Designing Workflows on the Fly Using e-BioFlow

Ingo Wassink[1,2], Matthijs Ooms[1], and Paul van der Vet[1,2,⋆]

[1] Human Media Interaction Group
University of Twente
Enschede, The Netherlands
[2] The Netherlands Bioinformatics Centre (NBIC)

Abstract. Life scientists use workflow systems for service orchestration to design their computer based experiments. These workflow systems require life scientists to design complete workflows before they can be run. Traditional workflow systems not support the explorative research approach life scientists prefer. In life science, it often happens that few steps are known in advance. Even if these steps are known, connecting these tasks still remains difficult.

We have extended the e-BioFlow workflow system with an ad-hoc editor to support on-the-fly workflow design. This ad-hoc editor enables an ad-hoc design of the workflow with no predetermined plan of the final workflow. Users can execute partial workflows and extend these workflows using intermediate results. The ad-hoc editor enables its users to explore data and tasks representing tools and web services, in order to debug the workflow and to optimise parameter settings. Furthermore, it guides its users to find and connect compatible tasks. The result is a new workflow editor that simplifies workflow design and that better fits the explorative research style life scientists prefer.

1 Introduction

Life scientists are used to work in an explorative research style, without having a clear hypothesis [1–4]. Data is used as a source of inspiration, and few steps are known in advance. A workflow system will better fit the life scientist's needs if it supports this explorative research style [5]. Current workflow systems separate the design and execution of the workflow, which has led to a trial-and-error approach in using them.

We have extended our workflow system, e-BioFlow [6], to an ad-hoc workflow system. An ad-hoc workflow system enables an ad-hoc workflow design, with a small or no predetermined plan of the final workflow [7]. e-BioFlow presents new interactions with workflow systems and supports the explorative research style life scientists prefer. Scientists can execute partial workflows. The data produced are explicitly present in the workflow model, can be inspected and used as sources of inspiration to decide on the next steps in the experiment. These data are available as input for new tasks or tasks already in the workflow. New

⋆ i.wassink@ewi.utwente.nl

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 470–484, 2009.

tasks can be inserted, connected to data produced by tasks in the workflow and executed in isolation. e-BioFlow simplifies workflow design, because it enables scientists to try things out and to insert tasks that may even be absent in the final workflow.

Even if the complete workflow model is known in advance, linking the parts is often difficult. Such problems are known as *plan composition problems* [8]. The real services to be used may be unknown and linking services often requires data conversion [9]. The ad-hoc editor will help the scientist to build the workflow. The scientist can run parts of the workflows and inspect intermediate results to test and debug the workflow, and to fine-tune parameter settings. The result of using the ad-hoc editor is a runnable workflow that can be stored as a generic file for future use. Due to e-BioFlow's support for late binding, it is independent of resources available at design time. Late binding means that tasks are abstracted from services until execution time. e-BioFlow can easily switch between alternative services without any change of the workflow model. Therefore, the workflow can be used as template for future experiments and shared with peers through web portals such as myExperiment [10, 11].

In this paper, we will first discuss the characteristics of an ad-hoc workflow editor. We will introduce our workflow system e-BioFlow. After that, e-BioFlow's ad-hoc workflow editor will be discussed. A use case will demonstrate the use of the ad-hoc workflow editor. Then, we will compare our approach to other systems that support ad-hoc workflow design. We will end with a discussion.

2 Ad-Hoc Editor: Characteristics

Although life scientists use data as sources of inspiration, workflow systems focus on tasks. The graph visualisation of the workflow consists of nodes representing the tasks and arrows representing the dependencies or data flows between the tasks. The data itself is absent and cannot be used to design the workflow. These workflow systems handle a *routine process-oriented mode*: the workflow needs to be designed in advance, before the workflow designer can run it [5]. Like in other visual programming environments, the workflow designer has to make many design choices without good data to direct his decisions [12]. This forces workflow designers to guess-ahead or to insert place-holders [8].

An ad-hoc workflow editor has characteristics of a traditional workflow editor, a workflow engine and a provenance system. It enables the workflow designer to execute partial workflows and extend them using the data produced by the tasks in the workflow that are already executed. It supports what Gibson et al. [5] call an *investigative data-oriented mode*.

Ad-hoc workflow systems have many advantages over traditional workflow interfaces:

- The tasks to be used are often unknown at design time. Tasks can be tried out in the ad-hoc editor.
- No need to know the complete workflow in advance, but extend and execute partial workflows.

- Speeds up of workflow design, because a small change in the workflow requires a rerun of just the tasks involved.
- Use intermediate results as sources of inspiration to decide on next steps of the workflow.
- No guess-ahead required about the data produced or consumed by the tasks.
- Fine-tune parameters and debug workflows by executing tasks in isolation.

Workflow systems have much in common with integrated development environments (IDE's) for visual programming languages and text-based programming languages. Most users of visual programming languages are not experts in programming and often do not want to be, but need to program for their daily working activities [4, 13], which is also true for most life scientists [14]. It is important that the visual language used matches the user's mental representation of the problem he wants to inspect [8, 12, 13]. The closer the programming world is to the problem world, the easier problem solving ought to be [8]. IDE's have implemented different techniques to help the programmer write correct program code through, among others, live editing, auto-completion and programming by demonstration. These three techniques are applicable to an ad-hoc workflow system as well. They will be explained in the context of workflow design.

The ad-hoc workflow editor explicitly presents the data to the workflow designer, which enables the designer to use these data to further design the workflow [2, 15]. The resulting environment supports what is called *live editing*, and is applied to textual programming languages [16]. A live editing environment supports explorative programming and gives programmers real-time feedback on the program's execution at edit time. The ad-hoc workflow system can be used as a live editing environment, but then to design workflows [15]. It enables the workflow designer to execute uncompleted workflows and gives feedback about the workflow's execution state by means of the data produced and consumed by tasks and about errors that may have occurred. In case of an error, the workflow designer can use the feedback to correct the workflow. In case of a successfully executed task, he can use the data produced to further design the workflow.

When data and input and output ports of tasks are syntactically and semantically typed, type information can be used by the ad-hoc workflow system to suggest new steps for the workflow design. The workflow editor has wizard-like functionality to help the workflow designer extend the workflow [15]. It supports what we call *guided workflow design*. The ad-hoc editor should support forward guiding, to propose tasks that can use the data produced as input [17], but also backward guiding, to find tasks that can produce the data required as input. Guided workflow design is close to the auto-completion functions found in many IDE's. Auto-completion helps the programmer, among others, to write correct programming code and to quickly discover methods [18].

Additionally the workflow system can guide data conversion. Data incompatibility forms a big problem in service composition [19–23]. Wassink et al. [9] have shown that at least 30% of the tasks in a typical bioinformatics workflow are devoted to data conversion. The workflow system can propose tasks that perform the data conversion required.

Some workflows are used only once, others repeatedly [11]. An ad-hoc workflow system should support both, workflows for one-time use and workflows intended for multiple-time use. The ad-hoc workflow system is a programming by demonstration environment. Programming by demonstration means the user shows what needs to be done, and the environment records these actions, generalises over them and translate them into a script [24]. A programming by demonstration environment acts like a macro recorder, but at the same time is able to recognise control structures such as iteration and conditional branching. In a workflow context, the workflow designer creates the workflow by demonstration; the ad-hoc workflow system abstracts from case specific properties, such as data and services, and translates the model into a template workflow.

Designing workflows by demonstration suits the dual mode of experiment design and experiment reuse. In the early phase of workflow design, scientists go through a fast cycle of hypothesis generation, experimentation, evaluation of the results and method selection [23]. After this phase, rationalisation is performed, in which scientists validate the results and formalise the process [5]. The ad-hoc workflow system enables the workflow designer to explore and to try things out in the early phase of workflow design. The result is a workflow that abstracts from concrete data and can be used as a template for future, similar experiments. The power of the template becomes even greater if the workflow system supports late binding, because then the workflow is independent of the resources used at design time.

3 e-BioFlow: Different Perspectives on Scientific Workflows

e-BioFlow [6] is an open source workflow system that provides its users a workflow editor and a workflow engine[1]. The workflow system uses a tabbed user interface to design and execute workflows and to analyse executed workflows. A tab is called a perspective. e-BioFlow contains six different perspectives at the moment:

Control flow perspective focuses on the order of tasks. It enables the workflow designer to model the order of task execution. The workflow designer can model sequential, parallel, iterative and conditional execution of tasks.

Data flow perspective is used to model data transfer between tasks, called pipes. Input ports and output ports contain type information (syntactical and semantical) about the data they respectively consume and produce.

Resource perspective is used to define the type of resources required to execute the task. The actual resource to execute the task is chosen at execution time of the workflow. The resources are called actors and are components that can execute tasks, such as invoking web services or executing scripts.

Workflow engine can execute workflows. It is responsible for scheduling tasks, performing the late binding and passing data between tasks. It is built on the YAWL engine [25], but supports late binding and passing data by reference.

[1] Available at: `http://www.ewi.utwente.nl/~biorange/ebioflow`

Provenance system automatically captures all process and data related information of workflow execution. It stores provenance data in Open Provenance Model [26, 27] compatible format. It contains a provenance browser, which is a graph visualisation to explore the provenance data.

Ad-hoc editor is able to perform ad-hoc design of the workflow. It will be discussed in more detail in the next section.

All perspectives except the provenance perspective directly communicate with the specification controller (Figure 1). This specification controller manages all workflows loaded into e-BioFlow. The perspectives send requests to the specification controller for a change in the workflow model when the user edits the workflow diagram. The specification controller applies the change and notifies all perspectives about the change.

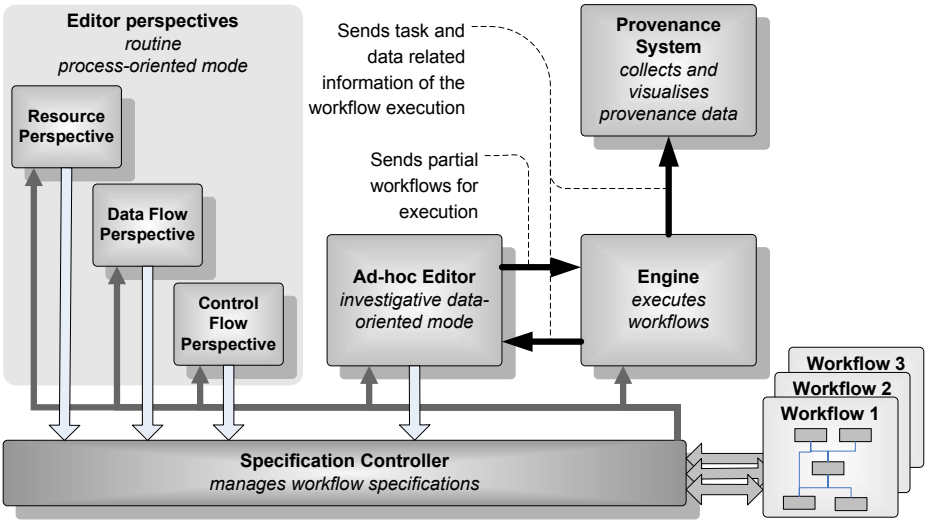


Fig. 1. All perspectives are registered to the specification controller to send and receive changes in the workflow model. The ad-hoc editor is a perspective that interacts with the engine.

The first three perspectives are introduced in previous work [6]. They are complementary: they edit the same workflow model, but each focuses on a specific aspect of the workflow. The ability to model control flow related information and data flow related information within a single workflow system makes e-BioFlow what is called a hybrid workflow system [28].

The engine can run the workflows managed by the specification controller in a routine process-oriented mode. It performs late binding using the task definitions. It tries to delegate the task to the default actor, if it is set and available, else it will try to find a compatible services. At the moment, e-BioFlow supports

three types of actors: i) actors that can invoke SOAP/WSDL or BioMOBY services, ii) actors that can execute scripts written in, among others, Java[2], Perl[3] and R [29], and iii) actors that can interact with the user. The provenance system communicates with the engine. It receives all information related to the workflow execution, and stores this information. The provenance browser can be used to interactively explore these data.

4 Ad-Hoc Workflow Design in e-BioFlow

The ad-hoc editor combines features of an editor, an engine and a provenance system. The ad-hoc editor uses the workflow models shared by the other perspectives. Like other perspectives, the ad-hoc editor uses the specification controller to receive notifications about changes in the workflow models and to request changes in the workflow model when the workflow designer edits the workflow. The three characteristics live editing, guided workflow design and workflow by example will be used to explain the ad-hoc editor in more detail.

4.1 Live Editing

At first sight, the ad-hoc editor looks similar to the data flow perspective: the workflow designer can drag and drop tasks into the workflow diagram and define outputs of one task to be input for others. However, using the ad-hoc editor, the workflow designer can select one or more tasks and instruct the ad-hoc editor to execute these tasks. When the workflow designer instructs the ad-hoc editor to do this, the ad-hoc editor creates a partial workflow of the selected tasks based on the original workflow model. It adds two user interaction tasks to this new workflow. The first task, called the *input-task*, is added to the start of the workflow. This task shows a dialog containing the input data already available and fields for the missing data. The user can modify the already available data and enter the missing data using drop-down boxes in the case there are fixed sets of valid options or else using text fields. The second task is called the *output-task* and is added to the end of the workflow to show the results of the tasks.

The ad-hoc editor uses the workflow engine to execute this partial workflow. It automatically captures the data produced during workflow execution. These data are visualised as circles called data items (Figure 2). The ad-hoc editor uses arrows from the output ports to the corresponding data items to present the *created-by* relations. The workflow designer can inspect the data by selecting the circles. At the moment, e-BioFlow can visualise many data formats, such as plain-text, XML, PDF-files, bitmap graphics and vector graphics.

The workflow designer can create a connection between a data item and a task's input port to define this data item to be input for that task. This relation is called a *used-by* relation. The ad-hoc editor automatically adds a pipe between the output port of the task that has generated the data item and the input port

[2] `http://www.java.sun.com`
[3] `http://www.perl.org`

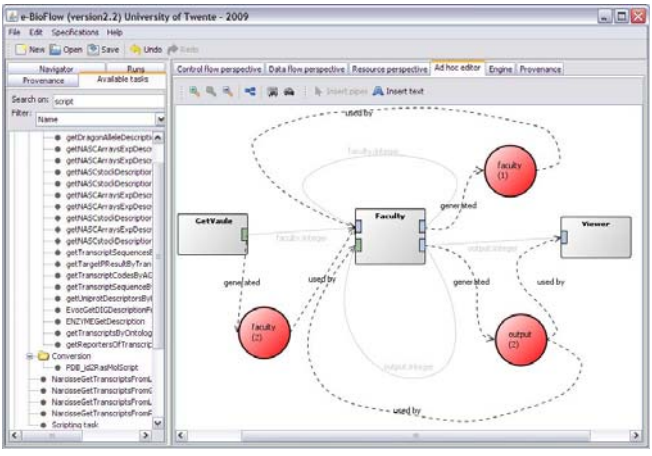


Fig. 2. A screenshot of the ad-hoc editor. Tasks are presented as boxes; data as circles connected to the output ports that have produced them and to the input ports that use them.

that uses the data item as input (Figure 3). When the user instructs the ad-hoc editor to execute this task, it uses this data item as input for that input port of the task. When multiple data items are defined to be input of the task, the input-task enables the user to choose which ones to use.

When an executed task has produced a data item related to an input port that is connected to an output port by means of a pipe, the ad-hoc editor automatically creates a used-by relation between the data item and the output port (Figure 4).

4.2 Guided Workflow Design

The ad-hoc editor helps the workflow designer to find new tasks to extend the workflow using the type information of data and the ports of tasks. When the workflow designer selects an input port of a task, the ad-hoc editor lists actors available and tasks already in the workflow that can produce compatible input. If the workflow designer chooses an actor from this list, the ad-hoc editor adds a task definition into the workflow for that actor. Additionally, it generates a pipe between the input port selected and the compatible output port of the new task. If the new task has multiple compatible output ports, the ad-hoc editor asks the workflow designer to which input port the pipe should be connected. If the workflow designer chooses a task already in the workflow, only the pipe is created.

In a similar way, the workflow designer can select output ports to find and add tasks or actors that accept the output data to as input. Data items can be selected to find and add tasks and actors that accept these data as input.

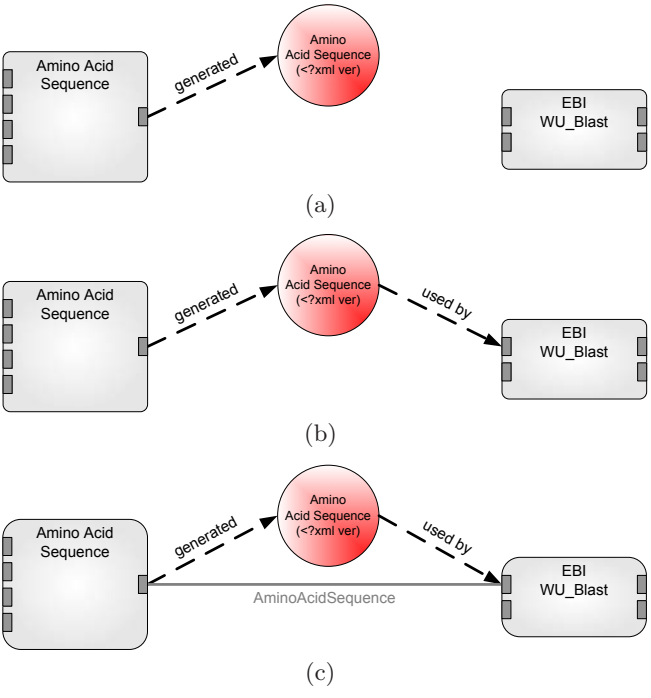


Fig. 3. (a) The "Amino Acid Sequence" task has generated a data item as output. (b) This data item is defined as input for the "EBI WU_Blast" task. (c) The ad-hoc editor automatically generates a pipe between the two tasks.

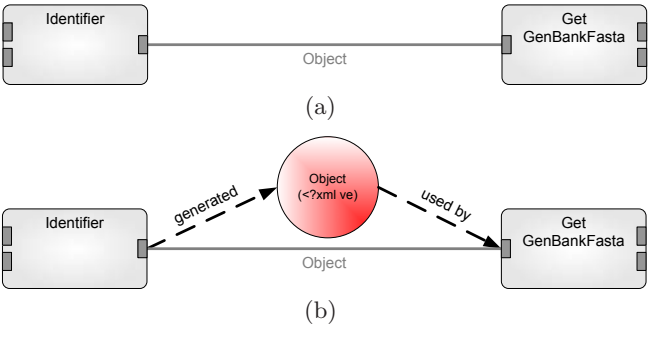


Fig. 4. (a) The output of the composer task "Object" is connected to the input of the task "Get GenBankFasta" task. (b) When the first task is executed, the ad-hoc editor automatically creates a data link between the data item produced and the next task.

Many web services use XML-based data structures as input or output, such as SOAP/WSDL and MOBY-S services. Different services use different formats, even for the same type of data. Creating these structures is a laborious and error-prone activity, especially when the data is hierarchical. The ad-hoc editor helps the workflow designer to build and parse these XML data structures by means of so-called composer tasks and decomposer tasks. The inputs of a composer task are the child elements, content and attribute values; the output is the XML structure built. The input of a decomposer task is the XML structure to be parsed; the outputs are the child elements, the content and the attributes. Multiple composers and decomposers can be chained to build or parse hierarchical XML structures. Composers and decomposers tasks are handled as normal e-BioFlow tasks, but are listed in separate categories when the workflow designer searches for compatible tasks. Although these composer and decomposer tasks do not solve all data conversion problems, they help the scientists to create XML structures and to reuse the contents of XML structures without programming.

4.3 Programming by Demonstration

Workflows designed in the ad-hoc editor can be edited directly using the other perspectives and vice versa. For example, when a new task is inserted in the ad-hoc editor, then this task is also visible in the other perspectives. Similarly, if a connection is made between a data item produced by a certain task and the input of another task in the ad-hoc editor, this is visible as a data pipe between the two tasks in the data flow perspective and vice versa. The relation is visible as a dependency relation in the control flow perspective, denoting the order of task execution. When the workflow is complete, it can of course also be run using the e-BioFlow workflow engine. The workflow can can be saved as a template for future experiments.

5 Use Case: Perform a Blat Operation

For the use case we introduce a fictitious bioinformatician named Tom, who wants to orchestrate web services to analyse a biological question. Tom wants to perform a sequence retrieval search against the zebrafish assembly for a set of 200 sequences. He uses the ad-hoc editor to construct a runnable workflow using a single sequence. Once the design of the workflow is finished, he will run the workflow for the whole set of sequences.

Tom searches for a Blat service [30], because it is a fast alternative for Blast. Soon, he finds the Blat service provided by Wageningen University, because this one provides fast access to the Ensembl [31] zebrafish assembly. He drags the Blat service to the workflow panel. The service requires two inputs, both MOBY-S objects. The first, named “User”, is required for session information; the second, called “BlatJob”, to provide the sequence and the database name. Tom does not know the XML structures required, and even does not want to. Luckily, the ad-hoc editor can help Tom to construct these complex data structures. Tom

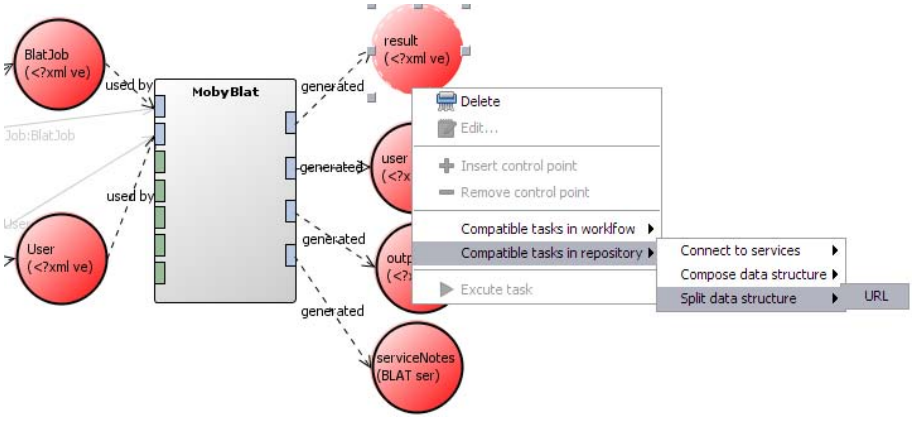
instructs the editor to add composer tasks for the "User" input by right clicking on this port. The ad-hoc editor shows compatible services and a composer task. Tom chooses the composer task and instructs the ad-hoc editor to execute it. The ad-hoc editor asks Tom to enter the e-mail address and password to construct the complex data structure. The service description tells Tom that any e-mail address and password will suffice. The ad-hoc editor shows the results of the composer task in the workflow panel. Additionally, two arrows are added to the workflow model, one connecting the composer's output port to the data item and one connecting the data item to the input port of the Blat task.

The "BlatJob" input is created in a similar way. This object is built of complex data input too (database and sequence information). The ad-hoc editor enables Tom to further compose these inputs. Tom instructs the ad-hoc editor to run these three composer tasks at once. The ad-hoc editor asks Tom to enter the database to be used and the sequence. The result is visualised as a red circle connected to the output port.

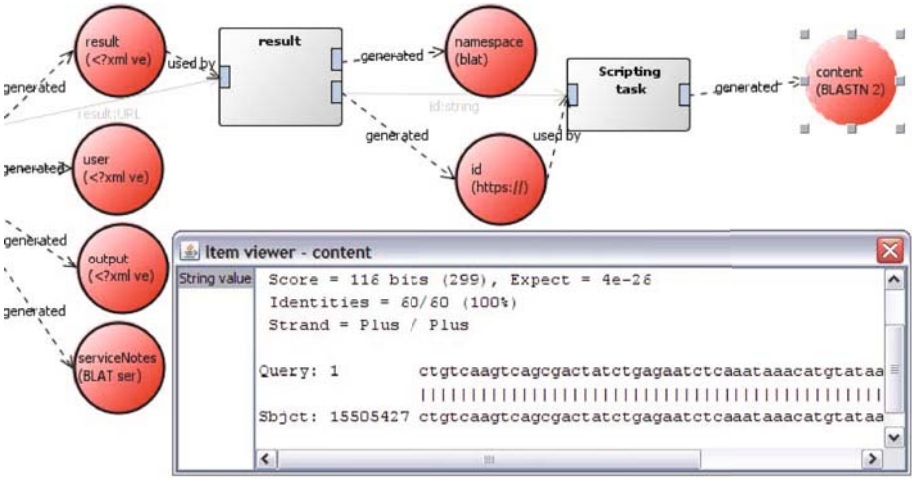
Now all the inputs required by the Blat task are available, Tom instructs the ad-hoc editor to execute the Blat task. The Blat service returns four outputs, namely a URL to the Blat report, a copy of the two inputs and the MOBY-S service notes. It seems that Tom has to download the Blat results using this URL, however, it is in MOBY-S XML format. By right-clicking on the URL data item, Tom selects the decomposer task for this MOBY-S object (Figure 5(a)). The data item is connected to the input of the decomposer automatically. Tom instructs the ad-hoc editor to execute this task. The output of the decomposer is the URL, this time in plain text.

The URL describes a location using a secure socket connection. Currently, e-BioFlow does not offer a task that can download content over a secure connection. Tom knows how to do this using Perl. He searches for a Perl task in the task panel and finds a "scripting task". Tom drags this task into the workflow panel. The scripting task has no inputs or outputs by default. When Tom selects the task, a configuration dialog pops to define the input (the URL) and the output (the Blat report) of the task, and the script to be executed. The scripting task requires Tom to select the language he wants to use. From the available languages, he chooses "Perl". Tom enters the code to be executed. The script panel supports syntax highlighting; the inputs and outputs are treated as normal variables, but are highlighted to distinguish them from the other variables.

When Tom has finished writing the code, he instructs the ad-hoc editor to run the scripting task. The ad-hoc editor shows an error message and complains about an unknown function. Tom reopens the configuration dialog of the scripting task and discovers he had forgotten to include a package. He inserts the import statement and re-executes the task. This time, the task runs successfully and returns the Blat report. The Blat report is in PSI format. Tom, however, wants the output in Blast format in order to inspect the alignment. He configures the Blat operation to generate the report in Blast format. He instructs e-BioFlow to rerun the Blat task, the decomposer and the download task. This time, the workflow generates the correct output (Figure 5(b)).



(a) Search for composer task.



(b) The result of the scripting task.

Fig. 5. Screenshots of the design of the use case workflow in the ad-hoc editor

Now Tom has designed a correct workflow using a single sequence, he can use it to perform the sequence-based search for the complete set of sequences.

6 Related Work

Few systems support ad-hoc workflow design, but none of them fulfill all the features mentioned above. We will mention five different systems, all handling a different approach in supporting ad-hoc workflow design.

Workflow by Example (WbE) [32]. WbE records the operations the user performs on a database and translate them into a workflow for future

applications in similar contexts. WbE, however, is task oriented and does not support direct data manipulation. Additionally, its focus is on automating querying databases instead of web service composition.

SeaHawk [17]. This tool provides an interface in which the user can explore data and request the tool for services that accept the data as input. Seahawk is not a workflow tool itself, but it can record the complete exploration history and export this to a Taverna workflow.

KNIME [33]. This is a data exploration system and mining. It provides access to many data analysis tools and they are presented as nodes of the workflows. KNIME enables its users to execute the tools in isolation and to explore the outputs. These outputs are, however, not explicitly presented in the graph. Additionally, KNIME does not support web services.

Data playground [5]. A Taverna [34, 35] extension that enable life scientists to "play" with MOBY-S services and to create small workflow snippets. This plugin system enables the scientist to transform these snippets to the standard process view of Taverna. Although the initial user experiences were promising, this extension is not further designed.

ADEPT2 [36]. ADEPT2 supports dynamic process adaptation Running processes can be modified at runtime. But ADEPT2 is directed to data-poor business administration workflows for which the workflow engine is idle most of the time during a run. Data, because it is scarce, cannot be used to decide about the continuation of the workflow.

e-BioFlow supports the workflow by example, the explorative reseach style of the Data playground and the guided experiment design of SeaHawk. It combines many features of these systems, and provides them through a single graphical user interface.

7 Discussion

The ad-hoc editor turns e-BioFlow into a workflow design and execution system that supports both the routine process-oriented mode and the investigative data-oriented mode to design workflows. The ad-hoc editor operates on the same workflow model and therefore workflows designed in this perspective can be edited using the other perspectives or run using the e-BioFlow engine. This editor enables the workflow designer to construct workflows in an explorative and intuitive way by giving real-time feedback of the state of the workflow and suggesting compatible actors, composers and decomposers. The workflow designer can run tasks in isolation, among others, to analyse intermediate results, to optimise parameters and to debug scripts.

The ad-hoc editor supports all actors provided by e-BioFlow. The ad-hoc editor fully supports the late binding capabilities of e-BioFlow. Workflows designed in this perspective are reusable templates for routine process-oriented mode to analyse other data sets and for sharing with other scientists through web portals such as myExperiment.

The use-case in this paper demonstrates a small but realistic scenario of using the ad-hoc editor to design a workflow. Many features the ad-hoc editor provides are shown, such as explorative design, composing and decomposing complex data structures, debugging the workflow and optimising parameter settings. The result of the use-case is a correct, runnable workflow that can be (re)used to perform a Blat analysis for a large number of sequences.

We [37] have tested the ad-hoc editor among life scientists and have found this editor to be a real improvement over traditional workflow editors.

Acknowledgement

This work was part of the BioRange programme of the Netherlands Bioinformatics Centre (NBIC), which is supported by a BSIK grant through the Netherlands Genomics Initiative (NGI).

References

1. Kell, D., Oliver, S.: Here is the evidence, now what is the hypothesis? BioEssays 26(1), 99–105 (2004)
2. Mahoui, M., Lu, L., Gao, N., Li, N., Chen, J., Bukhres, O., Miled, Z.: A dynamic workflow approach for the integration of bioinformatics services. Cluster Computing 8(4), 279–291 (2005)
3. Shao, Q., Sun, P., Chen, Y.: Efficiently discovering critical workflows in scientific explorations. Future Generation Computer Systems 25(5), 577–585 (2009)
4. Barga, R., Gannon, D.: Scientific versus business workflows. In: Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M. (eds.) Workflows for e-Science, pp. 258–275. Springer, Berlin (2007)
5. Gibson, A., Gamble, M., Wolstencroft, K., Oinn, T., Goble, C.: The data playground: An intuitive workflow specification environment. In: Cox, S. (ed.) E-SCIENCE 2007: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, Washington, DC, USA, pp. 59–68. IEEE Computer Society Press, Los Alamitos (2007)
6. Wassink, I., Rauwerda, H., van der Vet, P., Breit, T., Nijholt, A.: e-BioFlow: Different perspectives on scientific workflows. In: Elloumi, M., Küng, J., Linial, M., Murphy, R.F., Schneider, K., Toma, C. (eds.) 2nd International Conference on Bioinformatics Research and Development (BIRD), Vienna, Austria, pp. 243–257 (2008)
7. Wainer, J., Weske, M., Gottfried, V., Bauzer Medeiros, C.: Scientific workflow systems. In: Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems, Athens, Georgia, pp. 1–5 (1997)
8. Green, T.R.G., Petre, M.: Usability analysis of visual programming environments: A cognitive dimensions framework. Journal of Visual Languages & Computing 7(2), 131–174 (1996)
9. Wassink, I., van der Vet, P., Wolstencroft, K., Neerincx, P., Roos, M., Rauwerda, H., Breit, T.: Analysing scientific workflows: why workflows not only connect web services. In: Zhang, L. (ed.) SERVICES 2009 (Part I), Los Angeles, USA, pp. 314–321 (2009)

10. Goble, C., De Roure, D.: MyExperiment: Social networking for workflow-using e-scientists. In: Deelman, E., Taylor, I. (eds.) WORKS 2007, Monterey, California, USA, pp. 1–2 (2007)
11. Goderis, A., De Roure, D., Goble, C., Bhagat, J., Cruickshank, D., Fisher, P., Michaelides, D., Tanoh, F.: Discovering scientific workflows: The myExperiment benchmarks. IEEE Transactions on Automation Science and Engineering (2008) (Submitted)
12. Whitley, K.: Visual programming languages and the empirical evidence for and against. Journal of Visual Languages & Computing 8(1), 109–142 (1997)
13. Costabile, M., Fogli, D., Mussio, P., Piccinno, A.: Visual interactive systems for end-user development: A model-based design methodology. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 37(6), 1029–1046 (2007)
14. Kulyk, O., Wassink, I., van der Vet, P.E., van der Veer, G.C., van Dijk, E.M.A.G.: Sticks, balls or a ribbon? results of a formative user study with bioinformaticians. Technical Report TR-CTIT-08-72, CTIT, University of Twente, Enschede (2008)
15. Downey, L.: Group usability testing: Evolution in usability techniques. Journal of Usability Studies 2(3), 133–144 (2007)
16. Hundhausen, C., Lee Brown, J.: An experimental study of the impact of visual semantic feedback on novice programming. Journal of Visual Languages and Computing archive 18(6), 537–559 (2007)
17. Gordon, P., Sensen, S.: Seahawk: moving beyond html in web-based bioinformatics analysis. BMC bioinformatics 8(208), 1–13 (2007)
18. Robbes, R., Lanza, M.: How program history can improve code completion. In: Inverardi, P., Ireland, A., Visser, W. (eds.) 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), Aquila, Italy, September 2008, pp. 317–326 (2008)
19. Wroe, C., Goble, C., Greenwood, M., Lord, P., Miles, S., Papay, J., Payne, T., Moreau, L.: Automating experiments using semantic data on a bioinformatics grid. IEEE Intelligent Systems 19(1), 48–55 (2004)
20. Neerincx, P., Leunissen, J.: Evolution of web services in bioinformatics. Briefings in Bioinformatics 6(2), 178–188 (2005)
21. Belhajjame, K., Embury, S., Paton, N.: On characterising and identifying mismatches in scientific workflows. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 240–247. Springer, Heidelberg (2006)
22. Kappler, M.: Software for rapid prototyping in the pharmaceutical and biotechnology industries. Current Opinion in Drug Discovery & Development 11(3), 389–392 (2008)
23. Shon, J., Ohkawa, H., Hammer, J.: Scientific workflows as productivity tools for drug discovery. Current Opinion in Drug Discovery & Development 11(3), 381–388 (2008)
24. Lau, T.A., Weld, D.S.: Programming by demonstration: an inductive learning formulation. In: Maybury, M., Szekely, P., Thomas, C.G. (eds.) IUI 1999: Proceedings of the 4th international conference on Intelligent user interfaces, pp. 145–152. ACM Press, New York (1999)
25. van der Aalst, W., Aldred, L., Dumas, M., ter Hofstede, A.: Design and implementation of the YAWL system. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 142–159. Springer, Heidelberg (2004)
26. Moreau, L., Plale, B., Miles, S., Goble, C., Missier, P., Barga, R., et al.: The Open Provenance Model (v1.01). Technical report, University of Southampton (2008)

27. Moreau, L., Freire, J., Futrelle, J., Mcgrath, R., Myers, J., Paulson, P.: The open provenance model: An overview. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 323–326. Springer, Heidelberg (2008)
28. Shields, M.: Control- versus data-driven workflows. In: Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M. (eds.) Workflows for e-Science, pp. 258–275. Springer, Berlin (2007)
29. Ihaka, R., Gentleman, R.: R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics 5(3), 299–314 (1996)
30. Kent, W.: Blat–the blast-like alignment tool. Genome Research 12(4), 656–664 (2002)
31. Flicek, P., Aken, B., Beal, K., Ballester, B., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cunningham, F., Cutts, T., Down, T., Dyer, S., Eyre, T., Fitzgerald, S., Fernandez-Banet, J., Gräf, S., Haider, S., Hammond, M., Holland, R., Howe, K., Howe, K., Johnson, N., Jenkinson, A., Kähäri, A., Keefe, D., Kokocinski, F., Kulesha, E., Lawson, D., Longden, L., Megy, K., Meidl, P., Overduin, B., Parker, A., Pritchard, B., Prlic, A., Rice, S., Rios, D., Schuster, M., Sealy, I., Slater, G., Smedley, D., Spudich, G., Trevanion, S., Vilella, A., Vogel, J., White, S., Wood, M., Birney, E., Cox, T., Curwen, V., Durbin, R., Fernandez-Suarez, X., Herrero, J., Hubbard, T., Kasprzyk, A., Proctor, G., Smith, J., Ureta-Vidal, A., Searle, S.: Ensembl 2008. Nucleic Acids Research 36(Database issue), D707–D714 (2008)
32. Tomasic, A., McGuire, R., Myers, B.: Workflow by example: Automating database interactions via induction. Technical Report CMU-ISRI-06-103, Carnegie Mellon University (2006)
33. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The konstanz information miner. In: Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R. (eds.) Data Analysis, Machine Learning and Applications, pp. 319–326. Springer, Berlin (2008)
34. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Oxford Bioinformatics 20(17), 3045–3054 (2004)
35. Oinn, T., Li, P., Kell, D., Goble, C., Goderis, A., Greenwood, M., Hull, D., Stevens, R., Turi, D., Zhao, J.: Taverna/myGrid: Aligning a workflow system with the life sciences community. In: Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M. (eds.) Workflows for e-Science, pp. 300–319. Springer, Berlin (2007)
36. Reichert, M., Dadam, P.: Enabling adaptive process-aware information systems with ADEPT2. In: Research on Business Process Modeling. Information Science Reference, pp. 173–203 (2009)
37. Wassink, I., van der Vet, P., van Dijk, E., Veer, G., Roos, M.: New interactions with workflow systems. In: European Conference on Cognitive Ergonomics 2009 (ECCE 2009), Otaniemi, Finland (in press, 2009)