E-BioFlow: Different Perspectives on Scientific Workflows

Ingo Wassink[1], Han Rauwerda[2], Paul van der Vet[1], Timo Breit[2], and Anton Nijholt[1]

[1] Human Media Interaction Group, University of Twente
{i.wassink, p.e.vandervet, a.nijholt}@ewi.utwente.nl
[2] Micro Array Department, University of Amsterdam
{rauwerda, breit}@science.uva.nl

Abstract. We introduce a new type of workflow design system called e-BioFlow and illustrate it by means of a simple sequence alignment workflow. E-BioFlow, intended to model advanced scientific workflows, enables the user to model a workflow from three different but strongly coupled perspectives: the control flow perspective, the data flow perspective, and the resource perspective. All three perspectives are of equal importance, but workflow designers from different domains prefer different perspectives as entry points for their design, and a single workflow designer may prefer different perspectives in different stages of workflow design. Each perspective provides its own type of information, visualisation and support for validation. Combining these three perspectives in a single application provides a new and flexible way of modelling workflows.

1 Introduction

Workflow systems have proven to be successful for modelling both business processes [1, 2] and life science experiments [3–9]. The way workflow models are used in these two areas differs tremendously. Business workflow models are control flow-oriented (defining the order in which work has to be done), whereas life science workflow models are data flow-oriented (describing the information flow between tasks) [10]. However, there is growing demand for a more controlled approach to workflows in the life science domain [11]. Workflow systems in this domain lack the facilities to model advanced control structures such as conditional branches and iteration, and miss the functionality to easily switch between resources (also known as agents) for executing the tasks of workflow models. For example, we have developed the RShell plugin [3] for Taverna, which is now part of the standard Taverna distribution [6, 12, 13]. This plugin provides the RShell processor for executing R-scripts in a Taverna workflow. Our plugin requires a local or remote installation of R [14] in combination with RServe [15], which turns R into an R server. However, such a local installation of R in combination

[3] http://www.ewi.utwente.nl/ biorange/rshell

with the required R packages are not always available. Therefore, the specification of the location of the R server should be postponed until the workflow is actually executed.

Workflow systems support two tasks: workflow design and workflow enactment (as workflow execution is often called in the literature). The result of workflow design is a workflow model or workflow for short. The result of workflow enactment is actual execution of the model in the correct way. Design and enactment can in principle be performed by different systems as long as there is a common language to transfer the workflow model from the design system to the enactment system. In practice, most systems support both tasks.

Scientists have to show that their experiment conforms to quality standards in their field [16] whereas business modelers have to create consistent, optimized, and possibly automated business processes [1, 17, 18]. When workflows become complex, we need tools to manage this complexity, for example, to perform automatic validation of workflows. Workflow models enable formal checking from a *control flow perspective* (soundness [19]) as well as from a *data flow perspective* [20] and *resource perspective* [17, 18]. From a control flow-perspective, a suitable tool can also simulate the workflow to give the designer an idea of the course of events and the flow of data through the system. Workflow models, and diagrams in general, are very suitable as visualisation and communication means [21, 17, 22]. However, most workflow design systems mainly focus on structuring and connecting tasks; the visualisation aspect often gets little attention. Workflow visualisation is not limited to showing dependency relations between tasks, but can also show the types of data that flow between the tasks and what types of resources are able to perform the tasks. In existing systems, these different aspects (if supported at all) are often combined in a single diagram which results in cluttering of information [23]. This is counterproductive; if anything, visualisation should advance rather than hinder understanding.

We introduce *e-BioFlow*, a workflow design system that relies on an existing system to have the workflow enacted. It is available under the GNU General Public Licence through sourceforge [4]. E-BioFlow enables the user to model workflows from the three mentioned perspectives previously. E-BioFlow is inspired by the context of scientific collaborative environments, such as the e-BioLab [24]. The workflow system enables scientists to describe tasks in multidisciplinary life science experiments [25]. Workflow models for these types of experiments need to be flexible with respect to resources, which can be web services, scientists or machines in the laboratory. The models made by means of E-BioFlow can be enacted by the open-source workflow system Yawl [26]. In the next section, we will discuss the requirements of a workflow design system for modelling processes. After that, we will introduce our approach, e-BioFlow, the three perspectives it provides, and how these perspectives are related. The benefit of these perspectives will be illustrated using an example of a workflow that performs a simple sequence alignment. Our approach will be compared to related work and we will end with a discussion.

[4] http://sf.net/projects/e-bio-flow

2 Perspectives of a workflow model

Workflow models are often not designed for a single case but describe types of cases [18]. Each case is unique; cases can differ in the way tasks are executed, the data that flows between these tasks, but also the resources that execute the tasks. Therefore, a workflow model should provide a perfect balance between generalisation over the case type and adaptation to the specific cases.

Van der Aalst [26] and Jablonski and Bussler [17] distinguish three perspectives on workflows:

Control flow perspective: Tasks can seldom if ever be performed in an arbitrary order. The control flow perspective defines dependencies between tasks and the way tasks will be executed (sequential, parallel, conditional or iterative).

Data flow perspective: Tasks can consume and produce information. The data flow perspective defines these producer/consumer relations between tasks.

Resource perspective: Tasks can often be executed by a class of resources rather than by a single resource. The resource perspective defines the relation between tasks and the classes of resources that can execute them.

As we will explain later, these three perspectives are not orthogonal but interact and therefore deserve equal attention in a workflow design tool. Most workflow design systems, however, are either control flow-oriented or data flow-oriented [27]. Control flow-oriented workflows neglect the data flowing between tasks or only support them indirectly using task and net variables. Data flow-oriented workflows lack the presence of advanced control flow structures, such as conditional branching and loops. Workflow design systems that focus on both the control flow and data flow perspective are called hybrid workflow systems [27].

Ideally, a workflow can be reused for every instantiation of a certain case type. Therefore, it is important to abstract from resources and to delay resource-task binding until the workflow is enacted. E-BioFlow supports this and thus adds an important feature to the designer's toolkit.

3 E-BioFlow: a new type of workflow design system

E-BioFlow is a visual workflow design system that provides all three perspectives to users for designing their workflows. The three perspectives are explicitly present in e-BioFlow by means of different tabs in the user interface. The workflow designer is able to work in a single perspective at a time without being restricted to the functionality of a single perspective. Changes in one perspective are propagated to the two other perspectives wherever appropriate.

Every workflow has at least two tasks, namely the start and the end task. These two tasks are used respectively to provide the workflow's input data and to collect the workflow's output data. Hierarchy is a very important property of workflow models, because it helps to structure large diagrams and provides

a means for abstraction [19]. E-BioFlow supports hierarchical workflows; a task can be composite. We can choose to decompose a composite task into a sub-workflow, in which case the composite task is white boxed, but we can alternatively choose to ignore the way the composite task is structured, leaving it black boxed. The workflow specification is a container for workflows. However, in every container only one workflow is marked as the root workflow; the other workflows are decompositions of composite tasks. Two or more composite tasks can be white boxed to the same sub-workflow.

The concepts used in e-BioFlow to represent workflows are:

Task: A task is an abstraction of work to be done. This is also known as an activity [28].

Atomic task: An elementary representation of work [29].

Composite task: A task that can be black boxed or white boxed; in the latter case, we can also call it a sub-workflow.

Workflow: A workflow defines a set of tasks, the dependencies between the tasks, the data that flows among the tasks and the required capabilities to execute the tasks.

Specification: A specification is a container for workflow models. One of the workflows is the root model, the others are sub-workflows.

Dependency (Control Flow perspective): A relation between two tasks that defines enactment order: a certain task cannot start until another task has finished [27].

Dependency condition (Control Flow perspective): Every task has a start condition and an end condition describing, respectively, the way the task depends on prior tasks and the way it should activate next tasks [26].

Port (Data Flow perspective): A task can have multiple input and output ports for consuming and producing data, respectively. Ports are also known as parameters [23].

Object type (Data Flow perspective): The object type describes the type of information an input port accepts and an output port delivers.

Pipe (Data Flow perspective): A pipe defines a data dependency between two tasks, where data produced by the prior task is consumed by the next task [27].

Role (Resource perspective): A role describes the required capabilities to execute a task [28].

Actor (Resource perspective): An actor is a resource capable to fulfil a particular role and therefore to perform a certain class of tasks [28].

3.1 Three perspectives to design a workflow

The e-BioFlow language extends the Yawl workflow language [26]. Yawl is a formal workflow language based on the Petri net formalism. This formalism, originally introduced for representing concurrent processes, provides powerful analysis techniques to validate workflow [19]. Yawl enables one to model almost all workflow patterns described by Van der Aalst and others [30]. Workflows

designed in e-BioFlow can be enacted by the Yawl system [29]. The Yawl system itself also comes with a design tool that, however, only enables one to design control flow structures and does not explicitly consider the data flow and resource perspectives. E-BioFlow complements Yawl by adding these two perspectives. Next we will describe the three perspectives as they are offered to the designer by e-BioFlow. The use of these perspective will be illustrated using a simple life science case in section 3.2.

Control flow perspective Due to dependencies between tasks, there is often a specific order in which tasks can be executed. The control flow perspective describes these dependencies. Tasks can be executed in sequential, parallel, conditional and iterative order [18]. The way tasks depend on each other is described using conditions. The control flow perspective visualises both the dependencies between the tasks and the conditions on these dependencies.

A task is visualised as a box. A dependency between two tasks is visualised as an arrow from one task to the next. Like in Yawl, the way a task depends on others is controlled by *join* and *split* types. The split type defines the way a task should activate next tasks. The join type defines the way in which a task has to wait for prior tasks. The join and split types are attached to, respectively, the left side and right side of the task's box. Van der Aalst et al. [26] distinguish four different types of splits: SINGLE (a task has only one next task to be activated), AND (a task should activate all next tasks), OR (a task should activate one or more of the next tasks), and XOR (only one of the next tasks should be activated).

Similar types of join exist: SINGLE (a task depends on just one prior task), AND (a task has to wait for all prior tasks to finish), OR (a task has to wait for one or more of the prior tasks to finish), and XOR (a task has to wait for one of the prior tasks to finish).

The symbols used in the Yawl language are confusing and not easy to remember. Therefore, we have defined our own symbols, which are presented in figure 1. The visualisation of the "Single" type contains a single line, which shows that only one connection is allowed. The "And", "Or" and the "Xor" splits and joins are represented by the first letters of their meanings: 'A', 'O' and 'X' respectively.

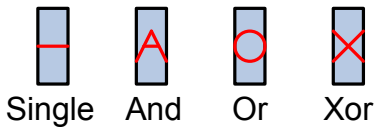


Fig. 1. Symbols representing the four different join and split types

Data flow perspective The data flow perspective is often seen in scientific workflow systems, where most tasks are executed by web services or computer applications. In a pure data flow representation, constructs such as loops are not included [27].

As in the control flow perspective, tasks are visualised as boxes. The input and output ports are represented as small horizontal lines, distributed over respectively the left and the right borders of the task box. If the number of input ports or output ports becomes large, it can be difficult to distinguish the ports. Therefore, the size of a task box grows proportionally to the number of ports. The names of the ports become visible when the user moves the mouse cursor over the port. Both input and output ports are tagged with the attributes *object type* and *cardinality*. The object type describes the type of data a port can consume or produce. To support a wide range of object types, e-BioFlow provides an abstract Java interface for object types which can be adapted to object types for a specific domain. By defining a repository, these object types can be fed to e-BioFlow. For example, we are currently implementing support for the BioMOBY [33] data types. The cardinality defines the amount of items that can be produced or consumed. Two types of cardinality are supported: *UNIT* and *COLLECTION*. The first means that one item at a time is produced or consumed; the latter that a set of items can be produced or consumed. Pipes are visualised as arrows between the corresponding output port and input port. Using the object types and the cardinality, e-BioFlow prepares for full data compatibility checking. If a pipe is valid, it is coloured black and labelled corresponding to its object types, otherwise it is coloured red.

Resource perspective E-BioFlow abstracts from resources by means of a ternary relationship between tasks, actors and roles. Actors are the real resources, such as web services. The role describes the abilities a resource is required to have to be able to perform the task [18]. Put simply, the role defines the *type* of service required. Roles can be played by different resources and resources can be able to play different roles, possibly at the same time [31, 18]. If a resource plays a certain role, it acts as a contractor and it is responsible for the work it accepts. The loose coupling between task and actor makes a workflow model reusable, even if some actors are not available [8]. However, a role description should contain enough information to choose a suitable actor for playing the role and executing the task [32]. An actor is able to execute a task if and only if it is able to fill the role assigned to that task [18]. In this view, the enactment engine is responsible to perform the actor-role binding while the designer only specifies constraints on the binding by means of roles. To do this, the engine needs a mapping function $f_{RA} : (Role \rightarrow Actor)$ to select a suitable actor for a given role. The implementation of this function is domain-dependent. It can be based on, for example, a repository of the available actors, sorted by the type of service they can deliver.

The workflow is visualised as a graph in the resource perspective, too. The visualisation of the resource perspective is closely related to the control flow

perspective, in order to keep the dependency relations in sight. However, the join and split condition information is left out. Each role is painted as a box around the task it is assigned to and contains the name of the role. Users can assign roles to tasks by dragging roles from a repository and dropping them on tasks.

In the current implementation, the resource perspective limits the user to only assign roles to atomic tasks on the ground that composite tasks will be expanded by the workflow enactment engine. An alternative would be to link every composite task to a role called "enactment engine" with the intended meaning that at execution the composite task is executed by the particular workflow enactment engine that happens to be running the parent workflow. The result would be a framework that encompasses both atomic and composite tasks. But it would also introduce a potential source of confusion, because for most atomic tasks a role entails a choice that will be made when the workflow is enacted. For composite tasks there is never such a choice: a composite task is always enacted in the framework of its parent task.

3.2 A simple life science case: sequence alignment

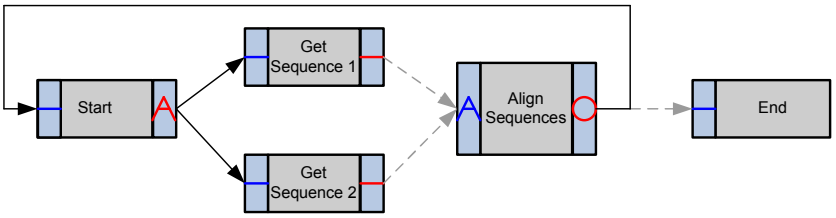
The three different perspectives will help the scientist to deal with the complexity of scientific workflows. Figure 2 shows the three different perspectives of a simple workflow for doing a sequence alignment. The workflow consists of a start and an end task, two tasks to collect the sequences and finally a task to actually perform the alignment. The control flow (Figure 2(a)) shows the order of the task execution. First, two sequences are collected and after that, an alignment is performed. When the alignment is performed, a next iteration of the sequence alignment is started or the end task is activated. The data flow (Figure 2(b)) shows only the data transfer between the tasks. The alignment task gets input sequences from both prior tasks; the end task gets the results of the alignment.

The roles of the tasks are shown in figure 2(c). The start and end task do not need roles, since these are used by the enactment engine to provide input data and collect output data. Both "Get Sequence" tasks require a sequence retrieval actor, such as an EBI retrieval service, so a sequence retrieval role is attached to each of the two tasks. An alignment role is attached to the "Align Sequence" task, for example by in-house software or again over a web service. The binding of the tasks to the actors will be done by the enactment engine.

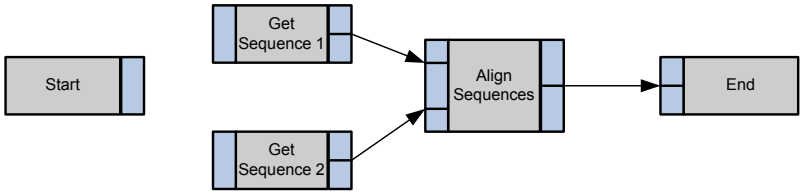
Each perspective complements the other perspectives and shows only limited information about the workflow in order to keep the workflow diagram usable and comprehensible.

3.3 Dependencies among the perspectives

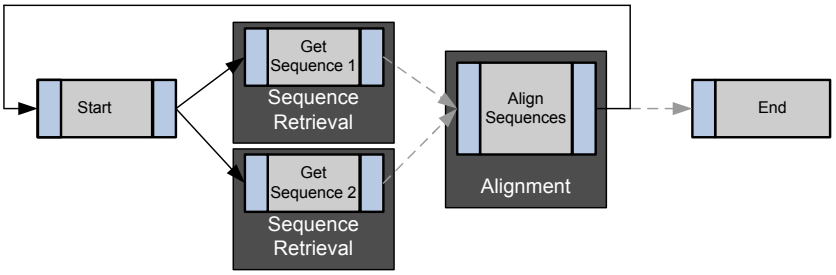
In all perspectives, the tasks of a workflow are represented as vertices of the graph. To simplify switching between the perspectives, tasks positions and task sizes remain the same in all perspectives.



(a) Control flow perspective



(b) Data flow perspective



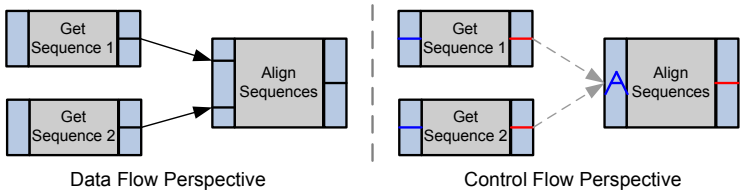
(c) Resource perspective

Fig. 2. An example of different perspectives for the workflow of a sequence alignment

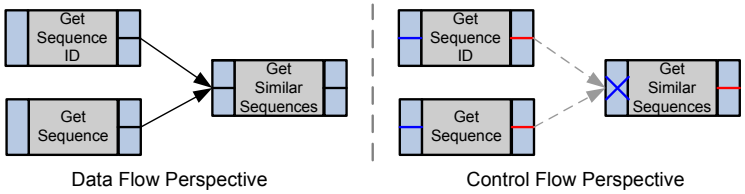
To illustrate the tight coupling between the perspectives, we will briefly discuss two scenarios. In one scenario, Figure 3(a), the designer has drawn two data pipes in the data flow perspective. E-BioFlow detects a dependency between the tasks involved, because in this example Align Sequences cannot start before the two "Get Sequence" tasks have delivered their data. Such a dependency is called an *inferred dependency* and it is inserted automatically in the control flow perspective as a dashed line. If the designer would later remove the data pipes, e-BioFlow automatically removes the dependencies in the control flow perspective.

In the other scenario, Figure 3(b), the designer has first inserted dependencies between the tasks in the control flow perspective. These are shown as solid lines. Later, the designer inserts data pipes in the data flow perspective. The solid lines in the control flow perspective are not affected because they are not inferred but inserted explicitly by the designer. For the same reason, if the designer later

removes the data pipes, the dependencies in the control flow perspective are not removed by e-BioFlow.



(a) The alignment task requires two inputs (left) and therefore it has to wait till both prior tasks have finished (right)



(b) The task that searches for similar sequences requires either a sequence id or a sequence as input (left) and therefore has to wait till one of the prior task has finished (right)

Fig. 3. Two examples showing the relationship between the data flow and the control flow perspective

Additionally, a relation exists between the resource perspective and the data flow perspective. The role definition depends more or less on the input and output types of a task, because not every actor can deal with all types of data. This means that the role description describes the ability to consume and to produce respectively the input and output data. Therefore, if an actor plays a role, it should be able to work with the input and output data [17]. The ability to check roles based on input and output types in BioMOBY fashion [33] is further work and requires more detailed descriptions of roles.

3.4 An architectural view and some implementation details

All three perspectives use, visualise and modify the same underlying workflow model. If this model is modified in a certain perspective, the other perspectives have to be notified to update their visualisations to reflect the change. Therefore, each perspective is registered to a software component called the *specification controller* (see figure 4). The specification controller works on top of the workflow specification. It has two main purposes.

First, it notifies all perspectives when the specification model is modified. These changes concern structural changes (i.e., a new task is inserted, a depen-

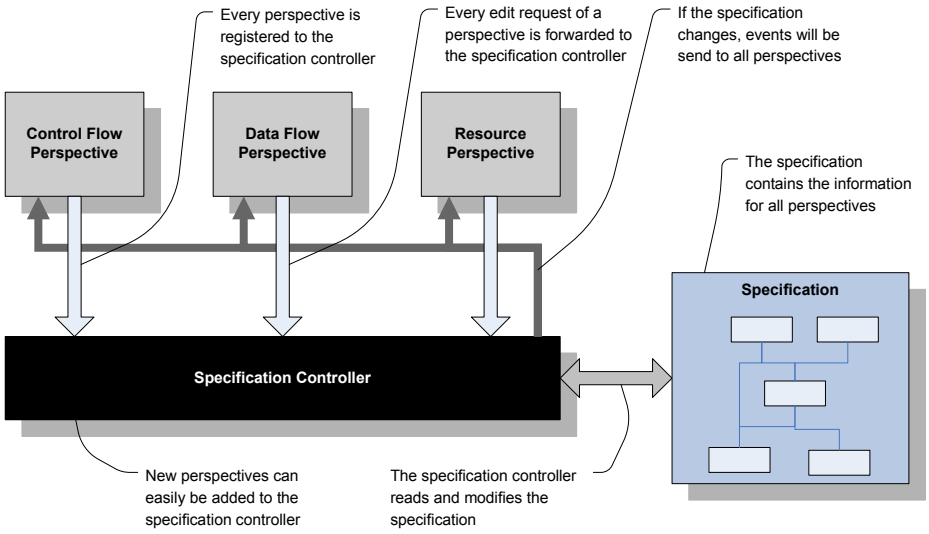


Fig. 4. The specification controller links the different perspectives to the specification

dency is removed) as well as graphical changes (i.e., a task is repositioned, the zooming level is changed or the graph is repositioned using scrollbars).

Second, the specification controller is the only component that is allowed to modify the specification. If an action is performed in a certain perspective, then this perspective sends a request to the specification controller to execute this action. Normally, the specification controller executes the action and sends a notification event to all perspectives. The specification controller also takes care of the undo/redo history.

Using a central specification controller for all perspectives, it is easier to introduce new components working on top of the workflow model, such as checkers for each perspective. It is possible to integrate a workflow enactment engine in e-BioFlow. This will result in a workflow environment that can help scientists to design, execute, (partially) redesign and re-start the workflow, which is ongoing work.

E-BioFlow is implemented in Java[5] and uses the JGraph[6] graph package. The default implementation of e-BioFlow is supplied with a limited set of object types and roles. However, it is not restricted to this limited set, because both object types and roles are accessed from repositories, which can easily be extended or replaced. New (local and remote) repositories can be created by extending existing repositories or adapting the provided abstract Java interfaces for these repositories. E-BioFlow has its own file format for storing workflow specifications. The control flow and the data flow perspectives of the specifica-

[5] http://www.java.sun.com (last visited: 31-01-2008)
[6] http://www.jgraph.com (last visited: 31-01-2008)

tions can be exported to the Yawl enactment engine format to execute workflow. The data flow perspective can be mapped to Yawl by translating the data flows to task variables, net variables and XPath expressions. The Yawl enactment engine does not support late binding, which hnders the implementation of our ideas on the resource perspective. We are working on this to remove this obstacle. Furthermore, e-BioFlow is able to export the control flow perspective in XPDL format [34], which is maintained by the Workflow Management Coalition (WfMC). E-BioFlow can import Yawl and Scufl, the language of Taverna [6, 35]. Due to the use of these central workflow formats, the workflow design system can be separated from the execution system [36].

4 Related work

The focus in life science is on data. The traditional design interfaces of business workflow systems do not fulfil the requirements of this domain. Therefore, most scientific workflow systems use a data flow oriented language for representing workflows. However, the problem with most of these systems is that they have difficulties to support advanced control structures. For example in Discovery Net [3] , SCIRun [7], Taverna [6, 12] and Knime [9] it is difficult if not impossible to construct advanced control structures, such as conditional branching and iterations. Triana [5] supports these control structures, however, like the other two tools, it has no ability to abstract from resources. From our point of view, this is cumbersome as resources, in particular web services, may be unavailable due to network errors, server overload, and similar problems. In Kepler [37, 8], Bowers and Ludäscher [38] have tried to tackle this problem by defining primitives for actor replacements. However, the replacement is still done at design time instead of instantiation time.

E-BioFlow is a hybrid workflow system. Another hybrid workflow system is JOpera [23]. Its designers do not speak of a workflow system but rather of a web services composition engine. For modern workflows, the distinction no longer matters because many resources are remote anyway. The convergence of the fields of workflow studies, web services composition, and scientific data processing is one of the more exciting developments in the field today. Like e-BioFlow, JOpera emphasises the visualisation aspects of workflow design. It enables the user to design workflows in two interacting perspectives, namely control flow and data flow, and also supports automatic detection of inferred dependencies. One key difference between e-BioFlow and JOpera is the fact that JOpera is not based on a formal model. Another key difference is the way resources are treated. JOpera only supports late binding using special constructions whereas e-BioFlow supports late binding by default. Of course, this results in special needs of the workflow enactment system, but we believe it is better to use late binding in order to keep a workflow reusable for different but similar experiments.

5 Discussion and future work

In life science during the past decade, the importance of computer-supported or dry-lab experimentation has sharply increased. Workflow models support dry-lab experiments because controlling and managing huge volumes of data is cumbersome if not impossible without them. Workflow models are used to automate these experiments and to manage the huge amount of data collected and generated during these experiments [39].

In most scientific workflow systems, it is neither possible to model human tasks nor to model machine tasks; only web service tasks can be modelled. Modelling human tasks and machine tasks would make it possible to model both the wet-lab and the dry-lab parts of a scientific experiment. LIMSs, traditionally used for the wet-lab part of an experiment, are very often based on built-in (and thus inflexible) workflows [40]. Modelling the wet-lab and dry-lab parts of an experiment in a single framework carries several advantages, among which the presence of a unified model of the entire experiment that is of help in designing the experiment and is a tool for automating the lab journal when the experiment runs. E-BioFlow is prepared for the task of modelling entire experiments. The control flow and data flow perspective each provides its own type of validation, which makes it easier to find inconsistencies in the model. Validation in the resource perspective is currently not available. Such a validation could be based on the types of inputs and outputs of a task and the specification of the role, containing the types of data it can respectively consume and produce. However, more investigation is required to support a formal validation in the resource perspective.

We suggest a redesign of the way workflow engines currently operate. The enactment engine is not only responsible for triggering tasks to start execution, but also for task assignments to actors, based on role descriptions. Currently, we are integrating the Yawl workflow engine into e-BioFlow, to be able to enact workflows designed by e-BioFlow within the e-BioFlow application. Some modifications of the Yawl workflow engine are needed to support the late binding of actors to tasks. As a proof of concept, we will use BioMOBY [33] to create a life science problem solving environment. The data types defined in BioMOBY's data ontology can easily be translated to the object types used in e-BioFlow. The service types of BioMOBY will be translated to roles in e-BioFlow. The hierarchy of service types in BioMOBY is not fully mature. Therefore, further investigation is required to support automated role-base selection of BioMOBY services. Instead of being only applicable to a specific problem, workflow models designed using e-BioFlow become a general solution for a group of problems, independent of specific resources. A workflow model may thus become an ideal scaffold for a problem-solving environment [41].

6 Acknowledgement

We like to thank Matthijs Ooms for his work related to e-BioFlow.

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. Georgakopoulos, D., Hornick, M., Sheth, A.: An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases **3**(2) (1995) 119–153
2. Santos, I., Valle, C., Raposo, A., Gattas, M.: A multimedia workflow-based collaborative engineering environment for oil & gas industry. In Brown, J., Cai, Y., eds.: Proceedings of the 2004 ACM SIGGRAPH, New York, USA, ACM (2004) 112–119
3. Rowe, A., Kalaitzopoulos, D., Osmond, M., Ghanem, M., Guo, Y.: The discovery net system for high throughput bioinformatics. Bioinformatics **19**(1) (2003) i225–231
4. Addis, M., Ferris, J., Greenwood, M., Li, P., Marvin, D., Oinn, T., Wipat, A.: Experiences with e-Science workflow specification and enactment in bioinformatics. In Cox, S., ed.: e-Science All Hands Meeting 2003, Nottingham, United Kingdom (2004) 459–466
5. Majithia, S., Shields, M., Taylor, I., Wang, I.: Triana: A graphical web service composition and execution toolkit. In Jain, H., Liu, L., eds.: IEEE International Conference on Web Services (ICWS'04), San Diego, California, USA, IEEE Computer Society (2004) 514–521
6. 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
7. Macleod, R., Weinstein, D., de St. Germain, J., Brooks, D., Johnson, C., Parker, S.: SCIRun/BioPSE: Integrated Problem Solving Environment for Bioelectric Field Problems and Visualization. In: Proceedings of the International Symposium on Biomedical Imaging, Arlington, VA, USA (April 2004) 640–643
8. Ludäscher, B., Altintas, I., Berkley, C., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience **18**(10) (2006) 1039–1065
9. Berthold, M., Cebron, N., Dill, F., Gabriel, T., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz information miner. In: Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007), Springer (2007) to appear.
10. Wainer, J., Weske, M., Gottfried, V., Bauzer Medeiros, C.: Scientific workflow systems. In: Proceedings of the NSF Workshop on Workflow and Process Automation in InformationSystems, Athens, Georgia (1997) 1–5
11. Kochut, K., Arnold, J., Sheth, A., Miller, J., Kraemer, E., Arpinar, B., Cardoso, J.: IntelliGEN: A distributed workflow system for discovering protein-protein interactions. Distributed and Parallel Databases **13**(1) (2003) 43–72
12. Oinn, T., Greenwood, M., Addis, M., Alpdemir, M., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M., Senger, M., Stevens, R., Wipat, A., , Wroe, C.: Taverna: Lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience Grid Workflow **18**(10) (2006) 1067–1100

13. Oinn, T., Li, P., Kell, D.B., 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: Workflows for e-Science. Springer (2007) 300–319
14. Ihaka, R., Gentleman, R.: R: a language for data analysis and graphics. Journal of computational and graphical statistics **5**(3) (1996) 399–414
15. Urbanek, S.: Rserve – a fast way to provide r functionality to applications. In Hornik K, Leisch F, Z.A., ed.: Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), Vienna, Austria (2003) 20–22
16. Barga, R., Gannon, D.: Scientific versus Business Workflows. In: Workflows for e-Science. Springer (2007) 258–275
17. Jablonski, S., Bussler, C.: Workflow management. Modeling concepts, architecture and implementation. Thomson, London, UK (1996)
18. van der Aalst, W., van Hee, K.: Workflow management: models, methods, and systems. The MIT Press, Cambridge, MA, USA (2002)
19. van der Aalst, W.: The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers **8**(1) (1998) 21–66
20. Belhajjame, K., Embury, S., Paton, N.: On characterising and identifying mismatches in scientific workflows. In Istrail, S., Pevzner, P., M.Waterman, eds.: Data Integration in the Life Sciences (DILS'06), Hinxton, UK (2006) 240–247
21. Reisig, W.: A Primer in Petri Net Design. Springer-Verlag, Berlin, Germany (1992)
22. Dourish, P.: Process descriptions as organisational accounting devices: the dual use of workflow technologies. In Ellis, C., Zigurs, I., eds.: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, Boulder, Colorado, USA, ACM Press 52-60 (2001) 52–60
23. Pautasso, C., Alonso, G.: The JOpera visual composition language. Journal of Visual Languages and Computing **16**(1–2) (2005) 119–152
24. Rauwerda, H., Roos, M., Hertzberger, B., Breit, T.: The promise of a virtual lab in drug discovery. Drug Discovery Today **11**(5-6) (2006) 228–236
25. van der Vet, P., Kulyk, O., Wassink, I., Fikkert, F., Rauwerda, H., van Dijk, E., van der Veer, G., Breit, T., Nijholt, A.: Smart environments for collaborative design, implementation, and interpretation of scientific experiments. In Huang, T., Nijholt, A., Pantic, M., Pentland, A., eds.: Workshop on AI for Human Computing (AI4HC), Hyderabad, India (2007) 79–86
26. van der Aalst, W., ter Hofstede, A.: YAWL: Yet another workflow language. Information Systems **30**(4) (2005) 245–275
27. Shields, M.: Control- Versus Data-Driven Workflows. In: Workflows for e-Science. Springer (2007) 258–275
28. Barthelmess, P., Wainer, J.: Workflow systems: a few definitions and a few suggestions. In Comstock, N., Ellis, C., eds.: Proceedings of the Conference on Organizational Computing Systems (COOCS'95), Milpitas, California, USA (1995) 138–147
29. van der Aalst, W., Aldred, L., Dumas, M., ter Hofstede, A.: Design and implementation of the YAWL system. In Goos, G., Hartmanis, J., van Leeuwen, J., eds.: 16th International Conference on Advanced Information Systems Engineering (CAiSE'04), Riga, Latvia, Springer-Verlag (2004) 142–159
30. van der Aalst, W., ter Hofstede, A.: Workflow patterns: On the expressive power of (Petri-net-based) workflow languages. In Jensen, K., ed.: Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPn 2002), Aarhus, Denmark, DAIMI (2002) 1–20

31. Sowa, J.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove (1999)
32. 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) (2004) 48–55
33. Wilkinson, M.D., Links, M.: BioMOBY: an open source biological web services proposal. Brief Bioinform **3**(4) (December 2002) 331–341
34. The Workflow Management Coalition: Workflow process definition interface – xml process definition language. Technical Report WFMC-TC-1025, The Workflow Management Coalition (2002)
35. Oinn, T., Addis, M., Ferris, J., Marvin, D., Greenwood, M., Goble, C., Wipat, A., Li, P., Carver, T.: Delivering web service coordination capability to users. In Feldman, S., Uretsky, M., eds.: Proceedings of the 13th international conference on World Wide Web, New York, NY,USA, ACM Press (2004) 438–439
36. Prior, C.: Workflow and Process Management. In: The Workflow Handbook 2003. Future Strategies Inc. (2003) 17–25
37. Altintas, I., Berkley, C., Jaeger, E., Ludäscher, B., Mock, S.: Kepler: An extensible system for design and execution of scientific workflows. In Hatzopoulos, M., Manolopoulos, Y., eds.: 16th International Conference on Scientific and Statistical Database Management (SSDBM'04), Santorini Island, Greece (2004) 423–424
38. Bowers, S., Ludäscher, B.: Actor-oriented design of scientific workflows. In Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O., eds.: Conceptual Modeling ER 2005. Volume 3716 of Lecture Notes in Computer Science., Klagenfurt, Austria, Springer (2005) 369–384
39. Greenwood, M., Goble, C., Stevens, R., Zhao, J., Addis, M., Marvin, D., Moreau, L., Oinn, T.: Provenance of e-Science experiments - experience from bioinformatics. In Cox, S., ed.: Proceedings of UK e-Science All Hands Meeting 2003, Nottingham, UK (2003) 223–226
40. Reuss, T., Vossen, G., Weske, M.: Modeling samples processing in laboratory environments as scientific workflows. In Wagner, R.R., ed.: 8th International Workshop on Database and Expert Systems Applications (DEXA '97), Washington, DC, USA, IEEE Computer Society (1997) 49–54
41. Gallopoulos, E., Houstis, E., Rice, J.: Computer as thinker/doer: Problem-solving environments for computational science. IEEE Computational Science and Engineering **1**(2) (1994) 11–23