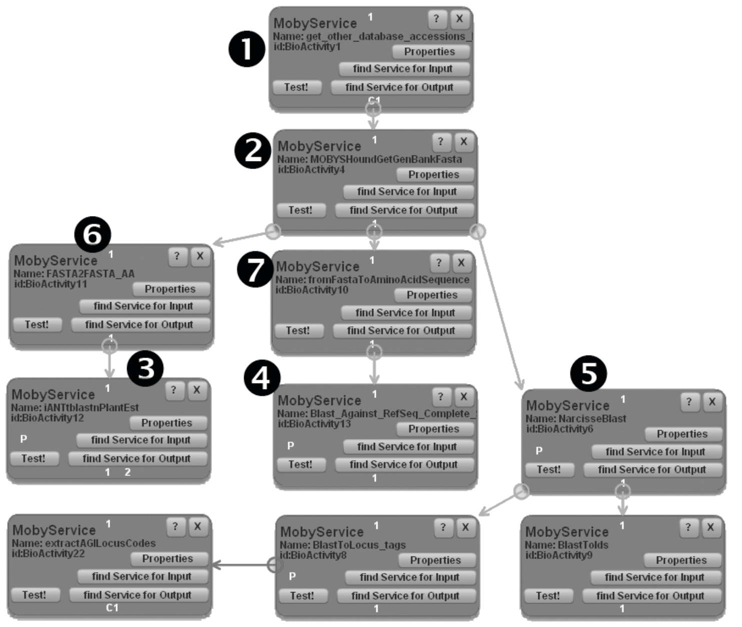

Fig. 2. Search for Orthologous Gene Sequences.

the output of all activities and sends the answer message. The interface of a workflow is given as a Tuple $\tau = \big(\iota, o(n_1), \ldots, o(n_{|N|})\big)$, where ι is the BioMoby type of its input message, and the $o(n_i)$ denote those message parts of its output message, which accord to the output of an activity. An $o(n_i)$ is a tuple of a message part name and a description of its BioMoby type. If n_j represents a CALVIN service, it yields a tuple $(o(n_{j,1}), \ldots o(n_{j,k}))$, which is added to τ instead of a single $o(n_j)$ Outputs of CALVIN services are treated as virtual nodes and can be used as input sources for other activities.

4 Architecture and Implementation

4.1 The CALVIN Architecture

The HOBBES and CALVIN clients have been implemented in Adobe Flex[7], which is based on Flash. Flex supports declarative GUI design in an XML language (MXML) with mapping to the Actionscript language. Its communication facilities support HTTP requests and can be enhanced with optional server side components called ”Lifecycle Data Services” (LCDS), which enable server-to-client notification. Client/server Communication in CALVIN is facilitated via LCDS *Remote Objects*, which map client-side Actionscript objects to server side Java objects. For every session, one *BioController* object holds a server-side workflow model. For communication with the *ActiveBPEL*

[7] `http://www.adobe.com/devnet/flex/`, last accessed: 24/10/2007.

engine, one *BPEL Engine Controller* is instantiated per session. The *MOBY Manager* remote object enables service queries via BioMOBY [7].

4.2 Transformation of CALVIN Workflows to BPEL Object Models

For each session, CALVIN holds a server-side and a more light-weight client-side object model, which communicate via transfer objects. When compiling and deploying a CALVIN workflow, it is transformed to a HOBBES BPEL Object Model (BOM), which is afterward compiled to a BPEL document. Before compiling the BPEL model, the BioMoby WSDL definitions have to be adapted by adding Partner Link Type definitions. CALVIN generates a WSDL definition for the process, including input and output messages whose message parts are annotated with BioMoby activity descriptions. The BOM is then generated by first importing all WSDL definitions and adding a global Flow activity with a Receive and a Reply activity. For every CALVIN activity, a Sequence activity is added, with an input and an output assignment, and a service invocation. Control flow edges are mapped to BPEL Flow Links, and data flow edges to Assign activities. If user interaction is required before or after the execution of an activity, CALVIN inserts code which polls a messaging singleton via XPath custom functions. The BOM is compiled to a BPEL file, which is saved to a temporary directory with WSDL files and other necessary files. A deployment script compresses the directory to an ActiveBPEL archive, which is then copied to the workflow deployment directory. After a CALVIN workflow has been deployed, it can be reused as an activity within other CALVIN workflows. Workflows can be reused as long as their WSDL definition is compatible with the CALVIN system.

5 Related Work

REMORA is a web-based DAG editor with BioMOBY access, where workflows are saved and executed on the server [9]. BioWMS contains a web-based workflow editor [10], presenting workflows as static HTML pages with embedded graphics, where changes lead to reloading the page. *Taverna* is a desktop-based workflow system for bioinformaticians [11], for composing and executing workflows, given in the Scufl language. No standardized Scufl specification exists and ”the language itself should be considered volatile”[8]. Lanzen and Oinn point out that ”in the standard version of the Taverna Workbench, a user cannot control the behaviour of a workflow once it is running” and have introduced an interaction extension with email notification [12].

6 Conclusion

We have presented the scientific workflow system CALVIN, which enables life scienctists to easily compose workflows. It has been implemented as a Rich Internet Application and supports semantic service composition. Workflows are compiled to the

[8] http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html, last accessed 30/09/2008.

standard language BPELand can be managed and executed on the server. In contrast to other systems, CALVIN targets biologists instead of bioinformaticians, and provides a user-friendly interface, where user interaction with workflows has been incorporated.

7 Acknowledgements

We would like to thank Dierk Wanke, Kenneth Berendzen, and Klaus Harter from the ZMBP for their helpful advice on processes in modern biology.

References

1. Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M.: Workflows for e-Science: Scientific Workflows for Grids. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
2. OASIS: Web Services Business Process Execution Language Version 2.0. `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf` (April 2007) last accessed 24/10/2007.
3. Held, M., Blochinger, W.: Structured collaborative workflow design. Future Generation Computer Systems **25**(6) (2009) 638–653
4. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the Challenges of Scientific Workflows. IEEE Computer **40**(12) (2007) 24–32
5. Barker, A., van Hemert, J.: Scientific Workflow: A Survey and Research Directions. In R. Wyrzykowski et al., ed.: Seventh International Conference on Parallel Processing and Applied Mathematics, Revised Selected Papers, volume 4967 of LNCS, Springer (2008) 746–753
6. Gibson, A., Gamble, M., Wolstencroft, K., Oinn, T., Goble, C.: The data playground: An intuitive workflow specification environment. In: E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, Washington, DC, USA, IEEE Computer Society (2007) 59–68
7. Wilkinson, M.D., Links, M.: BioMOBY: an open-source biological web services proposal. Briefings In Bioinformatics **3**(4) (2002) 331–341
8. The BioMoby Consortium: Interoperability with Moby 1.0 It's better than sharing your toothbrush! Briefings In Bioinformatics **9**(3) (2008)
9. Carrere, S., Gouzy, J.: REMORA: a pilot in the ocean of BioMoby web-services. Bioinformatics **22**(7) (2006) 900–901
10. Bartocci, E., Corradini, F., Merelli, E., Scortichini, L.: BioWMS: a web-based Workflow Management System for bioinformatics. BMC Bioinformatics **8 Suppl 1** (2007)
11. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics **20**(17) (2004) 3045–3054
12. Lanzén, A., Oinn, T.: The Taverna Interaction Service. Bioinformatics **24**(8) (2008) 1118–1120