

WORK FLOWS IN LIFE SCIENCE

INGO WASSINK

PhD dissertation committee:

Chairman and Secretary:

prof.dr.ir. A.J. Mouthaan, Universiteit Twente, NL

Promotors:

prof.dr.ir. A. Nijholt, University of Twente, NL

prof.dr. G.C. van der Veer, Open University, Heerlen, NL

Assistant-promotor:

dr. P.E. van der Vet, University of Twente, NL

Members:

prof.dr.ir. W.M.P. van der Aalst, Eindhoven University of Technology, NL

dr. T.M. Breit, University of Amsterdam, NL

prof.dr. W. Kruijer, University of Twente, NL

prof.dr. J.A.M. Leunissen, Wageningen University, NL

prof.dr. R.J. Wieringa, University of Twente, NL

Paranymphs:

bc. M.L. van Schie

ir. R.B. Trieschnigg

© Copyright 2010 by Ingo Wassink, Groenlo, the Netherlands

ISBN: 978-90-365-2932-7

ISSN: 1381-3617

DOI: 10.3990/1.9789036529327

Cover picture: Rehhütte Mühlrad, made by Immanuel Giel



Human Media Interaction

Human Media Interaction

The research reported in this thesis has been carried out at the Human Media Interaction research group of the University of Twente.



Center for Telematics and Information Technology (CTIT)

CTIT Dissertation Series No. 09-157

Center for Telematics and Information Technology (CTIT)

P.O. Box 217 – 7500AE Enschede – the Netherlands

ISSN: 1381-3617



Netherlands
Bioinformatics
Centre

NBIC Publication

This work is part of the BioRange program carried out by the Netherlands Bioinformatics Centre (NBIC), which is supported by a BSIK grant through the Netherlands Genomics Initiative (NGI). This thesis only reflects the author's views and funding agencies are not liable for any use that may be made of the information contained herein.



SIKS Dissertation Series No. 2010-02

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

WORK FLOWS IN LIFE SCIENCE

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. H. Brinksma,
on account of the decision of the graduation committee
to be publicly defended
on Thursday, January 14th, 2010 at 15:00

by

Ingo Wassink

born on September 7th, 1981
in Groenlo, the Netherlands

This dissertation is approved by
the promotor: prof.dr.ir. A. Nijholt,
the promotor: prof.dr. G.C. van der Veer,
and the assistant-promotor: dr. P.E. van der Vet

Acknowledgement

This is it; the result of four years hard work, but also a period with a lot of fun and many new experiences. I am happy to present you the results, this thesis. To write this thesis without the help of many people was difficult if not impossible. Therefore I will devote this section to the people that helped me establishing it. I like to thank them by means of a fictitious journey past these people, by starting at my office at the University of Twente to arrive at my home at Groenlo.

I will then start the journey at my office, room number 2120, to thank my room mate and project mate Olga Kulyk. We started our PhD projects at the same day. We had a nice time at this office, a lot of talks and laughings. Together with Dolf Trieschnigg, we followed the bioinformatics course at Nijmegen Radboud University. Together with Wim Fikkert, We went to SloßDagstuhl in Germany and to the British HCI 2006 in London, United Kingdom. The room next to ours is Paul van der Vet's. Paul, I like to thank you for being a great supervisor and assistant-promotor. Our daily walks at 9.30 o'clock were very nice but also very inspiring. The bread sticks we shared during lunch were almost as indispensable as your input was for my PhD project, papers and thesis. I also want to thank you for your ideas and comments about my work and your suggestions how to improve it.

Then, I will move to the room at the opposite of the corridor, which is among others, Dolf Trieschnigg's. Besides being a colleague for the last four years, you are a really good friend for more then twelve years now. In this time, we have had a lot of fun, but also many discussions, work related and non-work related. Sorry for the many times that I used the "Ik zie het probleem niet" argument (I don't see the problem) ;).

A few rooms next to Dolf's, I want to thank Anton Nijholt for being my promotor, the opportunity he gave to me for being a PhD student, the freedom he gave to me for doing my research and the nice time I had at the HMI group. At the secretaries' office, I like to

thank Charlotte Bijron and Alice Visser for their maybe invisible but essential support to enable me to do my research. I also want to thank Hendri Hondorp, for facilitating all ICT-related stuff and for his help in case of technical problems. A few rooms further on, there is another project mate I like to thank, Wim Fikkert. You were a nice colleague, with sometimes a completely different, but inspiring view on things. This led, of course, to many interesting discussions. Another colleague in the BioRange project I like to thank is Betsy van Dijk, for her support in the user studies we did.

Matthijs Ooms, I want to thank you for your help in designing the e-BioFlow workflow system, the improvements you made to e-BioFlow, the provenance system you have designed and the OligoRAP implementation in e-BioFlow. Without you, it was not possible to design, implement and test our workflow system e-BioFlow.

I like to thank Lynn Packwood for her help in the design of the NIWS prototype. Finally, I like to thank all other colleagues, for the fun we had during lunches and the nice talks we had near the coffee machine.

Then I will leave Enschede and take the train to Amsterdam. First I go to the Free University, to Gerrit van der Veer, my second promotor. I like to thank Gerrit for his inspiring talks, his advice and his help in the user study of the NIWS prototype. I also like to thank Elly Lammers. Without her, it was difficult if not impossible to organise meetings with Gerrit because of his full schedule. She was always in for a little talk about project and non-project related matter. Furthermore, I had a nice time with you two, Dhaval and Yujia in Helsinki, Finland during the ECCE 2009 conference.

I will continue to the University of Amsterdam. First of all to thank Timo Breit and Han Rauwerda of the MicroArray Department & Integrated Bioinformatics Unit, for their users' point of view on computer science applications and bioinformatics tools. Additionally, I want to thank you both for the papers we wrote together, the experiments we performed and for sharing our frustrations when implementing the workflow for mapping Vega-designed oligos to the Ensembl assembly ;). At the same university, I like to thank Marco Roos, Informatics Institute, for his help to find participants for the evaluation of the NIWS prototype and for inviting Matthijs and me for the Open Provenance Challenge meeting. From Amsterdam Schiphol, I take a flight to Manchester, to thank Katy Wolstencroft, for her comments on the service categorisation. Additionally, I like to thank Alan Williams for rewriting the RShell plugin to be compatible with Taverna 2.

Back at Schiphol, I take the train to Nijmegen. But first I take a stop at train station Ede-Wageningen to thank Pieter Neerincx and Jack Leunissen, Laboratory of Bioinformatics, Wageningen University and Research, for providing access to their grid web services (in particular the Blat and Bast services) and for providing OligoRAP as a test case for our workflow system. Pieter, I also want to thank you for your comments and suggestions

how to improve interfaces for life science tools, in particular the e-BioFlow user interface.

Once arrived in Nijmegen, I want to thank two persons. First of all, I like to thank Gerrit Vriend for allowing us to participate in the bioinformatics course at CMBI, Radboud University, Nijmegen and to perform a user study during this course. Second, I like to thank Martijn van der Bruggen for enabling me to evaluate the NIWS prototype at the Hogeschool Arnhem Nijmegen (HAN) during a Taverna course.

Then I finally travel to Groenlo, by taking the train from Nijmegen to Zutphen. As usual, I miss the train connection. Then, after 30 minutes, I can take another train to train station Groenlo-Lichtenvoorde. Once arrived, I have to finish my journey by bike. But first, I have to go to Zieuwent. First, to thank my mother for her love and the nice but late friday evenings we had when watching U2 videos. Second, I like to have a moment of silence for my father. Then, just before leaving Zieuwent, I want to hug Rex for being such a great dog.

Then I cycle to Groenlo, to thank my grandmother for being such a young and great grandmother. Of course, I like to thank Jurgen Röring for being family, a friend and especially for the survival trainings we had at difficult moments in my PhD project. I also like to thank Mark van Schie for being a friend for more than 14 years now; for the Cartan, Carcassonne and Stratego matches we played (especially those that I won). Last but not least, I want to thank Ellen Kamp (just Ellen for me) for being such a great girlfriend. I want to thank you for your unlimited love, your patience and your shoulder to lean on in hard times. Ellen, thank you.

Contents

1	Introduction	1
1.1	Bioinformatics: where biology meets computer science	1
1.2	Research context	2
1.3	Research approach and thesis outline	2
I	The life science domain	5
2	Getting to know the bioinformatics domain: an informal introduction	7
2.1	Introduction	7
2.2	User analysis and task analysis in bioinformatics	8
2.3	Questionnaire	9
2.4	Ethnographic observation	12
2.5	Interview	14
2.6	User profiles	16
2.7	Conclusion	18
3	The life science infrastructure	21
3.1	Introduction	21
3.2	Data, tools and web interfaces	22
3.3	Web service composition	25
3.4	Ontologies in the life science domain	28
3.5	Provenance	33
3.6	Workflow systems	33
3.7	Summary	40

II	Life science workflows	43
4	Workflows in scientific experiments	45
4.1	Introduction	45
4.2	Use case: mapping oligonucleotides to a genome assembly	46
4.3	Tasks in scientific workflows	50
4.4	Conclusion	58
5	Workflow reuse and service availability	61
5.1	Introduction	61
5.2	Related work	62
5.3	The data set	62
5.4	Approach	64
5.5	Results	66
5.6	Dealing with broken web services	70
5.7	Conclusion	71
III	e-BioFlow: a new type of workflow system	73
6	Designing workflows using different perspectives	75
6.1	Introduction	75
6.2	Related work	76
6.3	The e-BioFlow editor: a new type of workflow editor	78
6.4	A simple life science case: sequence alignment	87
6.5	Conclusion	89
7	The workflow engine as a movie set	91
7.1	Introduction	91
7.2	Behind the scene: related work	92
7.3	A workflow system as a movie set	94
7.4	The life science movie set	96
7.5	Conclusion	103
8	Capturing provenance data in e-BioFlow	107
8.1	Introduction	107
8.2	Related work	108
8.3	Provenance in e-BioFlow	111
8.4	Use case: OligoRAP	119
8.5	Conclusion	120

9	User investigation on ad-hoc workflow design	123
9.1	Introduction	123
9.2	Studies on scientific workflow systems	124
9.3	NIWS – an adaptive workflow system	124
9.4	Teach-back as a technique for hermeneutic analysis	126
9.5	Assessment of a design envisioning	127
9.6	Results	129
9.7	Conclusion	136
10	Designing workflows on the fly using e-BioFlow	139
10.1	Introduction	139
10.2	The characteristics of an ad-hoc workflow editor	140
10.3	Different perspectives in e-BioFlow	142
10.4	Ad-hoc workflow design in e-BioFlow	144
10.5	Use case: perform a Blat operation	148
10.6	Related work	151
10.7	Conclusion	152
IV	Conclusions	155
11	Conclusions	157
11.1	Summary	157
11.2	Discussion	161
11.3	Future work	162
	Appendices	165
A	Novice user questionnaire	167
B	Symbols used in Taverna workflows	177
C	Use case workflow	179
D	Using R in Taverna: RShell v1.2	181
D.1	Introduction	181
D.2	Implementation	181
D.3	Availability and requirements	184
D.4	Conclusions	184

E Mapping Vega-designed oligos to the Ensembl assembly	185
F Categorisation of local services	189
G NIWS questionnaire	191
Bibliography	199
Abstract	219
Samenvatting	223
SIKS dissertation series	227

List of Figures

2.1	The class room of the bioinformatics course	9
2.2	Preferred view of 3D visualisations of a protein	11
2.3	Layout of the meeting room	13
2.4	Visualisations in the life science domain	16
3.1	Exponential growth of life science data banks	23
3.2	ClustalW web interface	25
3.3	The MOBY-S data ontology	30
3.4	The INB data ontology	31
3.5	Program code vs. workflow implementation	36
3.6	Conceptual model of a workflow	37
4.1	A microarray scan	47
4.2	A workflow for microarray probe analysis	48
4.3	Size of the Taverna workflows stored at myExperiment	51
4.4	Workflow for analysing the workflows stored at myExperiment	52
4.5	Task usage in the workflows at myExperiment	53
4.6	Cumulative task usage aligned to the workflow size.	54
4.7	Use of local services, categorised by functionality	56
4.8	Invocation mechanisms used in workflows	57
5.1	Web services used in Taverna workflows	63
5.2	The SCUFL definition of a SoapLab service	64
5.3	A workflow for finding dead SoapLab web services	65
5.4	The SCUFL definition of a SOAP/WSDL operation	66
5.5	A workflow for analysing dead SOAP/WSDL operations	67

5.6	The SCUFL definition of a MOBY-S service	72
6.1	The e-BioFlow workflow system	78
6.2	Task representation in e-BioFlow	80
6.3	Join and split types used in e-BioFlow	81
6.4	Control flow vs. data flow	84
6.5	e-BioFlow's task panel	85
6.6	The specification controller	86
6.7	The different perspectives on a sequence alignment workflow	88
7.1	Relation between task, role and actor	94
7.2	The life science movie set	98
7.3	Mapping the data flow perspective to YAWL specification	100
8.1	OPM concepts	109
8.2	A workflow for asynchronous Blast service	114
8.3	The provenance browser	116
8.4	The provenance graph of a workflow execution	117
8.5	Perspectives applied to a provenance graph	118
9.1	The NIWS interface	125
9.2	NIWS scenarios	128
9.3	Iterative approach of designing a workflow	131
9.4	Use of worklets	132
9.5	Data transformation as a built-in feature	134
9.6	Scripting and search facilities	135
10.1	The architecture of e-BioFlow	144
10.2	The ad-hoc workflow editor of e-BioFlow	146
10.3	Automatic generation of "used by" relations	147
10.4	The automatic generation of pipes	148
10.5	Screenshots of the ad-hoc workflow editor	150
B.1	Symbols used in Taverna workflows	177
C.1	The expanded use case workflow	180
D.1	The RShell architecture	183
E.1	Number of probes per class	186

List of Tables

3.1	Publicly available life science data banks and databases	22
3.2	Comparison of the data ontology of MOBY-S and INB	32
4.1	Tasks used in the use case workflow	49
5.1	The influence of dead services on workflows	66
5.2	The number of broken SoapLab services	68
5.3	Missing WSDL files and broken SOAP/WSDL operations	68
5.4	Reasons for broken web services	69
7.1	Movie set terms applied to the workflow paradigm	95
7.2	Events generated during workflow execution	104
8.1	OPM profile for e-BioFlow	113
8.2	OligoRAP use case implemented in e-BioFlow	120
8.3	Number of elements created during an OligoRAP run	121
E.1	Classifications of the probes of a microarray chip	187
F.1	Categorisation of local services	190

Chapter 1

Introduction

1.1 Bioinformatics: where biology meets computer science

The introduction of Darwin's theory of evolution in 1851 has dramatically changed the life scientists' view on organisms. One of the central issues in this theory is inheritance: when two organisms reproduce, their children will inherit traits from both parents, such as appearance and behaviour. These inheritable traits are passed from one generation to the next, but never exactly the same.

Evolution is not only found in organisms, but also at the level of communities and organisations. In fact, the biology domain itself has undergone some evolutionary changes. In the nineteenth century, biology married with chemistry, resulting in a new domain called biochemistry, which studies the chemical processes in living matter. In the last decades, biochemistry has undergone a marriage with computer science, which resulted in the birth of the domain of bioinformatics. This young domain has become a subfield of biology that applies computer science technology to solve biological questions.

The bioinformatics domain has inherited many aspects of biology. Its main objective is to study organisms. Like biology, bioinformatics is an observational science [124]. The bioinformatician prefers to use an explorative research approach. She performs the experiments mostly at a computer by using computer programs and writing scripts. These computer based experiments are also known as in-silico experiments. The bioinformatician combines the information stored in the different databases and uses various tools to search, inspect, compare and combine these data. To her, computer science is just the tool to get access to and to gain insight into digital biological information. This might be the reason why the term is "bioinformatics" rather than "bio computer science".

As the amount and complexity of digital biological data grow, the requirements of bio-

informatics tools and databases have changed. The bioinformatician requires new applications to manage and run these complex experiments. Life science workflow systems promise to help the bioinformatician to setup and run complex experiments. Workflow systems enable a bioinformatician to translate his experiments into concrete models, called workflows. A workflow can be shared among peers, to exchange, reuse and adapt knowledge of an experiment. It can be visualised as a graph, whose nodes represent the tools to access or to analyse the data stored in databases and whose arrows define data transfer between nodes and the order of execution.

Although workflow systems should help bioinformaticians to do their experiments, many bioinformaticians experience difficulties using these systems. This thesis is devoted to discover what problems bioinformaticians have using these systems and why they have these problems, and to provide solutions to these problems.

1.2 Research context

This thesis is carried out in the context of the BioRange project. BioRange is funded by the Netherlands BioInformatics Centre (NBIC). The BioRange project formally started in 2005 to promote the collaboration of Dutch research institutes and universities active in the life science domain. This thesis is part of the subproject 4.2.1 “User interfaces for scientific collaboration” in the context of Virtual Laboratories for e-Science (VLe’s).

Within the Human Media Interaction (HMI) group, University of Twente, our main goal in this context is to design interactive interfaces that help life scientists perform their experiments. The appearance of these interfaces can be diverse, ranging from graphical applications that simplify access to and the control of life science resources, to gesture-based interfaces for controlling large, high resolution displays, to a situationally aware meeting environments for life science group meetings [205].

1.3 Research approach and thesis outline

This thesis investigates how bioinformaticians experience the use of workflow systems in their experiments. Current research in workflow systems mostly focuses on the technical aspects of workflow systems. Downey [56] and Gordon and Sensen [79] observed that life scientists experience many problems using these systems. Therefore, our main research question will be:

How can life science workflow systems help life scientists to design, run and capture their experiments?

This abstract research question is translated into concrete research questions:

- Who are the intended users of life science workflow systems?
- What problems do they experience using workflow systems and why?
- How can workflow systems be improved to better fit the life scientists' needs?

Answering these questions requires, first the analysis of the life science domain, the tools used and the workflows constructed, and second, the design of proposals of how to improve life science workflow systems.

The remaining of the thesis consists of ten chapters distributed over four parts. The chapters are ordered in a logical way, but can be read independently. The structure of the thesis is as follows.

The first part introduces the bioinformatics domain. Chapter 2 is an informal introduction for readers unfamiliar with the bioinformatics domain. It is based on our work in Kulyk and Wassink [116], Wassink et al. [215] and Kulyk et al. [117]. This chapter discusses interviews, observations and a questionnaire investigation we have performed among life scientists to gain insight into the daily working practices of bioinformaticians and the tools they use. The results are translated into user characteristics of novice users and expert users of bioinformatics applications. Chapter 3 gives an overview of the current life science infrastructure. It discusses the bioinformatics tools and databases available, and the problems bioinformaticians experience using these tools.

The second part focuses on the use of workflow systems for bioinformatics experiments. Chapter 4, which is based on our work in Li et al. [125], Wassink et al. [220], Wassink et al. [217], describes a use case workflow and discusses the tasks this workflow consists of. Furthermore, it analyses more than 400 workflows to gain insight into what a typical life science workflow consists of. Based on this analysis, it estimates the effect of data incompatibility problems in life science workflows.

Chapter 5 analyses the life span of web services used in life science workflow. It discusses why web services become unavailable and what the effects of these dead web services are on workflow reuse. Additionally, it proposes solutions from a workflow system point of view to deal with dead services.

The third part introduces our workflow system called e-BioFlow. This workflow system is designed to solve many problems mentioned in the prior chapters. Chapter 6,

which is based on our work in Wassink et al. [218], introduces the e-BioFlow workflow editor. This editor differs from other workflow editors in that it provides multiple perspectives on a workflow model. The bioinformaticians can use these perspectives to model both control flow and data flow related information. Additionally, it provides a perspective to design workflow specifications that are independent of the resources available at design time. Therefore, specifications designed in the e-BioFlow workflow editor are independent of web service locations.

Chapter 7 introduces the e-BioFlow workflow engine. This engine extends the YAWL workflow engine [200] and can run workflows designed in the e-BioFlow workflow editor. The engine supports late binding, which means that the actual resources to execute tasks are chosen at enactment time of a workflow. Chapter 8 is devoted to the provenance system of e-BioFlow. Provenance, which means origin, encompasses all the data and meta-data generated during an experiment. The workflow system forms an ideal environment to automatically capture the provenance data of an experiment, because it knows about all resources and all data involved. The provenance system in e-BioFlow uses the Open Provenance Model [138], an open specification for storing provenance data, to remain interoperable with other life science systems. e-BioFlow's provenance system provides a unique interactive provenance browser and query interface to explore provenance data. The e-BioFlow workflow engine is adapted to use the provenance archive as a cache to speed up the execution of tasks already executed.

We have designed a new workflow design interface, which enables the bioinformatician to design workflows ad-hoc. The goal of this new interface is to stimulate creative and explorative research. Chapter 9, which is based on our work in Wassink et al. [219], introduces the mockup implementation of this new interface called NIWS and discusses the user study we have performed using this mockup. The major contribution of this new interface is that users can run unfinished workflows and can use the data, which are explicitly present in the workflow, in their decision making process. The realisation of this new interface is discussed in Chapter 10. This chapter is based on our work in Wassink et al. [216].

The last part, conclusion (Chapter 11), summarises our contributions and discusses their strengths and weaknesses. Furthermore, we give directions for future research.

Part I

The life science domain

Getting to know the bioinformatics domain: an informal introduction

2.1 Introduction

The importance of bioinformatics in the life science domain has grown tremendously over the past years and is expected to do so for years to come. Various experts have to collaborate and to work with shared knowledge. They are forced to use complex scientific applications that require expertise they often do not have. The user interfaces of these tools need to be adjusted to the expertise their users have. Current bioinformatics interfaces are only suitable for expert bioinformaticians. They are too complicated for novice users such as bench biologists [51]. The users' cognitive load is overstretched by huge amounts of heterogeneous data, mutually inconsistent representations, and the complexity of and limited interaction with the user interfaces of bioinformatics tools [20]. A new generation of interactive visualisation interfaces has to meet user requirements as well as to improve the exploration of large amounts of heterogeneous data and to enhance knowledge construction [112; 44; 28]. Therefore, there is a need for user-centred interface design and evaluation in order to improve the effectiveness and efficiency of both visualisation systems and bioinformatics tools [131; 130; 51].

This chapter is devoted to understand the users of bioinformatics tools in their context of work, and how and why they use different information resources and tools. Understanding the users and their work is essential to provide information technology, in particular interactive visualisations [214]. This is also true for the life science domain [46; 215].

We conducted user analysis studies in real life settings in the bioinformatics domain

in order to gain insight into the needs and working practices of researchers active in this domain with various levels of expertise. These studies included a questionnaire, ethnographic observations and interviews. Different target groups were chosen for the studies in order to get different perspectives on users in the bioinformatics domain. For each user group, a different method of study was chosen, based on both the goal of the analysis and the characteristics of the target users. The results of these studies are translated into user characteristics for novice users and expert users of bioinformatics tools, and multidisciplinary teams in the life science domain.

The remainder of this chapter is organized as follows. The next section contains a brief review of related formative user studies in bioinformatics. The third section describes our method and three main target groups. Then, the results presented as user profile descriptions and design implications are discussed, followed by conclusion and discussion.

2.2 User analysis and task analysis in bioinformatics

Most published studies focus on the evaluation of the existing tools (for example Saraiya et al. [166]), but not on the user analysis to formulate requirements. Very few user analysis studies in the life science domain exists in the literature, and none concentrating on multidisciplinary collaboration in bioinformatics. Dunbar performed ethnographic observations and interviews to study cognitive mechanisms and complex thinking, albeit in molecular biology [58]. The only user analysis study in the bioinformatics domain we are currently aware of is the study reported by Barlett and Toms [20]. Their work is based on 20 interviews with bioinformatics analysts working on a functional analysis of a gene. Barlett and Toms proposed an information behaviour framework integrated with task analysis for studying patterns among bioinformatics experts.

Previous studies on creative and complex thinking of life scientists have shown that multidisciplinary in research teams stimulates the process of creative thinking and reasoning [58; 59]. Creativity may be stimulated by providing an interactive environment and an appropriate context to scientists [185]. According to creative thinking theory, there are three stages of creative problem solving: preparation, production and judgement [213]. Visualisation and interaction are part of the problem solving process and can support creativity in all three stages. Optimal perception of the information is especially important in the production stage to support the generation of hypotheses. The challenge at the judgement stage is to design visualization for an optimal perception of the information [213]. We will need to test and optimise the visualisation designs and interaction styles by performing user analysis and iterative evaluations [62; 114; 215].

2.3 Questionnaire

The first target group consists of novice users of bioinformatics tools. The aim of this part of our study is to gain more insight into how these users deal with bioinformatics problems and how do they use bioinformatics resources: What is their working strategy? What is their strategy of getting from the target question towards a conclusion? If they draw conclusions, do they use additional information to verify them? A multidisciplinary group of students taking a nine weeks introductory bioinformatics course at the Bachelor's level offered by the CMBI, Radboud University, Nijmegen, the Netherlands, participated in the questionnaire part of this user study. They had no experience with bioinformatics tools and therefore had no formed opinion about the usefulness of bioinformatics interfaces.

2.3.1 Procedure

Prior to the questionnaire, regular contextual, unobtrusive observations were performed during the weekly bioinformatics course. The environment where students had weekly practical course was a classroom consisting of multiple rows of tables with PCs (Figure 2.1). During this course, students learned how to use different types of bioinformatics resources. We wanted to gain insight into the daily practices of these novice users while they learned to use different bioinformatics tools and dealt with real-life problems.



Figure 2.1: The classroom where the weekly bioinformatics course took place.

The collected observations were translated into simple statements about the way novice users deal with practical bioinformatics problems using different types of visuali-

sations and on-line web resources, both data and tools. Based on these statements, a questionnaire was designed to check and refine these statements.

The questionnaire consisted of three parts: i) background information and general software usage questions, ii) questions on 3D visualization tools, and iii) questions on web-based databanks to obtain biological data. Additional space was left for extra comments after the second and the third part. Fourteen out of twenty-one questions used a 5-point Likert-scale, where '1' was presented as 'Agree strongly' and '5' as 'Disagree strongly'. Three questions were single-choice questions and another two were multiple-choice questions. The last two questions were ranking questions, where users had to rank the options by importance on a scale from 1 to 3. A pilot test with two course assistants was conducted. Based on their feedback, the necessary adjustments to the questions and the layout were made. The questionnaire can be found in Appendix A.

Students were asked to fill in the questionnaire at the end of the last course day. A course assistant explained the purpose of the study and emphasised that participation was anonymous and voluntary. Students were given an introduction on how the questionnaire was constructed.

It took 15-20 minutes for the students to answer the questions. In total 47 of the 52 students took part in the user study, resulting in a response rate of 90%. The respondents consisted of 21 females and 26 males. They were mainly Dutch and German and were all students of the Radboud University of Nijmegen. The students had different backgrounds (molecular science, chemistry and general natural science). The average age of participants was 21.5 years. Based on the user profile questions it became clear that students' level of experience with software tools was generally quite high. The majority of students used the Windows platform and multiple mail programs, web browsers, search engines, text editors, spreadsheets and instant messengers.

2.3.2 Results

The unobtrusive observations of students during the practical assignments of the introductory bioinformatics course showed that students often worked in groups of two to four on the assignments. The course assistants were often asked for explanations about both the material and how to use different bioinformatics tools. The atmosphere during the classes was very informal; active discussions were going on all the time. The students used a wide variety of different software tools simultaneously, e.g. mail program, spreadsheet, web search, messaging, games etc. In addition the electronic course material together with a paper study guide was used for practical assignments.

The exploration of 3D structures of proteins was very important in the course and was

performed a lot. Different tools were used, including Jmol ¹ and Chime ², but Yasara ³ was the most popular. The students easily recognised the structure of a protein. The 3D protein structures gave them necessarily information to reason about the function of amino acids of the proteins. The preferred 3D view in these tools was 'Ribbon' for a complete protein, and 'Balls and Sticks' for parts of a protein (Figure 2.2).

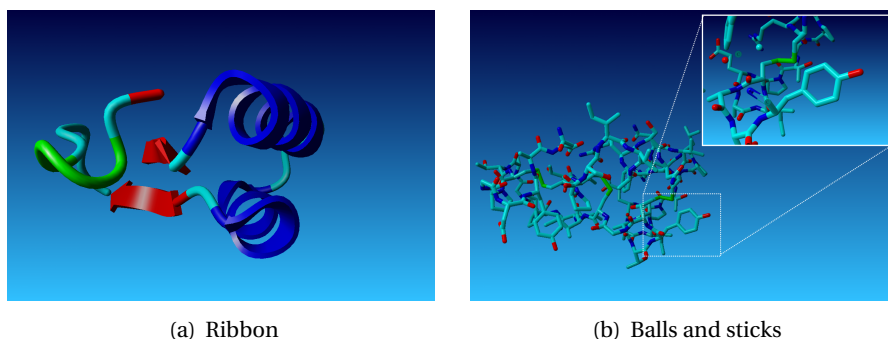


Figure 2.2: Preferred 3D views of proteins by novice users.

Exploration seemed easy for the students: using keyboard and mouse to rotate, zoom and point, they fluently explored the 3D structure. However, students complained that transforming, scaling and moving the molecule was difficult due to interactions that differed from other applications they were familiar with, such as photo editors and 3D computer graphics software.

The students had difficulties to select amino acids, because selection feedback was missing. Therefore, it was also hard to make certain amino acids or larger parts of a protein visible. The students used the functionality provided by the visualisation tools to apply alternative colour-codings for emphasising certain amino acids. They often used the menu options available to show and hide parts of the 3D molecule on demand. Navigating through menus was found difficult too, due to the large number of available options. For many options it was not clear how to use them and help was missing.

The students used mainly web portals to extract data from different databases, in particular MRS [91] and SRS [61]. In the use of online databases, it was often unclear what type of search the databases supported. Many non-ordered options were presented to

¹<http://jmol.sourceforge.net>, last visited: December 2009

²<http://www.symyx.com/products/software/cheminformatics/chime-pro/>, last visited: December 2009

³<http://www.yasara.org>, last visited: December 2009

the user to optimize the search, but also a lot of options were hidden. Novice users did not change options to optimise their search. They had difficulties in using these portals due to inconsistent use of data formats resulting into problems with the user interface.

Cross references were frequently used to obtain more information. Finding *reliable* information was the most important criterion for users when they searched for information. The absence of a history function was problematic when the users were redirected to a different web application.

2.4 Ethnographic observation

The second target group consisted of multidisciplinary teams collaborating on a joint scientific experiment. Such teams consists of scientists from different domains related to bioinformatics, among others molecular biology, chemistry and statistics. Teams of scientists are of special interest, since, as mentioned above, creativity of scientific thinking occurs in groups rather than individuals. The goal of the observation was to gain more insight into how researchers from different disciplines collaborate while solving biological problems and how they use technologies supported by the meeting room environment.

Collaborative creativity involves both individual and group working practices, which introduces a new level of complexity in understanding the target users and designing for their needs [48]. Novice users have little or no experience in collaboration with other researchers. Therefore, a multidisciplinary team of experts was chosen. This multidisciplinary team, consisting of three biologists, two statisticians and two bioinformaticians from different research institutes, had a regular project meeting at the University of Amsterdam, the Netherlands, in which they were discussing the influence of down-regulating and up-regulating genes possibly related to the p53 protein. Some participants were direct colleagues of each other. All participants knew each other from earlier meetings of the same project. The working atmosphere was informal. During the session people were drinking coffee and having candies.

2.4.1 Procedure

Figure 2.3 shows the layout of the meeting room. The whiteboard was behind the screen used for the video projector. The team members could not use the video projector and the whiteboard simultaneously.

Two observers were present during the meeting to make notes and to perform an audio recording. The participants were asked for permission to observe the meeting and to perform the audio recording. These observers sat down at the same table as the team members, but did not interrupt or take part of discussions.

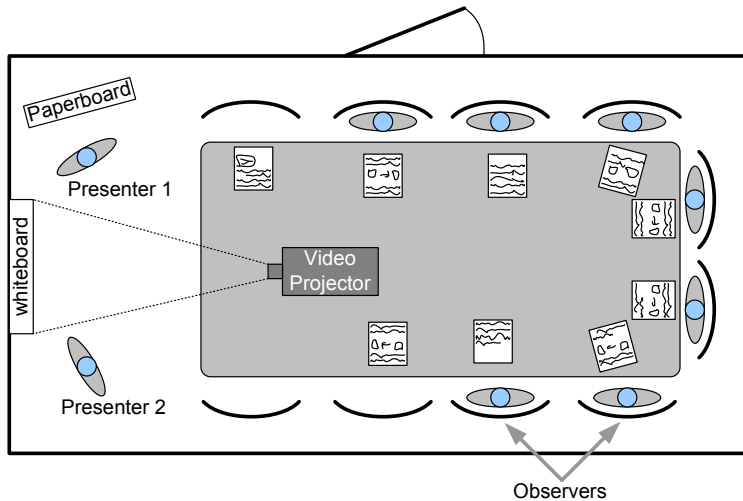


Figure 2.3: Layout of the meeting room where the observation took place.

The notes and the audio recordings were analysed to gain insight in how multidisciplinary teams in the life science collaborate to solve life science problems.

2.4.2 Results

The main goal at the observed meeting was to discuss the statistical steps for analyzing the obtained microarray data. Microarray analysis is a quantitative method to study the simultaneous activity of thousands of genes at a certain point in time. The biologists seemed to be the most active members during the meeting. There was no appointed leader or chairman, neither was there an appointed reporter. The schedule of the meeting was not written down on a paper or projected on the screen; everybody knew the schedule implicitly. The meeting had a clearly discernible structure of four parts: i) presenting the theory (1 hour), ii) presenting the practice (1 hour), and iii) discussion (1 hour), and iv) what to do next (0.5 hour).

Two statisticians together had performed a statistical analysis of the relevant microarray data. They started the meeting with presenting the theory and the practice of the statistical analysis using the video projector and a PowerPoint slideshow. One of the statisticians (statistician A) presented the theoretical part, after this, the other (statistician B) presented the practical part. During the presentation, the presenters could be interrupted for asking questions. This was done frequently. During the practical part, statistician A interrupted the statistician B many times to add additional theoretical information. During

the discussion, the video projector was not used.

The group did the experiment without a clear hypothesis, which is better characterised as a “try and see what happens” experiment. The type of experiments in which the hypothesis is formulated after the data are collected and analysed, is quite common in molecular biology [108; 192]. After the presentation, a discussion evolved about the task distribution and about the not unrelated issue of whether or not the goal of the experiment was to fit the results in a model. From one biologist’s point of view, this was not the case, which was a quite normal reaction, since creating a model of the results is something biologists not always aim at [108].

In general, discussions were very active. However, not all team members were always active during the meeting. It seemed that this depended on the experience with the subject at hand the group member ascribed himself or herself. Everyone was responsible for making his/her own notes. The project team, and one biologist in particular, generated a lot of new ideas, but these ideas had to be worked out in subsequent research and further meetings. It was clear from the observations that the project team could not remember what was agreed on during previous meeting.

Visualisations were frequently used during the meeting. They consisted of diagrams (frequency diagrams, scatter plots), but also custom-made sketches on the paper-board for showing abstract ideas or for explaining something. The pictures in the PowerPoint presentation were static bitmap images. To compare and to interpret diagrams, multiple diagrams were shown at the same time on a single slide. These diagrams differed in populations or in parameter settings. PowerPoint’s zooming function was used enlarge pictures, although this tool did not suit this kind of interaction. Everyone present knew how to interpret the diagrams, but the statisticians were the only ones who knew how to create them using statistical models such as Anova. The slideshow was made available for the participants after the meeting.

The statisticians used the paperboard as a big notepad, during the presentation and the discussion. The sketches they drew using this paperboard were schematic-based instead of text-based. They looked back to previous pages and they used new blank papers for overwriting/clean up certain parts of used papers. The paperboard was also used for writing down the schedule until the next meeting. Although these papers were kept after the meeting, the notes on it were not mailed to the other participants.

2.5 Interview

The third target group were bioinformaticians. Bioinformaticians are expert users of bioinformatics applications. These bioinformaticians were selected to gain insight into their

experiences in collaboration with researchers from other areas and their use of different tools.

The semi-structured interviews were held at the Centre for Molecular and Biomolecular Informatics (CMBI), Radboud University, Nijmegen, the Netherlands employing contextual inquiry technique [27]. In this research group, scientists with biology, molecular biology, bioinformatics and statistics backgrounds work together on various projects. So far, three researchers with different backgrounds working in bioinformatics (two PhD students and one post-doc) were interviewed. The three participants were male and were aged between 25 and 30 years, and were active in the bioinformatics domain for at least 2.5 years.

2.5.1 Procedure

The interview questions were focused on the participants' work practices and on the use of bioinformatics tools. Since observation showed that collaboration is essential in bioinformatics research, we also asked about experts' collaboration experience and opinion on how future technology in collaborative environments might influence collaboration.

The sessions were audio recorded with participants' permission. The full transcripts of interviews with these researchers are omitted for privacy reasons.

2.5.2 Results

One respondent said, "bioinformaticians are not computer scientists, who can build large software architectures, but they know how to program tools to extract biological meaning from databases". They live between the two worlds of biology and computer science. They have the necessary knowledge to collaborate with biologists, which is something computer scientists cannot do or at least not so smoothly. Bioinformaticians develop and use tools to collect huge amounts of data from (online) databases and to analyse these data using statistical techniques.

Bioinformaticians have to work with scientists from other disciplines, because they do not have all the required skills themselves. They are also often collaborating with industries that are highly interested in this type of research, such as pharma and food industries. These industries pose abstract research questions, which are translated by a team leader into several concrete research questions.

Statistical analysis is vital in bioinformatics research to compare and analyse the huge amounts of data stored in databases. One respondent said that at his department, the staff mainly uses MatLab ⁴ for doing statistical analysis. R [98] is a favourite statistical

⁴<http://www.mathworks.com/products/matlab>, last visited: December 2009

tool for many other bioinformaticians.

In addition, bioinformaticians use tools designed specifically for biological data analysis. These tools often provide a lot of parameters to customise their working. Although these parameters make the tools flexible, they also increase the complexity of the tools. As one respondent said this is often unavoidable. Only good documentation can help in understanding of the tool, but most tools lack this. Another respondent remarked that biologists use the default parameter settings most of the time, because they do not have much knowledge about the meaning of the parameters. Bioinformaticians have a different work style. They first try things out to verify a hypothesis or hunch, using the default parameters. If the hypothesis is more or less confirmed, then, second, they fine-tune the parameters to optimize the results.

Colleagues are important sources of information for learning how to use new bioinformatics tools. If they find a new tool by themselves, they test the tool and compare the results with those of familiar tools in order to establish a quality measure. Experience with, trust in, and perceived quality of tools are exchanged among bioinformaticians.

Visualisation of the data is very important to analyse the data. Bioinformaticians use for example, visualisations of 3D structures of proteins (Figure 2.4(a)), sequence alignment (Figure 2.4(b)) and of statistical data (Figure 2.4(c)). One of the interviewees mentioned that visualisation is often underestimated in life science.

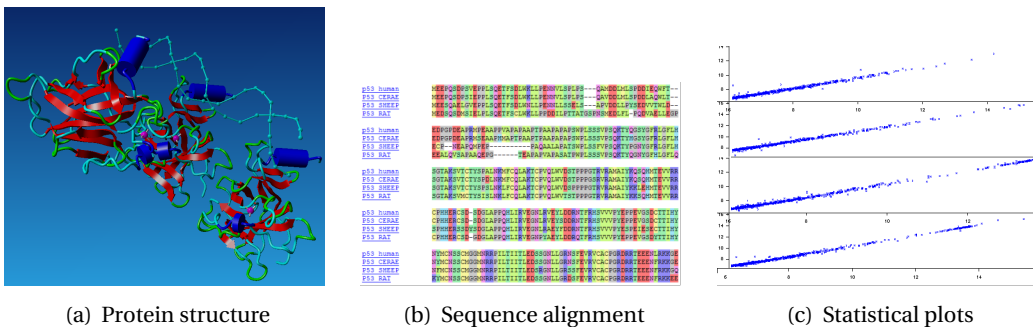


Figure 2.4: Visualisations frequently used in the life science domain.

2.6 User profiles

The user profiles represent two types of researchers using bioinformatics tools: biologists and bioinformaticians. Bioinformaticians can be seen as domain experts in this case.

Biologists are mostly novice users of bioinformatics resources; they are experts in doing wet-lab experiments.

2.6.1 Novice users

The novice users we have analysed are quite skilled and advanced in using general software tools. They, however, lack the programming skills that expert bioinformaticians have. They often do not directly understand how programs work. As a result, they are often discouraged from experimenting with these tools. One of the interviewees stated that biologists for this reason use only default parameters most of the time. This is confirmed by the questionnaire results; less than 22% of novice users change parameters to assess parameter influence on the result of an experiment.

More than 68% of the participants often use cross references for getting more details about experimental results. Novice users miss a history function when they are redirected to a different web application. They get also confused by many unstructured configuration options. Therefore, it is essential to provide an option to make information on demand visible. In addition, each bioinformatics database needs to clearly inform the users about what type of data it provides.

2.6.2 Domain experts

Bioinformaticians are expert users of bioinformatics tools. They know their way around in the vast (and growing) space of online bioinformatics resources, and they know about data handling and operating of bioinformatics databases [158]. They use diverse databases and tools to collect and analyse huge amounts of data and to draw conclusions from them.

Additionally, bioinformaticians have programming skills, and consequently, they understand how programs and tools work and they often know how to extend them. They create and use bioinformatics tools. This makes them less afraid to experiment with different tools and with parameter settings. Console applications are preferred over equivalent GUI or web-based interfaces, since a console allows them to customise all parameters. Their work style can be roughly characterised as follows: they first try things out to verify their hypotheses using the default parameters. When the hypothesis is more or less confirmed, they fine-tune the parameters to optimise the results. Plus, if there is no tool that suits their needs, they extend or adapt an existing tool, or make one themselves.

2.6.3 Multidisciplinary teams

Collaboration is performed in different forms, ranging from working together in the same physical place (in a meeting room but also at the same work floor) and at distance by means of published work, but also sharing tools and data.

Co-located meetings are preferred over virtual meetings. The co-located meetings can be greatly enhanced by a scientific collaborative environment, such as the e-BioLab [158; 205]. In such an environment, technologies, such as large interactive displays, can visualise different data sets or multiple but different views of a same data set at the same time to easily compare and interpret experimental results. Additionally, these displays can be used for discussing the experiment setup and keeping track of the experiment progress, decisions and action points. The interviewees think this can enhance creativity and stimulate discussion, although such displays should not overload users with a lot of results shown at the same time.

2.7 Conclusion

Most bioinformatics tools are very complex, even for domain experts, due to the huge amount of data involved, the large number of parameters that can be set and the lack of documentation to assist users in understanding the interface. Visualisation of biological data is very important in bioinformatics field. It encourages discussion of the design of an experiment and/or (intermediate) results and helps to assess the progress of an experiment. However, visualisation may currently be underestimated in bioinformatics.

The obtained results from our user studies provide a better understanding about the novice users' daily working practices with different bioinformatics tools. The participants of these studies are quite skilled in using general software tools. However, inconsistency in representation, complexity and limited interaction with user interfaces of bioinformatics tools combined cause information overload and time loss for users. Therefore, there is a need for user-centred interface design and evaluation in order to reduce the workload and improve the effectiveness and efficiency of both software tools and web resources use. In addition, it seems to be important to support researchers to keep track of their thoughts and ideas during the information search and analysis [115].

Multidisciplinary collaboration is an essential part of bioinformatics research. The focus is on co-located collaboration in a scientific collaborative environment. The target group for this environment will consist of multidisciplinary scientific teams. Such an environment will contain large interactive displays for presenting experimental results or project progress in order to improve collaboration.

The results of our analysis describe functional requirements for bioinformatics appli-

cations. The bioinformatician uses an explorative research approach. The tools they use in their experiments should support his approach. A history function is needed to keep track of the decision making process. Search functionality should be provided to help the bioinformatician find both biological data and tools. Finding reliable data and tools is the most important criterion used to make a selection. Good documentation of both data and tools is essential to choose from search results. The search engine, of course, should provide access to this information. Current bioinformatics user interfaces overstretch the cognitive load of their users. Better visualisation and support for statistics help life scientists to gain better insight into the large amount and complex biological data.

Although the number of participants limits the generalisability of the findings, the combination of regular observations with other user analysis techniques in real-life settings makes the contribution of this user analysis novel. Further studies with a larger sample from a more diverse population will reduce the current limitation and provide deeper insight into the working practices of novice users, expert users and multidisciplinary teams.

The life science infrastructure

3.1 Introduction

The life science domain has undergone a huge evolution since the introduction of in-silico experiments. The trend is towards “-omics” experiments, which stands for “whole”. For example, genomics refers to the whole genome of a species. Similarly, proteomics and transcriptomics refer to the whole proteome and the whole transcriptome of an organism. Large databases are built to store these “-omics” data [124]. A lot of software tools exist to collect and analyse these data. This chapter gives an overview of the different computer science technologies in the life science domain to perform in-silico experiments.

First, we will discuss the traditional approach, in which console applications and web interfaces are used. Then, we will move to a different approach where tools are translated into web services and where scripting is used to connect these web services. Standardising on data formats has become important to connect web services, especially when they are hosted by different organisations. We discuss the importance of ontologies in the life science domain. With the introduction of in-silico experimentation, a digital version of the lab-report is brought to the computer domain, called provenance data. Automatically capturing the data generated during in-silico experiments is important to assess the reproducibility of these experiments.

The last part of this chapter will be devoted to life science workflow systems. These systems simplify the design and execution of in-silico experiments. Workflow systems have opened many new challenges in the life science domain, such as easy sharing of experiments and automatic capturing and storing of experiment data. However, still, a lot of work has to be done to make them as intuitive as they are intended to be.

3.2 Data, tools and web interfaces

The amount of biological data digitally available grows exponentially since the late eighties and still does (Figure 3.1). These data are stored in large data banks¹, most of them publicly accessible (Table 3.1). The data describes among others (annotated) DNA, RNA and protein sequences, and structural information. Many of the data formats used to store these data stem from the early seventies. The PDB (Protein Data Bank) format [25; 92], for example, is developed in 1971, and uses a format optimised for punch cards. The data are often stored in so-called *flat files* or *ASCII files*. For example, the PDB records are text files with a fixed line length. Each line represents a record field of which the first word describes the field type and the remaining describes the data corresponding the field type.

The data stored in data banks are often uncurated, because curation is costly due to the techniques and expertise required to validate the data. One of the few exceptions is the UniProt/Swiss-Prot protein data bank [190], which has a large team of internal and external experts to ensure the quality of their database.

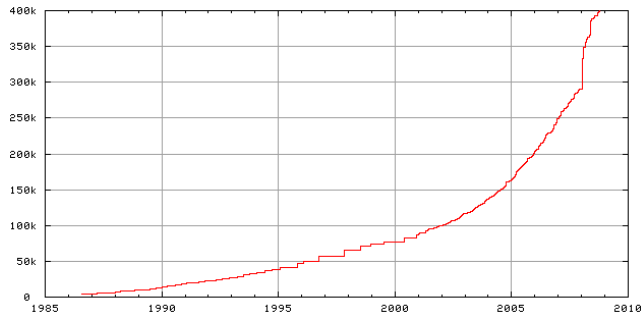
Table 3.1: A subset of publicly available data banks and data bases in the life science domain. Statistics collected at 1 November 2008. (NS = Nucleotide Sequences, AG = Annotated Genomes, PW = Pathways, MS = Macro molecular Structures, PS = Protein Sequences)

Data bank	Info.	Est.	ASCII	Annotation	Curated	# entries
Embl [43]	NS	1971	yes	human	no	137M
Ensembl [66]	AG	1982	yes	computer	no	146M
KEGG [105]	PW	1995	no	human	no	92K
PDB [92]	MS	1971	yes	human	no	50K
UniProt/Swiss-Prot [190]	PS	1971	yes	human	yes	400K
UniProt/TrEMBL [190]	PS	1999	no	computer	no	7M
VEGA [236]	AG	2004	yes	human	yes	64K

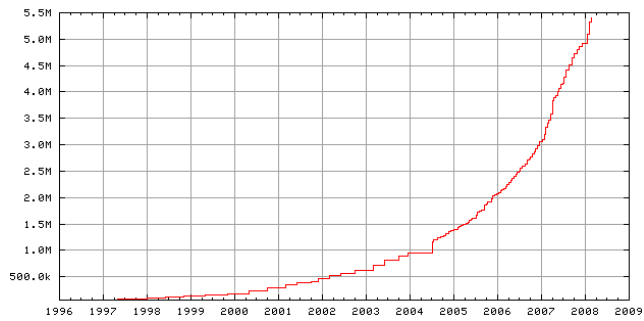
3.2.1 Tools and web interfaces

Tools have been developed to collect and analyse the data stored in these data banks [65]. Most of them are fast console applications. They are highly configurable to fine tune the algorithms implemented in these applications. Due to the large number of parameters that have to be set, they are often difficult to use, especially for less experienced users, as

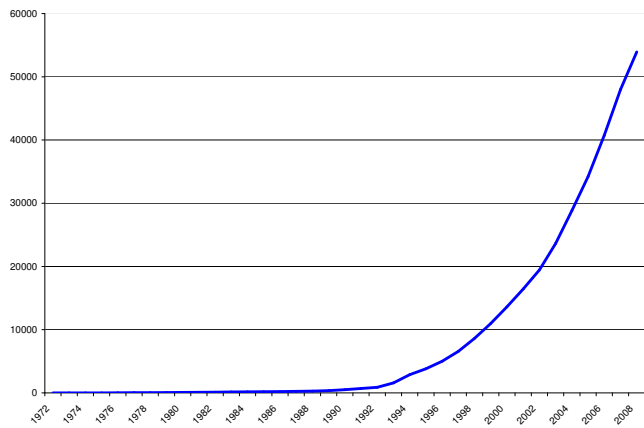
¹The term data bank is used instead of database to denote the usage of a file based storage instead of a database system.



(a) Size of the UniProt/Swiss-Prot protein bank.



(b) Size of the UniProt/TrEMBL protein bank.



(c) Size of the PDB data bank.

Figure 3.1: The exponential growth of three well-known data banks in the life science domain.

discussed in the previous chapter. Well known console applications are the Basic Local Alignment Search Tool (BLAST) [12] and ClustalW [194; 121]. BLAST is used to perform sequence based searches on protein and nucleotide databases. It returns sequences similar to the input sequence and a similarity score. ClustalW is a tool for comparing multiple sequences. It generates a sequence comparison report. Some tools have variants that provide a graphical user interface. ClustalWX [193; 121], for example, is a desktop variant of ClustalW.

Many organisations that host data banks provide web interfaces to the console applications. The console applications these web interfaces provide are preconfigured to have access to the data banks hosted by the organisations. So, the scientist does not need to have a local installation of the software and a local copy of the data bank. The web interfaces are often direct translations of the console applications, and are therefore still difficult to use for the novice user (Figure 3.2).

Within a single experiment, scientists use different tools, provided by different organisations [102]. Transferring data between these tools is difficult: scientists have to copy and paste data between the different interfaces, and possibly reformat the data. This approach is error prone, especially when data sets are large. Some organisations have web portals that provide access to all tools hosted by the organisation. Additionally, these portals store the outcomes of these tools, to simplify using these data as inputs for other tools provided by the web portal. Well-known portals that provide these facilities include the website of the European Bioinformatics Institute (EBI) [36]² and the Entrez website of the US National Center for Biotechnology Information (NCBI) [21]³. But when tools or databases are not accessible within the portal, then the scientist still has to copy and paste between web pages.

3.2.2 Scripting


Another approach frequently used for doing in-silico experiments is scripting. Popular languages in the bioinformatics domain are Perl, Java, Ruby and Python [44; 188]. The bioinformaticians use these for various activities, among others to connect tools, to collect data from the Internet and to perform statistical analyses. Scripting is used to mimic the browser's functionality in order to get access to web pages in a programmatic way. Such a scripts accesses web pages and parses these web pages using screen grabbing techniques. These techniques are unreliable; small changes in the presentation can results in errors [179; 128]. When the data is numerical, they are easy to parse. However, most text on web pages is human readable and therefore difficult to parse.

²<http://www.ebi.ac.uk>, last visited: December 2009

³<http://www.ncbi.nlm.nih.gov/Entrez>, last visited: December 2009

ClustalW2

ClustalW2 is a general purpose multiple sequence alignment program for DNA or proteins. It produces biologically meaningful multiple sequence alignments of divergent sequences. It calculates the best match for the selected sequences, and lines them up so that the identities, similarities and differences can be seen. Evolutionary relationships can be seen via viewing Cladograms or Phylograms.
[New users, please read the FAQ.](#)
 >> [Download Software](#)



YOUR EMAIL <input type="text"/>	ALIGNMENT TITLE <input type="text" value="Sequence"/>	RESULTS <input type="text" value="interactive"/>	ALIGNMENT <input type="text" value="full"/>
KTUP (WORD SIZE) <input type="text" value="def"/>	WINDOW LENGTH <input type="text" value="def"/>	SCORE TYPE <input type="text" value="percent"/>	TOPDIAG <input type="text" value="def"/>
MATRIX <input type="text" value="def"/>	GAP OPEN <input type="text" value="def"/>	NO END GAPS <input type="text" value="yes"/>	PAIRGAP <input type="text" value="def"/>
ITERATION <input type="text" value="none"/>		GAP EXTENSION <input type="text" value="def"/>	
OUTPUT OUTPUT FORMAT <input type="text" value="aln w/numbers"/>		PHYLOGENETIC TREE NUMBER <input type="text" value="1"/>	
OUTPUT ORDER <input type="text" value="aligned"/>	TREE TYPE <input type="text" value="none"/>	CORRECT DIST. <input type="text" value="off"/>	IGNORE GAPS <input type="text" value="off"/>
			CLUSTERING <input type="text" value="NJ"/>

Enter or paste a set of sequences in any supported format: [Help](#)

Upload a file:

Figure 3.2: The ClustalW web interface provided by the European Bioinformatics Institute (EBI) is a direct translation of the console application and provides more than twenty parameters to optimise the sequence alignment.

3.3 Web service composition

Web services form a solution to these problems. They are online programs that provide programmatic access to online resources. They are “a way to achieve software interoperability” [172]. Web service techniques open new challenges in the life science domain, because of the distributed nature of life science resources. Due to the availability of generic invocation protocols, bioinformaticians can easily access web services in their scripts.

Many frameworks today exist, specific and non-specific for the life science domain, to provide life science tools and algorithms as web services. We will discuss the frameworks most used in the life science domain: SOAP/WSDL, REST [64; 63], SoapLab [170] and MOBY-S [234; 233; 235; 188].

3.3.1 SOAP/WSDL

SOAP⁴ and WSDL⁵ is a widely used combination to provide access to web services. SOAP (Simple Object Access Protocol) is an XML based protocol. The WSDL (Web Service Description Language) is an XML based language used to describe web services. A WSDL file describes the interface to a web service, the data it works on and the protocol to access the service (almost always SOAP [152]).

EBI and the NCBI provide a lot of their applications as SOAP/WSDL services [155; 225]. But SOAP/WSDL is not specifically designed for the life science domain, and is used in a much wider area. Therefore, software libraries for almost any programming language are available, which makes it is easy to implement access to services that use these standards. A limitation of SOAP/WSDL is that WSDL lacks the facilities to describe the semantics of services and data [172; 128; 188]. Semantics annotation is important: it helps scientists to decide whether a service is suitable to fulfill a task [85] and it helps them to use services in the correct way [23]. Furthermore, when services are semantically annotated, tools can be built to support users to find and connect services in a correct manner [188; 128].

3.3.2 REST

REST (REpresentational State Transfer) [63] is a data format independent communication protocol based on HTTP. Like SOAP/WSDL, REST is designed not specifically for the life science domain. It becomes more popular in the life science domain. Since REST does not rely on message structuring, like SOAP does, it can better deal with large data sets.

Resources play a central role in REST and refer to data and operations. Uniform Resource Identifiers (URIs) are used to access and modify resources. For example, the URL <http://www.uniprot.org/uniprot/P04637> refers to the p53 Human protein, which has accession ID P04637. Similar, the p53 Mouse has accession ID P02340 and therefore can be accessed through the url <http://www.uniprot.org/uniprot/P02340>. Data can be exchanged in different forms in REST, such as plain text, xml or binary data. For example the URL <http://www.uniprot.org/uniprot/P04637.xml> refers to the p53 protein in XML format and the URL <http://www.uniprot.org/uniprot/P04637.txt> refers to the p53 record in text format.

Of course, there still needs to be an agreement between client and server about the formats used to exchange information. Currently, no description language exists to describe REST services, which complicates programmatic service invocation. A proposal

⁴<http://www.w3.org/TR/soap>, last visited: December 2009

⁵<http://www.w3.org/TR/wsdl>, last visited: December 2009

called Web Application Description Language (WADL) ⁶ is under development for REST, but it is rarely used yet.

3.3.3 SoapLab

SoapLab is a combination of a service registry of life science web services and a library to access these services. SoapLab has been developed to simplify the translation of existing bioinformatics console applications and algorithms into web services. The registry is annotated: it contains service descriptions and information about inputs and outputs of services. The annotations are maintained by EBI. New services cannot be added by third parties, but everyone is free to set up a new SoapLab registry.

The registered services can be accessed using client software API's available for Java and Perl or by means of automatically generated WSDL definitions. These WSDL definitions, however, do not provide all the information SoapLab provides.

3.3.4 Registries for web services

A problem with SOAP/WSDL and REST web services is that they are difficult to discover. A central registry, such as SoapLab provides, simplifies service discovery [179]. It helps scientists to find services based on service name, service type and the name of the authority that provide the service [17]. The UDDI (Universal Description, Discovery and Integration) ⁷ was developed to solve this problem for SOAP/WSDL services. It is a standard registry to discover web services described in WSDL. The approach was not successful and therefore OASIS, Microsoft and IBM gave up the standard in the beginning of 2006 [17].

Existing registries still require the initiative of authorities to register these services into these registries, otherwise services still remain invisible. Neerincx et al. [142] argue for a “Google for web services” and to use web crawler techniques to find and index web services. The situation is getting somewhat better, thanks to BioCatalogue [75] and EMBRACE [154]. Both are projects that aim to provide a curated registry of web services. BioCatalogue is a curated registry. It started with incorporating information from the MOBY-S Central, which will be discussed next, and the manual annotation generated for the Feta plugin for Taverna [127]. The EMBRACE registry continuously checks the status of registered services and automatically notifies service providers in case of a problem. These two projects collaborate with the view to migrating both registries. Whether manual curation of the web services can deal with high fluctuations in trends in bioinformatics and the rapidly growing number of bioinformatics web services, is questionable.

⁶<http://wadl.dev.java.net>, last visited: December 2009

⁷<http://www.oasis-open.org/committees/uddi-spec>, last visited: December 2009

3.3.5 MOBY-S

MOBY-S is a branch of the BioMOBY project intended to simplify the discovery process of and the access to life science services. The MOBY-S system is what UDDI should have been for SOAP/WSDL services: a registry to easily discover web services hosted by different organisations. The registry consists of three ontologies to describe web services, data types and namespaces. The ontologies are related: the data types are used in the service ontology to describe the input and output of the services. The namespace ontology is not a real ontology but rather a collection of namespaces. It is used to disambiguate data types and services with the same name. The registry, better known as the MOBY Central, is open access; everyone is free to register new services, data types and namespaces. The open access registry simplifies registering new services and finding these services. The drawback is that without a curator, such a registry easily gets plumbed with duplicated or outdated service and data definitions [81]. The MOBY-S ontologies will be discussed in more detail in section 3.4.

Turning existing applications into a MOBY-S service is simple and can be done with a few lines of Java code, due to the existence of libraries such as the one created by Gordon et al.[81]. Client libraries to access the MOBY-S framework are available for Perl, Java and Python. These libraries provide facilities to search for services based on service provider, service information and/or input and output data. Various tools are available to access the MOBY-S Central. GBrowse MOBY [232], REMORA [40] and SeaHawk [80] are some examples of tools that provide access to the facilities of the software library by means of a graphical user interface. These tools enable life scientists to “knit” BioMOBY services together.

3.4 Ontologies in the life science domain

Exchanging data plays an important role in the life science domain. Scientists combine information stored in sources provided by different organisations. These organisations more often than not use different and incompatible formats to represent the information, which complicates reusing it.

The lack on standardised formats has resulted into a situation where data transformation takes a large part of the daily activities in in-silico experiments. The different BLAST tools, for example, accept different standardised input formats, such as FASTA; the format of the output, named a BLAST report, is not standardised at all. The size of this problem is further discussed in Chapter 4.

When different organisations share their information and reuse that of others, ontologies can help. An ontology is part of knowledge representation, and is a social contract

between the different parties involved [178; 184].

3.4.1 The MOBY-S ontologies

As mentioned before, MOBY-S uses three ontologies to describe services and data. The namespace ontology is not a real ontology, but just a collection of namespaces. The other two ontologies, the data type ontology and the service ontology, are taxonomies. A closer look at these ontologies shows some but important problems that strongly limits the usability of this framework.

The data type ontology is very large. It has almost seven hundred concepts (Table 3.2). The main reason is that everybody can add new concepts. The BioMOBY Consortium provides some guidelines how to register and describe services and data types to limit the problems⁸. Following the guidelines, the ontology should have one root concept called “Object”. In reality, it contains almost forty additional root concepts. Many concepts are represented multiple times in the ontology. For example, basic types, such as plain text, formatted text and XML text, are present twice (Figure 3.3). Another problem is that syntactic information and semantic information is combined within the same taxonomy. The protein sequence concept, for example, is a sibling of the text-formatted concept. Semantics information refers to the conceptual meaning of data, for example a protein sequence. Syntactic information, on the other hand, refers to how data are represented, for example, using single-letter encoding [30]. Different syntactical structures can be used to represent the same semantic type.

Some improvements are made by the Spanish Instituto Nacional de Bioinformática (INB)⁹. They provide simpler, curated ontologies. The number of concepts in the data ontology is reduced to less than three hundred (Table 3.2), by removing a lot of redundancy (Figure 3.4). The taxonomy, however, still contains multiple root nodes and still combines syntactic and semantic information.

The service ontology categorises services based on the type of operation they perform. These service types are just place holders. Ideally, they should act as a contract of the service interface. A service type should define the data types the service should consume and produce, and the operation it performs, similar to what interface classes in object oriented programming languages are. Then, the services of a certain service type are implementors of the interface corresponding to the service type. This would closely match the original idea of the semantic web [172]. In general, this is not the case. Servi-

⁸http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Docs/MOBY-S_API/Perl/DesignAnObject.html, last visited: December 2009

⁹<http://www.inab.org>, last visited: December 2009

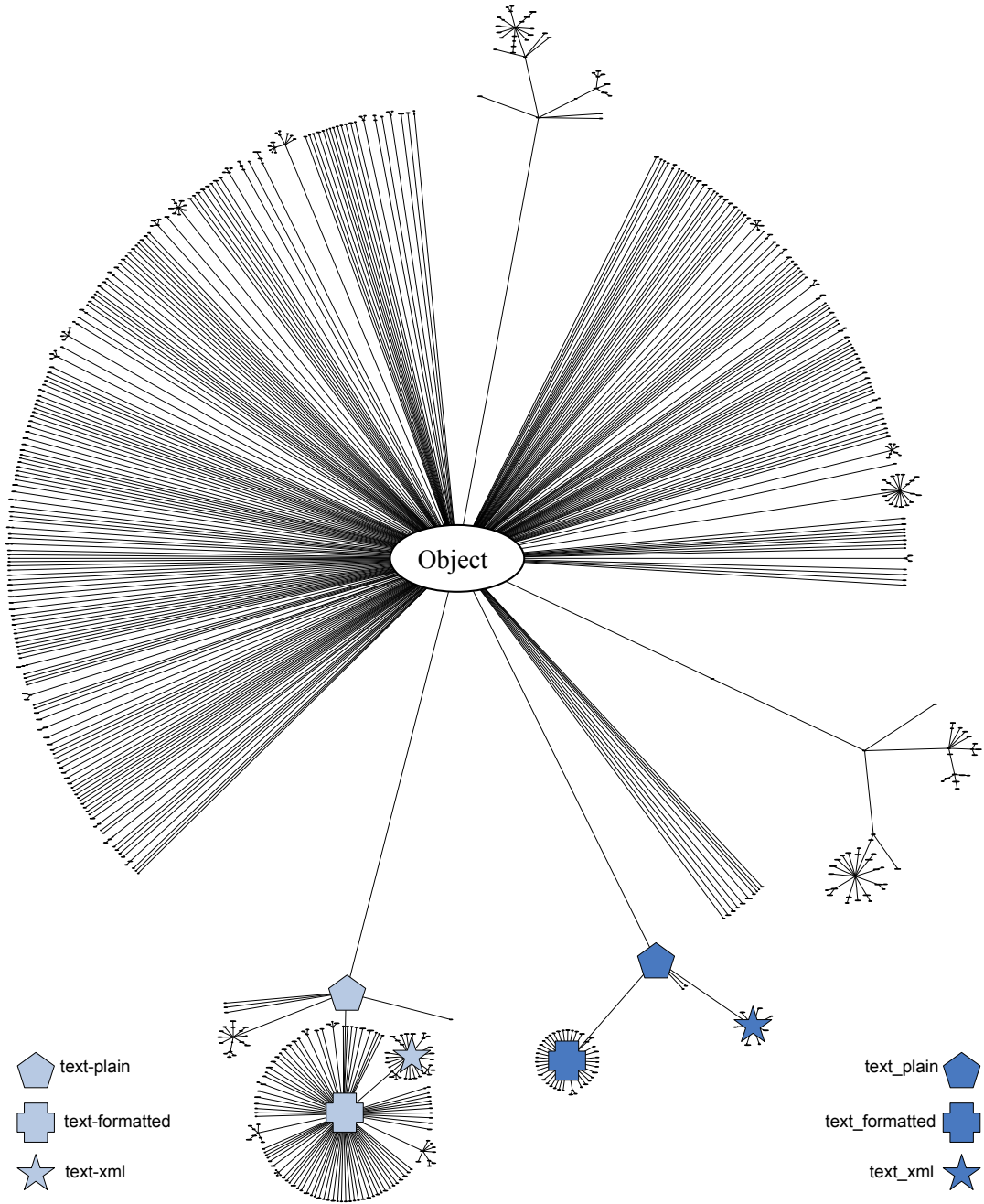


Figure 3.3: Tree structure of the MOBY-S data ontology. The ontology contains 698 nodes, of which 39 are parentless nodes (not shown). Multiple nodes exist to represent the same concepts. Three ambiguous concepts (plain text, formatted text and XML text) are emphasised using enlarged shapes. The data are collected in October 2008.

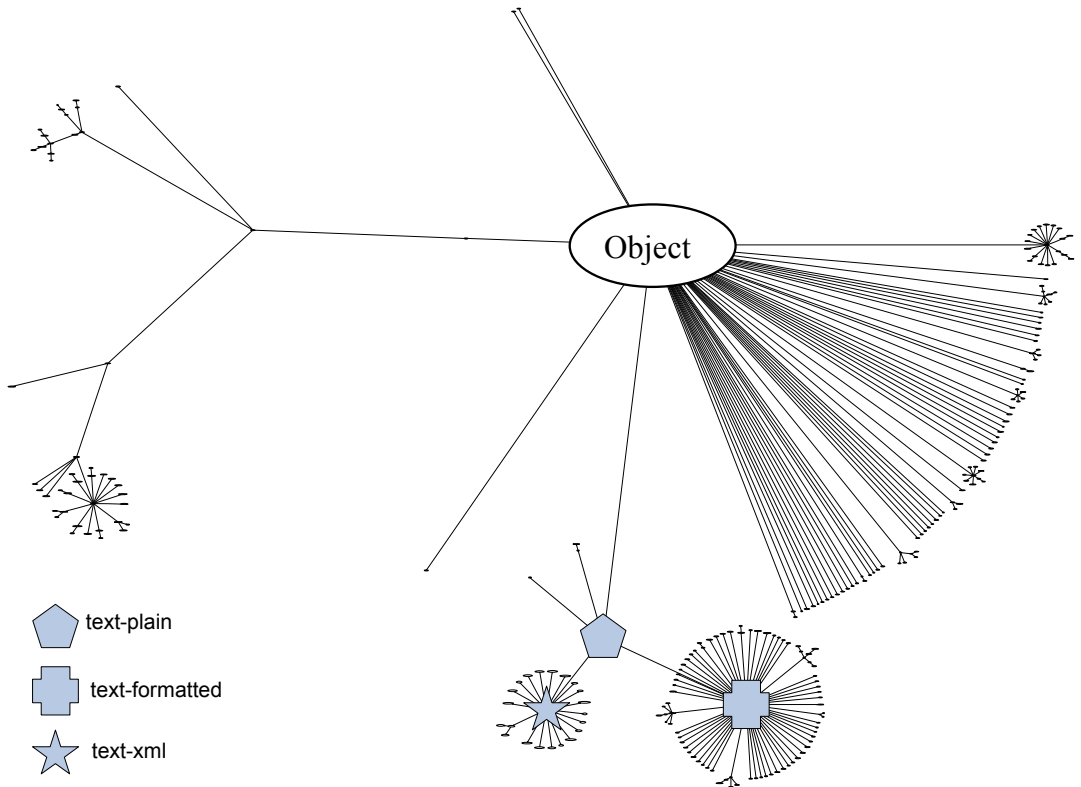


Figure 3.4: Tree structure representing the curated ontology of the Instituto Nacional de Bioinformática (INB). A lot of redundancy is removed. It contains 276 nodes, of which 15 parentless (not shown). The same concepts as in figure 3.3 are emphasised. The data are collected in October 2008.

Table 3.2: A comparison between the data ontologies of BioMOBY and the Spanish Instituto Nacional de Bioinformática (INB). The data are collected in October 2008.

	BioMOBY	INB
Number of concepts	698	276
Number of roots	40	16
Maximum tree depth	9	9
Mean tree depth	2.20	2.66
Standard deviation	2.70	3.16
Maximum number of children	265	59
Corresponding concept	Object	Object
Mean number of children	5.67	4.46
Standard deviation	26.22	11.91

ces of the same service type do not guarantee to consume the same inputs and to produce the same outputs, even if they point to the same application [239]. This is also true for the service ontologies provided by INB.

The main reason for these problems is that the MOBY-S Central is uncurated. Everyone is free to register new services and data types. It results in unstructured ontologies [128] and poorly annotated services [69]. Therefore, the term ontology is somewhat misplaced here. The registry of MOBY-S is realised by many one-sided agreements: Each organisation provides its own data types definitions and service definitions.

3.4.2 Other ontologies

The S-MOBY (Semantic Moby) [233] and Moby 2.0 [208], other branches of BioMOBY, promise to be solutions to describe web services and data semantically, but are still proposals. Another potential solutions is SAWSDL [211]. All of them can only succeed if stakeholders collaborate and come to an agreement instead of each creating ontologies in isolation [128]. Progress has been made at smaller scale. Groups working in the same area have developed ontologies to simplify data exchange and to create universal access to resources. Examples are the GeneOntology [189], TAMBIS [15], and the MGED ontology [226]. Each of them is applicable to a specific area of the life science domain. Most ontologies are, however, created in a quick and dirty way, which still results in data incompatibility between different groups [176]. This results in unstable ontologies in which frequently new concepts are added and existing concepts are removed [89].

3.5 Provenance

Life scientists have to provide a trace of their experiments in order to keep their results reproducible [174]. This means that a paper should include or refer to a complete trace of the experiment on which the conclusions of the paper are based [95]. Indeed, in the *Materials and methods* section of the average organic chemistry paper, for instance, one finds details such as the supplier of the chemicals and the brand and type of the apparatus used to assess the purity of the product.

In in-silico experimentation, there is a trend towards more complex experiments that produce more data [104]. “-Omics” research is a good example of this trend. For example, a single microarray experiment can yield up to a million data points, and both the wet-lab and dry-lab parts consist of several steps, each with its own choices and settings. Providing a trace for such experiments becomes very difficult. Yet, in this field as in any other field, the trace fulfills a vital function in the quality assurance of the scientific output [70]. In “-omics” research, the trace is commonly called *provenance*, elsewhere often *lineage* of scientific data. Its functionality is close to that of the traditional lab report. See [29; 175] for an overview.

The huge amount of data generated in in-silico experiments, requires a system that can automatically store provenance data in a universal and exchangeable format. Like the traditional lab journal, provenance should not only contain the data, but also the origin of the data (services) and release information about the tools, databases and algorithms used. The Minimal Information About a Microarray Experiment (MIAME) [34] is one of the approaches to create exchangeable reports, in this case about microarray experiments. The Open Provenance Model (OPM) [138] is more ambitious and tries to develop a standard for storing provenance. The OPM will be discussed in more detail in Chapter 8.

3.6 Workflow systems

Scientific workflow systems offer a new way for scientists to create and run their in-silico experiments. Workflows form an alternative to scripting for orchestrating web services. A workflow system is a tool to define and execute workflow specifications or just workflows [123]. The published literature on workflows and workflow systems is not entirely consistent in its terminology. We will mostly follow the terminology of Van der Aalst and Van Hee [203] and Jablonski and Bussler [99]. Thus, we speak about *processes* that are modeled as workflows. A workflow decomposes a process into *tasks*. In a *hierarchical workflow*, a task itself can be an entire workflow. Tasks that can be decomposed are called *composite tasks*; those that cannot *atomic tasks*. Atomic tasks are assigned to *resources*.

Hierarchy enables one to deal with large models and to define processes at different levels of abstraction [199]. The term *workflow editor* refers to the system to compose a workflow. The *workflow engine* refers to the system that can *instantiate* and *run* a workflow. A workflow can be instantiated multiple times. The workflow editor can be separated for the workflow engine, although a central format is required to exchange the workflow definition [156].

The role of a workflow system is twofold [55]: it co-ordinates tasks, and it helps to explain why these tasks have to be done and why they have to be done at that moment. According to a common definition, a workflow represents the activities involving the coordinated execution of multiple tasks by different resources, among others, machines, services, humans [164; 113; 183]. The workflow paradigm encompasses the so-called management paradigm: co-ordinating and delegating tasks without necessarily understanding their technical details [224]. Tasks are black boxes from the perspective of the workflow system. White and Fischer [227] define five functions of workflow specification: i) *work distribution* (define which tasks are responsible for what type of work), ii) *work routing* (send results of one task as input to other tasks), iii) *work prioritising* (what should be done first), iv) *work tracking* (who does what c.q. who is doing what), and v) *management reporting* (reporting provenance).

A workflow system structures work and therefore limits the freedom to order tasks for execution. It relies on other resources for completing its job. So, the perceived quality of a workflow system can never be better than that of its external resources. But still, a workflow system can form the ideal software architecture [186]: it is flexible, extensible and adaptable to new technologies and standards, due to its service oriented nature.

3.6.1 Scientific workflow systems

Workflow systems have been used in wet-lab experiments in which they are called Laboratory Information Management Systems (LIMS). The purpose of a LIMS is to offer traceability of every step in a laboratory experiment [132]. LIMS technology originated in analytical chemistry; it is now a mature technology offered by many commercial vendors. The typical LIMS has a representation of the workflow of at least one type of experiment in the sense that the steps of the experiment and their mutual relations are represented. The workflow representation, however, may be rather implicit. A workflow is either statically built into the system, or it can be changed but only at great effort [161]. Therefore, LIMSs are found to be inflexible.

There is continued interest in improving LIMS technology in the scientific community. LIMSs for specific applications currently under development cover genomics [100] and proteomics [140], but are for the moment only research prototypes. Two other re-

cent research systems address microarray experiments: BioArray Software Environment (BASE) [165] and Laboratory Information Management for Array Systems (LIMAS) [223]. They are designed to track, organise, and analyse the data produced by microarray experiments. The main limitation these systems share with the majority of LIMSs is that they cannot delegate tasks to external resources, be it through the Internet or otherwise. Recently, there is interest in workflow systems for so-called e-science and the grid [93; 94; 245]. Workflow systems are envisioned to connect and automate the steps of running programs remotely (e.g., BLAST), extracting data from web data resources, and combining the data with data from other sources [31; 147]. They are intended to help molecular biologists to choose from the huge number of data resources, web services and tools available [46], how to configure them [116], and how to combine these computational resources into a meaningful whole [9; 54]. New types of infrastructures such as *my*Grid [9; 74], GridLab [7], Ptolemy II [126], Pegasus [54], and VLAM [5; 24] address these problems. These infrastructures are also known as grids; their focus is on distributed computing. These projects all have at one time or other addressed the workflow problem. This has given rise to well-known tools for scientific workflows such as Taverna [145; 146; 147] (originated in the *my*Grid project), Triana [134; 187] (associated with the GridLab project), Kepler [11; 129] (built on top of Ptolemy) and WS-VLAM [229; 230]. Pegasus [54] and Knime [26], on the other hand, are designed with workflow functionality from the start.

The four workflow systems Taverna, Triana, Kepler and Pegasus focus on in-silico experiments. In other words, the workflow is a largely automated process involving access to data and computational resources over a Grid and/or over the Internet. Only Taverna originates in the bioinformatics community: its development is coordinated at the EBI. The other projects mention also applications in geology, astrophysics, and quantum chemistry, among others. This is perhaps the reason why Taverna is the most popular workflow system in the “-omics” community. In all four systems, stable and transparent access to remote resources has been given special attention. Each of the systems has particular strong points in this respect. Taverna has its Scavenger technology for automatically finding web resources at predefined locations [146]. It has support for many invocation mechanisms, among others, SOAP/WSDL, MOBY-S and SoapLab. Kepler’s Harvester component has almost equal functionality [11]. Both the Scavenger and Harvester technology help the scientist to easily discover and use web services. Triana and Pegasus put more focus on automatic workflow composition and the scheduling of tasks over the resources available on the grid.

A main advantage of a workflow over its scripting equivalent is that it can be visualised. Most workflow editor visualises the workflow as a graph. The web services accessible through the grid form the building blocks of the in-silico experiment and are the

nodes of the graph. [79; 174]. The arrows in life science workflow graphs represent data transfers between services and the same time they implicitly define the execution order of tasks. The resulting workflow is therefore often easier to understand than an equivalent script (Figure 3.5). Though, at the same time, often a big gap exists between the actual implementation and the scientists' conceptual model of the workflow [4; 95; 210; 86] (Figure 3.6). The scientist does not think about the resources available, but in terms of the tasks he wants to perform. So, the conceptual model focuses on the meaning of the workflow and abstracts from implementation details in terms of resources [210]. The explanation of the symbols used in Taverna workflows can be found in Appendix B.

```
import java.util.Map;

import org.biomoby.client.*;
import org.biomoby.shared.*;
import org.biomoby.shared.data.*;

public class GetGenBankff {
    public static void main(String[] args)
        throws Exception {

        // load moby central
        Central worker = new CentralImpl();

        // get service
        MobyService templateService =
            new MobyService("MOBYSHoundGetGenBankff");
        MobyService validService =
            worker.findService(templateService)[0];

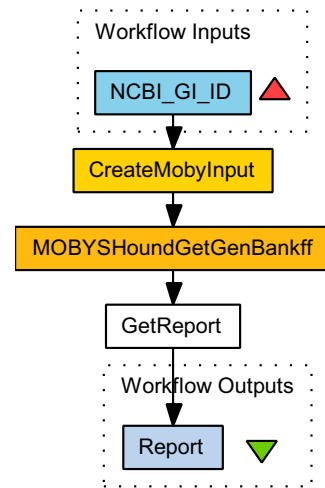
        // create input data
        MobyRequest mr = new MobyRequest(worker);
        mr.setService(validService);
        mr.setInput(new MobyDataObject("NCBI_gi", args[0]));

        // invoke service
        MobyContentInstance result = mr.invokeService();

        // return output
        MobyDataInstance dataInstance =
            result.get("1").getData()[0];
        Object content =
            ((Map) dataInstance.getObject()).get("content");

        System.out.println(content);
    }
}
```

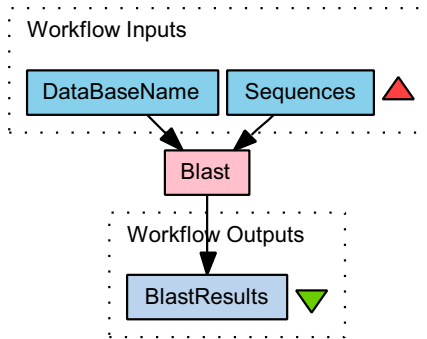
(a)



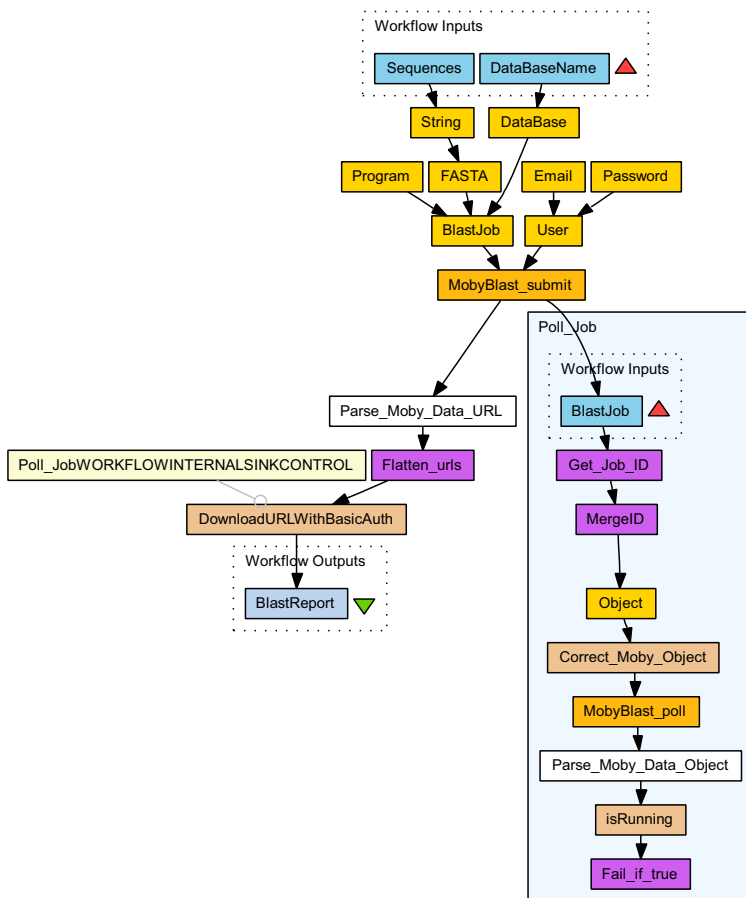
(b)

Figure 3.5: (a) Java code for invoking the MOBY-S GetGenBankFF service, (b) Taverna workflow invoking the same service.

Another advantage of using a workflow system instead of writing a script is that the



(a) The conceptual model of invoking a blast service.



(b) An implementation of invoking a blast service.

Figure 3.6: The implementation of a workflow in terms of services (b) is far more complex than the conceptual model (a).

a workflow system is an excellent tool for collecting, storing and sharing provenance data [84; 243; 175; 45; 16]. It has knowledge of the resources and data involved in a workflow execution. After all, LIMSs were designed for this reason. If one has a workflow system perform the automated steps of the experiment, provenance collection can be part of the execution. In the dry-lab environment, a workflow system can store both details about the particular instance that was run, and all data, including settings and intermediate results [4; 18]. Most scientific workflow systems provide the ability to automatically store provenance data, for example Taverna [242], Kepler [10], and Triana [134] can do this. Chapter 8 will go into further detail about provenance support for workflow systems.

3.6.2 Life science workflow systems vs. Business workflow systems

Workflow systems are not new: the business community uses workflow systems already for a long time for orchestrating business processes [18; 17]. There are many languages to describe these processes, of which the most important examples are Web Service Business Execution Process Language (WS-BPEL) [13], Event Process Chain (EPC) [109] and Yet Another Workflow Language (YAWL) [202]. Since 1993, the *Workflow Management Coalition* (WfMC) ¹⁰ organises meetings for academics, suppliers and customers of workflow systems. It has issued and maintains the XML Process Definition Language (XPDL) standard, an XML-based language for representing workflows ¹¹.

Both business workflows and scientific workflows describe processes, but they differ in the ways they are used [18; 70]. Business workflows are often data scarce. Within a single business workflow, services of a small group of service providers are used. These services are better standardised on data formats. The focus in business workflows is on the control of long term business processes [17; 18]. Therefore, most business workflow systems are control flow oriented.

Scientific workflow systems try to simplify the construction of an in-silico experiment. Life scientists prefer to think in terms of a data instead of processes [64; 78; 212; 17; 18; 195]. Therefore, most scientific workflows use a data flow oriented approach [17; 18]. These workflow systems need to be more flexible and require more exploration capabilities. The average scientific workflow is data intensive and has to deal more with heterogeneous data formats, because services provided by many different organisations are used. The life science workflow system has to be capable to deal with these large, heterogeneous data sets.

Ludäscher et al. [129] argue that there are significant differences between the control flow oriented workflows in business applications and the data-oriented workflows of sci-

¹⁰<http://www.wfmc.org>, last visited: December 2009

¹¹<http://www.wfmc.org/xpdl.html>, last visited: December 2009

entific computation. The control flow oriented workflow is more powerful with respect to control flow patterns, such as synchronisation, parallel execution and iteration [201]. These patterns form basic requirements for life scientists to model their experiments [4]. A data flow oriented workflow system has to incorporate special constructs to handle those cases in which the data flow does not determine execution order [1]. A control flow oriented workflow is more powerful than a data flow oriented one. It is easy to piggyback data flows on top of control flows because a data flow always entails a control flow. The converse is not directly possible, because there can be a control flow without associated data.

Most workflow design systems are either control flow-oriented or data flow-oriented. The ideal solution is a combination of data flow and control flow work flow systems, known as *hybrid workflow systems* [173]. A hybrid workflow system is more powerful than just either a control flow-oriented system or data flow-oriented system, since it provides a data-oriented interface to model workflows combined with advanced control flow structures [173]. In Chapter 6, we will discuss the hybrid workflow system called e-BioFlow, which we have developed for the design and execution of scientific workflows.

3.6.3 Formal aspects of workflow languages

When the number of tasks in a workflow grows and hierarchy is used, workflows can become complex. A workflow system that uses a workflow language based on a formal model can help its users to design valid workflows. These models can be checked from a control flow perspective and a data flow perspective. The language of Petri nets [153] is a well-known example of a formal (control flow) language. An important property from the control perspective, for example, is the *soundness property* [199], which can be explained as [206]: “A [workflow] is sound iff every [reachable state] can terminate properly.” Ouyang et al. [149] use Petri nets to check the soundness property of workflows designed in WS-BPEL workflows. The YAWL language mentioned above is a variant of the Petri net formalism. A tool with the same name, used to construct YAWL workflows [200], enables the user to check workflows written in this language on the soundness property.

Formal models can be used in scientific workflow languages, to model and check the data transfer between services. Data flow-oriented workflows can be checked on the type of data (both syntax and semantic) and the cardinality of data [23]. Coloured Petri nets [101] is one of the Petri net variants suitable to model and check these properties. In coloured Petri nets, data correspond to coloured tokens that travel through a graph of which the topology is determined by the control flow.

Taverna’s Simple Conceptual Unified Flow Language (SCUFL) is claimed to be based on Lambda calculus [197]. Taverna is a real data flow language, and hence, does not sup-

port iteration. Neither does it support recursion, in spite of Taverna's alleged foundation in Lambda calculus. Kepler uses a Process Network based model [103] as a formal model for synchronous data flows. Although both Taverna and Kepler claim to use a formal workflow language, neither enables its users to check their workflows on formal correctness, not from a control flow perspective, nor from a data flow perspective.

3.6.4 Workflow sharing

Workflows form a new type of resources themselves [85]. Scientists publish and share workflows to exchange knowledge and to construct similar experiments [70; 135]. The myExperiment web site [72; 76]¹² is designed with this in mind. myExperiment is a kind of Facebook or mySpace environment, where scientists can publish their workflows and reuse and repurpose (fragments of) existing workflows [78].

When scientists reuse, modify or embed workflows designed by others, a formal workflow language will be even more important. A workflow language based on a formal model can help a scientist to check the formal correctness of a workflow. When a workflow is repurposed, or embedded in another workflow, a formal model helps to check whether the formal correctness of the constructed workflow still holds.

A workflow that is shared need to be sufficiently generic and should serve as an executable template and an ontology at the same time. Workflow systems need to provide facilities to abstract tasks from resources (services) until instantiation time [85], which is known as *late binding*. When late binding is applied, linking tasks to resources is delayed from design time of the workflow until instantiation time. Late binding plays an important role in a service oriented architecture, such as workflow systems. If a service is unavailable or if better alternatives exist, it needs to be replaced [85; 78]. A workflow system that supports late binding benefits of a formal language. The workflow system can use the formal definition of tasks to find alternative, compatible services. Additionally, it can analyse the consequences of the soundness property of the workflow when no resources are available to execute the task [206; 207].

Workflow reuse and late binding will be discussed in more detail in Chapter 5 and Chapter 7.

3.7 Summary

The amount of biological data stored in online databases grows exponentially. Many console applications and web interfaces are available to collect and to analyse these data. The interfaces are only suitable for small scale experiments and do not fulfill the requirements

¹²<http://www.myexperiment.org>, last visited: December 2009

of “-omics” experiments. Life scientists prefer to create and use scripts to connect databases and algorithms, embedded as web services and tools. Transparent access to these services is difficult. Services use different protocols and different data formats. Furthermore, keeping a trace of the experiment in a universal data format is difficult and often ignored. The life science domain works towards the use of ontologies to model data and services. Most ontologies are created in isolation and different parties use their own formats. This complicates data exchange between the services. Therefore, a lot of time in an in-silico experiment is spent on data transformation. Problems of data incompatibility can only be solved if organisations agree on standards and use them [102].

To design, and run in-silico experiments, a workflow promises to be the ultimate experiment representation. It functions as a template for a type of experiment, ideally independent of the resources available at design time. A workflow can easily be exchanged with peers to share experiment knowledge [41]. A workflow system is the tool that can be used to design and run workflows. A workflow system has to fulfill many functional requirements to support the bioinformatician in performing his experiments. It needs to provide uniform access to the various bioinformatics web services hosted by different organisations using different protocols. The workflow system should provide search functionality to help the bioinformatician to find the required web services.

Current life science workflow systems are data flow-oriented, and therefore do not support advanced control flow structures. Business workflow systems support these control flow structures, but cannot deal with large, heterogeneous data sets. The solution ideal solution will be a hybrid workflow system, that is able to model both control flow-oriented aspects and data flow-oriented aspects of a workflow. Both can benefit from a formal model, which enables one to check the workflow on formal correctness. This is especially important when workflows become large or when workflows are shared.

Once a workflow is designed, the workflow can be run by the workflow engine. The bioinformatician would benefit from a provenance system that automatically stores information about the data and web services involved in the workflow run. The bioinformatician needs these data to keep his experiments reproducible. The workflow system should provide an interface to gain insight into provenance data in order to verify the experiment results.

Part II

Life science workflows

Chapter 4

Workflows in scientific experiments

4.1 Introduction

In the previous chapter, we have discussed what workflows and workflow systems are and what they are used for. A workflow system provides uniform access to tools and web services that can be used to build a workflow. This chapter is devoted to clarify what life science workflows consists of.

In 2005, Lord et al. [127] have investigated the bioinformatics web services available to *my*Grid users, based on the invocation mechanism used. They found that 25% of the services available are SoapLab services, 30% are MOBY-S services and 30% are REST services. Only 5% of the bioinformatics services available are SOAP/WSDL services. The remaining 10% of the services available are workflows themselves: *my*Grid enables the incorporation of workflows into larger workflows. Life science workflows can be complex with respect to the number of tasks. When workflows become more complex, hierarchy becomes important to keep the workflow comprehensible [53]. We expect large workflows to have more subworkflows than small workflows.

Different organisations use different formats to represent data. This has resulted in a situation where it is difficult, if not impossible, to pass data produced by one web service to another web service without restructuring, especially when the web services are hosted by different organisations [239; 142; 23; 106; 174]. Workflow systems provide local services and scripting tasks to help bioinformaticians to connect web services. Local services are tasks provided and executed by the workflow system itself. They can be compared to single statements in a programming language and enable “visual programming” within the workflow system. The drawback is that the number of tasks grows fast using these local services. Scripting tasks, by contrast, are tasks that are implemented by the workflow

designer by means of a script [125]. Scripting tasks are used to create tasks not available as web services and not provided by the workflow system as local services [106], to create interactive tasks [239] and to perform data transformations [174; 30].

In this chapter, we will illustrate the use of workflows in the life science domain by means of a real-life use case. The workflow in this use case judges the agreement that exists between two different genome assembly annotations. Later on, this workflow will be examined regarding the tasks it is composed of. Then we will broaden our scope to all Taverna workflows stored at the myExperiment website to perform a quantitative analysis of the tasks these workflows consist of. We will establish the types of tasks and the number of data transformations these workflows perform.

4.2 Use case: mapping oligonucleotides to a genome assembly

New techniques have been developed to analyse the complete genome, proteome or transcriptome in a single experiment. One of these techniques is called (DNA) microarray analysis [37]. A microarray analysis is a method, among others, to reveal absent or mutated genes. It can measure the activity of thousands of genes simultaneously [37; 182; 124]. Central in the microarray analysis is the microarray chip (Figure 4.1). A single chip contains up to a few thousand spots of oligos. An oligo is a chemically synthesised DNA molecule used as a probe to establish the activity of a specific gene.

The oligos of the microarray chip are synthesised using as guide the gene annotations stored in a DNA database [57; 182], such as Ensembl [66] and VEGA [236]. The information in different databases, however, does not necessarily agree. The Ensembl database contains computationally derived gene annotations. The Vega database is a manually annotated gene database based on laboratory experiments. Both databases contain gene annotations for different species, known as assemblies. For some species (mouse and human), the VEGA assembly is a subset of the Ensembl assembly. For other species this is not true, such as the zebrafish assembly.

It is essential that every oligo on a microarray chip represents a single gene. When it represents no gene at all, it will measure nothing. If it represents multiple genes, it is insufficient precise.

4.2.1 Workflow for mapping the oligos

Together with the MicroArray Department & Integrative Bioinformatics Unit (MAD-IBU) at the University of Amsterdam, the Netherlands and the Laboratory of Bioinformatics at Wageningen University, the Netherlands, we have designed a Taverna workflow to check the agreement between Ensembl and VEGA about a VEGA-designed zebrafish microar-

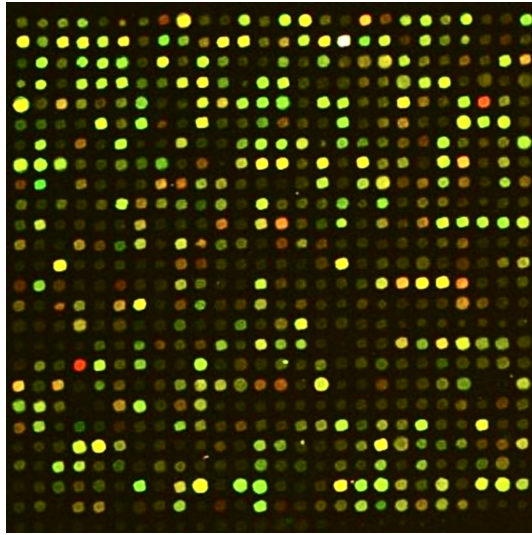


Figure 4.1: A scan of a microarray chip. The activity of genes in healthy and unhealthy tissue is measured. Green and red spots denote gene activity in either healthy or unhealthy tissue. Yellow spots denote gene activity in both tissues.

ray chip. The backbone of this workflow is presented in Figure 4.2. The explanation of the symbols used in Taverna workflows can be found in Appendix B. The entire workflow can be found in Appendix C. The workflow is available for download at <http://www.myexperiment.org/workflows/603>.

The first step of the workflow aligns all VEGA-designed oligo sequences on the zebra-fish genome using a Blast service and Blat service. Both are MOBY-S services provided by the Wageningen University. Blat [110] (Blast Like Alignment Tool) is a faster but less accurate variant of Blast. The result of each service is a set of hits. Each hit describes a location on the genome that looks similar to an input sequence (the oligo in this case) and a quality measure about this similarity. The workflow by default uses the Blat service. In case the Blat tool does not find hits, the Blast service is used. For each hit, a query to the Ensembl database is performed using the BioMart service [60] to retrieve transcript and gene information at the hit location. A script written in the R language [98] classifies the oligos among fifteen different classes using the Blat/Blast hits and the annotations found in Ensembl. The RShell plugin [125; 217] (see Appendix D) is used to embed and execute this script within the Taverna environment. The results of the classification are presented in Appendix E.

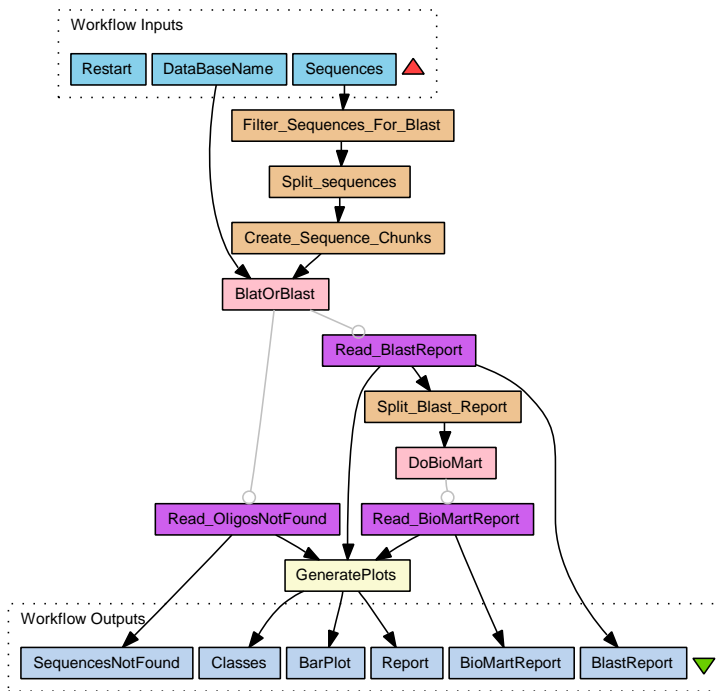


Figure 4.2: The collapsed workflow designed for mapping the oligonucleotides of the probes to the Ensemble database and Vega database.

4.2.2 Services used in the workflow

The workflow invokes three different web services: the Blat service, the Blast service and the BioMart service. Based on the description there would be four steps in the workflow:

1. Find regions on the genome that match an oligo sequence using Blat,
2. Find regions not found by Blat using Blast,
3. Find genes and transcripts at the regions found using BioMart,
4. Classify the oligos using the R script.

Interestingly, the entire workflow consists of 76 tasks (Table 4.1). One reason for this is that the Blast service is a so-called asynchronous web service. Performing a Blast operation therefore requires three service invocations: i) sending the request, ii) polling to

test whether Blast has finished the alignment, and iii) receiving the results. Between each step, the connection is lost between the web service and the client. The second step is repeated until the web service notifies the client it is finished. Asynchronous web services are used to prevent connection timeouts, because the Blast algorithm would take too much time for a synchronous web service invocation.

Table 4.1: Tasks used in the use case workflow.

Service type	Description	# tasks
Statistical analysis	R script for the statistical analysis	1
Database access	Access to the BioMart to find gene/transcript related information	1
Sequence search	Blast and blat services	2
Downloading	Download results from Blast and Blat service	2
Constant	Default values	6
Subworkflow	Workflows encapsulated in tasks	6
Coordination	Alternative branches workflow	13
Crash restore	Tasks to restore the workflow in case of a crash	14
Data transformation	Translate data into the desired format	31
Total number of tasks		76

The synchronous Blat operation requires two tasks instead of one. The first task sends the sequence to be aligned. The service returns a URL where to download the Blat report. This Blat report is hosted by a normal web server for load balancing of the server that hosts the Blat service. Invoking the BioMart web service requires only one task.

In the workflow, 31 tasks are related to data transformations. These tasks are local services and scripting tasks. Taverna provides many local services that can be used among others to perform string operations, to read and to write files, and to interact with users. Taverna provides two scripting tasks: the RShell processor [125; 217], to implement tasks in the R programming language and the BeanShell processor [125], to implement tasks in the Java programming language¹.

Most local services and BeanShell processors in the workflow are used to perform data transformations, or to affect the control flow of the workflow by iteration or conditional branching. Furthermore, 14 BeanShell processors are used to store intermediate results in case the workflow system crashes. Taverna (version 1.7) has memory problems with very large data sets (8157 oligos in this case). The extra tasks we inserted enable Taverna

¹The Taverna term *processor* refers to what we call a task.

to restore the workflow execution after a crash. The RShell processor in the workflow performs the classification.

Although this workflow may not be representative for all scientific workflows, it gives a good impression what a scientific workflow consists of. Even if the workflow description is simple and a few web services are involved, many additional tasks are required to invoke and connect tasks representing these web services and to affect the control flow of the workflow.

4.3 Tasks in scientific workflows

In this section we will examine the different types of tasks used in life science workflows. We will distinguish local services, web services, scripting tasks and subworkflows. Based on this distinction, we will establish a lower bound on the number of tasks dedicated to data transformations in life science workflows. Since different web services use different data formats we expect that many local services and scripting tasks are used to perform data transformations.

4.3.1 The data set

The study focuses on workflows designed in Taverna, because these workflows are easy to analyse for three reasons. First, Taverna is a very popular workflow system in the life science domain (>46.000 downloads since 2006 [51]), because it is open source and provides access to many web services. Second, workflows designed in Taverna are stored using the XML language Simplified Conceptual Uniform Flow Language (SCUFL) and are therefore easy to parse and to analyse. Third, workflows designed in Taverna are easy to collect due to the existence of the myExperiment website [72; 76]. The myExperiment website (>1.300 members since 2007 [51]) enables life scientists to share their workflows easily. Although the myExperiment website is not restricted to Taverna workflows, most of the workflows shared are Taverna workflows. The myExperiment site is online since October 2007 and at the moment of writing, it provides access to 415 Taverna workflows². The SCUFL files of the workflows are directly accessible through URLs. Each workflow has a unique identifier ranging from (1..N), where id=1 denotes the first workflow stored at myExperiment and id=N the last workflow stored at myExperiment.

If the SCUFL file contains subworkflows, the tasks these subworkflows consist of are analysed as well. In case multiple composite tasks refer to the same subworkflow, that subworkflow is analysed only once. The graph in Figure 4.3 presents the number of workflows sorted by size. The number of tasks per workflow ranges from 1 to 70 tasks. The

²The workflows are collected at December 11, 2008.

average workflow size is 8.8 tasks; the standard deviation is 11.7 tasks. The total number of tasks analysed is 3660.

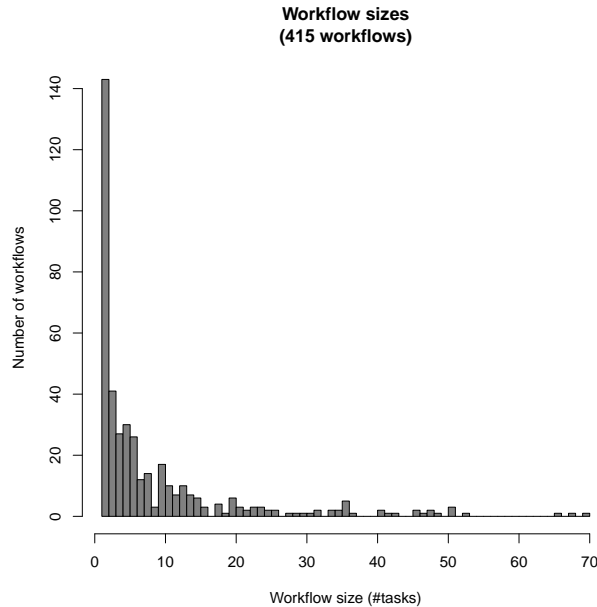


Figure 4.3: The distribution of the 415 Taverna workflows stored at myExperiment based on the number of tasks of the expanded workflows.

4.3.2 Approach

Ideally, we want to categorise the tasks in the workflows based on the way they are used in the workflow. Due to the huge number of tasks, manually annotating all the tasks is infeasible. To be able to tell something about the way the tasks are used, we have chosen to split the tasks into four categories: i) local services, ii) web services, iii) scripting tasks, and iv) subworkflows. These categories are also applicable to other workflows.

Based on our own experience, we expect that local services and scripting tasks are used more often for data transformations than the tasks in other categories. The analysis we have performed is implemented as a workflow, too (Figure 4.4). The workflow takes two inputs, the id of the first workflow and the id of the last workflow to be analysed. This workflow basically consists of four steps.

Generate_URLs: For all workflows stored at myExperiment with an id between first id and last id, the URL pointing to the location of the SCUFL file is generated. For example, the workflow with id 603 is located at <http://www.myexperiment.org/workflows/603/download>.

Analyse_Single_workflow: This is a composite task, in other words, a task that encapsulates a subworkflow. It downloads the SCUFL file of a workflow using the Taverna task “Get_web_page_from_URL” and uses the Taverna’s XPath task to collect the task types in the workflow.

Create_R_Table: The tasks collected by the XPath tasks are put in a table, in which each row describes the content of a single workflow. The first column contains the workflow identifier; all other columns describe the number of tasks used per task type.

Analyse_Workflows: The last step executes an R script to analyse the data stored in the table and to generate the plots used in this study.

The results of the analysis task are used as the workflow outputs. The workflow is available through the myExperiment site for download at <http://www.myexperiment.org/workflows/648>.

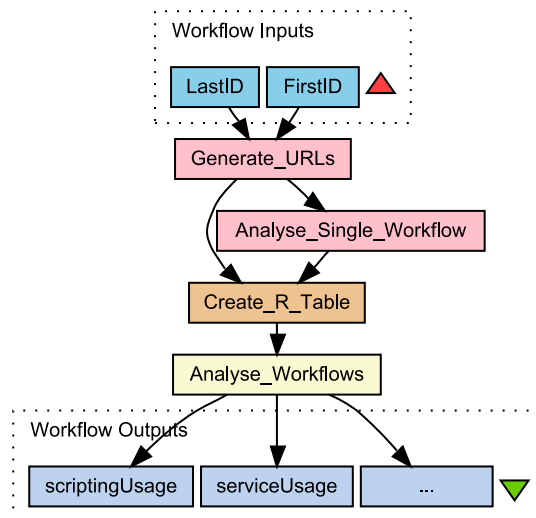


Figure 4.4: The workflow designed for analysing the workflows at myExperiment.

4.3.3 Results

Figure 4.3 shows a remarkably high number of workflows that consist of a single task (75 of the 415 workflows, 18%). A closer look at the myExperiment site clarifies that many of them are used as prototypes showing how to use certain local services and web services that provide a lot of configuration parameters. Other single-task workflows are used to share scripts. Taverna provides no functionality for easy sharing of scripting tasks, but it enables its users to import workflows published at the myExperiment website directly into the Taverna workflow system. Many Taverna users apply this approach in order to reuse scripting tasks.

Figure 4.5 shows the service type usage in all 415 workflows. About 57% of the tasks used in the workflows are local services and only 22% are web services. This is remarkable, since workflow systems, such as Taverna, are originally developed to connect web services. Scripting tasks count for 14% of the total number of tasks.

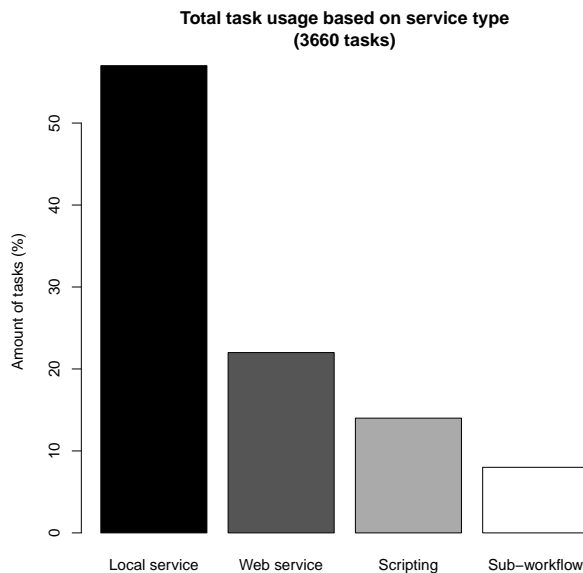


Figure 4.5: Task usage in all 415 workflows.

The diagram in Figure 4.6 shows the trends of task usage when the workflow size increases. As expected, subworkflows are seldom used in smaller workflows (workflows of size <10), but become more popular when the workflow size increases. About 8% of tasks

in the workflows are subworkflows. This closely matches the 10% of services provided as workflow measured by Lord et al. [127].

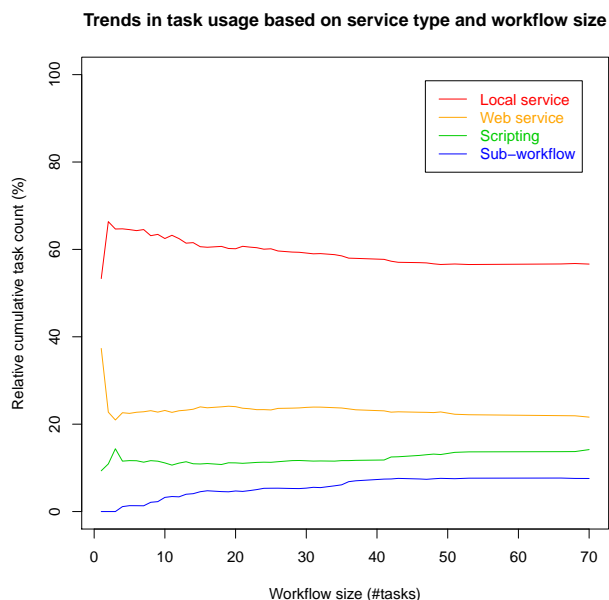


Figure 4.6: Cumulative task usage aligned to the workflow size.

The three categories local service, web service and scripting tasks will be discussed in more detail.

Local services

Taverna provides many local services that can be used to perform various types of actions. Hull has developed a website describing these local services³. Based on the description at this web site, we have categorised the local services on the type of actions they perform.

CDK: These tasks represent viewers, algorithms and analysis tools provided by the Chemistry Development Kit (CDK) Java library [180; 181].

Conditional: Tasks used to construct conditional branches, such as if-then-else constructs.

³<http://www.cs.man.ac.uk/~hull/shims.html>, last visited December 2009

Constant: A task that represents a constant value. These are used to assign default values to input ports of other tasks.

Database access: Tasks that are part of Taverna and provide access to well-known life science databases. For many of them, web service interfaces are available, but they have been developed quite recently.

Data transformation: Tasks to compose, decompose or translate data. For example, concatenating strings, composing XML data objects and transforming a sequence into FASTA format.

Operation: A task that creates new biological data based on its inputs. Examples are DNA to protein translation and calculating the complement DNA.

Testing: Tasks that are used to test the Taverna workflow environment, or a workflow designed in Taverna. For example, a task that generates a lot of strings.

User interaction: Tasks that interact with the user by means of dialog windows, such as the file selection dialog.

Util: General purpose task, such as file access.

Unknown: Tasks that are not categorised, because their function is not known.

Table F.1 in Appendix F shows how the Taverna's local services are put into these categories. Figure 4.7 presents the distribution of tasks among the different classes.

The data transformation tasks are responsible for 53% of the local services. This means that in total 30% of all the tasks are data transformation tasks, which indicates that data incompatibility is an important problem in the life science domain.

String constants tasks take a second place (28%). This is remarkable, because Taverna also enables its users to directly assign default values to input ports. The advantage of using a string constant task is that the default value is explicitly visible in the workflow. Both conditional branching services and util services take 5% of the local services. The residual 5% is shared among the remaining six categories.

Web services

Taverna by default supports the five invocation mechanisms listed below:

SOAP/WSDL: A widely used combination of the XML based web service description language WSDL [42] and the XML based protocol SOAP [32]. SOAP/WSDL is a general web service framework, popular inside and outside the life science domain.

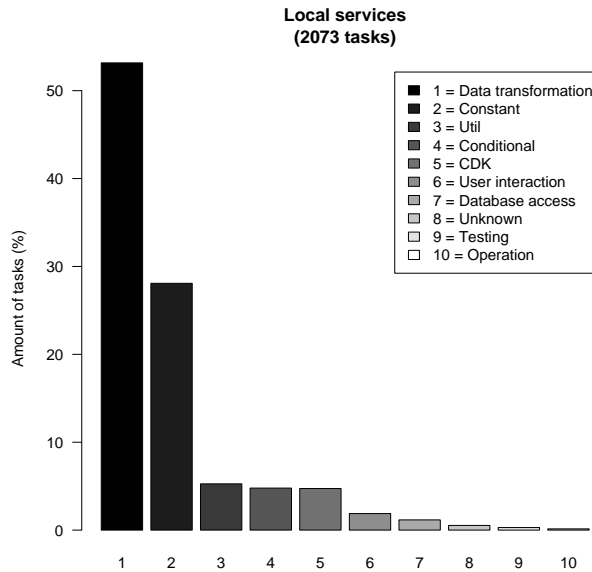


Figure 4.7: Use of local services, categorised by functionality.

SoapLab [170]: This framework is developed by EBI to easily encapsulate existing bioinformatics algorithms and tools as web services.

SeqHound [136]: The SeqHound services provide access to biological sequence and structure databases.

BioMart [60]: A query-oriented data management system that provides access to, among others, the Ensembl [181] database.

MOBY-S [234; 107; 188]: A branch of the BioMOBY consortium that simplifies the discovery process of life science services and the access to these web services by maintaining a central registry.

Figure 4.8 shows the use of the different invocation mechanisms in Taverna. The SOAP/WSDL web services are by far most popular (66%). SoapLab web services take a second place (24%). MOBY-S, BioMart and SeqHound web services are used to a lesser extent in Taverna. An explanation for this is the existence of other powerful applications that provide access to these web services too. BioMart web services are accessible through

the MartView web application; MOBY-S web services are accessible through various tools, such as Seahawk [80]. These tools often provide better facilities to search for web services and to connect them. A workflow system, such as Taverna, becomes the preferred tool when services using different invocation mechanisms need to be connected.

Another reason is that SOAP/WSDL web services and SoapLab web services support a greater number of functions. For example, BioMart provides only database access. Once data are collected from a database using a BioMart web service, there are plenty of things one can do to analyse the data, using numerous SOAP/WSDL web services and SoapLab web services, but not through BioMart.

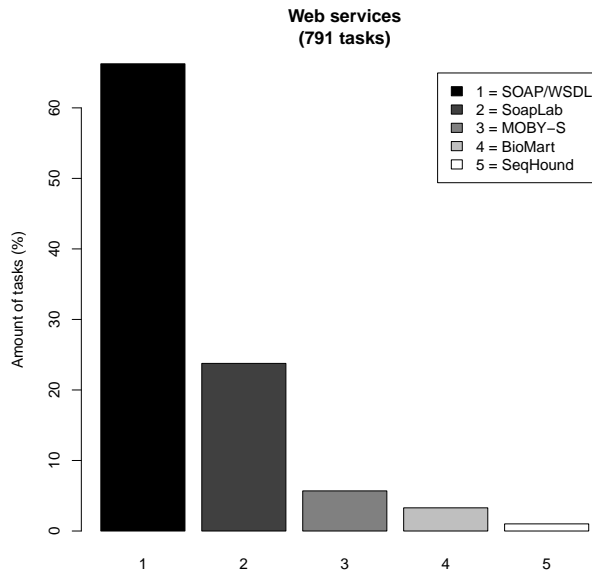


Figure 4.8: Different types of invocation mechanisms used in the workflows.

Scripting

As mentioned before, Taverna supports the two programming languages Java and R. Java provides many facilities to handle data and to create advanced user interfaces and therefore is useful to implement a wide variety of tasks. The BeanShell processor is very popular, 97% of the scripting tasks are implemented using this task. In the workflows stored at myExperiment, the BeanShell processor is used among others to perform data

transformation and to create interactive tasks.

The R language is specifically designed to perform statistical analysis. In Taverna, all the RShell processors (3% of the scripting tasks) are used to perform statistical analyses and to create plots.

4.4 Conclusion

Scientific workflow systems have been developed to simplify the construction of in-silico experiments by providing a graphical user interface by which end-users can orchestrate web services into complex workflows [51; 70]. Ideally, the workflow designer does not have to program, but can construct workflows by dragging and dropping tasks representing web services into the workflow diagram and connecting them. The use case has shown that workflows are more than just a set of connected web services. Many additional tasks are required in the workflow to invoke web services, to steer workflow execution and to perform data transformations.

Dealing with data incompatibility problems forms a large part of the workflow designers' daily activities [23; 79]. This is confirmed by our quantitative study: about 30% of the tasks in the workflows analysed represent data transformation activities. This is just a lower limit. Based on discussions with bioinformaticians, we can give three reasons why the real amount of data transformation tasks is even higher.

1. A lot of BeanShell processors are used to perform data transformations.
2. Some web services are implemented to perform data transformations. Not all scientists know how to program in the languages supported by Taverna. These scientists prefer to program in a language they already know, such as Perl. For them, it is easier to implement a new web service and to register this web service to the MOBY-S system. This newly created data transformation web service is directly accessible through Taverna, even for other people. In the MOBY-S Central, these web services are marked as "Conversion" web services.
3. Some services are not intended to perform data transformations, however, some users use them for this purpose. For example, a bioinformatician can send multiple requests that only differ in the output format requested to the same database. In such a case, the database web service is used to perform data transformation rather than collecting new data.

The analysis performed is restricted to Taverna workflows, but probably can be generalised to other workflow systems and even to scientific programming. The problems with

data incompatibility can only be solved if different organisations agree on standards [168]. However, as long as people are free to define their own standard, they will. The problem of data incompatibility is recognised in the life science domain [30; 168]. However, the size of the problem has never been investigated before.

Although its primary goal is connecting local and web services [4; 18; 69; 51], a scientific workflow system should not be seen as just a web service composition environment. It is a visual programming environment that aims to reduce the programming effort required by the scientists [53]. Support for hierarchical workflows in these systems is essential to help workflow designers manage large and complex workflows.

A workflow system has its advantages with respect to programming languages. It can and should guide workflow designers to perform data transformation. Many workflow systems therefore provide local tasks, also known as *shim* services [96], to transform closely related data. For example, it can provide local services to compose and decompose complex data structures. The MOBY-S plugin [107] for Taverna provides so-called composer and splitter tasks to compose and decompose MOBY-S objects. In similar fashion, Taverna has composer and splitter tasks to compose and decompose XML data used by SOAP/WSDL services.

Support for scripting is an essential requirement of workflow systems. Workflow designers use scripting to implement tasks not available as local services and web services. Ideally, a workflow system supports multiple scripting languages to enable the workflow designer to choose the programming language he is familiar with. Web sites for sharing workflows, such as myExperiment, form a valuable source for sharing these scripts. Workflow designers can reuse scripts of others and prevent them from developing these tasks themselves. A deeper analysis of the scripting tasks currently available can help to find out what kind of data transformations are required and how a workflow system can help its users to perform them.

Workflow reuse and service availability

5.1 Introduction

Previous chapters have shown that workflow systems help life scientists to find and connect services. At the same time, they improve reproducibility of experiment results. Scientists can share their workflows to exchange knowledge of experiments and reuse workflows created by others [76; 70; 51]. Through web sites like myExperiment [72; 76], life scientists all over the world now can easily share their workflow experiments with peers at a single place. Existing workflows form a good starting point for new, but similar experiments. At the same time, workflows have become new means of publications. For example, Son et al. [177], Li et al. [125] and Wiggins et al. [231] put their workflows on myExperiment to prove the validity of their experiments.

Workflow reuse, however, is not trivial. The workflow specification is tied to web services used at a time in the past, namely when these workflows were designed. These web services are often hosted by third parties, which means that the workflow designer has no influence on the continued existence and accessibility of the web service [22]. We expect three main reasons a service may become inaccessible: First, a web service can become (temporarily) unavailable, among others when the service is down, or has restricted access. Second, the interface of the service may change [142]. The *new* service may have different inputs and outputs or may use different data types. Third, the service location may change, which means that the service is still alive, but at a different location, including a different web server port. When the workflow description refers to the physical locations of services, these problems all result in broken workflows. The workflow user needs to repair these problems before she can run the workflow.

The problem that services are unavailable is well-known [22], but its size is not in-

investigated before for workflows. In this chapter, we will analyse the accessibility of web services used in life science workflows. First we will discuss related work. Next, we will discuss the experiment setup and the results of the experiment. Then, we will discuss how workflow re-users can deal with these problems and which solutions workflow systems can provide. We will end with conclusions.

5.2 Related work

Goderis et al. [76] have performed a survey among 24 bioinformaticians to gain insight into the practice and requirements of workflow sharing solutions. They report that three-quarter of the participants had experienced some services not to be available when they reused workflows. The main reasons are services being local to the workflow designer or services being down. These services need to be replaced by alternative but equal services, hosted at a different location [22]. A service is an equal alternative if its syntactic and semantic description are the same [240].

Problems also arise when the service definition changes. The workflow system can still invoke the service, but probably delivers the wrong input data or receives data from the service it cannot deal with. Most services are described in service definition files, such as WSDL files, or in central registries, such as BioMOBY's MOBY-S Central [234; 235; 188]. Service providers are expected to keep the service information up-to-date, but sometimes change their services without updating the service definitions [142]. Even if the service provider keeps the service definition updated, changes in the service definition will go unnoticed until the workflow is executed. The workflow re-user has to adapt the workflow to the new service definition.

Workflow sharing solutions, such as myExperiment, can help the workflow designer to find broken workflows. The web services used in the workflow can be checked against, among others, changes in the interface, availability, permissions and performance [73]. This information should be presented to the workflow user, to help him to decide how reusable a workflow really is.

5.3 The data set

The myExperiment web portal forms the ideal resource for collecting workflows intended to be shared among peers. Although myExperiment allows to share workflows specified in any workflow language, we will limit our analysis to the workflows designed in Taverna [146; 147; 148], because most workflows stored at myExperiment are designed using this tool [218], and Taverna uses a simple XML language called SCUFL (Simple Con-

ceptual Unified Flow Language), which is easy to parse.

The previous chapter has shown that SOAP/WSDL and SoapLab services together cover 90% of the web services used in Taverna workflows (Figure 5.1). SoapLab [170] is a framework developed by EBI to easily encapsulate existing bioinformatics algorithms and tools as web services. SOAP/WSDL is a widely used combination of the XML-based web service description language WSDL [42] and the XML-based protocol SOAP [32]. SOAP/WSDL is a general web service framework, popular inside and outside the life science domain. Other service invocation mechanisms Taverna supports are MOBY-S [234; 107; 188], SeqHound [136] and BioMart [60]. These three invocation mechanisms are not taken into account, because they together are responsible for only 10% of the web services used.

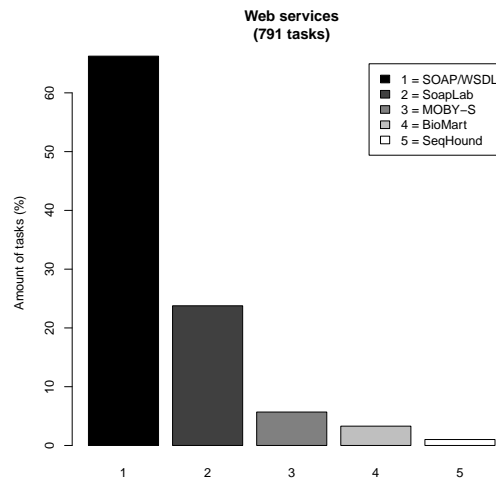


Figure 5.1: Web services used in Taverna workflows.

The SCUFL files of the workflows at myExperiment are directly accessible through URLs. Each workflow has a unique identifier X , ranging from $(1 \dots N)$, where $id=1$ denotes the first workflow stored at myExperiment and $id=N$ the last workflow. The SCUFL file of workflow X can be downloaded at: <http://www.myexperiment.org/workflows/X/download>.

5.4 Approach

SoapLab services and SOAP/WSDL services are analysed separately. Each analysis is implemented as a Taverna workflow itself.

5.4.1 Analysing SoapLab services

The workflow to analyse the SoapLab services starts with downloading all Taverna workflows stored at myExperiment using Taverna's download task¹. As mentioned before, the workflow specifications are defined in the XML-based SCUFL language. These workflow specifications are therefore easy to parse using XPath expressions. The XPath task in Taverna is used to collect the XML elements describing the SoapLab services used in each workflow. These XML elements contain URLs to the SoapLab services and the operations to be performed (Figure 5.2).

```
<s:processor name="embossversion">
  <s:description>Writes the current EMBOSS version number</s:description>
  <s:soaplabwsdl>
    http://www.ebi.ac.uk/soaplab/emboss4/services/utills\_misc.embossversion
  </s:soaplabwsdl>
</s:processor>
```

Figure 5.2: The SCUFL definition of a SoapLab service in a Taverna workflow.

In the final step of the analysis workflow, the existence of each SoapLab service is checked using the URL. For each web service, the workflow saves the following information: i) the name of the service, ii) the number of times it is used, iii) the workflow that uses the service, and iv) the availability of the service. Figure 5.3 shows the structure of this workflow.

5.4.2 Analysing SOAP/WSDL services

The workflow for checking the existence of the SOAP/WSDL services looks like the previous workflow. It starts again with downloading all Taverna workflows from myExperiment. This time it uses the XPath task to collect all the XML elements that describe SOAP/WSDL services. The SOAP/WSDL service description consists of a URL pointing to the WSDL location and the operation name (Figure 5.4). The WSDL file describes the

¹Taverna normally uses the term processor to refer to what we call a task.

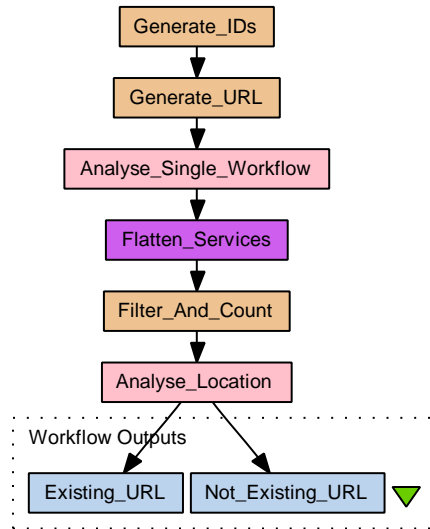


Figure 5.3: The workflow that searches for dead SoapLab web services in Taverna workflows at myExperiment.

interface to the web service in terms of the data types used, the operations provided and the binding of operations to physical locations of web servers. A WSDL file can define multiple operations.

Checking the accessibility of a SOAP/WSDL service is split into checking the existence of the WSDL file and checking the accessibility of the operation. The workflow saves for each operation the following information: i) the name of the operation, ii) the number of times it is used, iii) the workflows that uses the service, iv) whether the WSDL file exists, and v) whether the operation is accessible.

The operations not found in the WSDL file are checked by hand, because some WSDL definitions contain import statements of other WSDL files. The workflow is not able to deal with this. Figure 5.5 shows the structure of the workflow to check SOAP/WSDL services.

Both workflows do not depend on external resources, because all tasks are implemented using Taverna's local tasks and BeanShell scripting tasks. The workflows are stored at myExperiment and can be downloaded at:

For testing SoapLab services: <http://www.myexperiment.org/workflows/773>.

For testing SOAP/WSDL services: <http://www.myexperiment.org/workflows/774>.

```

<s:processor name="run_eFetch">
  <s:arbitrarywsdl>
    <s:wsdl>
      http://eutils.ncbi.nlm.nih.gov/entrez/eutils/soap/eutils.wsdl
    </s:wsdl>
    <s:operation>
      run_eFetch
    </s:operation>
  </s:arbitrarywsdl>
</s:processor>

```

Figure 5.4: The SCUFL definition for accessing a SOAP/WSDL operation in a Taverna workflow.

5.5 Results

To limit the influence of coincidence, the measurement was performed monthly, from February 2009, up until and including August 2009. Table 5.1 presents the effect of the inaccessible services on all Taverna workflows available through myExperiment at the specific dates. It is worth noting that multiple tasks can refer to the same SoapLab service or SOAP/WSDL operation. The number of SoapLab tasks remains almost stable over the past few months; the number of broken SoapLab tasks grows in the last few months. The number of SOAP/WSDL tasks grows slightly, just like the number of broken SOAP/WSDL tasks.

Table 5.1: The influence of inaccessible services on the tasks and workflows, measured over eight months in 2009.

	Feb	Mar	Apr	May	Jun	Jul	Aug
# Analysed workflows	421	425	429	435	447	454	460
# with SoapLab or SOAP/WSDL services	235	238	242	246	250	257	263
# SoapLab tasks	184	184	184	184	185	185	185
of which not runnable	9	9	9	9	9	15	15
# SOAP/WSDL tasks	540	550	563	574	576	601	607
of which not runnable	69	76	88	90	79	80	84
# Broken workflows	25	26	34	37	33	35	38
due to SoapLab tasks	5	5	5	5	5	6	6
due to SOAP/WSDL tasks	22	23	31	34	30	31	34

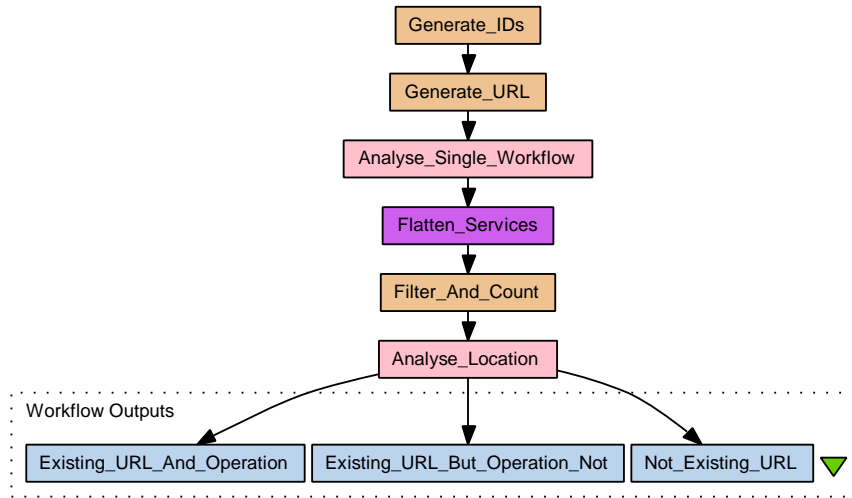


Figure 5.5: The workflow that searches for dead SOAP/WSDL operations in Taverna workflows at myExperiment.

The total number of broken workflows is less than the sum of the number of workflows broken due to SoapLab tasks and the number of workflows broken due to SOAP/WSDL tasks. The reason is that three workflows in the data set have broken SoapLab tasks as well as broken SOAP/WSDL tasks. In these seven months, the number of broken workflows has grown from 6% to about 8%. When only the workflows that have SoapLab or SOAP/WSDL services are considered, the number broken workflows has grown from 10% to 14%. All broken workflows are at least ten months old. The results for SoapLab and SOAP/WSDL will be discussed below in more detail.

5.5.1 SoapLab services

In total, 185 SoapLab tasks are used in the workflows stored at myExperiment. These tasks refer to 55 different SoapLab services. These services are provided by six organisations; 46 of the 55 services used (91%) are provided by the European Bioinformatics Institute (EBI) and Manchester University. Table 5.2 shows the trend of service availability over 7 months.

Nine out of fifty-four SoapLab service are inaccessible (17%) in the first six months. In the seventh month, July, the number of inaccessible services increases from 9 to 15 (27%). These 6 new dead web services were all hosted at the same domain, <http://www.o2i.it>. These services are dead, because this domain name does not exist anymore.

Table 5.2: The number of SoapLab services and those that are inaccessible, measured over eight months in 2009.

	Feb	Mar	Apr	May	Jun	Jul	Aug
# Service providers	6	6	6	6	6	6	6
# SoapLab services	54	54	54	54	55	55	55
# inaccessible	9	9	9	9	9	15	15

5.5.2 SOAP/WSDL services

The SOAP/WSDL services are used far more often than SoapLab services (Table 5.1). This is due to the large number of SOAP/WSDL service providers and the many libraries to create and to access SOAP/WSDL services.

Like for SoapLab tasks, multiple Taverna tasks can refer to the same SOAP/WSDL operation and a single workflow can have multiple SOAP/WSDL tasks. If two tasks represent different operations, they can still refer to the same WSDL file, because a single WSDL file can define multiple operations. Table 5.3 shows the number of SOAP/WSDL operations used in tasks in Taverna workflows stored at myExperiment over the seven months. The inaccessible services are split into missing WSDL files and operations that are inaccessible due to a change in the service definition.

Table 5.3: The number of WSDL files, missing WSDL files and broken operations used in Taverna workflows, measured over eight months in 2009.

	Feb	Mar	Apr	May	Jun	Jul	Aug
# Service providers	43	44	45	47	48	50	51
# WSDL files	105	108	111	116	117	121	124
# WSDL files inaccessible	16	18	19	22	22	23	25
# Operations	199	203	209	212	216	227	233
# with WSDL file inaccessible	29	41	49	51	44	45	49
# with operation undefined	8	1	1	1	1	1	1

In June, the number of broken SOAP/WSDL task decreased by 7. These seven services are up and running again. One of them, however, is accessible through a different web server port; the corresponding task in the workflow has been modified to use this new location.

During the first measurements, eight operations were found undefined in the WSDL file of the corresponding web service. This is probably due to updated service definitions.

Table 5.4: Why the broken SoapLab web services and SOAP/WSDL operations are inaccessible at August 2009.

Reason	# SoapLab	# SOAP/WSDL
Hosted locally	0	13
Port changed	3	0
Location changed	7	5
Service definition changed	0	1
Service down	5	31
Total	15	50

These eight operations were all defined in the same WSDL file and referred to the GoViz web service. This web service was located at EBI and provided operations for the analysis of Gene Ontologies (GO) [14]. During the next measurements, the WSDL file of this web service was not accessible. Contact with the developers clarified that this service was removed. During the second to fourth experiments, only one operation was found not to be defined in the WSDL file of the web service.

5.5.3 Why are these services inaccessible?

The service providers of the services that are down have been contacted to gain insight into why these services have become unavailable. Table 5.4 summarises the causes why the SoapLab services and SOAP/WSDL services found at the latest measurements are inaccessible.

Many inaccessible SoapLab services are available at new locations. For example, the author of the services hosted at <http://www.o2i.it> explained that these services are now available at <http://bioinformatics.istge.it>. Some other SoapLab services are hosted at the same location, but through a different web server port. Changing this port number in the workflow definition will result in a working workflow.

The most important reason why SOAP/WSDL services have become inaccessible is that they are not hosted anymore. The death of a single web service results in multiple broken operations. For example, the death of the GoViz web service causes 8 broken operations used in workflow tasks. Most web services were no longer hosted because they were not used anymore. They were often designed for a single experiment.

Thirteen operations are inaccessible because the 6 corresponding SOAP/WSDL services are hosted locally, at the workflow designer's computer (*localhost*) or elsewhere in the local area network. Maybe the workflows that use these web services are not meant to be

reused as such, but only as a template. Installing and using web services locally is done frequently [76]. It improves speed and gives the user more control to customise to specific needs [142], because web servers have to share CPU-usage among many processes. A web service is often noticeably slower than its local variant. Workflows that make use of these services require the re-user to install these services locally or to replace these tasks to make these workflows runnable.

5.6 Dealing with broken web services

Although dead services currently is a small issue in workflow design and workflow sharing, it can form a big problem when workflow sharing becomes popular. Over just seven months, the number of workflows broken because of a dead service has risen from 25 to 38, an increase of more than 50%. The organisations that supply web services often differ from the web service users. The user therefore has little influence on the existence of a web service and its interface. To keep their workflows runnable, workflow designers have to replace broken tasks. The workflow system should help the user to find and repair these tasks. However, most workflow systems only enable their users to replace dead web services by deleting the task representing this dead web service, inserting a new task and restoring the data connections. Alternatively, the user can use a text editor to repair the workflow by manually rectifying the URL the task is pointing to. The latter only works if the interface of the alternative service is the same.

A partial solution would be a workflow system that supports late binding. Specifications made in such a workflow system are not tied to specific web services [85]. Tasks in a workflow system supporting late binding do not refer to real resources, but only describes required capabilities of a resource to execute the task. A task then describes, among others, the data to be consumed and produced and the type of operation to be performed. Of course, the task description should contain enough information to decide which services are suitable to execute the task. The workflow is independent of web services available at design time, because the actual resource is chosen at enactment time by the workflow engine. The workflow engine is still able to run a workflow when services are hosted at a different location or through a different web server port.

Late binding can be useful in many situations; we will describe five of them. First, in the life science, many retrieval services are available, but they do not all provide access to the database required. Therefore, one may want to replace the retrieval service with one that has access to the database required. This is often the case for the many Blast services available. These services all perform the same type of operation, namely sequence based search, but provide access to different genome assemblies. Second, newer services may exist that use faster algorithms or have higher quality. Third, services can be temporarily

overloaded. Then it is better to switch to mirror services. Fourth, some organisations frequently provide new versions of databases they host. Ensembl [66], for example, releases a new version approximately every two months, but also provides access to many older versions of the database. It depends on the situation whether the workflow user wants to switch to the newest version of the database or to older ones. Five, late binding helps the user to easily switch to a local copy of the service. Running the web services locally helps to deal with dead web services, but also to speed up service execution and prevents sending large data sets to and fro between web services. Late binding will only work if equivalent web services have the same interface, which is not always the case [6].

Late binding is not new in business workflows. The WS-BPEL language [13], for instance, has support for late binding. Only a few scientific workflow systems support late binding. Taverna enables its users to define *alternate* services. Such an alternate service will only be invoked when the original service is down. The RShell plugin in Taverna [125; 217] enables its users to easily switch between local and remote R installations. Although both solutions help the workflow designer to deal with broken services, the solutions are still made at design time instead of runtime. The BioMOBY plugin for Taverna [107] supports late binding through the MOBY-S Central. Tasks representing MOBY-S services are defined by the triplet {MOBY-S Central location, web service name, service authority}, that uniquely defines the web service to be executed (Figure 5.6). The web service location itself is stored in the MOBY-S Central. When the Taverna workflow executes a BioMOBY task, the MOBY-S Central redirects the service invocation to the real web service. A change in the service location therefore does not require a change in the workflow using this service and does not require the user to notice this. A disadvantage is that the user cannot easily switch to equivalent services hosted. Problems arise when the MOBY-S Central is (temporarily) down; then all MOBY-S services become inaccessible.

Kepler [11; 129] supports switching between alternative services by defining primitives for actor replacements [31]. Defining these primitives is still done at design time of the workflow. JOpera [151] supports late binding too, but like Kepler, it requires the workflow designer to use special constructs. Pegasus [52; 54] supports late binding through workflow segmentation. The workflow designer describes the workflow at an abstract level. At enactment time, the workflow undergoes a series of refinements until all refinements can be mapped to resources.

5.7 Conclusion

Web services are used in the life science domain to provide programmatic access to databases and tools. Workflow systems help scientists to easily connect these services and to create in-silico experiments that can be shared with peers. However, workflow sha-

```

<s:processor name="MobyBlat">
  <s:description>
    BioMOBY web service wrapper for the command line query tool
    'BLAT'.
  </s:description>
  <s:biomobywsdl maxretries="30" retrydelay="500">
    <s:mobyEndpoint>
      http://moby.ucalgary.ca/moby/MOBY-Central.pl
    </s:mobyEndpoint>
    <s:serviceName>
      MobyBlat
    </s:serviceName>
    <s:authorityName>
      www.bioinformatics.nl
    </s:authorityName>
    <s:Parameter s:name="out">blast8</s:Parameter>
    <s:Parameter s:name="q">dna</s:Parameter>
    <s:Parameter s:name="minScore">0</s:Parameter>
    <s:Parameter s:name="minIdentity">0</s:Parameter>
    <s:Parameter s:name="maxIntron">1000000</s:Parameter>
  </s:biomobywsdl>
</s:processor>

```

Figure 5.6: Taverna supports late binding for MOBY-S tasks. It saves the location of the MOBY-S Central, the service name and the authority.

ring is only useful if all the resources used in the workflow remain available. Goderis et al. [76] have shown in their user study that workflow re-users have experienced difficulties in reusing workflows of others. We have shown in this chapter why these problems arise and what the effects are. Although the analysis is limited to Taverna workflows stored at myExperiment, the problem itself occurs in a wider range of workflow systems. There are many reasons why web services are inaccessible. Services can be (temporarily) down, have restricted access, or are moved to new locations.

To make workflows containing inaccessible services runnable, the workflow re-user has to replace these services. Late binding promises to be a solution for many situations. It is a functional requirement of a workflow system, because it delays task resource binding until enactment time. Workflows designed in such a system are independent of the resources available at design time. Of course, these workflow systems still require at least one suitable web service to be available at enactment time.

Part III

e-BioFlow: a new type of workflow system

Designing workflows using different perspectives

6.1 Introduction

This is the first chapter of five about the e-BioFlow workflow system. e-BioFlow is inspired by the context of scientific collaborative environments, such as the e-BioLab [158]. It is an extensible visual workflow system that focuses on the design and execution of reusable workflows and the analysis of the provenance data generated during the execution. This chapter focuses on the e-BioFlow workflow editor. Chapter 7 discusses e-BioFlow's workflow enactment engine. The design and implementation of the provenance system for e-BioFlow are discussed in Chapter 8. e-BioFlow provides an ad-hoc interface for explorative workflow design. The mockup implementation and user evaluation of this interface will be discussed in Chapter 9. Chapter 10 discusses the design and actual implementation of this ad-hoc system.

The e-BioFlow editor enables the workflow designer to describe the tasks in multidisciplinary life science experiments. 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 [205]. The e-BioFlow editor enables the workflow designer to model both data flow related information and control flow related information. e-BioFlow thus is a *hybrid workflow system* [173]. As shown in Chapter 5, resources used in workflow models can become unavailable or are accessible only within a specific domain. Therefore, it is important to abstract from resources and to delay resource-task binding until workflow enactment. e-BioFlow is developed to support late binding. Workflows desig-

ned in e-BioFlow are independent of the location of the resources available at design time, because e-BioFlow postpones task-resource binding until execution time. A workflow designed in the e-BioFlow editor is a template for a group of experiments. So, experiments can have equal structure, but still can use different resources.

In the next section, we will discuss the requirements of a workflow design system for modelling processes. After that, we will introduce our approach, the e-BioFlow workflow system, the perspectives it provides, and how these perspectives are related. The benefits 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.

6.2 Related work

A workflow systems supports two tasks, namely, workflow design and workflow enactment. The result of workflow design is a workflow model, which is a template often not designed for a single case but for a class of cases [203; 51; 77; 238]. In the life science domain, a workflow functions as a problem solving environment; each workflow describes a specific type of experiment. The workflow engine performs the enactment of a workflow model. Enactment encompasses the instantiation of a workflow into a specific case, and the execution of this case. Each case is unique, although cases may share the same workflow model. Cases can differ in the way tasks are executed, the data that flows between these tasks, but also the resources used to execute the tasks. Therefore, a workflow model should provide a perfect balance between the generalisation of the case type and the adaptation to the specific cases.

Workflows have proven to be successful to model business processes. [68; 8; 203]. The way workflow models are used in the life science domain and the business domain, however, differs tremendously. Business workflow models are control flow-oriented and focus on the order in which work has to be done. The focus in life science is on data, so the traditional design interfaces of business workflow systems do not fulfill the requirements of the life science domain [212]. The life science workflow models are data flow-oriented, describing the information flow between tasks. However, at the same time, there is growing demand for a more controlled approach to workflows in the life science domain [111; 144; 77]. Most existing workflow systems in the life science domain, such as Taverna [146; 147; 148] and Kepler [11; 129], lack the facilities to model advanced control structures such as conditional branching and iteration (loops).

Some others, such as Triana [134], support these control structures, but still miss the key functionality to perform late binding. Bowers et al.[11; 31] have tried to tackle this problem in Kepler [129] by defining primitives for actor replacements. However, the re-

placement is still done at design time instead of instantiation time. As mentioned in the previous chapter, late binding in Taverna is only available through the RShell plugin [125; 217] and the BioMOBY plugin for Taverna [107]. JOpera [151] is an example of a hybrid workflow system. A key difference between JOpera and the system we present, e-BioFlow, is that the first supports late binding only using special constructs and the latter supports late binding by default.

Van der Aalst [202] and Jablonski and Bussler [99] 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. These perspectives can be used to model, visualise and check workflows.

Workflow models, and diagrams in general, are very suitable as visualisation and communication means [160; 99; 55]. However, most workflow design systems mainly focus on structuring and connecting tasks; the visualisation aspect often gets little attention [79]. 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 [151]. This is counterproductive; if anything, visualisation should advance rather than hinder understanding.

Formal checking of workflows is important, especially when the workflow becomes complex. Scientists have to show that their experiment conforms to quality standards in their field [17] whereas business modelers have to create consistent, optimized, and possibly automated business processes [68; 99; 203]. Workflow models enable formal checking from all the three perspectives, control flow perspective (soundness [199]), data flow perspective [23; 197; 244] and resource perspective [99; 203; 22].

6.3 The e-BioFlow editor: a new type of workflow editor

The e-BioFlow system provides three perspectives: control flow, data flow and resource perspective, through a tabbed graphical user interface (Figure 6.1). Each perspective is represented by a tab. Additionally, e-BioFlow provides tabs for the enactment of workflows (Chapter 7), the analysis of provenance data (Chapter 8) and explorative workflow design (Chapter 10). The workflow designer is able to work in a single perspective at a time without being restricted to the functionality of a single perspective. As section 6.3.4 will explain, changes in one perspective are propagated to the other perspectives wherever appropriate.

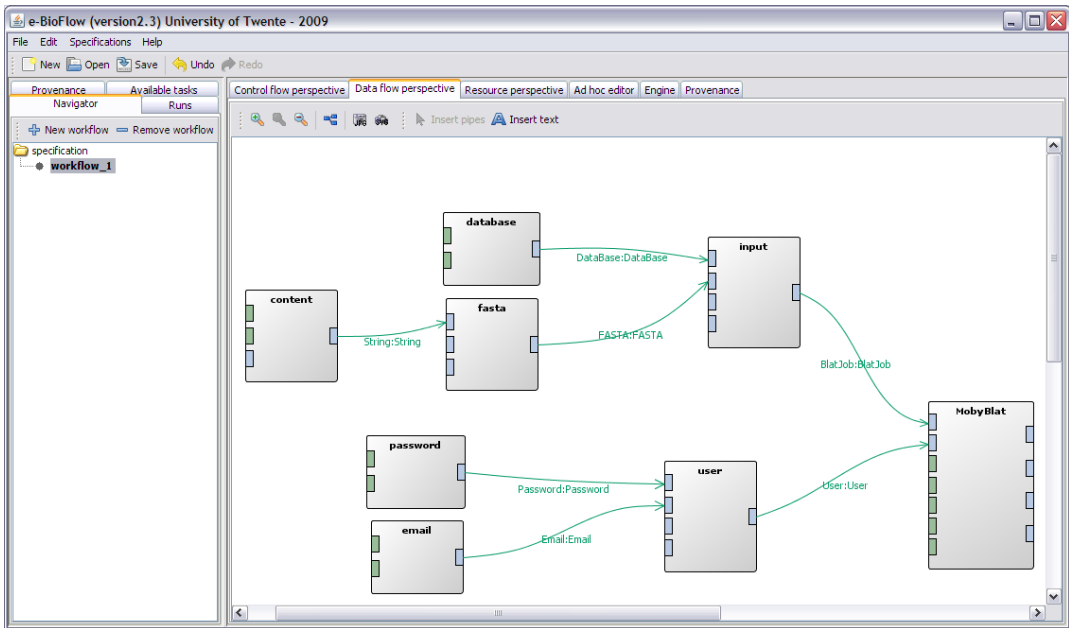


Figure 6.1: The e-BioFlow workflow system.

The workflow language that e-BioFlow uses, the e-BioFlow language, extends the YAWL workflow language [202]. 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 workflows [199]. YAWL enables one to model all workflow patterns described by Van der Aalst et al. [201]. The YAWL system itself comes with a design tool that, however, only enables one to design control flow structu-

res and does not explicitly consider the data flow and resource perspectives. e-BioFlow complements YAWL by adding the data flow perspective and the resource perspective.

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

Task: A task is an abstraction of work to be done. This is also known as an activity [19]. Taverna calls this a processor [146].

Atomic task: An elementary representation of work [200].

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

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 subworkflows.

Dependency (Control Flow perspective): A relation between two tasks that defines enactment order: a certain task cannot start until another task has finished [173].

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 [202].

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 [151].

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 [173].

Role (Resource perspective): A role describes the required capabilities to execute a task [19; 68].

Actor (Resource perspective): An actor is a resource capable to fulfill a particular role and therefore to perform a certain class of tasks [19].

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.

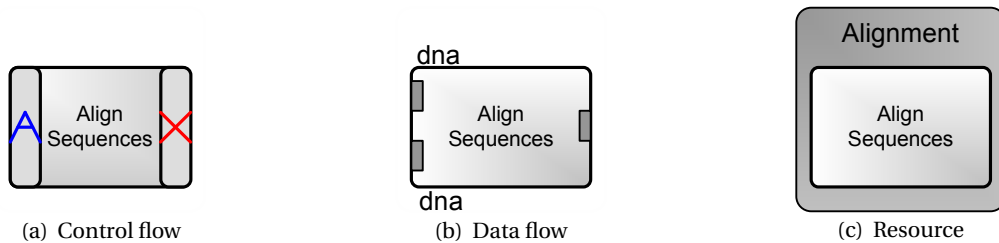


Figure 6.2: Task box representation in the three perspectives.

e-BioFlow supports hierarchical workflows; a task is either atomic or composite. Hierarchy is a very important property of workflow models. It helps to structure large diagrams and provides a means for abstraction [199]. The user can choose a task to be composite, in which case the task is white boxed into a subworkflow, but can alternatively choose to ignore the way a task is implemented, leaving it black boxed. The e-BioFlow editor is flexible with respect to task expansion. Multiple composite tasks can be white boxed to the same subworkflow. The user does not need to know in advance whether a task will be expanded into a subworkflow.

Next we will describe the three perspectives as they are offered to the workflow designer by e-BioFlow. The use of these perspective will be illustrated using a simple life science case in section 6.4.

6.3.1 Control flow perspective

The control flow perspective visualises tasks as boxes. There is often a specific order in which tasks can be executed due to dependencies between tasks. Tasks can be executed in sequential, parallel, conditional and iterative order. The control flow perspective visualises the dependencies as arrows from one task to the next.

Multiple tasks can depend on the same task and a single task can depend on multiple tasks. 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 (Figure 6.2(a)). Van der Aalst et al. [202] 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).

Condition statements are used to determine the tasks to be activated in case of a XOR

split or an OR split. These statements are attached to the dependencies between tasks. The e-BioFlow language uses the XPath language to define them, because it extends the YAWL language which uses XPath too. Conditional statements in e-BioFlow can refer to data consumed or data produced by the task. Not everyone is an expert in XPath or knows about the XML document structure used by e-BioFlow. Therefore, next to a traditional text editor, the control flow perspective provides a wizard to help the user create the desired XPath expressions. Using this wizard, the user can select a logical test operator and the data to perform the test on. The XPath expression generated can be updated using the wizard, but also by hand.

The join types are similar to the split types: 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 6.3. 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.

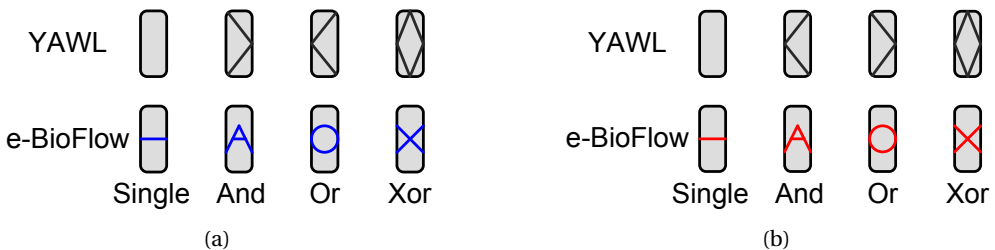


Figure 6.3: Symbols representing the four different join types (a) and split types (b).

Since the e-BioFlow language extends the YAWL language, all workflow patterns supported by YAWL are also supported by the e-BioFlow workflow language. For example, like YAWL, e-BioFlow supports the various multiple instance patterns. The cancellation pattern, on the other hand, is supported but cannot be modelled using the control flow perspective yet. Support for this pattern requires an update of the control flow perspective.

6.3.2 Data flow perspective

Tasks are visualised as boxes, as in the control flow perspective. The input ports and output ports are represented as small boxes, distributed over, respectively, the left and the right borders of the task box (Figure 6.2(b)). The names of the ports become visible when the user moves the mouse cursor over the port. The user can add and remove ports using the configuration dialog of a task. The size of a task box grows proportionally to the number of ports to keep the ports distinguishable.

Each input and output port has a name, an *object type* and a *cardinality*. The object type and cardinality together define the data a task can produce or can consume. The object type describes the type of data a port can consume or produce and contains syntactic and semantic information. The object types are collected from repositories, such as WSDL files and MOBY-S Centrals. The level of detail in which an object type is described depends on the information stored in these repositories. The cardinality defines the number of items a task produces or consumes. Two types of cardinality are supported: *UNIT* and *COLLECTION*. The first means one item at a time is produced or consumed; the latter a set of items can be produced or consumed. e-BioFlow supports deeper levels of collections through new object types. Although the current version of the e-BioFlow workflow engine does not support streaming, this property can be used to apply streaming. The data flow perspective uses pipes to model data transfer between tasks. A pipe is visualised as an arrow between the corresponding output port and input port. e-BioFlow supports data compatibility checking using the object types and the cardinality. A pipe is normally coloured black. In case the object types or the cardinalities of the input port and output port do not match, the arrow is coloured red, denoting this pipe to be invalid.

6.3.3 Resource perspective

e-BioFlow uses a ternary relationship between tasks, actors and roles to support late binding. An actor is the real resource, such as a web service. The role describes the abilities an actor is required to have to be able to perform the task [203]. Put simply, the role defines the *type* of service required. Roles can be played by different actors and actors are able to play different roles, possibly at the same time [68; 178; 203]. If an actor 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 [129]. A role description should contain enough information to choose a suitable actor for playing the role and executing the task [239]. An actor is able to execute a task if and only if it is able to fill the role assigned to that task [203]. In this view, the designer uses the editor to only specify constraints on the binding by means of roles. The workflow engine is responsible to perform the actor-role binding. How the workflow

engine can do that is described in the next chapter. The workflow designer can set the preferred actor to play the role and to execute the task in the resource perspective. At execution time, the engine will choose the preferred actor if available, and else, it will look for alternative actors.

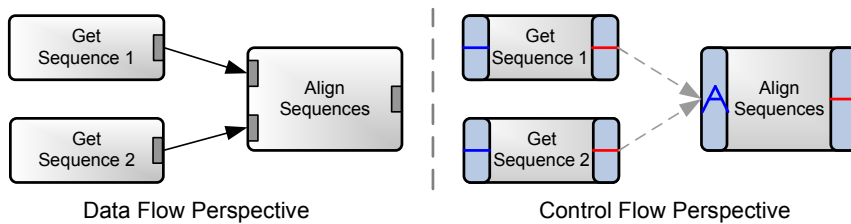
The workflow is visualised as a graph in the resource perspective, too. The visualisation of the resource perspective is closely related to the control flow perspective, in order to keep the dependency relations in sight. However, the join and split condition information is left out. Each role is painted as a box around the task it is assigned to and contains the name of the role (Figure 6.2(c)). Users can assign roles to tasks by dragging roles from a repository and dropping them on tasks.

The current implementation of the resource perspective limits the workflow designer to only assign roles to atomic tasks. An alternative would be to link every composite task to a role called “enactment engine” with the intended meaning that subworkflows can be run by workflow engines different from the one that happens to be running the parent workflow. Subworkflows could use different execution strategies, similar to Kepler, where directors are used to specify the execution strategy for each workflow [31]. 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: the e-BioFlow engine is built on top of the YAWL engine and therefore a subworkflow is always enacted by the engine of its parent task.

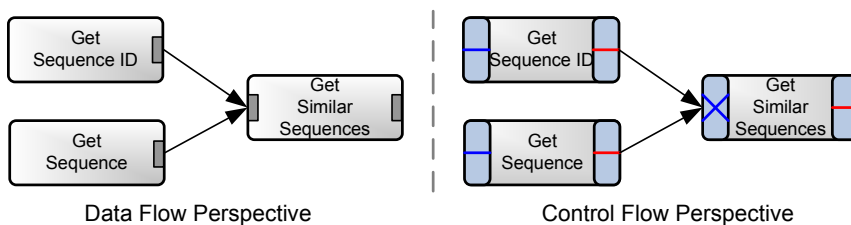
6.3.4 Linking the perspectives

All three perspectives represent the tasks of a workflow as vertices of the graph. To simplify switching between these perspectives, task positions and task sizes remain the same in all perspectives. Additionally, the zoom level and the focus point of the graph are coupled among the three perspectives. As mentioned above, the three perspectives focus on different but related information of the workflow model. To illustrate the tight coupling between the perspectives, we will briefly discuss two scenarios. In one scenario, Figure 6.4(a), the workflow designer has drawn two data pipes in the data flow perspective. Although the workflow designer has not modelled a dependency, there is a dependency between the “Align Sequences” task and the two prior tasks. The “Align Sequences” task cannot start before the two “Get Sequence” tasks have delivered their data. e-BioFlow automatically detects these dependencies, called *inferred dependencies*. When it detects an inferred dependency, it visualises the dependency as a dashed line in the control flow perspective. If the workflow designer would later remove the data pipes, e-BioFlow removes the inferred dependencies in the control flow perspective automatically.

In the other scenario, Figure 6.4(b), the designer has first inserted dependencies between the tasks using the control flow perspective. These are shown as solid lines. Later, the designer inserts data pipes using the data flow perspective. The solid lines in the control flow perspective are not affected, because the dependencies they represent 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 the e-BioFlow editor.



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



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

Figure 6.4: Two examples showing the relationship between the control flow perspective and the data flow perspective.

The e-BioFlow editor helps the user to determine the join and split type of tasks. By default, the join and split type are both set to SINGLE. If a task depends on multiple prior tasks, the editor automatically sets the join type to AND, XOR and OR. The choice depends on whether, from a data flow perspective point of view, data is required from two tasks (for example Figure 6.4(a)) or only from one task (for example Figure 6.4(b)). When all but one dependencies are removed, the join type is automatically set to SINGLE. The split type of a task is automatically set to SINGLE, if just one task follows up this task, and to AND if two or more tasks follow up this task. The workflow designer is able to change the join and split type.

A relation exists between the resource perspective and the data flow perspective. The role definition depends more or less on the input and output data types of a task, because not every actor can deal with all types of data [22]. 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 [99]. e-BioFlow helps the workflow designer to construct correct task definitions with respect to input ports and output ports.

The workflow designer can create task definitions by prototyping. e-BioFlow presents all available actors in a tree structure based on the operation they perform, called the *task panel* (Figure 6.5). This panel is positioned at the left side of the workflow panel in the editor. The task panel provides a search box to search for actors based on service name, operation type, input and output data and authority. Only actors that match the search criteria are shown. The workflow designer can drag and drop actors from the task panel to the workflow panel. This actor will be used as a prototype to generate the task, from a data flow perspective to create the corresponding ports, and from a resource perspective to attach the corresponding role. Additionally, the actor is set as the preferred actor to execute the workflow. The workflow engine uses this as a hint for actor selection.

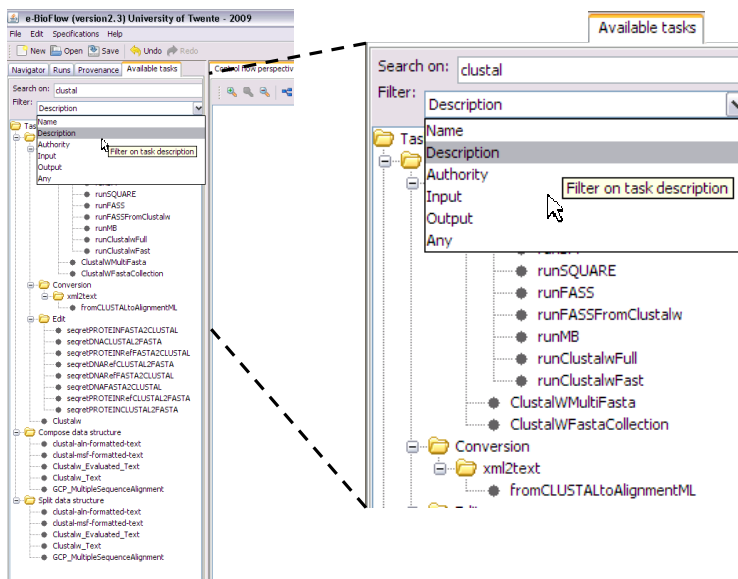


Figure 6.5: The task panel in e-BioFlow's user interface and its enlargement. The task panel enables the workflow designer to search tasks based on the name, description, authority, inputs and outputs. Only tasks that match the criteria are shown.

6.3.5 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* (Figure 6.6).

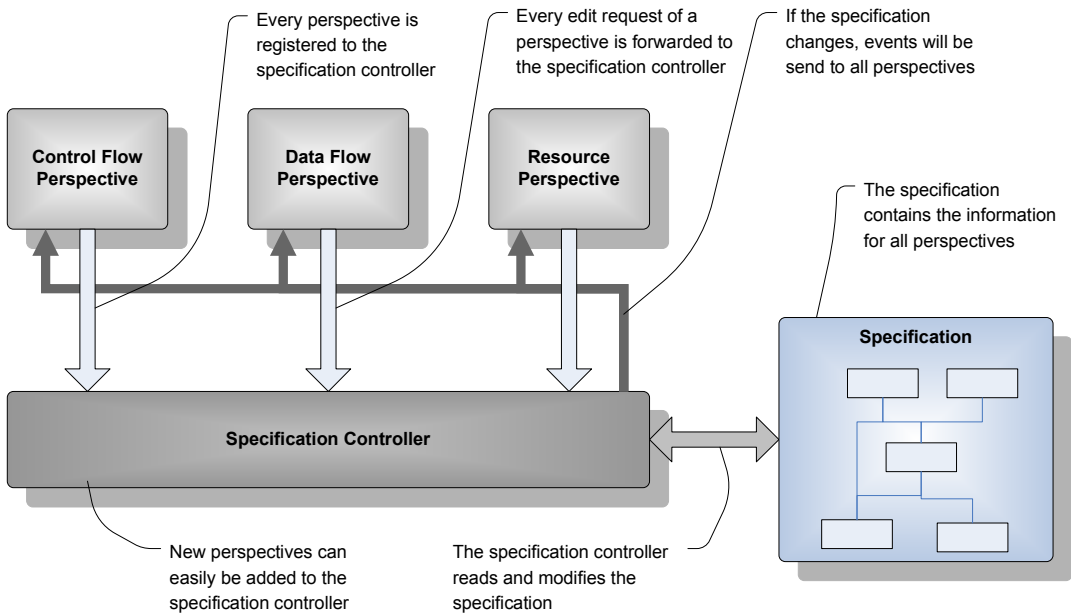


Figure 6.6: The specification controller links the different perspectives to the specification.

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 or a dependency 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 an event to all perspectives to notify that the workflow model has changed. 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. When these components are registered to the perspective, they automatically receive all notification events. A specification controller belongs to a single workflow specification. The e-BioFlow editor can handle multiple specification controllers to have its users open specifications simultaneously.

e-BioFlow is implemented in Java¹ and uses the JGraph² graph package. The default implementation of e-BioFlow is supplied with a limited set of object types, roles and actors, but is not restricted to this set. It supports plugins through the Java Plugin Framework³. Developers can easily extend e-BioFlow with new perspectives without affecting the original code. Object types, roles and actors are accessed from repositories, which can easily be extended or replaced as well due to e-BioFlow plugin-based architecture. New (local and remote) repositories can be created by extending existing repositories or adapting the provided abstract Java interfaces for these repositories. The current implementation has support for BioMOBY's MOBY-S services and SOAP/WSDL services; both implemented as plugins. Both plugins support multiple repositories (MOBY-S Centrals/SOAP/WSDL locations) in parallel. They will be further discussed in the next chapter.

e-BioFlow uses its own file format for storing workflow specifications. Control flow and data flow related information of a specification can be exported to the YAWL enactment engine format to execute a workflow. The data flow related information is translated into YAWL's task variables, net variables and XPath expressions.

e-BioFlow is able to export control flow related information to the open XML Process Definition Language (XPDL) format [191], which is maintained by the Workflow Management Coalition (WfMC). e-BioFlow can import YAWL and SCUFL, the language of Taverna [146; 145].

6.4 A simple life science case: sequence alignment

Jane uses e-BioFlow to perform a series of sequence alignments. She has designed a workflow that retrieves two sequences and uses them as inputs for the alignment task. The alignment report is shown to the user and the user is asked whether she wants to align another sequence.

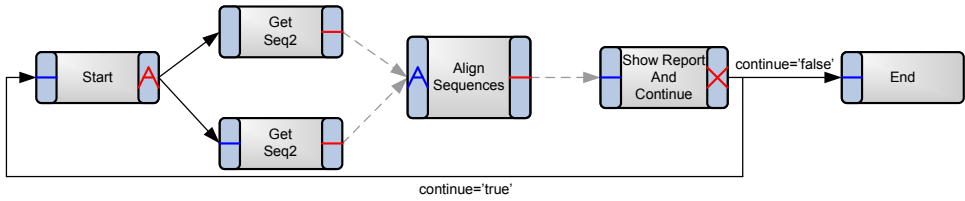
The three perspectives of the workflow Jane has designed are presented in Figure 6.7. Each perspective shows limited but complementary information about the workflow to keep the visualisation usable and comprehensible. The control flow (Figure 6.7(a)) shows

¹<http://www.java.sun.com>, last visited: December 2009

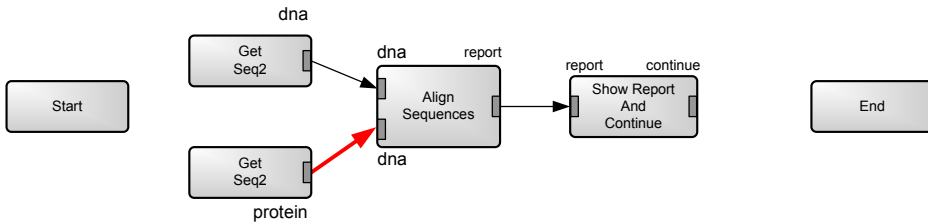
²<http://www.jgraph.com>, last visited: December 2009

³<http://jpf.sourceforge.net/>, last visited: December 2009

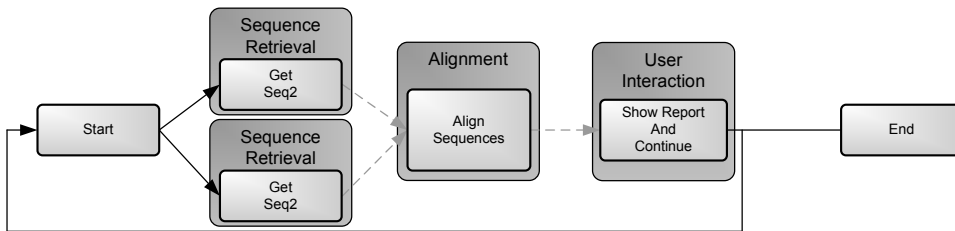
the order of task execution. The start task activates both tasks to retrieve sequences. The sequence alignment has to wait until these two sequences are collected, denoted by an AND join. When the alignment is performed, the results are shown. The XOR split of this task defines that either a next iteration of the sequence alignment is started or the end task is activated. The dotted arrows denote dependencies inferred from pipes present in the data flow perspective.



(a) Control flow perspective



(b) Data flow perspective



(c) Resource perspective

Figure 6.7: The different perspectives on a sequence alignment workflow.

The data flow (Figure 6.7(b)) shows the data transfer between the tasks. The alignment task gets input sequences from both prior tasks; the alignment report is sent to the task that shows the results. This task delivers a 'continue' value as output, which is used to determine whether a next iteration should be done. The bold red arrow between the "Get Seq2" task and the "Align Sequences" task denotes a data incompatibility problem. Jane

has to correct this problem.

The roles of the tasks are shown in figure 6.7(c). The start and end task do not have roles, since these are used by the enactment engine to provide input data and to collect output data. Both tasks for getting the input sequences 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. This task can be executed by, for example, in-house software or again a web service. The task that shows the results requires a user interaction actor. Of course, the binding of the tasks to the actors will be done by the workflow engine.

6.5 Conclusion

The user interface of the e-BioFlow workflow editor provides a control flow perspective, a data flow perspective and a resource perspective. Using these three perspectives, the workflow designer can model both control flow and data flow related information of a workflow and design workflows that are independent of resources.

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 from a control flow perspective is possibly through exporting the workflow to the format YAWL uses. Embedding YAWL’s validation tools into e-BioFlow is future work. Validation from a data flow perspective is possible directly from within the e-BioFlow editor. Validation in the resource perspective is currently not available, because more investigation is required to support a formal validation in this perspective. But, the e-BioFlow editor helps the workflow designer to construct correct task-role relationships by means of prototyping. The existence of web portals, such as myExperiment, introduces a new opportunity, namely workflow sharing. Workflow sharing, however, becomes difficult if workflow specifications are tied to specific resources, as we have already seen in the previous chapter. The e-BioFlow language and the e-BioFlow editor provides facilities to design templates for experiments independent of the resources available at design time. The workflow designer is able to model almost all aspects of a scientific workflow using the three perspectives provided by the editor.

The 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. But it still requires 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. In the next chapter, we will discuss the e-BioFlow engine, which fully supports late binding and can execute the workflows designed in the e-BioFlow editor.

The workflow engine as a movie set

7.1 Introduction

Workflow systems facilitate in-silico life science experiments by providing bioinformaticians access to various (online) scientific resources and helping them connect these resources. Due to workflow sharing solutions, such as myExperiment [72; 76], workflows have become resources themselves for sharing and exchanging knowledge of an experiment. Bioinformaticians reuse workflows for constructing and executing similar experiments [70]. Furthermore, workflows are increasingly accepted as means for publication to share experiments and experiment results [70].

A workflow specification often contains physical locations of the resources used. As shown in Chapter 5, resources can be (temporarily) unavailable, can be replaced or moved to a different location. This complicates workflow reuse [85]. Delaying the choice of resources until instantiation time would be a solution [70], and is known as *late binding* [179; 239].

In the previous chapter, we have introduced the e-BioFlow workflow editor [218]. This editor is a so-called *hybrid workflow editor*: it can model aspects from both the control flow perspective and the data flow perspective of a workflow. With late binding in mind, we introduced a third perspective in that chapter, called the resource perspective, to distinguish between the task to be executed and the resource to execute the task. Although the workflow designer creates workflows by dragging and dropping real resources, such as web services, into the workflow graph, the specification is not bound to these resources. These resources are used as examples to construct the task definition in the workflow. The resources are set as preferred or suggested ones, but the workflow enactor can still search for alternatives hosted by different organisations at runtime.

In this chapter, we introduce the e-BioFlow workflow engine, which can enact the workflows designed in the e-BioFlow editor. The architecture of the engine is based on the movie set metaphor to provide an easy-to-understand and extensible structure to implement late binding. The e-BioFlow engine has been arranged with support for different types of bioinformatics web services, different scripting languages, and user interaction. The engine is an integrated part of the e-BioFlow workflow system.

We will first give an overview of related work and introduce the terms used in the rest of the chapter. Second, we present the Movie Metaphor and, third, we will show how it is applied to the e-BioFlow engine. We will end with a discussion and future work.

7.2 Behind the scene: related work

Only a few life science workflow systems currently support late binding. ESCOGITARE supports late binding, but is built on the Globus Toolkit [67] and therefore it requires services to be compliant with the WS-Resource Framework (WSRF) [82]. KNIME [26], Pegasus [54] and Triana [134; 187] all focus on grid computing in which computationally intensive jobs are scheduled among a cluster of computers. This is somewhat different from the definition we use, according to which late binding denotes the abstraction of tasks from specific web services until execution time. JOpera [150; 151] supports late binding, but requires special constructs in the workflow design to model this.

Our solution borrows terms from the movie set. Kepler [31; 11] uses terms from the movie set too, but not to support late binding, but to be flexible with respect to execution models. An execution model refers to the way actors exchange data and is called a director. Kepler provides different directors, such as the continuous time director and the discrete event director. The term actor is used to refer to resources. However, no distinction is made between the actor and the task it executes, so the choice of actors to execute a task is made at design time of the workflow. Kepler supports late binding by means of actor replacement [129], but requires the workflow designer to use special constructs at design time.

Adams et al. [3] have extended the YAWL [200] system to support worklets. A worklet is a workflow that handles one specific task in a larger, composite workflow. The use of worklets enables the design of abstract, reusable workflows that can be used for different cases with a similar high level structure. The extension is implemented as a new YAWL environment. The YAWL engine delegates tasks to environments. This worklet environment contains a registry of the worklets available. Once the engine activates a task, the worklet environment dynamically chooses the most appropriate worklet. The worklet is also executed by the YAWL engine, but as a separate workflow instance. A major diffe-

rence between the worklet approach and the approach we present is that this approach operates at workflow level by substituting tasks by subworkflows. Our approach operates at resource level instead.

Workflow systems provide access to different type of resources, of which the most important are web services, local services, scripting engines and user interaction. Among the web services, SOAP/WSDL services are most popular, possibly because of the high number of these services available and the multitude of libraries available to access them. YAWL and WS-BPEL are examples of business workflow systems that provide access to SOAP/WSDL services. Taverna [146; 147; 148], Kepler [11; 129] and Triana [134; 187] are well-known examples of life science workflow systems that can access such services.

The MOBY-S framework [234; 233; 235; 188], developed by the BioMOBY Consortium, is specific for the life science domain. Software packages for Java and Perl are available to access this registry. These software packages provide facilities to search for a service based on service description, service type, service provider, and the data type it produces or consumes. Many life science applications provide access to MOBY-S services, such as Gbrowse MOBY [232], REMORA [40], Seahawk [80], but at the moment Taverna is the only workflow system that provides access to MOBY-S framework [107].

As discussed in Chapter 4, scripting is used in workflows for many reasons, among others, to perform data transformation [174; 30], to implement tasks not available as web services or local services [106], and to create interaction tasks [239]. Many workflow systems have support for scripting languages. The Pilot system [106] provides scripting facilities by means of its PilotScript language. This language, however, requires its users to learn a whole new scripting language. Scripting will be easier and less error prone, if users can choose the scripting languages they are already familiar with. Most workflow systems have connectivity to existing programming languages [18]. Taverna enables its users to program snippets of Java code by means of the Java BeanShell processor [125]. The I³ system [33] supports three broadly used programming languages, namely Java, Ruby and Perl. An example of an activity that cannot be packed into a single web service is statistical analysis, because it highly depends on the experiment at issue. In the life science domain, R [98] is a language much used to perform statistical analysis [116]. Therefore, we have created RShell [125; 215], a plugin for Taverna to add support for this language (Appendix D). For the same reason, the I³ system and KNIME support this language.

Restructuring data before passing it to another task is essential [174] and is often done by creating small scripts. The MOBY-S plugin for Taverna provides a different approach. It has so-called *composer* and *splitter* tasks, which are local services that help the workflow designer to automatically compose and decompose the XML structures defined in the MOBY Central without scripting [107]. Automatically composing and decomposing XML

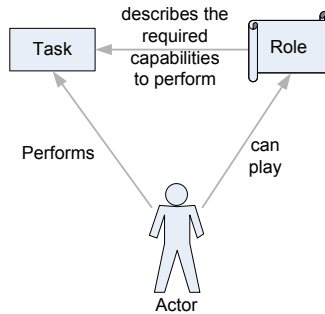


Figure 7.1: The role attached to a task describes the required capabilities of an actor to be allowed to execute a task.

data structures is less error prone than scripting. For similar reasons, Taverna provides facilities to split and compose XML data structures defined in WSDL files.

User interaction is employed in scientific workflows to show intermediate results to the users for making a selection, or for giving some feedback. The scripting facilities can also help to construct interactive tasks [125], but programming a user-friendly interactive task can be difficult for the less experienced programmer.

ESCOGITARE [119] and Taverna [120] both have so called *interaction services* to help the designer to construct user interaction tasks. The idea is borrowed from business workflows, such as Lynx [209], where the user gets notified by an e-mail message when an interaction service is activated. This message contains a URL to a dynamically generated web page to interact with the user. These types of service are highly configurable to support a large set of interactions. To construct such services, the user still needs to program.

7.3 A workflow system as a movie set

The Movie Metaphor is introduced in the Dutch Driving Simulator [221; 222] to create an easy-to-understand architecture that supports dynamic generation of traffic scenarios. In the Movie Metaphor, the world is seen as a movie set in which actors play roles and execute tasks conforming to that role (Figure 7.1). We will demonstrate that this metaphor can be applied to a workflow system to create an easy-to-understand architecture for a workflow engine.

The workflow specification is the script of the movie. The designer of the workflow specification is, in fact, a script writer.

Instantiating a workflow specification can be compared to making a movie. On the

Table 7.1: Terms in the movie set and their usage in the workflow paradigm.

Term	On the movie set	In workflow context
Script	The story describing the movie.	A (hierarchical) workflow specification. A workflow specifies the tasks to be executed, the order of execution, and the exchange of information between tasks.
Scene	A unit of actions, taken at a single location.	A (sub)workflow, consisting of tasks constituting a higher level task.
Actor	A person with the capabilities to play a certain role.	A resource, such as a web service or script engine, that is able to perform a certain task.
Role	Specification of a character and its tasks.	A specification of the type of actor required to perform a task.
Director	Person who directs the movie.	The workflow engine; it selects and schedules tasks.
Casting director	Person responsible for selecting actors based on the role descriptions.	The component that selects actors based on the role attached to the task.

movie set, the services are the actors and the workflow engine is the director. The role describes the type of actor required to execute the task [218]. The director controls the set; it selects the tasks to be executed based on the script. The director does not work in isolation but gets help from the casting director to select the actors to execute tasks (known as actor assignment [19]). Like the script of a theatrical play, a workflow specification can be performed (instantiated) more than once and each performance can have different actors involved. Table 7.1 presents an overview of the terms we have borrowed from the movie set, what they stand for and how they can be interpreted in a workflow context.

The actors, director and casting director interact. Based on the script, the director determines the tasks to be executed. For each of these tasks, the director asks the casting director for an actor to execute it. The casting director searches for an actor in so-called *actor repositories*, which function as casting agencies. Multiple actors can be available to play the same role. Based on the task and the role attached to it, the casting director selects a capable actor and delivers it to the director. In workflow terms, a suitable service (web service, command line tool or an interactive application) is selected based on the role (type of service) and the task (operation type, input data and desired output data). The director delegates the task to the selected actor, which then executes the task. Besides

selecting tasks and delegating tasks to actors, the director is also responsible for providing and accepting the data consumed and produced by actors.

7.4 The life science movie set

In e-BioFlow, the architecture of the workflow engine has been arranged to provide access to various life science resources. Actors are available to execute SOAP/WSDL and MOBY-S services, scripts written in three different languages and an interaction task. Additionally, e-BioFlow provides composers and decomposers to construct and parse XML structures used by the web services. The system can be extended with, among others, support for other protocols and scripting languages.

7.4.1 Using YAWL as a director

The implementation of the director is built on an existing workflow engine called the YAWL system [200]. YAWL is an open source workflow engine, written in Java. The YAWL engine can run workflows described in the control flow language with the same name, YAWL (Yet Another Workflow Language) [202]. As mentioned above, the engine schedules and delegates tasks of a workflow to so-called environments and therefore closely matches the director functionality. These environments normally access web services. Each environment usually represents a single protocol. To add support for new protocols, one has to implement a new environment. A task definition in YAWL contains a reference to the service to be invoked. The engine uses this reference to determine to which environment the task needs to be delegated. The corresponding environment uses the reference to invoke the service. When it has finished the service invocation, it notifies the engine and sends the results to the engine.

The language YAWL is normally used to describe business processes. The adoption of a business workflow language into the life science domain is not unusual. ESCOGITARE [119], for example, uses BPEL (Business Process Execution Language) to benefit from the flexibility that business workflow languages provide. In contrast to BPEL, YAWL is a formal workflow language [35], which becomes of special importance when workflows become large. At the moment, YAWL lacks functionalities important for the life science domain:

- Late binding is not supported, because service references are hard-coded in the task definition. These references are used by the engine to determine the environment to invoke the service.

- The engine uses XPath expressions in conditional branching to determine the branches to activate. Not all workflow designers are familiar with XPath.
- The YAWL engine has problems with large data sets, because data is stored in working memory using an XML data structure.
- In its current state, YAWL only supports SOAP/WSDL services. YAWL is not able to access other protocols, such as the MOBY-S services, and to invoke scripting engines, such as the R interpreter.

Although YAWL in its current state does not fulfill the requirements to execute life science workflows, it is a better starting point than data flow oriented systems, because of the control flow patterns it supports. The use of environments makes the YAWL engine highly adaptable to support a variety of protocols.

An environment of actors

The current implementation of e-BioFlow uses the original YAWL engine code with little modifications. We have created a new environment for YAWL called the *actor environment*, which takes care of all tasks to be executed. The actor environment contains all actors. Each actor is a protocol-dependent service invoker, and represents a single service. It is possible that two actors represent the same service, but use different protocols. As mentioned before, the actors are stored in actor repositories. In general, all actors within the same repository use the same invocation protocol. Thanks to e-BioFlow's plugin architecture, new repositories can be implemented to add actors that support different invocation mechanisms.

Figure 7.2 illustrates the different actor repositories currently provided by e-BioFlow and the relations between the actors, the director and casting director.

When the director enables a task, it requests the casting director to find a suitable actor. The casting director searches in the various actor repositories in the environment using the role attached to the task. Three situations can happen. i) Workflow designers can have defined a preferred actor for a task. If the casting director finds this preferred actor, it returns this actor. ii) If no preferred actor is set or available and only a single actor is suitable, the casting director selects this actor. iii) If multiple actors are available, the casting director asks the user to make a choice. Automatic service selection is only appreciated if a service needs to be selected from among a set of mirrors, because the bioinformatician wants to keep in control of the resource selection. But in practice, it is difficult to determine the most up-to-date service or mirror [142]. The user can set the chosen actor to be the preferred actor for this role in this workflow run. The director delegates the task to the actor chosen.

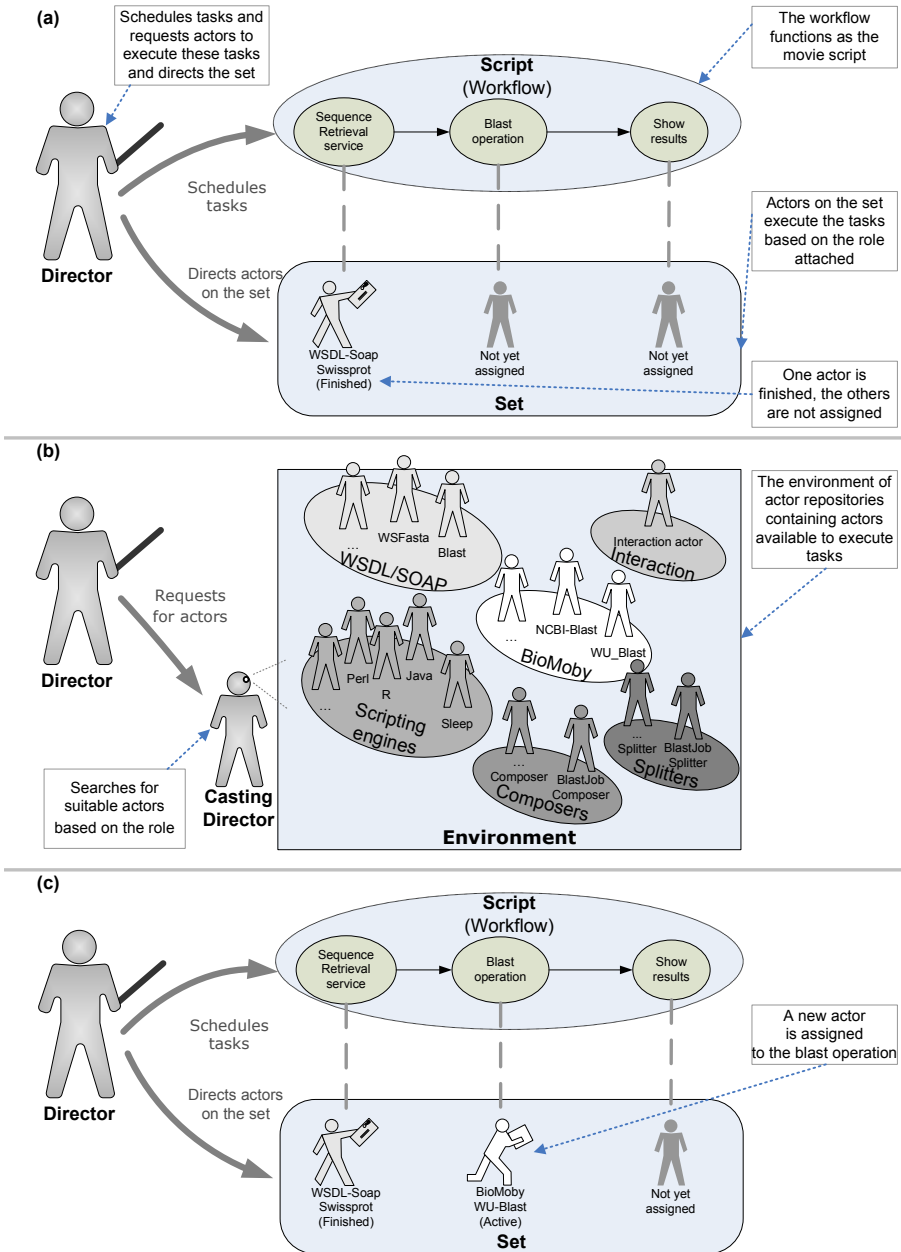


Figure 7.2: The life science movie set: the director directs the workflow and schedules tasks (a). It requests the casting director for actors when a task becomes enabled. The casting director searches for suitable actors based the role description (b). The selected actor executes the task (c).

More advanced actor selection procedures can be added by implementing a new casting director. For example, actors can be selected using a kind of priority list, based on preference, quality or availability, just like bioinformaticians normally do [116].

Passing data between actors using references

YAWL uses workflow variables as intermediate storage locations to pass data between tasks [200]. These workflow variables form a shared memory among the tasks in a workflow and can be accessed by all tasks. Internally, the YAWL engine uses a single XML document per workflow stored in main memory for book-keeping the workflow variables. In the YAWL editor, the workflow designer has to define XPath expressions when a task needs to get access to the workflow variables in the XML document. The way YAWL handles data is not suitable to deal with the large data sets the life science domain works with, for three reasons. First, the amount of data generated during the run of a large life science workflow will not fit in main memory. For example, the use case in the next chapter generates almost 3GB data. At the moment, an average desktop computer cannot keep this amount of data in main memory. Second, every time the YAWL engine reads a variable, it creates a copy of the data contained in a workflow variable. Third, when the data stored in a workflow variable is XML itself, YAWL recognises this as being part of the XML document structure instead of the value of a workflow variable. As a result, the data can be written, but not read.

Before a specification designed in e-BioFlow is passed to the YAWL engine, e-BioFlow translates it to the format YAWL uses. The e-BioFlow engine uses pipes to model data transfer between two tasks. Every pipe connects a single output of a task to a single input of another task; multiple pipes can exist between tasks. At instantiation time, e-BioFlow translates each pipe to a workflow variable and creates XPath expressions to write to and to read from this workflow variable (Figure 7.3).

e-BioFlow prevents creating multiple copies of the data by using references to these data. In case of a data transfer between two tasks, e-BioFlow writes the references to the data produced into the workflow variables, instead of the data itself. The data is stored in a central storage place named the *data manager*. This data manager creates and returns references to the data it stores. The actor that requires the data as input gets references to the data from the engine. The actor can use these references to access the data. It requires no knowledge about implementation details of the data manager. e-BioFlow currently provides two data managers. The first keeps all data in main memory. The second uses a PostgreSQL database¹ to store and retrieve data; the data is only in memory when retrieved from or sent to an actor.

¹<http://www.postgresql.org>, last visited: December 2009

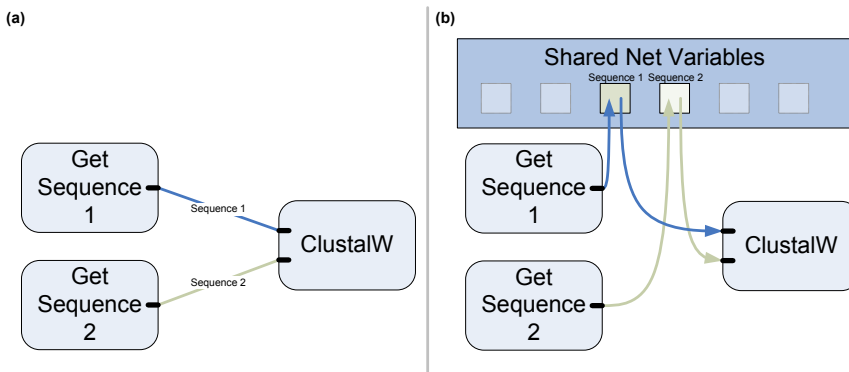


Figure 7.3: The pipes used in e-BioFlow to define data transfer between tasks (a) are translated to the workflow variables used in YAWL (b). ClustalW [194] is a life science web service to compare DNA or protein sequences.

The use of references keeps the data out of the engine and therefore solves the three problems mentioned above. Our solution also introduces a new problem. When conditional branching is used in a workflow specification, XPath expressions are used to decide which branches should be activated. Most XPath expressions refer to the data stored in workflow variables. Due to the use of references, the real data is not present in the workflow anymore. We have modified the YAWL engine to support references to evaluate XPath expressions. The e-BioFlow editor helps the workflow designer to construct these XPath expressions without requiring him to have knowledge about the syntax they have and YAWL's internal XML document for workflow variables.

7.4.2 Actors that invoke web services

In its current state, e-BioFlow has support for SOAP/WSDL and MOBY-S services. As mentioned before, SOAP/WSDL services are described in WSDL files. e-BioFlow has no support for a central registry for SOAP/WSDL services, such as UDDI, since they are rarely used in the life science domain. But, e-BioFlow is able to handle multiple WSDL files. WSDL locations can be added to e-BioFlow's configuration file. A WSDL file contains definitions for data types, operations and bindings of operations to physical locations on the web. All operations defined in the WSDL file are translated to roles and are registered to e-BioFlow. If multiple WSDL files provide the same operation definition, only one role is registered for that definition. For each WSDL location, an actor repository is created. The bindings are translated to actors; these actors are stored in the corresponding actor repository.

Support for MOBY-S services is implemented in similar fashion. e-BioFlow has support for multiple MOBY Centrals. A MOBY Central contains a taxonomy of service types and services. Service types can have children, which are either service types or services; services are always childless. The service types are translated to roles and are registered to e-BioFlow. If two Centrals contain the same service type, only one role representing that service type is registered. In the user interface, the parent-child relationship of service types is visualised in the list of available roles, to simplify the discovery of roles. An actor repository is created for each MOBY Central. All the services within a Central are translated to actors, which are stored in the corresponding actor repository.

7.4.3 Actors that can execute scripts

e-BioFlow contains a single role to denote scripting tasks, named the “scripting role”. It is used for all scripting languages that e-BioFlow supports. This scripting role is a *configurable role*: the workflow designer can specify the input the actor should consume, the output it should produce, the script it should execute and the language the script is written in. The definition of the script language contains the name and possibly a version number. Both are used by the casting director to select a suitable actor for scripting tasks. In this way, e-BioFlow supports multiple installations of interpreters for the same programming language. This can be very useful when a specific version of a language is required or when one installation provides libraries which other installations do not.

By default, e-BioFlow has actor implementations for the four scripting languages Java BeanShell ², R [98] ³, Perl ⁴ and Sleep ⁵:

Java BeanShell The Java BeanShell actor can execute BeanShell scripts. BeanShell is a Java dialect to be executed by a Java package that supports dynamic execution of Java code without the need to compile. It does not require the user to install additional software, because it is completely written in Java.

R: The R actor can execute scripts written in the R language. It is based on the implementation of the RShell plugin for Taverna [125; 217] (Appendix D). It uses the RServe library [198], which turns the R interpreter into an R server. The R server uses a TCP/IP connection to enable external programs to send data and scripts to the R interpreter and to retrieve results. Our R actor is implemented as a client for this server. Each time a script has to be executed, it sends the input data and the script

²<http://www.beanshell.org>, last visited: December 2009

³<http://www.r-project.org>, last visited: December 2009

⁴<http://www.perl.org>, last visited: December 2009

⁵<http://sleep.dashnine.org>, last visited: December 2009

to the server as a request operation. The R server executes the script and returns the results to the R actor. Due to the TCP/IP connection used, e-BioFlow can access remote installations of R with the RServe library installed. Multiple R actors can exist side by side in e-BioFlow, which can be useful when different R interpreters have different libraries installed.

Perl: To run Perl scripts, a local installation of the Perl interpreter is required. At startup, e-BioFlow automatically searches for Perl interpreters at default locations. Paths to interpreters can also be specified manually. Each interpreter found is represented is registered as a CGI-servlet to the Jetty server ⁶ and represents a Perl actor. The Jetty server is a web server component also used by the YAWL engine embedded in e-BioFlow. e-BioFlow supports multiple Perl interpreters at the same time. The Perl actor uses XML-based Remote Procedure Calls technology (XML-RPC) ⁷ to communicate with the interpreter.

Sleep: If no Perl installation is available and no special libraries are required, one can also use the Sleep language, which has almost the same syntax as Perl. This interpreter is completely written in Java and therefore does not require additional software to be installed. Furthermore, it provides direct access to Java classes.

Of course, e-BioFlow is not restricted to these scripting languages. The reason for picking these languages is that they are used by many bioinformaticians and they show different approaches to support scripting within e-BioFlow. In the near future, adding new scripting languages will become even simpler due to the existence of the “Scripting support for Java Platform” ⁸.

7.4.4 Actors that help to pass data

Reformatting data exists at two levels: restructuring complex data structures and translating the encoding scheme that is used. A complex data structure is a composition of other data structures. Restructuring data is solved in e-BioFlow for both complex WSDL data types and complex MOBY-S data types. Both data types are XML based. e-BioFlow provides composer and decomposer roles to help the workflow designer compose and decompose XML data structures. A composer role defines the inputs required to construct the complex data, and the output, which is the complex data. A decomposer role defines the required input, which is the complex data, and the outputs, which are the sub-fields of the complex data. The composer and decomposer roles handle only one level of

⁶<http://www.mortbay.org>, last visited: December 2009

⁷<http://www.xmlrpc.com>, last visited: December 2009

⁸<http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting>, last visited: December 2009

complex types. If a nested complex data type needs to be further composed or decomposed, composer roles and decomposer roles can be chained to reach the desired level of composition or decomposition. A composer actor and a decomposer actor are available to play these roles and to automatically compose and split these complex data types.

The automatic composition of WSDL and MOBY-S data types is relatively simple, because both are XML documents. A huge number of generic XML builders and parsers are available to compose and decompose these structures [2; 168]. If two services use the same type of data but different encoding schemes, decomposing and composing will not help. This is normally done by scripting. Of course, the scripting facilities provided by e-BioFlow can help to do this.

7.4.5 An actor that interacts with the user

e-BioFlow provides a user interaction role and a user interaction actor to play that role. The role defines a task to be an interaction task. The interaction actor presents a dialog with fields showing the incoming data and fields in which the user can enter the outgoing data of the task. This actor can visualise different types of data, among others, plain text, XML, and graphical data such as JPG, PNG, SVG, and PDF.

This one-size-fits-all solution makes it very easy to define user interaction tasks in a workflow specification. New interaction actors can be added to support more advanced interaction patterns, such as navigating through the input data and making selections. This is future work.

7.4.6 Recording the movie

The director generates 12 different types of events during the workflow enactment (Table 7.2). Listeners can be registered to the director to receive these events. Receiving these events is used among others to visualise the progress of the workflow enactment and to implement the provenance system. Chapter 8 will discuss the provenance system for e-BioFlow.

7.5 Conclusion

The hybrid workflow editor e-BioFlow discussed in the previous chapter has been complemented with the e-BioFlow engine. Its architecture is based on the Movie Metaphor to create an easy to understand and extensible architecture that supports late binding by default. The responsibility of the components in the workflow execution environment fairly match the staff involved in movie making. The tasks in the workflows designed in

Table 7.2: An overview of the different events the director sends during workflow execution, categorised per level of the workflow specification.

Level	Event	Description
Specification	Instantiated	A specification is instantiated.
	Started	The specification is started; the main workflow of the specification can be instantiated now.
	Finished	The complete specification including the workflows and tasks are finished.
	Canceled	The whole specification is canceled.
Workflow	Instantiated	The main workflow or a subworkflow has been instantiated.
	Started	The workflow is started; the first task of the workflow will now be started.
	Finished	The workflow, including all its tasks, is finished.
	Canceled	The execution of the workflow is canceled.
Task	Instantiated	A task is instantiated. Now, the casting director will search for a suitable actor.
	Started	A suitable actor is found, the input data is available and the task is started.
	Finished	The task is finished and the output data is available.
	Canceled	The task is canceled.
Dependency	Followed	This dependency is followed to activate the next task.

e-BioFlow are not directly coupled to resources. Therefore, these specifications are easier to reuse than workflow specifications made in other workflow systems. The approach of Adams et al. [3] can perfectly complement our approach. This will result in a workflow system that is not only able to perform late binding by choosing a resource at runtime, but also by choosing worklets that represents subworkflows. We are aware of the large number of scientific and business workflow systems available. To prevent creating *yet another workflow system* from scratch, the YAWL engine is used to fulfill the role of the director. This engine has been modified to be able to deal with large data sets.

New protocols, scripting languages and user interaction tasks can easily be implemented and connected to e-BioFlow due to its plugin structure. The e-BioFlow system has been arranged to provide access to various life science resources. It can access many life science web services due to the support of SOAP/WSDL and MOBY-S services. e-BioFlow has support for different scripting languages, so workflow designer can choose the language he prefers. A generic user interaction actor helps the workflow designer to create interaction tasks in workflows. Data transformation takes a large part of the activities of workflow designers. The composer and decomposer tasks and scripting possibilities provided by e-BioFlow help them to perform these data transformations.

Capturing provenance data in e-BioFlow

8.1 Introduction

Provenance data comprise the complete derivation history of experiment results. These data are as important as the experiment itself, because they are indispensable for assessing the quality of the experiment results [70]. Provenance data make experiments reproducible, simplify the discovery of changes in the underlying data and pay credit to the owners of data and resources [71; 87; 175]. The provenance data are suitable for the inspection of (intermediate) results [84], for simulating the workflow model [163] and for the validation of experiment results [237; 137]. Provenance data are only useful if they are interchangeable with peers. This is only feasible, if a standardised provenance model is used [38; 4; 87; 18; 195]. Workflow systems are ideal platforms for automatically storing provenance data [16]. They can manage what is called a process-oriented provenance model [241; 175]. They ‘know’ which resources are accessed, when they are accessed and what data are exchanged between the resources [163]. That is the reason why well-known workflow systems such as Taverna [242], Kepler [10], and Triana [134] all support the automatic capture of provenance data.

Despite the high interest in provenance, it is still an open research field [237]. Davidson and Freire [50] point out four major open challenges in storing and using provenance data: i) interoperability, ii) analyse and visualise provenance data, iii) information overload, and iv) database storage for workflow provenance.

In this chapter, we will introduce e-BioFlow’s provenance system. This system is ba-

sed on the Open Provenance Model (OPM), an open standard for storing and exchanging provenance data. We will first discuss related work. Then we will demonstrate how e-BioFlow's provenance system is used as cache to improve the speed of workflow enactments. e-BioFlow's provenance browser provides the means to explore the provenance data stored by the provenance system. The implementation of the provenance system will be tested with the OligoRAP use case. We will end with a discussion.

8.2 Related work

At the moment, most workflow systems use their own data format to store provenance data. As a result, the provenance data captured using one of these systems cannot be read and analysed in other systems. Only a few approaches exist to establish a standardised provenance data format. One of them is the Minimal Information About Microarray Experiments (MIAME) [34]. MIAME is specifically designed to capture provenance data of microarray experiments, and therefore is limited to this type of experiments. Scientific workflow systems require a generic provenance model, because they provide access to many different resources and are not limited to microarray experiments.

The Provenance Recording Protocol (PReP) [87] is an implementation-independent protocol that defines how provenance should be captured in service-oriented grids. It describes the messages that should be exchanged between the resources and the provenance system and how they should be recorded. Its goal is to make data interoperable and shareable among different parties by differentiating the provenance system from the main application. An actual implementation of PReP is found in PReServ [88]. PReServ is a back-end for storing provenance data. This back-end can be connected to applications that have implemented the Provenance Recording Protocol.

Recently, the Open Provenance Challenge has resulted in a proposal for storing and sharing provenance data, among others but not limited to workflow systems. The Open Provenance Challenge is a collaboration between various disciplines including the life science domain. Their proposal, called the Open Provenance Model (OPM) [139; 138]¹ is a generic model, that defines core relations between the concepts processes (tasks), artifacts (data) and agents (resources). In contrast to the PReP, it does not describe how to capture the data, but only what to store. Provenance collected conforming to the OPM can be represented as a directed acyclic graph. The nodes represent the concepts mentioned above; edges represent causal dependencies between these concepts, such as “was generated by” and “used by” (Figure 8.1). The OPM provides so-called accounts to represent alternative views on OPM concepts. Two accounts can reflect the same set of

¹<http://openprovenance.org/>, last visited: December 2009

concepts, where one represents an abstract view (black box) and the other a detailed view (white box) of these concepts. A formal definition of the OPM is given in Kwasnikowska et al. [118].

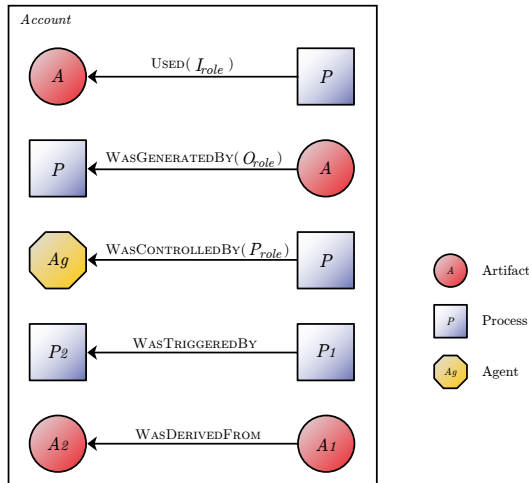


Figure 8.1: OPM concepts and possible relations between them.

The Open Provenance Challenge has developed XML and RDF schemas for the serialisation of OPM data. Serializing all provenance data, including the data passed between tasks, into a single file can result in very large files, which will end in scalability problems [175]. Therefore, how to include data in an OPM XML serialization is left undefined at the moment.

A difficulty of the OPM is that it is meant to be open. Open, in this context does not only mean providing an interchangeable format, but also providing an adaptable format, in order to keep it applicable as widely as possible. How provenance data of a workflow are translated into the concepts OPM defines is not part of the OPM specification, because the OPM intends to be technology-agnostic. The actual meaning of these concepts needs to be described in a so-called *OPM profile*. The profile is a dictionary that is required to interpret the provenance data stored in OPM format. Every provenance system supporting OPM uses an OPM profile, albeit implicitly. When OPM data are exchanged between different systems, an explicitly defined OPM profile becomes essential to interpret these data. At the moment, the only explicitly defined OPM profile we are aware of is the profile defined by Kwasnikowska et al. [118]. They have defined a formal translation of their NRC DataFlow Model into the OPM, but an actual implementation is not reported.

The OPM specification is still a working draft. In May 2009, the third Open Provenance

Challenge workshop was held in Amsterdam. The main goal of this workshop was to test interoperability between the OPM implementations of the parties involved. We participated in this workshop to discuss the OPM implementation in e-BioFlow and our findings about the current state of the OPM. Four major problems have been discussed during this workshop: i) a standardised OPM profile, ii) how to include the data transferred between tasks, iii) how to express dependencies between tasks in case no data transfer is involved, and iv) how to visualise provenance data.

8.2.1 Provenance as cache

Caching in workflow systems helps to speed up the execution of tasks in workflows that have been executed previously. Caching for example is useful in case a workflow crashes or when the workflow designer wants to test parts of the workflow without rerunning the entire workflow. The enactment of a workflow can take several hours, or even days. If the workflow system without a workflow restore mechanism crashes, all tasks have to be re-executed. For this reason, the workflow discussed in Chapter 4 has many tasks to restore the workflow state in case of a crash. Ideally, the workflow system itself provides the facilities for this. Provenance systems are very suitable to be used as cache. These systems have all the information of previous runs that can be used to speed up future executions of tasks.

The VisTrails [39] is a pipeline system for the design and maintenance of interactive visualisation pipelines. The provenance system of VisTrails is a combination of a file-based system and database system: raw data are stored as files; meta-data (i.e., the trail specification and instance information) are stored in the database. This system uses a caching mechanism to prevent redundant computations and to speed up overlapping parts of the pipeline. Nodes of the pipeline are treated as input-output functions. When a node is activated, the VisTrails system checks if for the given input an output is stored in the cache. If this is the case, the node is replaced by a node that fetches the data automatically from the cache. VisTrails enables its users to define nodes to be “non-cacheable”. The outputs of these nodes are not saved.

Altintas et al. [10] have implemented a variation of the VisTrails algorithm to provide a caching mechanism for their Kepler system. This variation uses the provenance archive as a cache. The cache enables smart re-runs: results of earlier task instances are fetched from the provenance archive and used as output of new task instances. This is only applicable, of course, when a task is executed with equal inputs and the tasks are deterministic. In Kepler, the nodes of the workflow are called actors. An actor refers to the service or tool to be invoked. When smart re-runs are enabled, Kepler automatically replaces deterministic actors by “Stream Actors”. The “Stream Actor” fetches the necessary

data from the provenance archive to produce the output the original actor would have produced. Similar to VisTrails, Kepler enables the workflow designer to mark actors to be “non-cacheable”. The data of these actors are not replaced by “Stream Actors”.

Other caching schemes extend service invocation protocols directly, such as the SOAP extension by Seltzsam et al.[169]. The Taverna Webservice Data Proxy uses a similar concept. It is developed to keep large data sets out of the Taverna engine². It can also be used to store intermediate results to serve as a cache in order to speed up the re-execution of workflows.

8.2.2 Provenance visualisation

Equally important to provenance system is a provenance visualisation system, which we call a provenance browser. This provenance browser should help its users to explore the tasks executed, the actors involved and the data generated during a run [87]. Additionally, the provenance browser needs to provide access to different levels of detail with respect to hierarchical workflows. It should be interactive. By clicking on the nodes of the provenance graph, the users should be able to inspect task and actor related information and the data passed between the tasks.

A provenance browser is not only useful to understand and check the enactment of the workflow, but also during debugging workflows to discover problems in case of a service failure. Combined with caching, it can greatly improve the speed and quality of the design of workflows.

Taverna provides a provenance system that enables the user to inspect both end results and intermediate results of tasks in a workflow. Although its provenance system collects and stores the provenance data at all levels the execution of an hierarchical workflow, the provenance browser is currently limited to exploring the top level workflow. Zhao et al. [242] have designed a query interface for analysing the provenance data stored by Taverna. This query language is used to address queries specified during the first Open Provenance Challenge. Many other workflow systems plan to implement a provenance query and visualisation system, among others, Altintas et al. [10] and Kwasnikowska et al.[118].

8.3 Provenance in e-BioFlow

The provenance system for e-BioFlow uses the OPM model to store and retrieve provenance data. The engine, which is discussed in the previous chapter, has been designed to give real-time feedback on the workflow’s execution state by means of an event-listener

²<http://www.cs.man.ac.uk/~sowen/data-proxy/guide.html>, last visited December 2009

interface. The provenance system is implemented as an event-listener to the engine. It translates all events in OPM compatible format and stores them into the database. How the workflow enactment is captured and translated into OPM concepts is defined in the OPM profile (Table 8.1).

This profile is designed to be applicable to other workflow systems as well. At the moment, the profile has one aspect that for some other workflow systems may be problematic. In order to keep it simple, e-BioFlow treats inputs and parameters the same. Other workflow systems, such as Taverna, distinguish between these two. Parameters are used to tune tasks to change the way input data are transformed into output data. The distinction between inputs and parameters is, however, not always clear. Most Blast services require an input sequence and a database to search in. Some Blast services treat the database value as a parameter, others as an input.

Additionally, the relation “was triggered by” is used somewhat differently than defined by Moreau et al. [139]. Following the OPM specification, this relation means an artifact is involved, but it is not clear which artifact. In the profile we propose, this relationship refers to a control flow dependency. Artifacts, therefore, do not necessarily have to be involved.

This profile is successfully employed in e-BioFlow. The provenance data, however, is stored in neither XML nor RDF, but into a PostgreSQL database to improve performance. Storing the provenance data in a database system is robust and allows optimised querying of the provenance archive. The database tables are direct translations of the OPM concepts. The artifacts table contain references to the *real* data already stored in the same database by the data manager discussed in previous chapter. Not only to OPM concepts, but also the complete workflow specification is stored, because this specification is an essential part of the provenance data.

e-BioFlow can export provenance traces into OPM XML format. At the moment, the XML file of a provenance trace contains reference ID's to the artifacts in the database to keep the XML file comprehensible. Access to the database is required to get the real values of the artifacts.

8.3.1 Provenance as cache

e-BioFlow's engine has been adapted to use the provenance system as cache. When the engine enables a task, it first checks whether caching is enabled for this task. If caching is enabled, it checks whether the provenance archive contains a previously executed instance of the task with equal inputs. If this is the case, it returns the output values retrieved from the provenance archive. Otherwise, the service is executed as usual.

A requirement for using the cache is that the tasks are deterministic. The workflow is

Table 8.1: The OPM profile used in e-BioFlow to store and read provenance data.

OPM concept	Meaning in workflow context
Process	This concept is created when a task is started. The process describes the task name, task description, start time, end time and whether the task has successfully finished.
Agent	This concept contains the information about the invoked resource: the location of the resource and eventually the authority and version.
Artifact	The artifact represents a data item that is passed between different tasks. It contains a time stamp when the data are created, and the data itself.
Used by(A, T, P)	Task T has used artifact A as input for port P.
Was generated by(A, T, P)	Task T has generated artifact A as output of port P.
Was triggered by(T1, T2)	A tasks T1 is triggered to start by another task T2. Tasks can only be triggered by other tasks due to the existence of a dependency in the control flow perspective.
Was controlled by(T, A)	A task T is executed by the actor A.
Account	An account contains all information of the processes, agents, artifacts and relations captured during the workflow execution. For composite tasks, two additional accounts are defined. The first one, called <i>generic account</i> , contains the composite task and is a black box of the process. The second, called <i>refined account</i> is the corresponding white box and represents the entire process. It contains all the concepts processes, agents and artifacts related to the subworkflow.
Refines (A1, A2)	The refinement relation defines account A1 to be a refinement of account A2.
Overlaps (A1, A2)	Account A1 represents the parent workflow of a subworkflow represented by A2. The result of this relationship is a tree of accounts, conforming the workflow hierarchy.

not able to determine this, since tasks and actors are treated as black boxes. Therefore, e-BioFlow requires the workflow designer to mark a task to be “cacheable”. The workflow designer can do this for both atomic tasks and composite tasks. A composite task can be deterministic even if it hides non-deterministic tasks.

For example, the workflow in Figure 8.2 is a subworkflow of a composite task that represents the invocation of an asynchronous Blast service. The first task submits a Blast job to the Blast service. The second task performs a poll operation to test whether the Blast service has finished the Blast job. This task is repeated until the service is finished. Then, the third task retrieves the results. Although the intermediate results of the polling task and the number of loops might differ due to server performance balancing, the black boxed composite task can be marked as cacheable, as long as the underlying Blast algorithm and database remain unchanged.

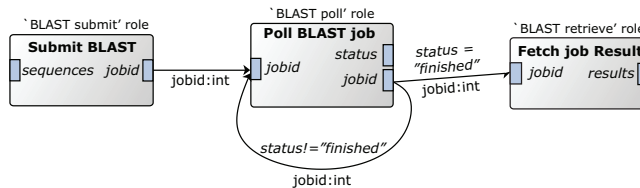


Figure 8.2: A workflow that invokes an asynchronous Blast service.

It is important to be careful in marking tasks being cacheable, because caching non-deterministic tasks might result in incorrect workflow execution. Caching the results of the polling task in the previous example can result in a livelock, for example.

8.3.2 Provenance visualisation

The provenance system comes with an interactive provenance browser (Figure 8.3). This browser allows easy navigation and exploration through the provenance traces in the provenance graph. Its user interface consists of four components:

Graph panel: This panel shows the provenance graph. The graph panel is interactive. The user can click on the elements to gain more information.

The graph panel uses the hierarchy of accounts to address levels of detail. In its initial state, it only shows the root workflow. The refined accounts are hidden until needed. The user can click on a generic account to expand it to its refinement.

The user can further expand the graph by right-clicking on the elements. A popup menu will appear that contains options to make neighbouring elements visible.

This way, the user can explore the provenance graph by showing detail-on-demand. The user can inspect interesting elements, such as an artifact or task, by clicking on them. A viewer will pop up containing the properties of the element. In case of a process, it will show process name, process description, the start and finish time, and actor related information (authority and location). In case of an artifact, the viewer will show the artifact value. It is able to show text, XML and graphical data (among others, PNG, PDF, SVG).

Tree panel: The tree panel is an alternative to explore the provenance graph. Accounts are presented as internal nodes or branches. The processes, artifacts and relations are leaves.

Selecting an account in the tree panel shows the account in the graph panel; selecting an artifact or process in the graph panel puts focus on the node in the graph panel.

Query panel: The query panel allows querying of the provenance trace. It can be used to quickly find a set of elements, such as all artifacts produced as output of port “Blast report”. The elements that match the query are shown in the graph panel.

At the moment, the query panel supports MySQL based queries on the provenance archive. This panel enables the user to find and select interesting nodes with a single query. A limitation is that it requires the user to have knowledge about the underlying database model. A more intuitive query interface is future work.

History panel: This panel shows a table containing all provenance traces stored in the provenance archive. The user can select a trace in order to visualise it in the graph panel and the tree panel. New runs automatically appear in this history panel.

When the user selects a trace in the history panel, the provenance system will load provenance data into the provenance browser. Additionally, it loads the original workflow in the editor perspectives and the engine. This way, the user can easily rerun and modify the workflow.

Figure 8.4 shows the graph panel of the refinement of the generic “Blast” account. Only the parts of the total OPM graph the user is interested are shown. This way very large traces can be navigated, such as the provenance trace that will be discussed in the use case, which contains over 200,000 elements and relations.

Chapter 6 discussed the different perspectives e-BioFlow provides on the workflow. The provenance browser provides analogous perspectives on the provenance graph (Figure 8.5). The user can easily switch between these perspectives using buttons to hide information-on-demand.

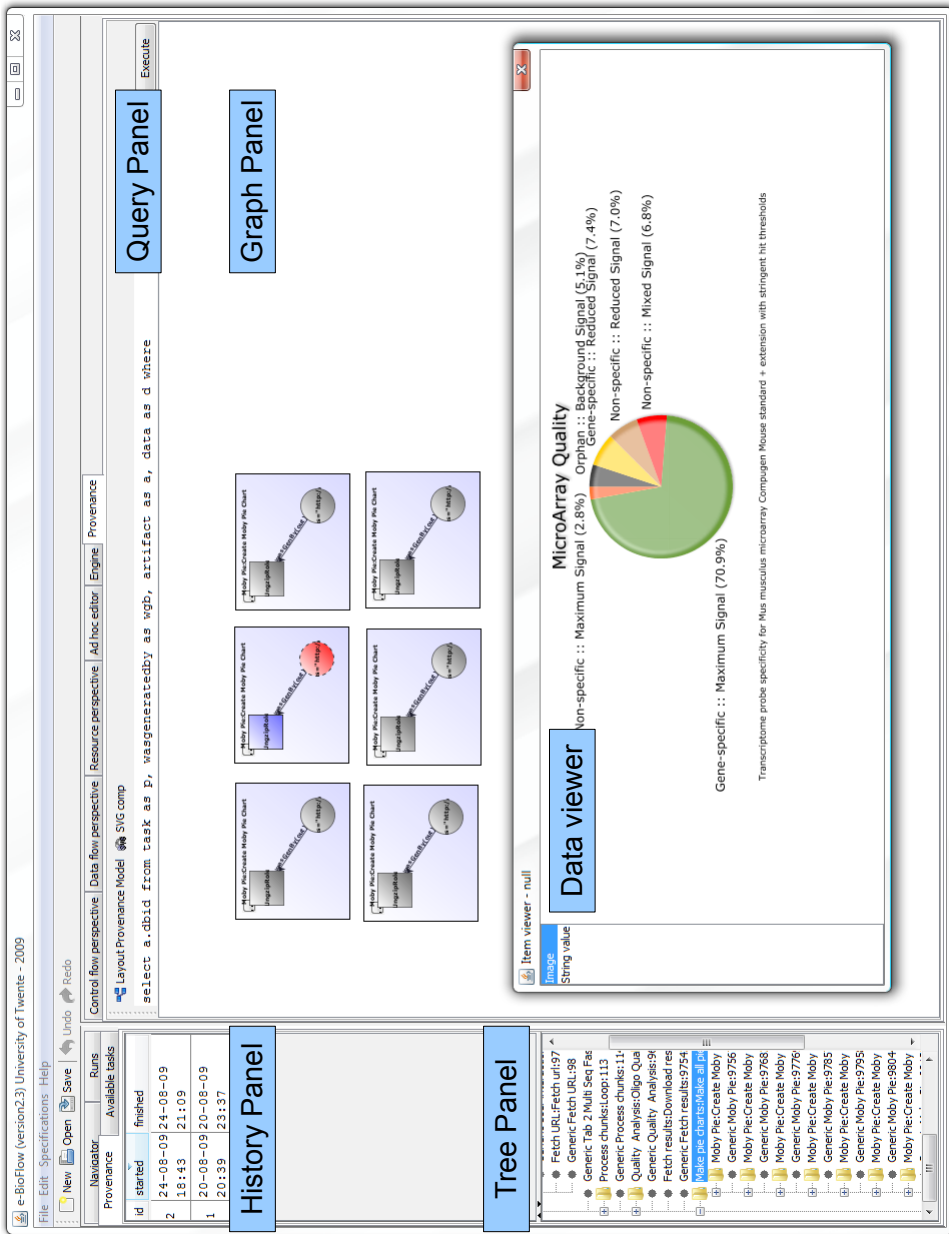
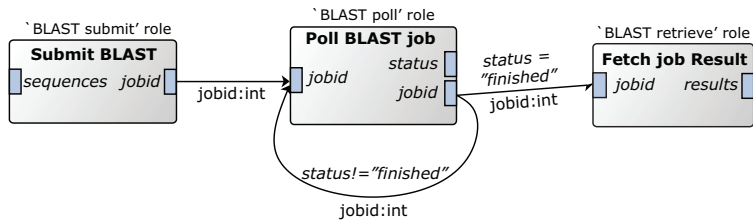
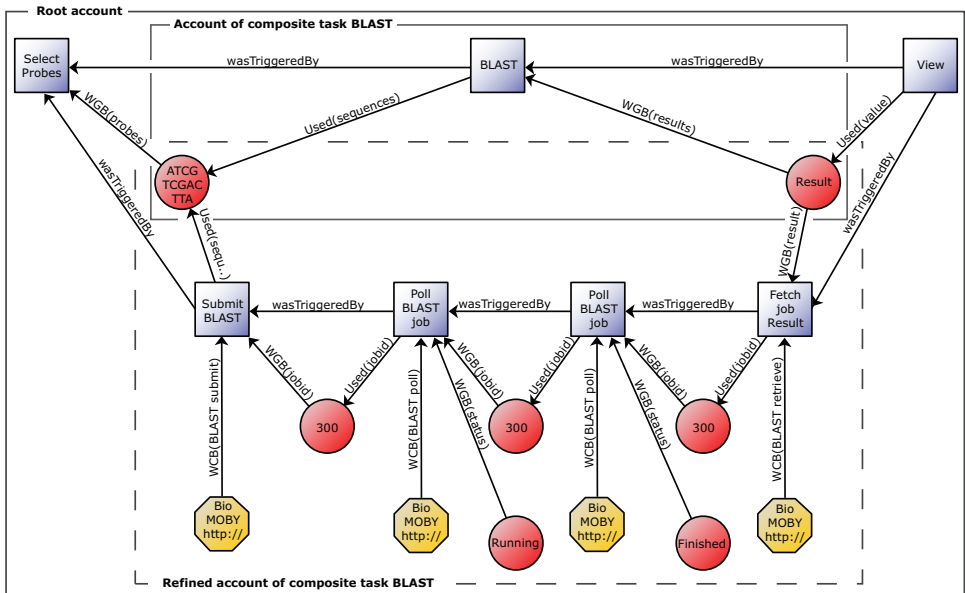


Figure 8.3: The provenance browser contains four panels: i) the graph panel, ii) the tree panel, iii) the query panel, and iv) the history panel. The graph panel shows parts of the provenance graph that match the query. The data viewer shows the contents of the selected data item in the graph (the red circle). The value of the selected data item is a pie chart.



(a)

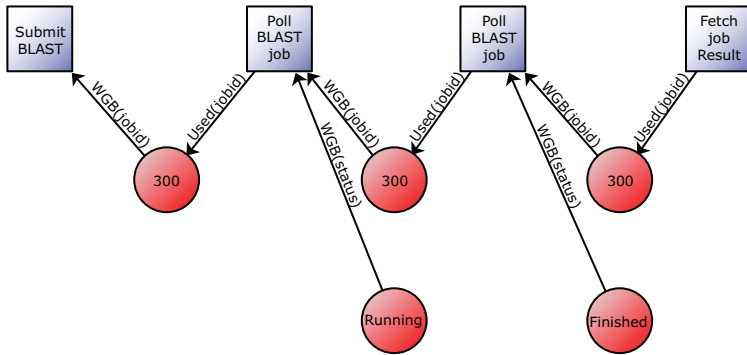


(b)

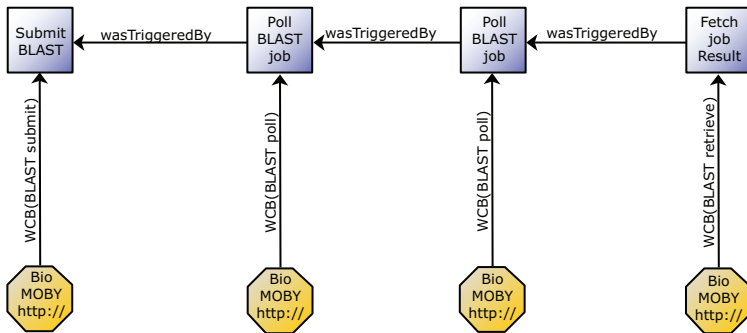
Figure 8.4: (a) The hierarchical workflow performing a Blast operation using a polling mechanism. (b) The provenance visualisation in e-BioFlow using accounts to address different levels of details.



(a) The control flow perspective shows only processes and “was triggered by” relations”.



(b) The data flow perspective shows only artifacts, processes, and the “was generated by” and “used by” relations.



(c) The resource perspective shows the processes, agents, “was triggered by” and “was controlled by” relations.

Figure 8.5: Perspectives applied to a provenance graph.

The combination of different panels, the perspectives in the graph panel and the different levels of detail help the user explore large provenance graphs. Of course, if the number of tasks in a (sub)workflow becomes large, the corresponding account will become large and difficult to explore. A good design of the workflow by means of subworkflows can limit clutter of the provenance graph. When workflow designers use hierarchy at design time to keep the specification simple and arranged, this automatically contributes to a finer level of granularity of the generated provenance data.

8.4 Use case: OligoRAP

To test our prototype, we have chosen to use the OligoRAP [143; 141]. OligoRAP, which stands for “Oligo Re-Annotation Pipeline”, is a data-intensive life science application. It automatically checks the target specificity of oligonucleotide probes and updates their annotations. A high-quality oligonucleotide probe library is an essential component of microarray-based gene expression experiments. In order to maintain this quality, probes in a library have to be updated when new sequences or annotation data are released.

OligoRAP was originally written entirely in Perl as modular system [143]. It consists of various MOBY-S services (among others, Blast, Blat and OligoQualityAnalyser services) and a client that orchestrates and invokes these services. The oligos are processed in chunks containing a maximum of 250 sequences each. The result of an OligoRAP run consists of XML files that provide detailed information per oligonucleotide and a quality assessment of the whole array. An OligoRAP run in the Perl client for a microarray of the mouse consisting of 20K+ oligos (80 chunks) takes about 6 hours (Table 8.2).

OligoRAP has been casted as an e-BioFlow workflow. It is used as a data intensive test case for both the e-BioFlow engine and provenance system. The workflow consists among others of MOBY-S tasks to invoke the MOBY-S services, composers and splitters to build and parse MOBY objects, and Perl scripting tasks to execute parts of the original client. The workflow variant is optimised using parallelism to process multiple chunks simultaneously. The whole workflow consists of 149 tasks of which 35 are composite tasks referring to one of the 15 subworkflows.

Unfortunately, the servers this workflow accesses are not able to handle the load of executing all the synchronous Blat services at once. The workflow pattern “interleaved parallel routing” [201] has been applied to the workflow to limit the number of simultaneous invocations of these web services. This pattern states that for multiple instances, only one instance can be active at one time, but the order of instances is chosen non-deterministically.

The workflow implementation has been enacted under three different circumstances: i) without caching and provenance enabled, ii) with provenance enabled, iii) with

Table 8.2: The time required to run the OligoRAP case, as a Perl implementation, and as a workflow implementation in e-BioFlow without provenance, provenance enabled and using the provenance archive as cache.

	Perl Client	e-BioFlow client	e-BioFlow, with provenance	e-BioFlow, cache enabled
Duration	5h:56	2h:50	3h:03	2h:25
Data generated	3.3Gb	2.5Gb	2.9Gb	1.3GB

provenance and cache enabled. The results of these runs are compared the original Perl client (Table 8.2). All runs were performed on a commodity PC.

The workflow implementation is about two times faster than its Perl equivalent. The speed improvement of the workflow implementation can be ascribed to the parallel processing of chunks. The size of data generated in the workflow variant of OligoRAP with provenance disabled is 0.8GB smaller than the size in the Perl client. The 2.5GB data generated by the e-BioFlow client covers only the data passed between the tasks; the Perl client, in contrast, also saves process information.

When provenance is enabled in e-BioFlow, workflow enactment is about 13 minutes slower than the run without provenance. It is slower because it performs many data base accesses to store OPM elements (Table 8.3). The total amount of data generated grows to 2.9GB, which is an increase of 0.4GB (21%). These data cover both the data passed between tasks and data stored according to the OPM profile.

In the last run, caching was enabled for the tasks that represent the slow Blat and Blast web services. This run was 25 minutes faster than the run with only provenance enabled. When caching would have been enabled for other tasks as well, the run might be even faster. Of course, the speed improvement by using the provenance archive as cache highly depends on the data in the cache. This run made optimal use of the provenance archive as cache, because the input data for the workflow is equal to the input data of the previous run. The amount of data stored in the provenance archive is also less than the run without cache, because data retrieved from the provenance archive are stored only once.

8.5 Conclusion

The OPM promises to be an interchangeable provenance storage format. The way the data is stored and can be read, however, is ambiguous due to missing standards for OPM

Table 8.3: Number of elements created during the OligoRAP run with provenance enabled.

Element	Count
Data	48,124
Task	36,704
Artifact	64,329
Agent	33,835
Was generated by	56,063
Was triggered by	53,080
Used by	50,710
Was controlled by	33,835
Account	5,739
Refines	2,869
Overlaps	5,738
Total	407,231

profiles. In this chapter, we propose an OPM profile for workflow systems. Using this profile, provenance data captured in workflow systems can easily be read and interpreted in other workflow systems. Although this profile is currently only used in e-BioFlow, it can serve as a standard for other workflow tools. Scientists will benefit from such a standard profile, because using it, they can easily share provenance data with peers.

Implementing the provenance system in e-BioFlow is really straightforward due to the event-listener model implemented in the engine. The provenance system is implemented as a plugin. Therefore, e-BioFlow can easily be configured to run workflows with provenance disabled.

Like Kepler, e-BioFlow uses its provenance archive as cache. The provenance system is used as a cache to speed up future workflow runs. This can be extremely useful during the design of the workflow or when a workflow crash occurs. A major difference between Kepler and e-BioFlow is that e-BioFlow leaves the workflow definition unchanged. The engine is extended to check whether the data is available in cache. The use case in this chapter shows the overall performance improvements by implementing software as a workflow and by using the provenance archive as cache. The use case is illustrative, but not complete: the performance increase of implementing software as a workflow of course highly depends on the implementation quality of both the workflow and its scripting equivalent. The advantage of the workflow model is the support for parallelism, modular design, better maintenance, and the automatic capturing of provenance data.

The interactive provenance browser provides an interface to explore and analyse the information about the executed process and generated data. This interface supports detail-on-demand: using the different panels, the user can search for specific elements and show them in the graph panel. The graph panel supports different levels of detail through hierarchy by means of the accounts. The query interface improves the speed to find interesting nodes in the workflow graph. At the moment it uses the MySQL language which requires the user to have knowledge of the underlying database system. An easier-to-use query language is future work.

Three of the four challenges mentioned by Davidson and Freire [50] have been addressed in this chapter. First, thanks to the definition of an OPM profile and the implementation of the OPM model, interoperability is found with other workflow systems that will use the OPM model to store provenance data. Second, the provenance browser enables its users to visualise and analyse the provenance data. By providing different panels, different levels of detail and query facilities, the provenance browser addresses detail-on-demand to prevent information overload. Third, the provenance system stores all provenance data in a database system and can export these data into an OPM XML file. However, embedding the whole data into this XML file in an efficient, feasible way still remains an open issue.

User investigation on ad-hoc workflow design

9.1 Introduction

Workflow systems are developed to help bioinformaticians deal with the complexity of designing and running in-silico experiments. Their chief appeal lies in the fact that they provide easy access to tools and services provided by different groups and using different protocols.

Building a workflow, however, is a difficult job. The bioinformatician has to choose the right services and, when services are connected, to deal with data incompatibility problems between services [49; 220]. The situation is even more complicated because in current workflow systems, the complete workflow needs to be designed in advance before it can be run. In practice, however, the complete setup of the experiment is often not known in advance [4; 69]. In such cases, the bioinformatician wants to decide on the next step of the experiment using the outcomes of steps that have been finished [102].

We propose a new design system for e-BioFlow, called an ad-hoc editor. This editor enables the bioinformatician to design and execute partial workflows. This system will better fit the explorative working approach of the bioinformatician. The outputs of the tasks in the partially designed workflow are explicitly presented in the editor. They can be inspected to decide how the workflow will be extended and can be used as input for new tasks. The important question is, of course, will such a system satisfy the bioinformatician? To answer this question, we embarked on a systematic design approach: i) we analyzed the domain problem; ii) we developed a view on a solution (ad-hoc workflow

editor); iii) we developed a first draft design; iv) we evaluated our envisioning; and subsequently (this chapter), v) we have built a full blown implementation (next chapter).

This chapter will describe the draft design of our ad-hoc workflow editor named NIWS (New Interactions in Workflow Systems), and the evaluation of our envisioning. For the evaluation, we investigated the design's relevance and usability with bioinformaticians familiar with workflow systems, by applying the teach-back technique, a hermeneutic method to provoke the users to externalize their mental models [157]. The implementation of the ad-hoc workflow editor in e-BioFlow will be discussed in the next chapter.

We will first give an overview of existing user studies of workflow systems. We will describe NIWS. Then, we will explain the teach-back technique. After that, we will describe our empirical investigation with professional participants. We will discuss our results and we will end with a reflection.

9.2 Studies on scientific workflow systems

Much research has been done on scientific workflow systems, though, only few consider the usability of these systems. There is often a big gap between the level of detail that is relevant for a life science problem and the level of detail required for the implementation of the experiment as a workflow [4; 210; 86]. Gordon et al. [79] performed a user study to test the usability of the Taverna workflow system. They found functionality problems due to the exploratory nature of life scientists. They need to interact with the workflow during the actual experiment. Direct interaction enables them to try parameter settings and to debug workflows [4].

Downey [56] performed a user study to test the usability of the Kepler workflow system. One of the main features workflow users found missing in this tool is a real-time debugger of the workflow and the possibility to inspect intermediate results to make further decisions. The workflow system should guide its users to construct the workflow. Additionally, participants requested data directly being visible in the workflow diagram.

Gibson et al. [69] provided a first implementation and evaluation of an ad-hoc workflow editor. Their system enables workflow designers to design and execute partial workflows and to reuse the intermediate results to further design the workflow. The results of the user study were promising; however, the system was not further developed.

9.3 NIWS – an adaptive workflow system

In chapters 6–8, we have discussed the editor, the engine and the provenance system of our workflow system e-BioFlow. This workflow system previously only supported the

classical approach in which a complete workflow has to be designed in advance. NIWS is the mockup implementation of the ad-hoc extension for e-BioFlow to support and stimulate explorative experiment design and execution.

Designing and running workflows in NIWS is intended to be easier than in classical workflow systems. Tasks can be executed in isolation by pressing the play button in the task box (Figure 9.1). Input ports and output ports of the tasks are present at respectively the top and the bottom of the task box. The data consumed and produced by the tasks is explicitly present in the workflow as circles. The user can inspect these data and use them as sources of inspiration for how to further design the workflow. Data can be defined to be input for new tasks. NIWS does not require the user to rerun the complete workflow, but only the inserted and modified tasks.

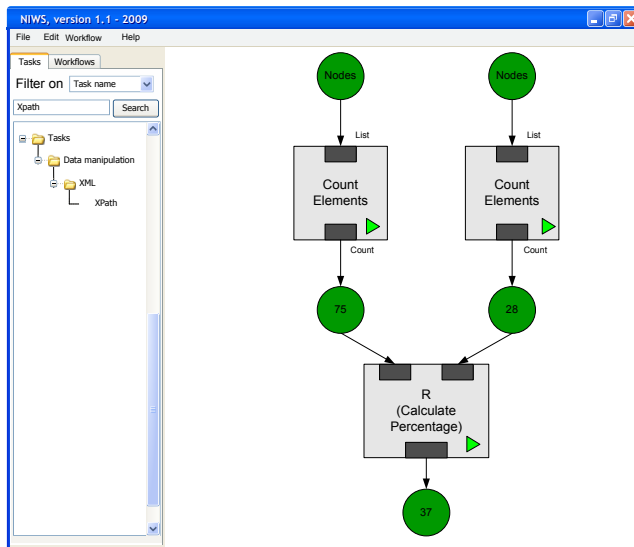


Figure 9.1: The NIWS editor presents data explicitly. At the left side, it provides a search engine for easily finding tasks.

Finding suitable tasks is difficult. NIWS has a search engine to help its users find a task based on its name, the type of operation it performs, its inputs and outputs, and the authority that hosts the service the task represents.

But NIWS is more: it supports guided analytics. Based on the type of data in the workflow, it suggests tasks that can take these data as input. This helps the user to find compatible tasks in a quick manner, but at the same time it forms a source of inspiration of possible directions in the experiment. In a similar way, NIWS can help to deliver the input

required for a certain task by suggesting tasks that can produce the right data.

To connect tasks, users often have to parse and build complex data structures. NIWS helps its users doing this for XML structures. It provides so-called composer and decomposer tasks to build and to parse XML structures. In case of a composer, the user only has to provide the attribute values of the XML; NIWS will automatically build the required XML structure using these values. In case of a decomposer, NIWS will return the attribute values of an XML structure.

9.4 Teach-back as a technique for hermeneutic analysis

People working with complex systems need a mental model of the system in order to: i) plan use, ii) actually interact with, iii) understand and assess the effect of the interaction, and iv) understand the meaning of unexpected system actions. Mental models are knowledge structures inside people's minds, based on learning the semantics of the system and its context ("what-is" knowledge), experiencing the dialogue with the system ("how-to" knowledge), and understanding the representations of the system state, system actions and system feed-back (the "vocabulary" of the interaction).

Mental models actually develop based on a current need for, for example, acting or explaining to a colleague, in a current context, with or without the system being at hand. Since mental models are "mental", we cannot directly observe or register them. Hermeneutics is a philosophical method in which an analyst develops understanding of the meaning an object (e.g., an artifact) has for a certain person or a certain group of people.

We apply the teach-back technique [157] for our hermeneutic analysis: We introduce prospective users of our design (professional bioinformaticians) to our early design ideas (use cases represented as realistic scenarios by introducing a realistic user persona, a typical context of use and a relevant task). We then ask these users to teach back their understanding of the system to an imaginary colleague. In order to teach back, we pose, both, "what-is" questions and "how-to" questions, the latter in different degrees of similarity with the use cases shown in the scenarios. In order to record the externalized mental representations, we ask our users to write down (scribble, use key words and full text at will) their teach-back.

To interpret these representations, we first develop a scoring schema and fine tune this to a level where independent analysts reach agreement to an acceptable level. We aim at a level comparable with inter-rater reliability accepted for psychological personality measurement techniques.

9.5 Assessment of a design envisioning

The mockup of NIWS is an animated slideshow presentation containing a narrative of a bioinformatician performing experiments, showing text and sketchy mockups of the system. A voice-over reads the text in the slideshow to make the presentation vivid and realistic.

The presentation contains two scenarios that show various features of the envisioned system and suggest new possibilities when using this system. The scenarios are based on real-life situations in bioinformatics, but worked out using our system ideas. The first scenario describes the design of a workflow in NIWS that can analyse the tasks another workflows consist of (Figure 9.2(a) and (b)). This analysis workflow is used among others to determine the number of scripting tasks in a workflow. The second scenario discusses the design of a workflow in NIWS to perform a Blast alignment search against the *Danio rerio* (zebrafish) genome (Figure 9.2(c) and (d)). The presentation of the scenarios takes about ten minutes.

The questions about NIWS consist of one “What is” question, probing a semantic mental model, and three “How to” questions, probing procedural mental models. In the first question, the participant is asked to explain to an imaginary colleague Tom, who is familiar with workflow systems but does not know NIWS, what NIWS is. In the three “How to” questions, the participant is asked to explain to Tom how to perform a particular task using NIWS. These tasks are not explicitly covered by the scenarios, but using NIWS could be inferred from them in relation to the individual participant’s mental model. The questions are presented in Appendix G. The questions are distributed on paper. To respond, participants can write, scribble, make drawings, etc. The participants get five minutes to answer each question. They are, however, not allowed to discuss or to ask questions, since we are interested in what they believe the system can do. We do explicitly mention that the questions are not to test the participants’ knowledge: there are neither right nor wrong answers. Participation is anonymous and voluntary. All participants are rewarded for participation with a 1 GB USB key.

9.5.1 Participants

In total, there were 50 respondents, originating from different countries, though most of them were Dutch. The participants had different backgrounds (biology, bioinformatics, chemistry, computer science) and their expertise in using workflows in life science experiments differed from beginner to experienced user. These respondents were recruited during six sessions: during visits at life science research groups, courses in the Taverna

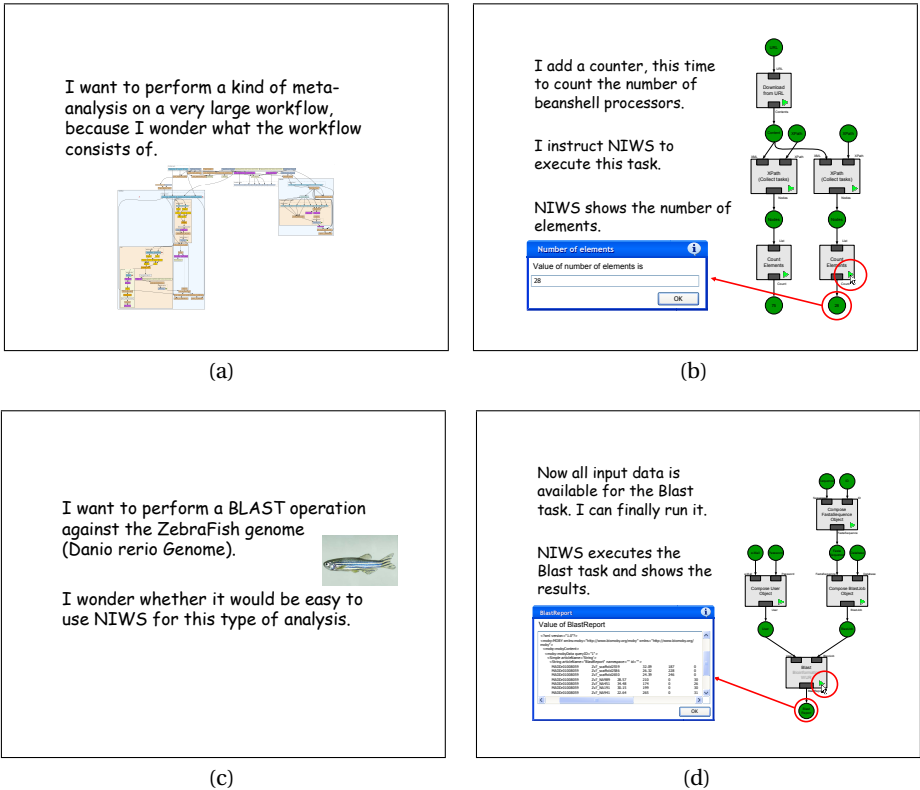


Figure 9.2: Screenshots of the scenarios presented to the participants. In the first scenario, someone wants to analyse a workflow stored at myExperiment using NIWS (a). The partial workflow designed for the analysis is shown in (b). In the second scenario, someone wants to perform a Blast operation against the Zebrafish genome assembly (c). The partial workflow is presented in (d).

workflow system and a meeting of the BioAssist Group ¹. The size of the groups ranged from 1 to 20 persons.

A strict protocol was handled in these sessions in order to keep the experiment reproducible. In each session, an experimenter was present to start the scenarios, to distribute and collect the protocols, and to manage the time. No information about NIWS was given to the participants other than the scenarios. The participants received the reward when they handed in the form.

9.5.2 Scoring the protocols

The result of teach-back consisted of both creative and corrective feedback. Creative feedback encompassed new features the participants expected to exist based on the scenario. In corrective feedback, the participants mentioned features they did not like, expected not to work, or wanted to be improved.

The feedback was analyzed regarding three levels of the system: i) feedback related to the functionality: what the participants believed the system could do and what its limitations would be, ii) feedback related to the dialogue, and iii) feedback related to the representation of the workflow experiment and the system interface.

A scoring scheme was set up to analyze the forms in an unambiguous and reproducible way. This scheme consisted of rules and examples how to categorize the feedback. To set up this scheme, two analysts separately analysed five forms. They discussed their findings with a third analyst and built the scoring scheme. The two analysts separately scored another three protocols and compared their scorings to test the agreement on the scoring scheme, which confirmed interpretation and scoring reliability. Consequently a single analyst was sufficient to score the remaining 42 protocols.

9.6 Results

The results were grouped along the scoring levels. In subsection 9.6.1 we showed the results (illustrated by examples from the protocols) for functionality:

Corrective feedback: Functionality (“what-is” knowledge) as indicated in the scenarios that we found back in the protocols and that is consistent with the scenarios, as well as functionality understood by the participants that is inconsistent with our scenarios, and indications of functionality aspects not appreciated by the participants.

Creative feedback: We will show examples of functionality found in the protocols not mentioned in the scenarios, that makes sense as extensions of the design.

¹<http://www.nbic.nl/support/bioassist>, last visited: December 2009

In subsection 9.6.2 we will report on corrective feedback and creative feedback regarding the dialogue (“how-to” knowledge) of NIWS, and in 9.6.3 we will do the same for feedback regarding the representations. On this last aspect we need to keep in mind that the scenarios as presented by us are describing our NIWS design ideas at a global level, focusing on the functionality, and hinting the dialogue, but being vague on the actual representation of the system interface and on the users’ actions.

9.6.1 Functionality

Most respondents reacted positively on the system presented. Many of them mentioned that NIWS was like other workflow systems, but then more intuitive, simpler or easier to use. As one said, “a big plus is that you can add additional processing anywhere in the chain, without having to re-run everything as it caches intermediate results”. Another respondent mentioned “You don’t have to rerun the workflow every time. Therefore you will save a lot of time”. It was also easier to use for beginners: “NIWS is this new workflow system that has this cool feature of giving you hints when you don’t know what to do. Ideal for beginners like me ;-).” However, one respondent said the questions were easier to solve without using a workflow system.

Many respondents picked up the idea of designing workflows step by step. Intermediate results could be used to further design the workflow. “The nice thing is that one can execute every process in isolation and that one can inspect the outputs of the workflows at any moment.” NIWS enabled one to design workflows iteratively: design the partial workflow, test and debug it and then extend this workflow (Figure 9.3). One respondent described this as “kneading” the workflow.

Eight respondents proposed a two step approach to design and run workflows in case large data sets are analyzed. First, design a workflow using a small example data set. Second, when the design is finished, run the workflow for the entire data set. Another respondent suggested to create a workflow for one data item, and to embed this one into a larger workflow that runs it for each data item of the complete set in parallel (Figure 9.4).

The respondents expected NIWS to be able to save and load workflows, also those designed by others. These workflows could be reused and adapted for different cases. Six respondents expected that the workflow presented in scenario 1, to analyse other workflows, was a built-in feature of NIWS instead of just an example workflow designed in NIWS.

To find services, 27 respondents recommended using the search facility of NIWS, though some of them found the use of this facility to be unclear. One respondent expected the search function to be smart: meta-data can be used to further refine the search. For example, the database name could be used to find Blast services that had access to

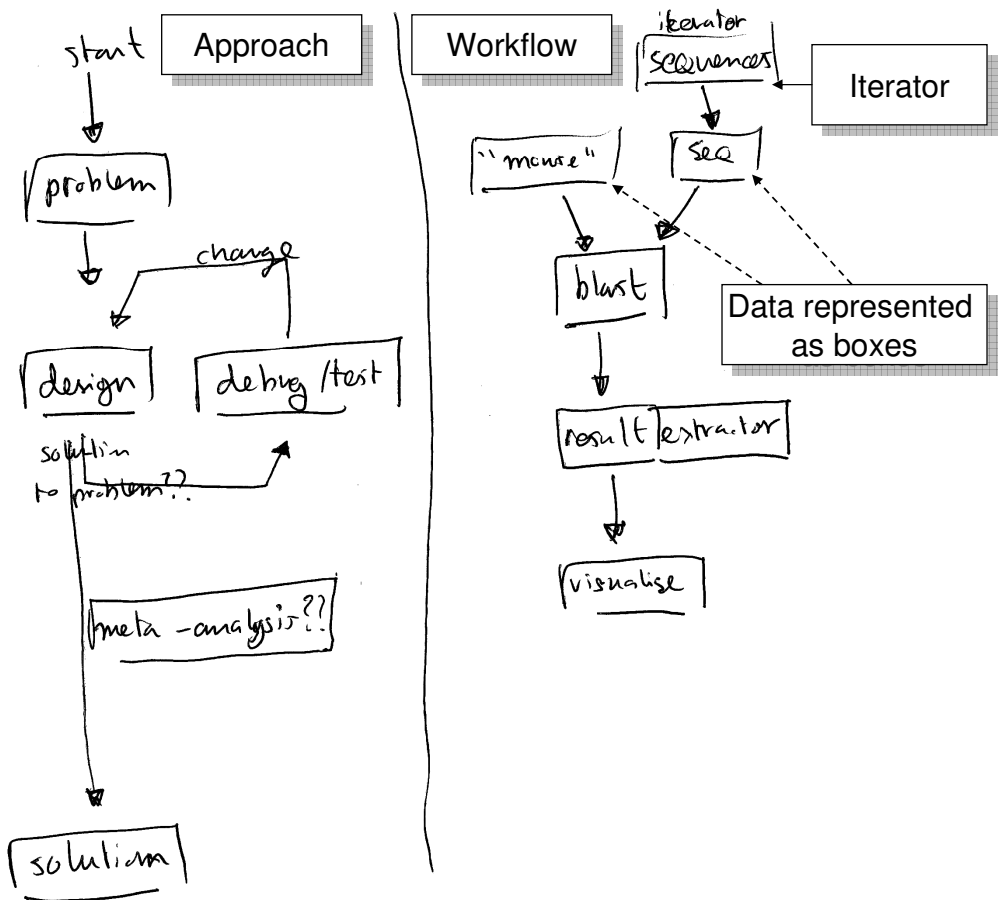
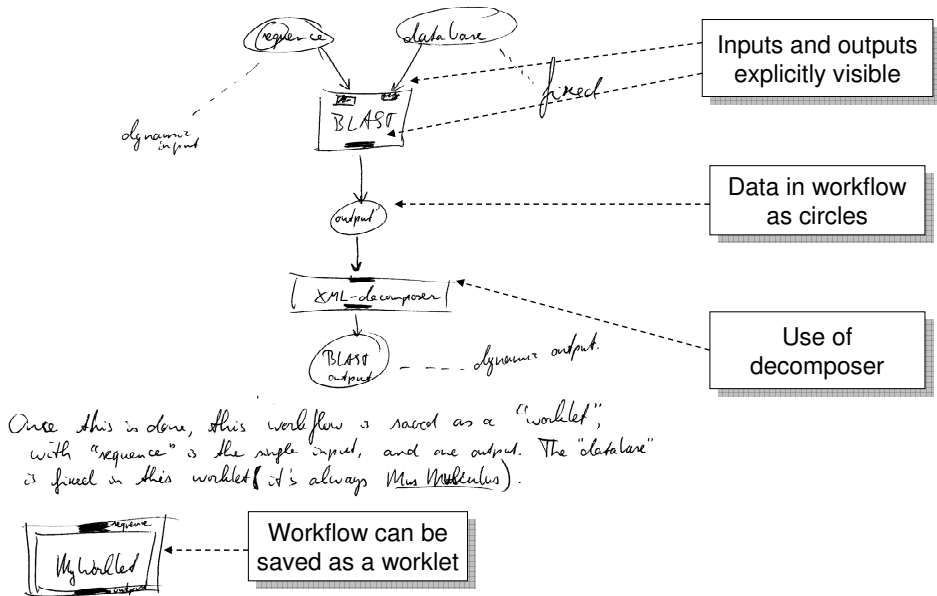
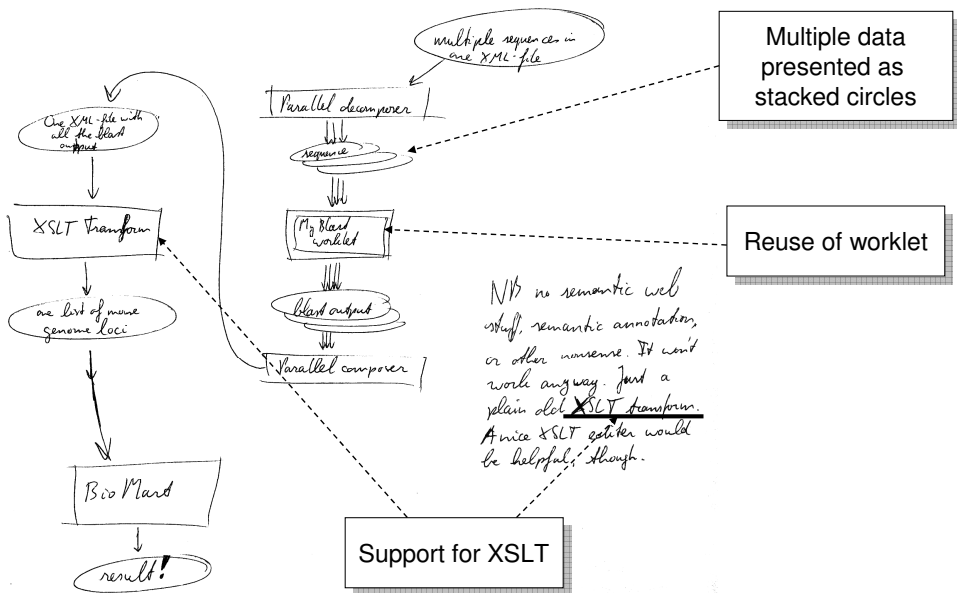


Figure 9.3: This respondent has drawn the iterative approach at the left side. The workflow is drawn at the right side.



(a)



(b)

Figure 9.4: (a) This respondent has drawn a worklet. (b) The worklet is embedded in a larger workflow.

that database. Others recommended using external resources, such as Google or colleagues, to find services. NIWS was expected to provide access to many different types of web services, among others, BioMOBY, REST, XML-RPC and SOAP/WSDL.

The feature of NIWS to suggest services that can take data available in the workflow as input was picked up by 11 respondents. Three of them explicitly mentioned that they expected the suggested services to be compatible with the data in its current format; so no data transformation should be needed.

NIWS's functionality to automatically compose and decompose XML data was found useful by many respondents. Sixteen respondents even expected these facilities to solve all data format problems, and data transformation to be a built-in feature of NIWS (Figure 9.5). Others, however, were skeptic about the automatic data transformation facilities: "If this went well, e.g. if you would never experience data compatibility issues, is questionable, because the output of one service needs to know what kind of format is expected as input of the other service". Some respondents expected support for scripting facilities, including query languages, to perform data transformations (Figure 9.6). These scripting facilities could also be used to affect the control flow of the workflow. Others recommended searching for external data transformation services that would hopefully transform the data into the right format.

9.6.2 Dialogue

The scenarios showed the drag and drop facilities of NIWS. Two respondents expected copy and paste functionality to be available for easily reusing parts of workflows. A few respondents expected the option to embed workflows previously designed or designed by others in larger workflows.

Many respondents had picked up the feature to use the play button in the task box to run a task in isolation. From the scenarios, it was not clear whether tasks upstream in the workflow would be executed automatically. One respondent supposed this to be the case. NIWS would ask its user to enter missing data. In case only a fixed set of options was valid as input, NIWS would list them to let users choose from them. One respondent mentioned it would be nice if users could also choose from data already in the workflow.

Many respondents perceived that composer tasks and decomposer tasks could be added by right-clicking on respectively the input and output ports of a task to feed correct input or parse output. One respondent described this as some magic: "The Blast service needs some magic before we can use it, so we must tell [NIWS] to do its magic. The result is two boxes which we can give our [user] name and sequences". He referred to the composition tasks to deliver the correct XML input. Some other respondents expected right-clicking on ports was used to set default input values. Two respondents expected NIWS to

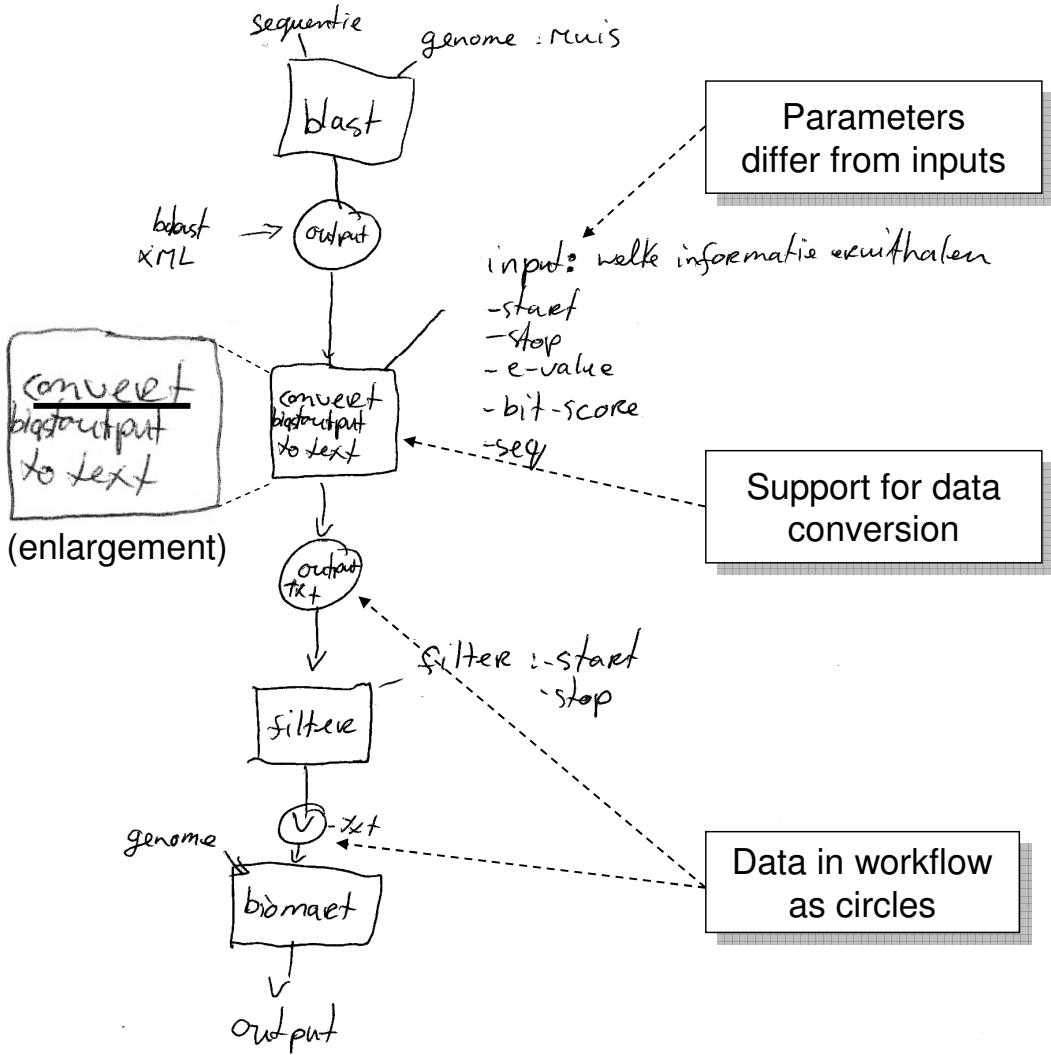


Figure 9.5: This respondent expects data transformations as builtin facilities of NIWS.

First load the simple BLAST search we made earlier.
 Add an iteration, so we can loop over a set of sequences.
 Let NIWS help you to select the file name or files.

so:

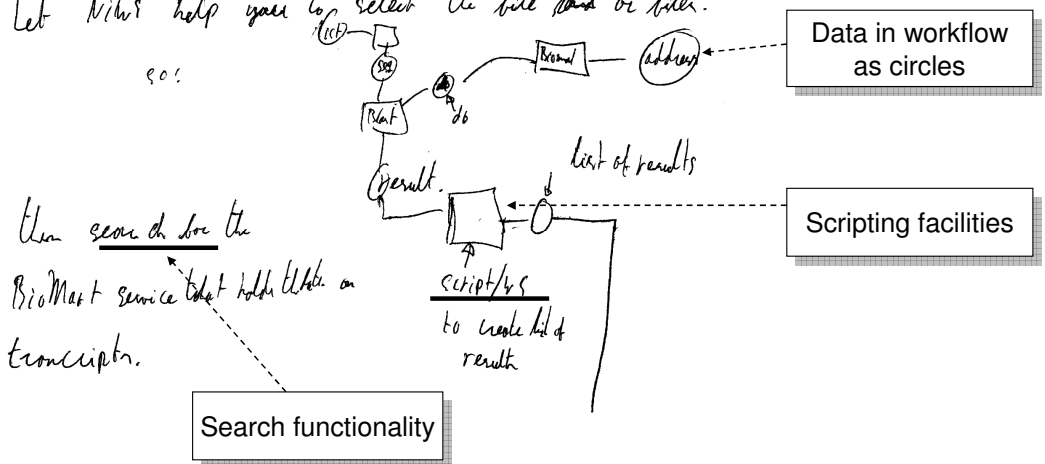


Figure 9.6: This respondent uses the search facilities of NIWS and expects scripting facilities to be available.

add composer tasks and decomposer tasks automatically. Two other respondents declared that building and parsing hierarchical XML structures could be established by a chain of composition or decomposition tasks.

NIWS was expected to warn about the existence of data compatibility problems. “The system will give an error if the outputs don’t match the inputs. When that is the case (and yes, this will happen) write a small converter script to process the output.” To create these scripts, a good editor with auto-completion and syntax highlighting would be desirable. Four respondents expected tasks could be configured to return a specific output format and to solve data transformation this way.

One respondent recommended limiting the amount of mouse interaction required: “It looks like a lot of clicking is required, for every composer and decomposer and to execute a task, you have to click.”

9.6.3 Representation

In total, 28 respondents used drawings to explain Tom how to use NIWS. In many drawings, the tasks boxes had inputs and outputs explicitly visible at top and bottom. In the scenarios, data were presented as circles, which made the workflow graphs look like coloured Petri nets [101]. Data were explicitly present in drawings of 19 respondents. These data were connected by arrows coming from output ports and arrows going to input ports. Some respondents drew data using boxes instead of circles. So, the difference between these two symbols seemed to be unclear. One respondent used stacked circles to represent collections of data in case a task returns multiple data items (Figure 9.3).

9.7 Conclusion

This chapter describes the design and evaluation of our early design ideas of an ad-hoc workflow editor. Using a scenario-based mockup implementation called NIWS, we were able to design and evaluate this interface in an easy and fast way in an early stage of the design.

The mockup implementation was presented to life scientists from different backgrounds with different levels of expertise with workflow systems. These scientists were asked to give creative and corrective feedback by applying the teach-back technique. We found that NIWS is a significant improvement over traditional workflow interfaces. Many respondents put high value on the ability to inspect intermediate results to further design the workflow. Helping with data transformations and finding and suggesting services are other features these respondents put high value on.

Besides positive feedback, respondents gave feedback about desired functionality of a workflow system, even of aspects not shown in the scenarios. Furthermore, the respondents gave directions to improve the interaction with the system presented and with other workflow systems.

The results of this study were used to develop an interactive implementation of NIWS in our workflow system e-BioFlow. This implementation will be discussed in the next chapter.

Chapter 10

Designing workflows on the fly using e-BioFlow

10.1 Introduction

Bioinformaticians are used to work in an explorative research style, without having a clear hypothesis [108; 133; 171; 17]. Few steps are known in advance and data are used as sources of inspiration. A workflow system will better fit the bioinformatician's needs if it supports this explorative research style [69]. Current workflow systems separate the design and execution of the workflow, which has led to a trial-and-error approach in using them or not using them at all.

The previous chapter discussed the design and evaluation of a mockup system for an ad-hoc workflow editor. The participants of the evaluation study expected this new workflow editor to be easier in use than traditional workflow editors. Based on the results of the evaluation, we have extended our workflow system, e-BioFlow, to an ad-hoc workflow editor. An ad-hoc workflow editor enables an ad-hoc workflow design, with a small or no predetermined plan of the final workflow [212]. e-BioFlow presents new interactions with workflow systems and supports the explorative research style bioinformaticians prefer. The workflow designer can execute partial workflows using this new editor. 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 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 workflow designers 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. This problem is known as the *plan composition problems* [83]. The real services to be used may be unknown. Furthermore, linking services often requires data transformation [220]. The ad-hoc workflow editor will help the workflow designer to build the workflow. He can run parts of the workflow and inspect intermediate results to test and debug the workflow, and to fine-tune parameter settings. The result of using the ad-hoc workflow editor is a runnable workflow that can be stored as a generic workflow model 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 in the workflow model. Therefore, the workflow can be used as template for future experiments and shared with peers through web portals such as myExperiment [72; 76].

In this chapter, 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 this new editor. Then, we will compare our approach to other systems that support ad-hoc workflow design. We will end with a discussion.

10.2 The characteristics of an ad-hoc workflow editor

Although bioinformaticians 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 are 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 [69]. Like in other visual programming environments, the workflow designer has to make many design choices without good data to direct his decisions [228]. This forces workflow designers to guess-ahead or to insert place-holders [83].

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. [69] call an *investigative data-oriented mode*.

Ad-hoc workflow editors 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 workflow editor.

- There is no need to know the complete workflow in advance. One can extend and execute partial workflows.
- It speeds up workflow design, because a small change in the workflow requires a rerun of just the tasks involved.
- Intermediate results can be used as sources of inspiration to decide on next steps of the workflow.
- No guess-ahead is required about the data produced or consumed by the tasks.
- One can fine-tune parameters, test 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 [17; 47], which is also true for most bioinformaticians [117]. It is important that the visual language used matches the user's mental representation of the problem he wants to inspect [83; 228; 47]. The closer the programming world is to the problem world, the easier problem solving ought to be [83]. 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 editor 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. It enables the designer to use these data to further design the workflow [133; 56]. The resulting environment supports what is called *live editing* [97]. A live editing environment supports explorative programming and gives programmers real-time feedback on the program's execution at edit time. Hundhausen et al. [97] applied live editing to textual programming languages to help novice programmers implement their programs. The ad-hoc workflow editor is a live editing environment, but then to design workflows [56]. It enables a 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 editor to suggest new steps for the workflow design. The workflow editor can have a wizard-like functionality to help the

workflow designer extend the workflow [56]. This is what we call *guided workflow design*. The ad-hoc workflow editor should support forward guiding, to propose tasks that can use the data produced as input [80], 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 [162].

Additionally, the workflow system can guide data transformation steps. Data incompatibility forms a big problem in service composition [239; 142; 23; 106; 174]. In Chapter 4, we have shown that at least 30% of the tasks in a typical bioinformatics workflow are devoted to data transformations. The workflow system can propose tasks that perform the data transformations required.

Some workflows are used only once, others repeatedly [76]. An ad-hoc workflow editor should support both workflows for one-time use and workflows intended for multiple-time use. The ad-hoc workflow editor is a programming by demonstration environment. Programming by demonstration means that the user shows what needs to be done, and the environment records these actions, generalises over them and translates them into a script [122]. 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 editor 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, bioinformaticians go through a fast cycle of hypothesis generation, experimentation, evaluation of the results and method selection [174]. After this phase, rationalisation is performed, in which they validate the results and formalise the process [69]. The ad-hoc workflow editor 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.

10.3 Different perspectives in e-BioFlow

Though e-BioFlow is already discussed in the previous chapters, we will give a short overview of the components it currently consists of. The ad-hoc workflow editor shares many characteristics with the editor perspectives, the engine and the provenance system. e-BioFlow has a tabbed user interface to design and execute workflows and to analyse exe-

cuted workflows. A tab is called a perspective and is used to design or run workflows or to analyse the results of a run. 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 enactment 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 [200], but supports late binding and passing data by reference.

Provenance system automatically captures all process and data related information of workflow runs. It stores these data in Open Provenance Model [139; 138] compatible format. It contains a provenance browser, a graph visualisation to explore the provenance data.

Ad-hoc workflow 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 10.1). 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.

The first three perspectives 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 [173].

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

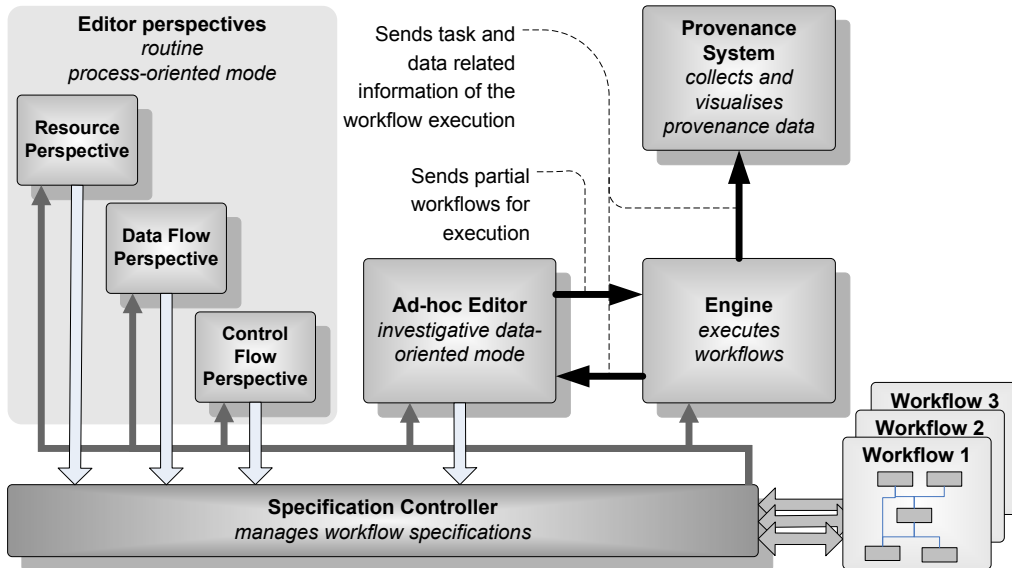


Figure 10.1: All perspectives are registered to the specification controller to send and receive changes in the workflow model. The ad-hoc workflow editor is a perspective that interacts with the engine.

compatible services. At the moment, e-BioFlow supports four types of actors: i) actors that can invoke SOAP/WSDL or BioMOBY web services, ii) actors that can execute scripts written in, among others, Java¹, Perl² and R [98], iii) actors that can interact with the user, and iv) actors that can compose and decompose XML data structures.

The provenance system communicates with the engine. It receives all information related to the workflow execution, and stores this information in a database. It provides a provenance browser and a query interface to interactively explore and query these data.

10.4 Ad-hoc workflow design in e-BioFlow

The ad-hoc workflow editor uses the workflow models shared by the other perspectives. Like the other perspectives, it 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, gui-

¹<http://www.java.sun.com>, last visited: December 2009

²<http://www.perl.org>, last visited: December 2009

ded workflow design and workflow by example will be used to explain the ad-hoc workflow editor in more detail.

10.4.1 Live editing

At first sight, the ad-hoc workflow 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. But, using the ad-hoc workflow editor, the workflow designer can select one or more tasks and instruct the editor to execute these tasks. When the workflow designer instructs the editor to do this, the ad-hoc workflow 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, invisible to the user. 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 selected tasks.

The ad-hoc workflow 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 10.2). Each circle contains the name of the output port by which it is produced and a short representation of the data it holds. The ad-hoc workflow editor uses arrows from the output ports to the corresponding data items to present the *generated* relations. The workflow designer can inspect the data by selecting the circles. The editor will show a dialog containing the data. At the moment, it 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 workflow editor automatically adds a pipe between the output port of the task that has generated the data item and the input port that uses the data item as input (Figure 10.3). When the user instructs the ad-hoc workflow 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 one 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 workflow editor automatically creates a used-by relation between the data item and the output port (Figure 10.4).

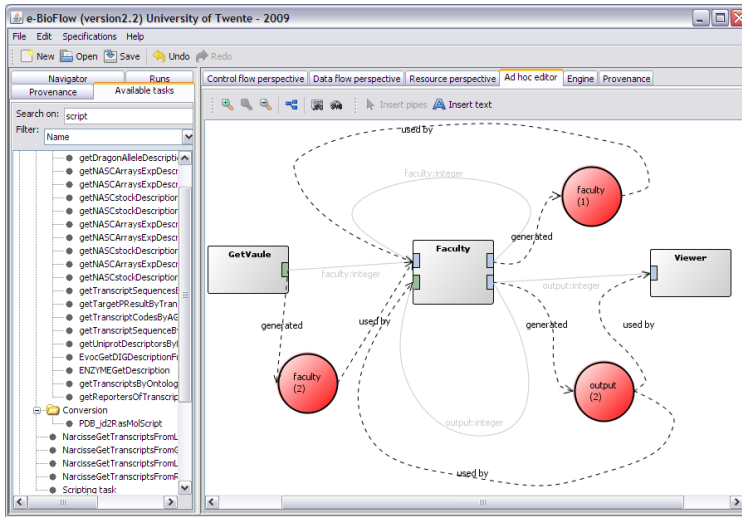


Figure 10.2: A screenshot of the ad-hoc workflow editor. Tasks are presented as boxes; data as circles connected to the output ports that have produced them and to the input ports that use them.

10.4.2 Guided workflow design

The ad-hoc workflow editor helps the workflow designer 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 workflow 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 editor adds a task definition into the workflow for that actor and sets this actor to be the preferred actor to execute the task. 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 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 as input. Data items can be selected to find and add tasks and actors that accept these data as input.

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 workflow editor helps the workflow designer to build and parse these

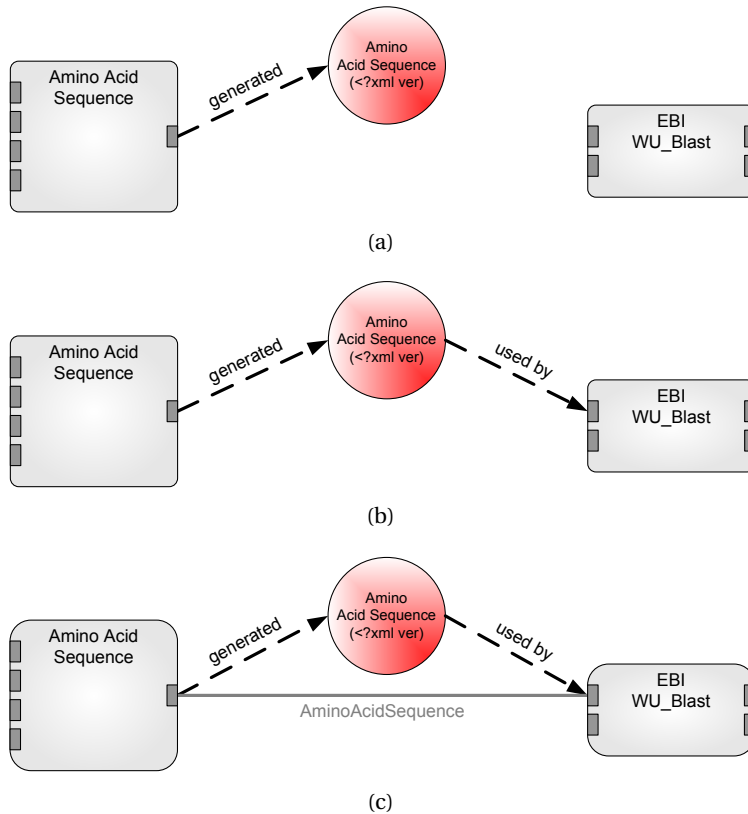


Figure 10.3: (a) The “Amino Acid Sequence” task has generated a data item as output. (b) This data item is defined as input for the “EBI WU_Blast” task. (c) The ad-hoc workflow editor automatically generates a pipe between the two tasks.

XML data structures by means of 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 incompatibility problems, they help the workflow designer to create XML structures and to reuse the contents of XML structures without programming.

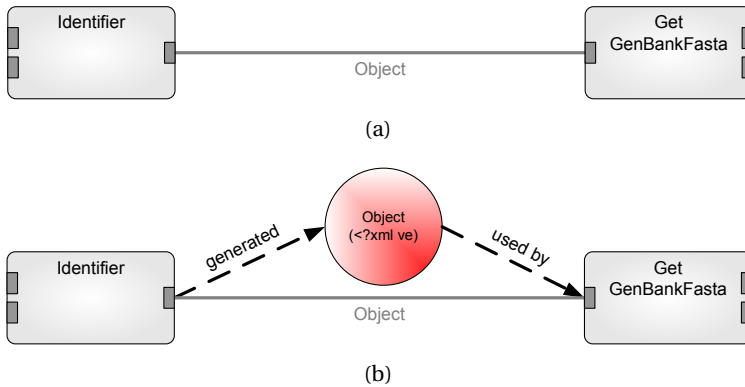


Figure 10.4: (a) The output of the composer task “Object” is connected to the input of the task “Get GenBankFasta”. (b) When the first task is executed, the ad-hoc workflow editor automatically creates a data link between the data item produced and the next task.

10.4.3 Programming by demonstration

Workflows designed in the ad-hoc workflow editor can be edited directly using the other perspectives and vice versa. For example, when a new task is inserted in this 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 workflow 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. Additionally, the workflow can be saved as a template for future experiments.

10.5 Use case: perform a Blat operation

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

Maria searches for a Blat service [110], because it is a fast alternative for Blast. Soon, she finds the Blat service provided by Wageningen University, the Netherlands. This one

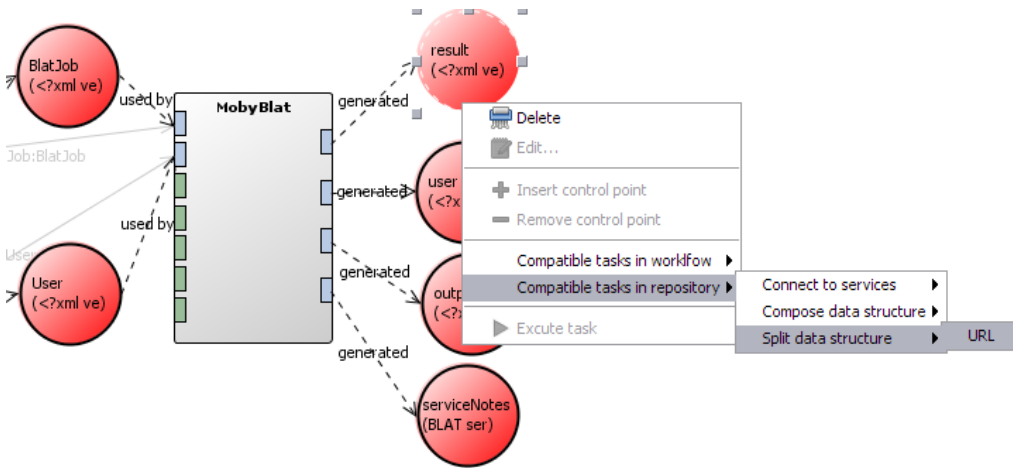
provides fast access to the Ensembl [66] zebrafish assembly. She drags the Blat service to the workflow panel. The service requires two inputs, both MOBY-S objects. The first, named “User”, is required for session information; the second, called “BlatJob”, to provide the sequence and the database name. Maria does not know the XML structures required, and even does not want to. Luckily, the ad-hoc workflow editor can help her to construct these complex data structures. Maria instructs the editor to add composer tasks for the “User” input by right clicking on this port. The editor shows compatible services and a composer task. Maria chooses the composer task and instructs the ad-hoc workflow editor to execute it. The editor asks her to enter the e-mail address and password to construct the complex data structure. The service description tells Maria that any e-mail address and password will suffice. The 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 workflow editor enables Maria to further compose these inputs. Maria instructs the editor to run these three composer tasks at once. The editor asks her to enter the database to be used and the sequence. The results are visualised as red circles connected to the output ports.

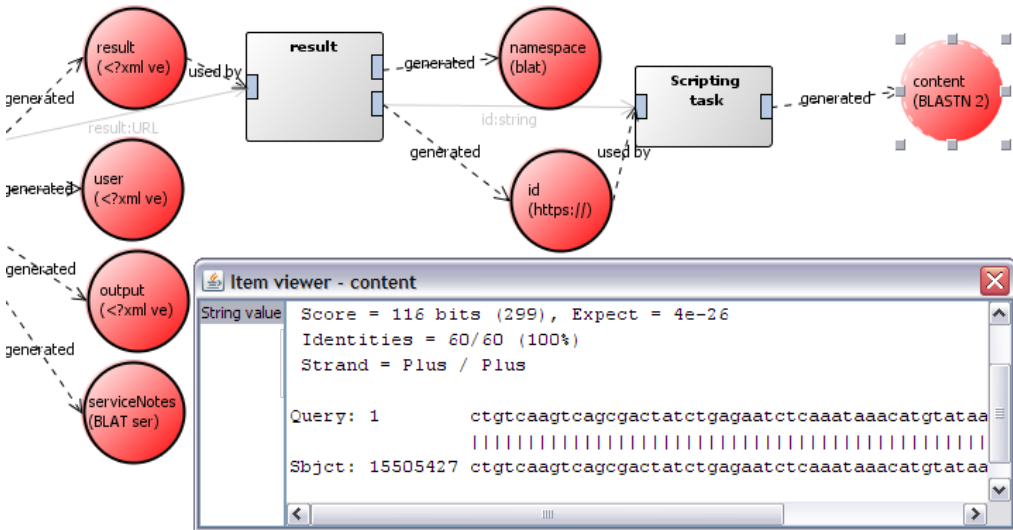
Now all the inputs required by the Blat task are available, Maria instructs the ad-hoc workflow editor to execute the Blat task. The Blat service returns four outputs, namely a URL to the Blat report, a copy of the two inputs and the MOBY-S service notes. It seems that Maria has to download the Blat results using this URL, however, it is in MOBY-S XML. By right-clicking on the URL data item, Maria selects the decomposer task to parse this MOBY-S object (Figure 10.5(a)). The data item is connected to the input of the decomposer automatically. Maria instructs the ad-hoc workflow 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. Maria knows how to do this using Perl. She searches for a Perl task in the task panel and finds a “scripting task”. Maria drags this task into the workflow panel. The scripting task has no inputs or outputs by default. When Maria 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 Maria to select the language she wants to use. From the available languages, she chooses “Perl”. Maria 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 Maria has finished writing the script, she instructs the ad-hoc workflow edi-



(a) Maria searches for a decomposer task to decompose the result data structure containing the URL.



(b) The result of the scripting task is a Blast report.

Figure 10.5: Screenshots of the design of the use case workflow in the ad-hoc workflow editor.

tor to run the scripting task. The editor shows an error message and complains about an unknown function. Maria reopens the configuration dialog of the scripting task and discovers she has forgotten to include a package. She 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. Maria, however, wants the output in Blast format in order to inspect the alignment. She configures the Blat operation to generate the report in Blast format. She instructs e-BioFlow to rerun the Blat task, the decomposer and the download task. This time, the workflow generates the correct output (Figure 10.5(b)).

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

10.6 Related work

Schonenberg et al. [167] distinguish four types of flexibility. i) Flexibility by design: the ability to model alternative paths in a process definition at design time. ii) Flexibility by deviation: the ability to handle occasional situations for a process instance to deviate from the process definition without altering its process definition. iii) Flexibility by underspecification: the ability to run incomplete process definitions through late modeling and/or late binding. iv) Flexibility by change: the ability to modify a process definition at runtime such that one or all currently existing process instances are migrated to the new process definition.

Most business workflow systems and life science workflow systems support the first type of flexibility through conditional branching. Some business workflow systems support dynamic adaptation of a single process instance (flexibility by deviation) and dynamic change of a process definition (flexibility by change). ADEPT2 [159] is one of them. ADEPT2, however, does not support flexibility by underspecification. Flexibility by underspecification plays a less important role here, because data is scarce in business workflows and therefore cannot be used to decide about the continuation of the workflow. InConcert supports flexibility by change and also flexibility by underspecification, but does not support flexibility by design [204].

Flexibility by underspecification is an essential property of a scientific workflow system. Although InConcert supports flexibility by underspecification, it does not explicitly present data in the workflow graph. As the investigation reported in the previous chapter has shown, that scientists put high value on being able to inspect the data. e-BioFlow supports flexibility by underspecification. Its engine can perform late binding, and the ad-hoc editor can run incomplete workflow models and data is part of the incomplete workflow graph. As shown in Chapter 6, e-BioFlow also supports flexibility by design through split and join types.

A few other systems support ad-hoc workflow design. None of them fulfill all the features mentioned above. We will mention four different systems, all choosing a different approach in supporting ad-hoc workflow design.

Workflow by Example (WbE) [196] records the operations the user performs on a database and translates 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 [80] provides an interface in which the user can explore data and can request for services that accept the data as input. Seahawk is not a workflow system itself, but can record the complete exploration history and export this to a Taverna workflow.

Data playground [69] is a Taverna extension that enables 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.

KNIME [26] is a data exploration and mining system. It provides access to many data analysis tools. These tools are presented as nodes of workflows. KNIME enables its users to execute the tools in isolation and to explore the outputs. These outputs are, however, not explicitly present in the graph. Additionally, KNIME does not support web services.

e-BioFlow supports the workflow by example, the explorative research 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.

10.7 Conclusion

In the previous chapter, we have presented a mockup implementation of an ad-hoc workflow editor, called NIWS, to life scientists. They have found this editor to be a real improvement over traditional workflow editors. The ad-hoc workflow editor of e-BioFlow is the realisation of NIWS. It 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 workflow 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 on the state of the workflow and by 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 workflows.

The ad-hoc workflow editor supports all actors provided by e-BioFlow and 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 others through web portals such as myExperiment.

The use case in this chapter demonstrates a small but realistic scenario of using the ad-hoc workflow editor to design a workflow. Many features the ad-hoc workflow 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.

Part IV

Conclusions

Conclusions

The research reported in this thesis was realised while closely working with life scientists in the field. In the first chapter, we have introduced the main research question: How can life science workflow systems help life scientist to design, run and capture their experiments? In this last chapter, we first summarise the previous chapters of this thesis. In the second section, we will discuss our results and will give an answer to the main research question. We will close this chapter with future work.

11.1 Summary

Gaining insight in the daily working practices of the potential users, is an essential part of a software design process. Therefore, the Chapters 2–5 are intended to establish functional requirements of life science tools, and in particular life science workflow systems.

Who are the intended users of life science workflow systems?

This thesis starts with an analysis of the daily working practices of life scientists in Chapter 2. Based on the analysis, we can distinguish novice from expert users of bioinformatics tools. Most bioinformatics tools are not suitable for novice users. The biologist, for example, is an expert in biology, but is not quite skilled in bioinformatics tools. The bioinformatician, on the other side, is a scientist who has a background in biology, but at the same time knows how to apply computer science technologies to use these tools. The bioinformatician has experience in using console applications and scripting. He can connect different tools and perform data transformations when needed.

Despite their knowledge of computer science, even bioinformaticians experience difficulties using and connecting the multitude of life science tools available today. They need to find and to have access to various tools, available in different forms, such as web pages, command line tools and web services (using different invocation mechanisms). These tools are often difficult to use, because documentation about these tools and the parameters to fine-tune the algorithm implemented are often missing.

The bioinformaticians often create scripts to connect these tools. This is, however, not straightforward, because they have to deal with different data formats different tools use, especially when they are provided by different organisations. This has resulted in a situation where transforming between data formats takes a large part of the daily working activities. As shown in Chapter 3, different organisations do not standardise on data formats and the interfaces of tools. So, standardising on data formats is not an option. Bioinformatics tools can and should help bioinformaticians doing these transformations.

When the bioinformatician runs the experiment, he has to store a complete trace of the experiment, called provenance. Provenance data make experiments reproducible, simplify the discovery of changes in the underlying data and pay credit to the owners of data and resources [71; 87; 175]. It is important that these data are stored in an interchangeable format and that tools are available to visualise and explore provenance. Only then it is possible to share provenance data with peers.

problems do life scientists experience using workflow systems and why?

Life science workflow systems promise to support bioinformaticians in using tools for experimentation. They are meant to simplify the design, enactment and analysis of life science experiments. A major advantage of using a workflow system instead of programming a script to connect tools is that workflow systems provide uniform access to resources that bioinformaticians require. The workflow designer does not even have to be aware of the protocol used. A workflow system provides a graphical programming environment in which these resources are provided as building blocks for life science experiments. The bioinformatician uses the workflow system to connect these building blocks and to model data transfers between the resources. A workflow is often easier to understand and to share than its scripting equivalent, because of these visualisation capabilities.

The effect of not standardising on data formats recurs in life science workflow systems. Chapter 4 shows that only 20% of the tasks in the Taverna workflows stored at myExperiment represent web services. About 30% of the tasks used in these workflows perform data transformations. Scripting tasks, which are tasks implemented by the workflow designer by means of a script, are responsible for another 14% of the tasks. It is ironical, since life science workflow systems are meant to take scripting away. In practice, however,

support for scripting is an essential requirement in a workflow system, because not all required tasks are provided by the workflow system or available as web services. Therefore, the workflow system should offer support for various scripting languages, enabling the workflow designer to choose the scripting language he is familiar with and easily reuse those created by others. Tasks to automatically compose and decompose complex data structures are important ingredients of workflow systems. They help the workflow designer to build and parse these complex data structures without programming.

Once the workflow designer has finished the design of a workflow, he can save the workflow for future experiments or share it with peers. Web sites such as myExperiment help life scientists doing this. However, the workflow highly depends on the resources its tasks represent; the workflow designer often has no influence on the existence of these resources. Resources may go down or moved to a new location, mainly because the project has ended or the services are not used anymore. The workflows that use these resources become broken. Chapter 5 shows that 14% of the Taverna workflows at myExperiment that use web services have become broken due to dead or moved services. In order to reuse these workflows, the workflow (re)user has to replace these services with alternatives, if available.

A workflow system that supports late binding offers a solution for these problems, at least in the case when services are moved or alternatives are available. Workflows designed in such a workflow system are independent of the resources available at design time. The choice of resources is postponed to enactment time. The success of late binding depends on the degree in which service providers standardise on the interface of web services to make web services replaceable. When similar web services provide similar interfaces, then a workflow system that supports late binding can easily switch between these web services without affecting the workflow model.

How can workflow systems be improved to better fit the life scientists' needs?

Based on the functional requirements, we propose our workflow system e-BioFlow. The ideas implemented in e-BioFlow are based on collaborations and discussions with bioinformaticians and the shortcomings we have experienced using existing workflow systems for a couple of experiments. To prevent implementing yet another workflow system, the implementation of e-BioFlow is based on YAWL. The workflow editor of e-BioFlow, presented in Chapter 6, enables the workflow designer to design both control flow and data flow related aspects of a workflow. The workflows designed in the e-BioFlow workflow editor are independent of the resources available at design time.

The actual binding of tasks to resources is performed by the e-BioFlow engine, which is presented in Chapter 7. The e-BioFlow engine extends the YAWL engine. In contrast to

YAWL, the e-BioFlow engine supports late binding. When the engine activates a task, it searches among the available resources for a suitable resource to execute the task. The e-BioFlow engine provides access to different types of web services and scripting languages, and supports user interaction. It is in no way limited to these resources, but can easily be extended with support for other types of web services and scripting languages due to its plugin-based architecture. The engine can deal with large data sets, because it passes data by reference. The actual data is stored in a database management system until needed to execute a task. In this way, this engine is able to run large life science workflows with several gigabytes of data involved.

The engine is designed with provenance in mind. During the run of a workflow, it generates events covering the execution state of the workflow. These events can be captured, among others, to provide real time feedback about the workflow's execution state and to store provenance data. Chapter 8 discusses e-BioFlow's provenance system. This provenance system automatically stores the events generated by the workflow engine during a workflow run. It stores these events using the interchangeable Open Provenance Model (OPM). By default, the provenance data are stored in a database, called the provenance archive, to improve the performance. The engine can use this provenance archive as cache to speed up future execution of tasks. Caching is a very useful feature of a workflow engine to quickly test and debug workflows at design time, or to quickly restore the workflow execution in case of a workflow crash. The provenance system enables its users to export provenance traces stored in the provenance archive into OPM compatible XML format to share them with peers.

The provenance system provides a provenance browser and a query interface, to explore and query the provenance traces stored in the provenance archive. By using the detail-on-demand principle, the bioinformatician can collapse and expand parts of the workflow run. In this way, he is able to explore large workflow runs. The power of e-BioFlow's engine and provenance system is demonstrated with a life science use case workflow consisting of 135 tasks and 19 sub workflows, with almost 3GB of data involved.

Current life science workflow systems require their users to design the complete workflow in advance. We have designed a new workflow editor which enables bioinformaticians to apply their explorative research approach within a workflow system. This new workflow editor enables ad-hoc workflow design: the workflow designer can design the workflow step-by-step by running unfinished workflows. Intermediate results are explicitly present in the workflow model. The workflow designer can use them as sources of inspiration to further design the workflow. The ad-hoc workflow editor helps the workflow designer in the design process by suggesting compatible tasks, i.e., tasks that can produce the data required or tasks that can take the data available as input. This helps

the workflow designer to prevent data incompatibility problems between services, but at the same time helps the workflow designer to find tasks to extend the workflow.

We have presented our initial design ideas through a mockup implementation called NIWS. Chapter 9 discusses our study among 50 life scientists with various expertises in workflow systems. This study has shown that many participants were very enthusiastic about this new workflow editor. They expect it to be much easier to use than classical workflow editors. The study resulted in both corrective and creative feedback at the level of functionality, dialogue and representation. The feedback at the functional level was used to realise a working implementation of an ad-hoc workflow editor for e-BioFlow, which is discussed in Chapter 10. Meeting the requirements at the dialogue level and the representation level is future work for a next iteration in the user-centered design. The ad-hoc workflow editor is closely connected to the other components of e-BioFlow: once the workflow designer has finished the workflow design, he can save it for future use or run it as a traditional workflow using the e-BioFlow engine.

11.2 Discussion

As mentioned in Chapter 3, workflow systems are not new. They have been used for decades to model business logic and to orchestrate web services. Also in the life science domain, workflow systems are not new. Laboratory Information Systems (LIMSs) exist for a long time, and they all have implemented a workflow, albeit hard-coded. A LIMS coordinates the lab apparatus and manages the data involved. Business workflows and LIMSs have in common that the entire workflow model must be known in advance.

Many life science tools are available as web services. The use of a workflow system to orchestrate these services to create in-silico experiments seems logical. As a result, current life science workflow systems are based on business workflow systems and LIMSs. These workflow systems do not fulfill the requirements bioinformaticians have, because they do not support the explorative research approach of the bioinformatician. Developers of workflow systems claim to apply the data flow paradigm to their workflow systems, because data plays a central role in the decision making process in an in-silico experiment. The life scientist uses data as sources of inspiration and to verify hypotheses. Nevertheless, the real data is still invisible in these workflow systems. The scientist is expected to predict the behaviour of web services and to model the entire experiment in advance, without data.

Though many researchers argue not to create yet another workflow system, we have decided to create our own playground. Adapting existing workflow systems often requires rethinking basic concepts in the architecture, because many problems are the results of decisions in the early stage of design. For example, adding support for iteration through

loops and conditional branching in a data flow paradigm is only possible through special constructs or hacking. Similarly, extending a workflow system with support for late binding is difficult if the workflow system ties workflow specifications to the locations of the resources used to design the workflow.

The well-considered architecture of e-BioFlow has resulted in an extensible framework. The traditional editors, the engine, the provenance system and the ad-hoc editor are all implemented as plugins of the base architecture. We prevented reinventing the wheel by reusing the YAWL engine to perform the execution logic. This engine has been adapted to support passing data by reference, to perform late binding, and to invoke various types of resources, to support large, data-intensive in-silico experiments of bioinformaticians.

Even though the engine is control flow based, data play a central role in the user interface. In the provenance browser and the ad-hoc editor, data are explicitly visible. The ad-hoc editor of e-BioFlow, a realisation of what Gibson et al. [69] called a virtual data playground, enables the bioinformatician to think in terms of data instead of processes. By explicitly presenting the data, it challenges scientists to play with data, and stimulates them to test services and to test different parameter settings.

11.3 Future work

Current users of life science workflow systems are bioinformaticians, for similar reasons that users of web services are often web service developers themselves [141]. Although our workflow system e-BioFlow fulfills many functional requirements of a workflow system, such as explorative design and late binding, it still requires its users to have knowledge of scripting, web services and algorithms. The ultimate goal is to provide a workflow interface that enables even the less experienced life scientists, such as the biologists, to design and run in-silico experiments.

The next step in design will be establishing the requirements at the interface level of a workflow system. Now the functional requirements of a workflow system are met, the focus should be on how to interact with workflow systems (dialogue level) and how to present information (representational level). The e-BioFlow system can be used as a working prototype for gaining insight into these requirements: let life scientists experience the new interfaces for the design and execution of workflows and the analysis of provenance data. This again requires close collaboration between computer scientists and life scientists. Only by applying a user-centered design, one can design solutions that better fit the life scientists' needs [215; 51].

The life science community, in turn, should actively promote the design and use of standards for data formats and web service interfaces, and the documentation of tools, to

simplify the use of these tools and the construction of workflows.

The result will be a new language (workflow representation) that enables the design of high-level workflow models. These high-level workflow descriptions should be closer to conceptual models life scientists have about experiments and therefore reduce the cognitive load of these scientists [210]. This new language should enable the life scientist to think in terms of steps he wants to perform instead of the services he needs to use. In current workflow languages there is often a big gap between the level at which a life scientist thinks about the life science problem and the workflow implementation (Chapter 4).

Adams et al. [3] have introduced a technique to design abstract workflows to describe the treatment processes of patients. This technique may also be applicable to design conceptual workflows in life science. Central in their approach are worklets. A worklet is a small workflow that covers all the actions required to perform a higher level task (or step). A worklet black boxes implementation details from a workflow designer's point of view. The workflow system should maintain a catalogue of workflows. It can have multiple worklets available to concretise the same task. The concretisation of tasks into worklets is done at enactment time by the workflow engine, similar to e-BioFlow's late binding facilities.

Worklets depend to a lesser extent on the standardisation of web service interfaces. They can be used in life science workflow systems to describe and implement classes of tasks. For example, worklets can help to provide a standardised interface for alignment tasks in a workflow. Then, the abstract alignment task encompasses all alignment resources, including (a)synchronous Blast services and Blat services. At design time, the workflow designer can define an alignment task in a workflow without having to choose the actual alignment algorithm. This solution enables the workflow designer to describe high level workflows that are better reusable for similar cases. Additionally, it will simplify the replacement of dead services and limit the amount of data transformation required. Its success, of course, highly depends on the degree in which community standardises on worklet interfaces. And as long people are free to use their own standards, they will.

Appendices

Appendix **A**

Novice user questionnaire

Questionnaire

Please fill in the first 4 questions and then read the next block. It is **important that you answer all of them**. Thank you.

You may answer the questions in Dutch

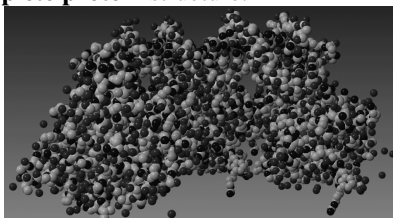
- 1 Age:.....
- 2 Gender:
 - Female
 - Male
- 3 Study background:.....
- 4 Which software tools do you use?
Please underline which software you use or specify which other program you use:
 - a. Operating System: Windows / Linux / Unix / Apple
 - b. Text editor: Word / OpenOffice / LaTeX / Other
 - c. Spreadsheet: Excel / OpenOffice / Lotus Notes / Other
 - d. Browser: Internet Explorer / Firefox / Netscape / Opera / Other.....
 - e. Mail program: Outlook Express / Outlook / Bat / Eudora / Thunderbird / Hotmail /GMail / Other
 - f. Search Engine: Google / Altavista / Yahoo / Other
 - g. Other frequently used software:

On the following pages you will be asked questions about your experiences regarding use of bioinformatics applications during the course. There are no right or wrong answers; we are interested in your personal opinions and experiences. Do not think about questions for a long time, but try to rely on your first reaction. It is no problem if you are not sure about this. Just try to give the answer that *you* think is most suitable. This questionnaire is completely anonymous and the results will not be associated with your name.

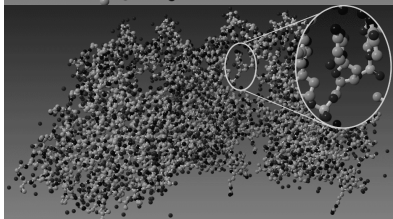
Part I: 3D Visualization tools: Yasara, Jmol, Chime

1 I prefer the following 3D view of a **complete protein** structure:

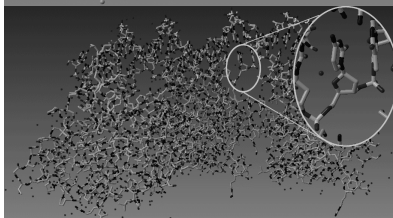
Balls



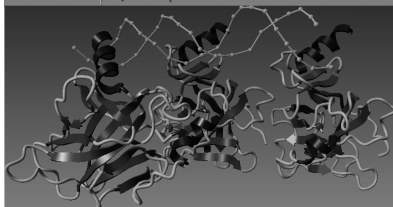
Balls and sticks (See the enlarged part -->)



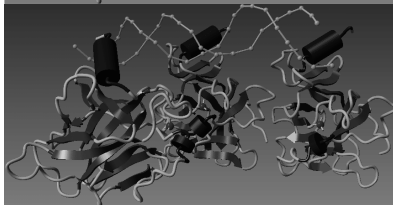
Sticks (See the enlarged part -->)



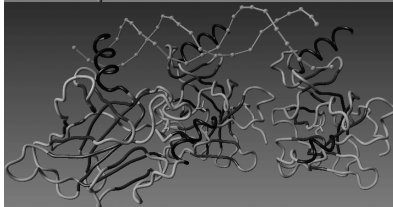
Ribbon



Cartoon



Tube



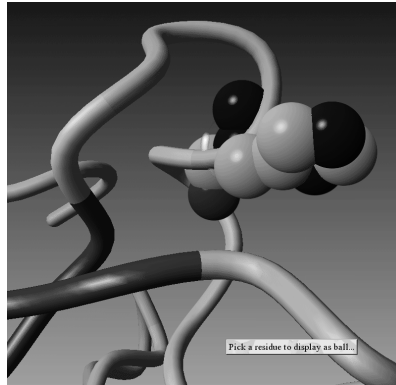
I have no preference

It depends on (please specify)

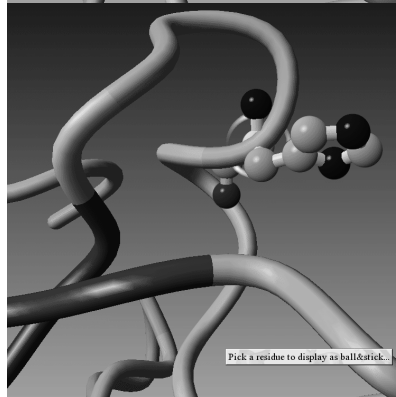
.....
.....

2 I prefer the following 3D view for a **part of a protein** structure:

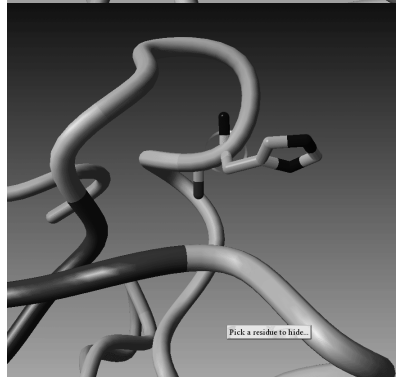
Balls



Balls and sticks



Sticks



I have no preference

It depends on (please specify)

.....
.....
.....

- 3 I often use an option to make only a selection of the protein structure visible.
 disagree strongly disagree neutral agree agree strongly
- 4 It is easy to recognize residue's structure in a protein.
 disagree strongly disagree neutral agree agree strongly
- 5 I often use several 3D visualization tools (Yasara, Chime, JMol) to get more insight into a molecule.
 disagree strongly disagree neutral agree agree strongly
- 6 I often make only the required information of an interesting residue visible (for example, only side chain of the interesting residue).
 disagree strongly disagree neutral agree agree strongly
- 7 The 3D structure of a protein gives me information about what the function of a residue is.
 disagree strongly disagree neutral agree agree strongly
- 8 The 3D position of an residue in a protein in combination with my background knowledge about the residue always gives me enough information to determine a possible function of a residue.
 disagree strongly disagree neutral agree agree strongly
- 9 When I know the position of a residue, I often use additional resources (like access to other databases, Google, etc.) to gain more information about the protein.
 disagree strongly disagree neutral agree agree strongly
- 10 When I know the function of an **entire protein**, I often use additional resources (like access to other databases, Google, etc), to verify my conclusions.
 disagree strongly disagree neutral agree agree strongly
- 11 When I know the function of a **part of the protein**, I often use additional resources (like access to other databases, Google, etc), to verify my conclusions.
 disagree strongly disagree neutral agree agree strongly
- 12 I search first for existing information about the protein in sources on the internet before trying to discover more about the protein myself.
 disagree strongly disagree neutral agree agree strongly

13 I often use an option to hide some irrelevant part of the protein.

disagree strongly disagree neutral agree agree strongly

Please write here your extra **comments** about the about the 3D Visualization tools:

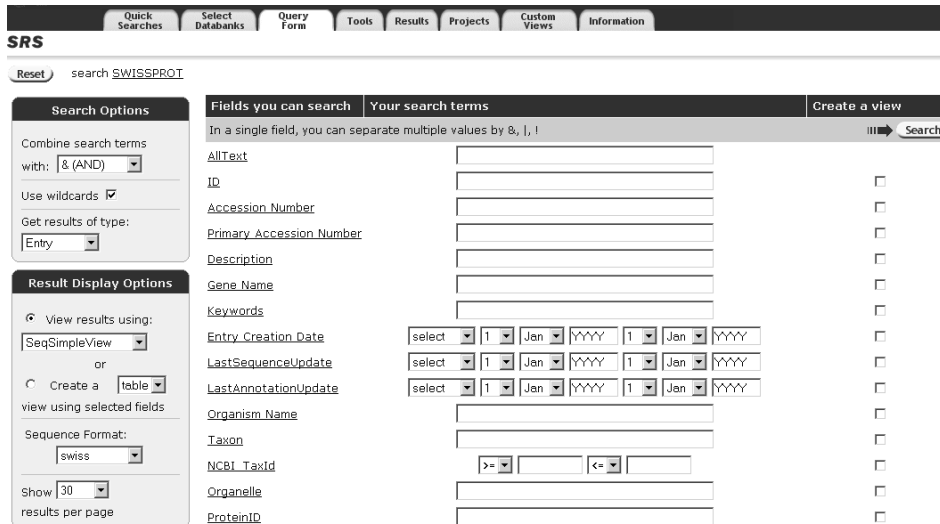
Part II: SRS & MRS

1 I often change search options (e.g. “**Blast options**” for BLAST search, as on the figure below) to optimize my search.

disagree strongly disagree neutral agree agree strongly

2 I use the extended query form (for example in SRS to find previous annotations of a protein in order to see the history of this protein annotation, as on the figure below).

- Often
- Sometimes
- Never



3 When I receive results from a tool (e.g. for sequence alignment), I try to change parameters to see how it will influence the results.

disagree strongly
 disagree
 neutral
 agree
 agree strongly

4 The tools often give cross references to other databases with additional information. I often use these cross references to get more information.

disagree strongly
 disagree
 neutral
 agree
 agree strongly

5 When I am not satisfied with the information that I find in the SwissProt database, I use the following additional sources (**more than one option** can be chosen):

- Search engines
- Cross references
- Search manually in other databases
- Knowledge from other students
- Other
-
-
- None

6 When I search for information about a disease related to a protein, I use the following additional sources to get more information (**more than one option** can be chosen).

- Search engines
- Omim cross references
- Other cross references
- Other
-
-
- None

7 When I look for information, the most important to me is:

Rank the options by importance to you (1..3)

- Detailed answers
- Reliable answers
- Non-redundant answers
- Easy-to-use answers (standard format, like Fasta used in ClustalW)
- Well-documented answers (with respect to the traceability of their origin)

8 When I use cross-references, I use the cross reference according to:

Rank the options by importance to you (1..3)

- The kind of information I want to get
- The reliability of the source which is going to provide the data
- The fact that I know whether the cross-reference has been added manually
- The fact that I know whether the cross-reference has been added automatically (e.g. by computer systems)
- Other.....
-
-

Comments:

Please write here your extra comments about **MRS&SRS** or any other comments:

You have finished the questionnaire. Thank you very much!

Appendix B

Symbols used in Taverna workflows

Taverna uses five different symbols in the graphical representation of a workflow. The tasks are represented as boxes. The inputs and outputs of tasks are hidden by default. Boxes have different colours depending on the invocation mechanism the task uses, such as MOBY-S or SOAP/WSDL. Tasks related to workflow inputs and workflow outputs are surrounded by a dotted box. Data connections are represented by arrows between task boxes. In case of a dependency between tasks without data being involved, a circled arrow is used.

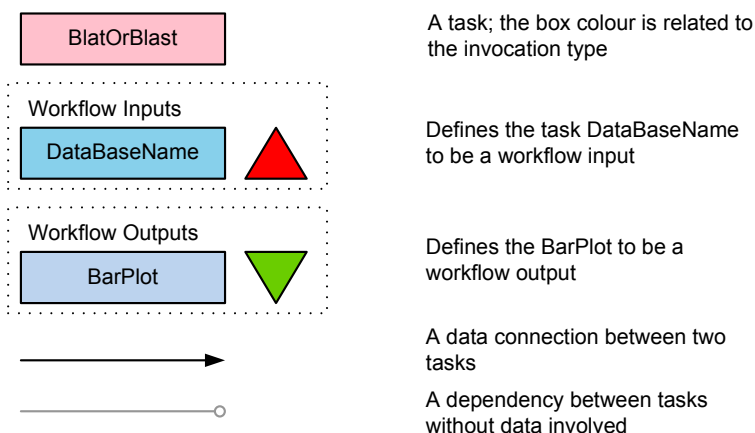


Figure B.1: Taverna uses 5 different symbols in its workflow language.

Appendix **C**

Use case workflow

Appendix **D**

Using R in Taverna: RShell v1.2

D.1 Introduction

The open source workflow system Taverna [148] provides access to and integration of many life science web services. Not all desired data-analysis procedures are available as web services. Therefore, support for scripting is essential. By default, Taverna provides scripting in Java. Many scientists are not familiar with Java and prefer other languages, such as R, a statistical language [98]. Taverna had limited support for R.

We developed an R plugin for Taverna: RShell, which removes the limitations of R support in Taverna workflows. RShell can be incorporated in a Taverna workflow as a processor for executing R scripts. In this appendix, we present a detailed description of the RShell functionality and architecture. Additionally, we introduce new functionality of our current RShell version 1.2.

D.2 Implementation

RShell is based on a client-server architecture. It requires a local or remote installation of the R-interpreter with the Rserve library [198] installed. The Rserve library turns the R-interpreter into a server, which enables other applications to communicate with the R-interpreter by means of a socket connection. From here on, the R-interpreter with the Rserve library installed will be denoted as the *R server*. RShell uses the Java library named REngine to establish and maintain connection between Taverna and the R server. To execute R scripts, the RShell processor sends the script and the input data via the RShell session manager to the R server, which delegates the script to the R-interpreter and sends

the results back to RShell (Figure D.1). This new version is fully compatible up to and including R 2.9.

D.2.1 Configuring the RShell processor

The RShell processor is highly configurable: the user can define the script to be executed, the input ports to feed the relevant data, and the output ports to extract the created data. The user can configure the RShell processor by invoking the RShell processor in Taverna's "advanced model explorer". This will show a configuration dialog containing four tabs for: i) the script, ii) the input ports, iii) the output ports, and iv) the connection settings.

Syntax highlighting is available in the scripting tab, to help the user write correct R code. The keywords are highlighted in blue; the inputs and outputs of the RShell processor are highlighted in pink.

RShell supports multiple input ports and output ports. Each port has a name, corresponding to the variable name it represents, as well as a type, corresponding to the type of data it holds. RShell uses the data type to determine how to exchange data with Taverna. RShell supports booleans, numeric values (both integers and floating point numbers), strings, and, vectors of numerics and strings. Inputs and outputs of these data types can directly be used as variables in the R script. RShell provides the text-file data type to handle large data. Ports of this type can be read and written as normal R tables. The PNG data type and, since version 1.2, the PDF data type are available for graphical output. PNG can be used for bitmap graphics; PDF for vector graphics. PNG and PDF outputs are handled as graphics devices in the standard R fashion. Input ports and output ports can be defined using the input ports and output ports tab.

RShell can execute any R script as long as the required libraries are installed in the R server. By default, the RShell processor is configured to use a locally installed R server serving at address *localhost*, port *6311*. It can be configured to use a remote installation of R instead, using the connections tab. This can be useful when, the user is not able to install R, a central installation of R with a specific set of installed libraries is used, or R is installed in a grid environment. Multiple R servers can be accessed within the same workflow.

D.2.2 Persistent sessions

RShell supports persistent sessions to prevent unnecessary data transfer. The user can enable persistent sessions in the connection tab. When these are enabled, all input data, output data and script variables will be kept in memory of the R-interpreter until the whole workflow execution is finished. Multiple RShell processors in the same workflow

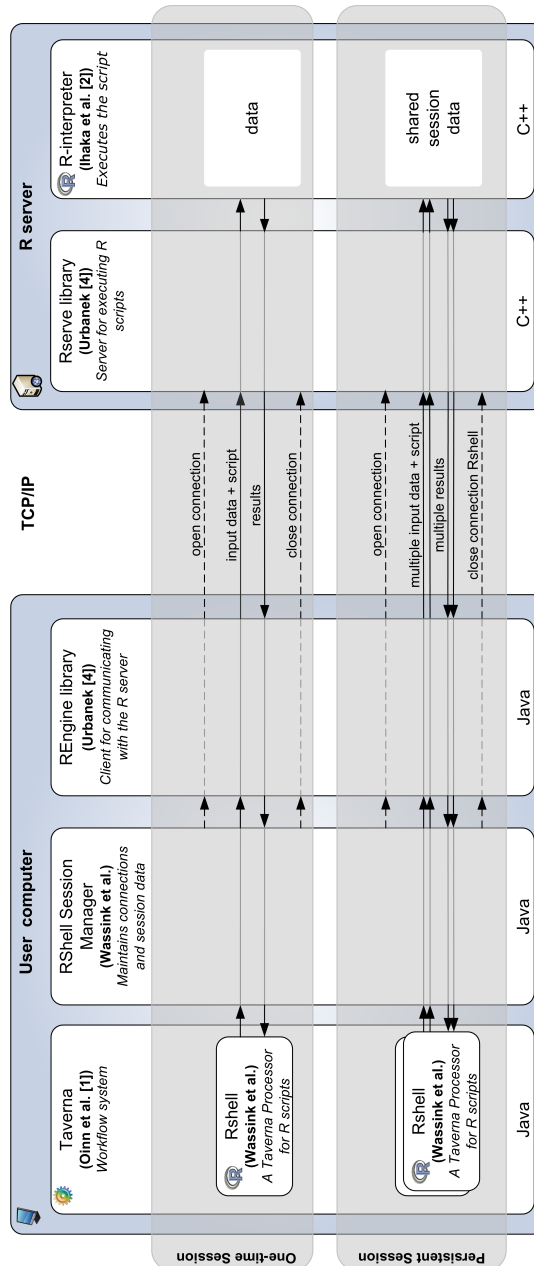


Figure D.1: Each RShell processor communicates with the R-Interpreter through the RShell Session Manager. The RShell Session Manager sets up and maintains the connection with the R interpreter with the RServe library installed through the REngine java library.

are able to use the data provided by previously used processors, without requiring data links between these processors, as long as they use the same R server.

Although persistent sessions can be very useful, they have three limitations: i) sessions can only be used among RShell processors, ii) RShell processors involved in a single persistent session have to access the same R server (but RShells devoted to a different session can access other R servers), and iii) it takes extra effort to keep a provenance log of data kept at the R interpreter. Taverna manages all data consumed and produced by processors. By using persistent sessions, Taverna is not aware of the data generated by one RShell processor and used by another RShell processor. If the user wants to record data generated by an RShell processor, he can do that by defining an output port for that variable.

D.3 Availability and requirements

Project name: RShell v1.2

Project home page: <http://ewi.utwente.nl/~biorange/rshell/>

Operating system(s): Any (Java)

Programming language: Java

Other requirements: Java 1.6, Taverna 1.3+ and R with the Rserve library installed. Taverna, R and Rserve are all open source and freely available.

Licence: GNU GPL

D.4 Conclusions

With the introduction of our RShell plugin, scripting in R is now available for Taverna. RShell has a configurable processor that is able to execute any R script that can be executed by the available installation of the R-interpreter. The client-server architecture enables a centralised installation of the R-interpreter containing all frequently used libraries used by an organisation. The support for persistent sessions in RShell 1.2 helps to prevent data transfer overload.

RShell 1.0 comes with the standard installation of Taverna. RShell 1.2 can be downloaded as a plugin for Taverna and contains several improvements, such as support for vector graphics and persistent sessions.

Mapping Vega-designed oligos to the Ensembl assembly

A microarray with 15k probes of 60-mer oligonucleotides has been designed on gene sequences from Vega ¹ and Ensembl ² that are also known in the Zebrafish Information Network ³ of the genome DNA-sequence assemblies. For zebrafish, the VEGA set is not a subset of the Ensembl set. To judge the agreement that exists between the different assembly annotations, we mapped the Vega-designed probes onto the Ensembl assembly in the following way. All probe sequences are aligned to the Ensembl assembly. Hits with an e-value below $1.5e-4$ are considered to be able to contribute to the hybridisation signal on the microarray [90]. Next, for each hit, a query is performed to check which genes and/or transcripts are present at the hit location in the genome. Finally, each probe with at least one hit is further classified based on the number of hits, genes and transcripts. For this classification, an additional lower cut-off is applied (e-value below $1e-12$) and the possibility of the occurrence of intron-spanning probes has been considered. Therefore, probe sequences that show two or more hits located close together on the genome, and that constitute a continuous stretch of more than 57 nucleotides on the probe, are labeled intron-spanning probes. For the alignment, BioMoby Blat and Blast services at WUR ⁴ are used. Gene and transcript finding is performed by the Ensembl 51 Genes BioMart service ⁵. The RShell processor is used to implement the classification and the

¹http://vega.sanger.ac.uk/Danio_rerio, last visited: December 2009

²http://www.ensembl.org/Danio_rerio, last visited: December 2009

³<http://zfin.org>, last visited: December 2009

⁴<http://www.bioinformatics.nl>, last visited: December 2009

⁵<http://www.ensembl.org/biomart>, last visited: December 2009

visualisation in the workflow.

Table E.1 and Figure E.1 show the result of the classification. About 38.3% of the oligos have a single hit on the Ensembl assembly and represent a single transcript (class 0). Another 3.0% of the oligos were linked to multiple transcripts of a single gene (class 5–7). The oligos that match multiple genes, class 11, are responsible for 5.6% of the total number of oligos. For 45% of the oligos, no genes were found although they have a hit on the genome assembly (class 12). These probes target on Vega transcripts that are absent in the Ensembl annotation.

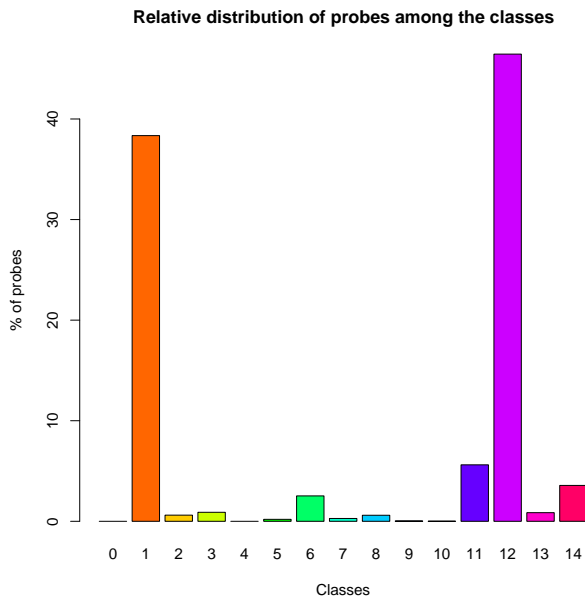


Figure E.1: Number of probes per class.

Table E.1: Classifications of the probes of a microarray chip including the description of the classes.

Class	Description	# probes	% probes
0	no hit	0	0.0
1	single hit, single transcript, single gene	3128	38.3
2	multiple hits, single transcript, single gene, intron spanning	50	0.6
3	multiple hits, single transcript, single gene, possible intron spanning ^a	73	0.9
4	multiple hits, single transcript, single gene, no intron spanning	0	0.0
5	multiple hits, multiple transcripts, single gene, intron spanning	16	0.2
6	multiple hits, multiple transcripts, single gene, possible intron spanning ^a	206	2.5
7	multiple hits, multiple transcripts, single gene, no intron spanning	23	0.3
8	single hit, does not meet additional criteria ^b	49	0.6
9	multiple hits, single transcript, do not meet additional criteria ^b	3	0.0
10	multiple hits, multiple transcripts, do not meet additional criteria ^b	1	0.0
11	multiple hits, multiple genes	458	5.6
12	no transcript found but hit(s) meet additional criteria ^b	3789	46.5
13	no transcript found and hit(s) do not meet additional criteria ^b	70	0.9
14	multiple hits, single transcript, single gene plus hit without transcript found and hits meet additional criteria ^b	291	3.6

^aOligo below e-value cut-off 1e-12, but also intron spanning criteria met.

^bAdditional criteria: either e-value below 1e-12 or intron spanning.

Appendix **F**

Categorisation of local services

Table F.1: The local services are put in 9 categories. The last category, “Unknown”, contains all uncategorised services.

Class	Services	
Conditional	FailIfFalse	FailIfTrue
Constant	stringconstant	
Data transformation	biomobyobject biomobyparser ByteArrayToString DecodeBase64 EncodeBase64 ExtractMobyData FilterStringList FlattenList GenBankParserWorker PadNumber RegularExpressionStringList SliceList	SplitByRegex StringConcat StringListMerge StringSetDifference StringSetIntersection StringSetUnion StringStripDuplicates XMLInputSplitter XMLOutputSplitter XPathTextWorker XSLTWorker
Interaction	AskWorker ChooseWorker SelectWorker	TellWorker WarnWorker
Operation	ReverseCompWorker	TranscribeWorker
Retrieval	NucleotideFastaWorker NucleotideGBSeqWorker NucleotideINSDSeqXMLWorker NucleotideTinySeqXMLWorker ProteinFastaWorker	ProteinGBSeqWorker ProteinINSDSeqXMLWorker ProteinTinySeqXMLWorker PubMedEFetchWorker SwissProtParserWorker
Testing	EmitLotsOfStrings TestAlwaysFailingProcessor	TestSometimesFails
Util	apiconsumer ConcatenateFileListWorker EchoList ExtractImageLinks FileListByExtTask FileListByRegexTask	LocalCommand SQLQueryWorker TextFileReader TextFileWriter WebImageFetcher WebPageFetcher
Unknown	alternate arbitrarygt4	helpurl knowarcjanitor

Appendix **G**

NIWS questionnaire

Introduction

We are investigating the usability of workflow systems and we have some ideas how to improve them. But to really improve these systems, we need your help too. Two scenarios showing sketchy mockups of the interface have been designed in PowerPoint to share our ideas with you. We would like to know what you think of them.

Based on these scenarios, we will ask you 4 questions. You will get five minutes time to answer each question. It is important to know that **these questions are not used as a test of your knowledge about workflow systems or this system in particular and you are not expected to learn anything.** We are only interested in what you think of the system in order to validate and improve our ideas.

In total, it will take about 30 minutes time. You will get a 1 GB USB stick as reward for participating this session. Please do not turn the page to the next question until I instruct you to do so.

Thank you in advance.

Background information

I am a:

- Biologist
- Bioinformatician
- Chemist
- Computer scientist
- Mathematician
- Other, namely:

My experience with workflow systems

- I have only seen them
- Little experienced I have used them only for a couple of experiments
- Experienced, I use them regularly to do my experiments
- Very experienced, I use them for almost all experiments

Question 1

We would like to know how you imagine the NIWS workflow system after seeing these scenarios.

For that reason we ask you the following question: Please, explain to your colleague Tom, who is regularly using workflows, but who is not familiar with this new system, what is NIWS.

You can use text, drawings, sketches, keywords etc.

Question 2

Tom has found 200 interesting sequences and wants to perform a sequence based search against the mouse (*Mus Musculus*) genome. Tom is curious about this system NIWS and wants to use it. Please explain him how to do using NIWS.

You can use text, drawings, sketches, keywords etc.

Question 3

Tom wants to do align two sequences using NIWS. Please explain him how to find a sequence alignment service and how to perform the sequence alignment using NIWS.

You can use text, drawings, sketches, keywords etc.

Question 4

Tom wants to perform a sequence based search against the (mouse) Mus Muscles assembly using a Blast service for a few hundred sequences. He wants to use a BioMart service to find out which transcripts and which genes are located on the locations found.

Tom doesn't know how to connect different services in NIWS. He is afraid that data may be incompatible. Please explain to him how to construct the workflow in NIWS.

You can use text, drawings, sketches, keywords etc.

Bibliography

- [1] G. Abram and L. Treinish. An extended data-flow architecture for data analysis and visualization. In G.M. Nielson and D. Silver, editors, *IEEE Conference on Visualization*, pages 263–270, Atlanta, GA, USA, 1995. IEEE Computer Society Press.
- [2] F. Achard, G. Vaysseix, and E. Barillot. XML, bioinformatics and data integration. *Bioinformatics Review*, 17(2):115–125, 2001.
- [3] M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, pages 291–308, Montpellier, France, 2006.
- [4] M. Addis, J. Ferris, M. Greenwood, P. Li, D. Marvin, T.M. Oinn, and A. Wipat. Experiences with e-Science workflow specification and enactment in bioinformatics. In S.J. Cox, editor, *e-Science All Hands Meeting 2003*, pages 459–466, Nottingham, United Kingdom, 2004.
- [5] H. Afsarmanesh, R.G. Belleman, A.S.Z. Belloum, A. Benabdelkader, J.F.J. van den Brand, G.B. Eijkel, A. Frenkel, C. Garita, D.L. Groep, R.M.A. Heeren, Z.W. Hendrikse, L.O. Hertzberger, J.A. Kaandorp, V. Kalletas, E.C. and Korkhov, C.T.A.M. de Laat, P.M.A. Sloot, D. Vasunin, A. Visser, and H.H. Yakali. VLAM-G: A grid-based virtual laboratory. *Scientific Programming*, 10(2):173–181, 2002.
- [6] A. Akram, D. Meredith, and R. Allan. Evaluation of BPEL to scientific workflows. In S.J. Turner, B.S. Lee, and W. Cai, editors, *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 269–274, Washington, DC, USA, 2006. IEEE Computer Society.
- [7] G. Allen, A. Goodale, T. Radke, M. Russell, E. Seidel, K. Davis, K. N. Dolkas, N. D. Doulamis, T. Kielmann, A. Merzky, J. Nabrzyski, J. Pukacki, J. Shalf, and I.J. Taylor. Enabling applications on the grid: a GridLab overview. *International Journal of High Performance Computing Applications*, 17(4):449–466, 2003.
- [8] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and limitations of current workflow management systems. *IEEE Expert*, 12(5):68–74, 1997.

- [9] M.N. Alpdemir, A. Mukherjee, N.W. Paton, A.A.A. Fernandes, P. Watson, K. Glover, C. Greenhalgh, T.M. Oinn, and H.J. Tipney. Contextualised workflow execution in myGrid. In P.M.A. Sloot, A.G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Advances in Grid Computing - EGC 2005*, pages 444–453, Amsterdam, The Netherlands, 2005. Springer.
- [10] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance collection support in the Kepler scientific workflow system. In L. Moreau and I. Foster, editors, *International Provenance and Annotation Workshop (IPAW 2006)*, volume 4145 LNCS, pages 118–132, Chicago, IL, USA, 2006. Springer Verlag.
- [11] I. Altintas, C.H.D.J.E. Berkley, E.J.M. Jaeger, B. Ludäscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In M. Hatzopoulos and Y. Manolopoulos, editors, *16th International Conference on Scientific and Statistical Database Management (SSDBM'04)*, pages 423–424, Santorini Island, Greece, 2004.
- [12] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. A Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.
- [13] A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Golland, N. Kartha, C.K. Liu, S. Thatte, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language version 2.0. working draft. Technical Report WS-BPELTC 12791, OASIS Standard, 2008.
- [14] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene Ontology: tool for the unification of biology. *Nature America Inc.*, 25(1):25–29, 2000.
- [15] P.G. Baker, A. Brass, S. Bechhofer, C.A. Goble, N.W. Paton, and R. Stevens. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics*, 16(2):182–183, 2000.
- [16] R.S. Barga and L.A. Digiampietri. Automatic generation of workflow provenance. In L. Moreau and I. Foster, editors, *International Provenance and Annotation Workshop (IPAW 2006)*, pages 1–9, Chicago, IL, USA, 2006.
- [17] R.S. Barga and D.B. Gannon. Scientific versus business workflows. In I.J. Taylor, E. Deelman, D.B. Gannon, and M.S. Shields, editors, *Workflows for e-Science*, pages 258–275. Springer Verlag, Berlin, DE, 2007.
- [18] A. Barker and J. van Hemert. Scientific workflow: A survey and research directions. In R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics*, pages 746–753, Gdansk, Poland, 2007.
- [19] P. Barthelmeß and J. Wainer. Workflow systems: a few definitions and a few suggestions. In N. Comstock and C.A. Ellis, editors, *Proceedings of the Conference on Organizational Computing Systems (COOCS'95)*, pages 138–147, Milpitas, California, USA, 1995.
- [20] J.C. Bartlett and E.G. Toms. Developing a protocol for bioinformatics analysis: An integrated information behavior and task analysis approach. *Journal of the American Society for Information Science and Technology*, 56(5):469–482, 2005.

- [21] A.D. Baxevanis. Searching the NCBI databases using Entrez. *Current protocols in bioinformatics*, 1:Unit 6.10, 2006.
- [22] K. Belhajjame. Addressing the issue of service volatility in scientific workflows. In B. J. Krämer, K. Lin, and P. Narasimhan, editors, *Service-Oriented Computing – ICSOC 2007*, pages 377–382, Vienna, Austria, 2008.
- [23] K. Belhajjame, S.M. Embury, and N.W. Paton. On characterising and identifying mismatches in scientific workflows. In S. Istrail, P.A. Pevzner, and M. Waterman, editors, *Data Integration in the Life Sciences (DILS'06)*, pages 240–247, Hinxton, UK, 2006.
- [24] A.S.Z. Belloum, D.L. Groep, Z.W. Hendrikse, L.O. Hertzberger, V. Korkhov, C.T.A.M. de Laat, and D. Vasunin. VLAM-G: A grid-based virtual laboratory. *Future Generation Computer Systems*, 19(2):209–217., 2003.
- [25] F.C. Bernstein, T.F. Koetzle, G.J.B. Williams, E.F. Meyer Jr., M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank. a computer based archival file for macromolecular structures. *Journal of Molecular Biology*, 112(3):535–542, 1977.
- [26] M.R. Berthold, N. Cebron, F. Dill, T.R. Gabriel, T. Kötter, T. Meini, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. KNIME: The konstanz information miner. In C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, editors, *Data Analysis, Machine Learning and Applications*, chapter KNIME: The Konstanz Information Miner, pages 319–326. Springer Verlag, Berlin, DE, 2008.
- [27] H. Beyer. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann, San Francisco, USA, 1997.
- [28] D. Bolchini, A. Finkelstein, V. Perrone, and S. Nagl. Better bioinformatics through usability analysis. *Bioinformatics*, 25(3):406–412, 2008.
- [29] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys*, 37(1):1–28, 2005.
- [30] S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In S. Istrail, P.A. Pevzner, and M. Waterman, editors, *Data Integration in the Life Sciences*, chapter An Ontology-Driven Framework for Data Transformation in Scientific Workflows, pages 1–16. Springer Verlag, Berlin, DE, 2004.
- [31] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In L.M.L. Delcambre, C. Kop, C.M. Heinrich, J. Mylopoulos, and O. Pastor, editors, *Conceptual Modeling – ER 2005*, volume 3716 of *Lecture Notes in Computer Science*, pages 369–384, Klagenfurt, Austria, 2005. Springer.
- [32] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, and D. Winer. SOAP. Technical Report NOTE-SOAP-20000508, W3C, 2000.
- [33] J. Boyle, C. Cavnor, S. Killcoyne, and I. Shmulevich. Systems biology driven software design for the research enterprise. *BMC Bioinformatics*, 9(295):1–24, 2008.

- [34] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C.A. Ball, H.C. Causton, T. Gaasterland, P. Glenisson, F.C. Holstege, I.F. Kim, V. Markowitz, H. Matese, J.C. and Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron. Minimum Information About a Microarray Experiment (MIAME)-toward standards for microarray data. *Nature Genetics*, 29(4):365–371, 2001.
- [35] A. Brogi and R. Popescu. From BPEL processes to YAWL workflows. In M. Bravetti, M. Núñez, and G. Zavattaro, editors, *Web Services and Formal Methods*, pages 107–122. Springer Verlag, Berlin, 2006.
- [36] C. Brooksbank, G. Cameron, and J. Thornton. The European Bioinformatics Institute's data resources: Towards systems biology. *Nucleic Acids Research*, 33(Database issue):D46–D53, 2005.
- [37] P.O. Brown and D. Botstein. Exploring the new world of the genome with DNA microarrays. *Nature genetics*, 21(1):33–37, 1999.
- [38] P. Buneman, S. Khanna, and WC. Tan. Why and where: A characterization of data provenance. In J. Van den Bussche and V. Vianu, editors, *Proceedings of the International Conference Database Theory - ICDT 2001*, pages 316–330, Berlin, DE, 2001. Springer Verlag.
- [39] S.P. Callahan, J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, and H.T. Vo. Vistrails: visualization meets data management. In V. Hristidis and N. Polyzotis, editors, *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747, Chicago, Illinois, USA, 2006. ACM.
- [40] S. Carrerre and J. Gouzy. REMORA: a pilot in the ocean of biomoby web-services. *Bioinformatics*, 22(7):900–901, 2006.
- [41] J.G. Chin, L.R. Leung, K. Schuchardt, and D. Gracio. New paradigms in problem solving environments for scientific computing. In K. Hammond, Y. Gil, and D. Leake, editors, *7th International Conference of Intelligent User Interfaces (IUI 2002)*, pages 39–46, New York, NY, USA, 2002. ACM Press.
- [42] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. WSDL. Technical Report NOTE-wsdl-20010315, W3C, 2001.
- [43] G. Cochrane, R. Akhtar, P. Aldebert, N. Althorpe, A. Baldwin, K. Bates, S. Bhattacharyya, J. Bonfield, L. Bower, P. Browne, M. Castro, A. Cox, F. Demiralp, R. Eberhardt, N. Faruque, G. Hoad, M. Jang, T. Kulikova, A. Labarga, R. Leinonen, S. Leonard, Q. Lin, R. Lopez, D. Lorenc, H. McWilliam, G. Mukherjee, F. Nardone, S. Plaister, S. Robinson, S. Sobhany, R. Vaughan, D. Wu, W. Zhu, R. Apweiler, T. Hubbard, and E. Birney. Priorities for nucleotide trace, sequence and annotation data capture at the Ensembl Trace Archive and the EMBL Nucleotide Sequence Database. *Nucleic Acids Research*, Database Issue(36):D5–D12, 2008.
- [44] J. Cohen. Bioinformatics-an introduction for computer scientists. *ACM Computing Surveys*, 36(2):122 – 158, 2004.
- [45] S. Cohen, S. Cohen-Boulakia, and S.B. Davidson. Towards a model of provenance and user views in scientific workflows. In U. Leser, F. Naumann, and B.A. Eckman, editors, *Data Integration for the Life Sciences (DILS'06)*, volume 4075, pages 264–279, Cambridge, UK, 2006. Springer-Verlag.

- [46] S. Cohen-Boulakia, S.B. Davidson, and C. Froidevaux. A user-centric framework for accessing biological sources and tools. In S. Istrail, P.A. Pevzner, and M. Waterman, editors, *Data Integration in the Life Sciences (DILS'05)*, pages 3–18, San Diego, CA, USA, 2005.
- [47] M.F. Costabile, D. Fogli, P. Mussio, and A. Piccinno. 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.
- [48] T. Coughlan and P. Johnson. Interaction in creative tasks: Ideation, representation and evaluation in composition. In R.E. Grinter, T. Rodden, P.M. Aoki, E. Cutrell, R. Jeffries, and G.M. Olson, editors, *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems*, pages 531–540, Montréal, Canada, 2006. ACM Press.
- [49] V. Curcin, M. Ghanem, and Y. Guo. Web services in the life sciences. *Drug Discovery Today*, 10(12):865–871, 2005.
- [50] S.B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In J. Wang, editor, *SIGMOD international conference on Management of data, SIGMOD 2008*, pages 1345–1350, Vancouver, Canada, 2008. ACM.
- [51] D. De Roure and C.A. Goble. Software design for empowering scientists. *Software, IEEE*, 26(1):88–95, 2009.
- [52] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbee, R. Cavanaugh, and S. Koranda. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1):25–39, 2003.
- [53] E. Deelman, D.B. Gannon, M.S. Shields, and I.J. Taylor. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2008.
- [54] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Keselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J.C. Jacob, and D.S. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.
- [55] P. Dourish. Process descriptions as organisational accounting devices: the dual use of workflow technologies. In C. Ellisand and I. Zigurs, editors, *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 52–60, Boulder, Colorado, USA, 2001. ACM Press.
- [56] L.L. Downey. Group usability testing: Evolution in usability techniques. *Journal of Usability Studies*, 2(3):133–144, 2007.
- [57] D.J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J.M. Trent. Expression profiling using cDNA microarrays. *Nature Genetics*, 21(1 SUPPL.):10–14, 1999.
- [58] K. Dunbar. How scientists really reason: Scientific reasoning in real-world laboratories. In *The Nature of Insight*, pages 365–395. The MIT Press, Cambridge, MA, 1995.
- [59] K. Dunbar. How scientists think: On-line creativity and conceptual change in science. In *Creative Thought: An Investigation of Conceptual Structures and Processes*, pages 461–493. American Psychological Association, Washington DC, USA, 1997.

- [60] S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber. BioMart and Bio-conductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21(16):3439–3440, 2005.
- [61] T. Etzold, A. Ulyanov, and P. Argos. SRS: information retrieval system for molecular biology data banks. *Methods Enzymol*, 266:114–128, 1996.
- [62] S. Faisal, P. Cairns, and B. Craft. InfoVis experience enhancement through mediated interaction. In E. Zudilova-Seinstra and T. Adriaansen, editors, *ICMI'05 Workshop on Multimodal Interaction for the Visualization and Exploration of Scientific Data*, pages 3–9, Trento, Italy, 2005.
- [63] R.T. Fielding. *Architectural styles and the design of network-based software architectures in Information and Computer Science*. PhD thesis, University of California at Irvine, 2000.
- [64] R.T. Fielding and R.N. Taylor. Principled design of the modern web architecture. In C. Ghezzi, M. Jazayeri, and A.L. Wolf, editors, *Proceedings of the 22nd International Conference on Software Engineering (ICSE'00)*, pages 407–416, Limerick, Ireland, 2000. ACM Press.
- [65] C. Fields. Data exchange and inter-database communication in genome projects. *Trends in Biotechnology*, 10(1-2):58–61, 1992.
- [66] P. Flicek, B.L. Aken, K. Beal, B. Ballester, M. Caccamo, Y. Chen, L. Clarke, G. Coates, F. Cunningham, T. Cutts, T. Down, S.C. Dyer, T. Eyre, S. Fitzgerald, J. Fernandez-Banet, S. Gräf, S. Haider, M. Hammond, R. Holland, K.L. Howe, K. Howe, N. Johnson, A. Jenkinson, A. Kähäri, D. Keefe, F. Kokocinski, E. Kulesha, D. Lawson, L. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, A. Prlic, S. Rice, D. Rios, M. Schuster, I. Sealy, G. Slater, D. Smedley, G. Spudich, S. Trevanion, A.J. Vilella, J. Vogel, S. White, M. Wood, E. Birney, T. Cox, V. Curwen, R. Durbin, X.M. Fernandez-Suarez, J. Herrero, T.J.P. Hubbard, A. Kasprzyk, G. Proctor, J. Smith, A. Ureta-Vidal, and S. Searle. Ensembl 2008. *Nucleic Acids Research*, 36(Database issue):D707–D714, 2008.
- [67] I. Foster. Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.
- [68] D. Georgakopoulos, M.F. Hornick, and A.P. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.
- [69] A. Gibson, M. Gamble, K. Wolstencroft, T.M. Oinn, and C.A. Goble. The Data Playground: An intuitive workflow specification environment. In S.J. Cox, editor, *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 59–68, Washington, DC, USA, 2007. IEEE Computer Society Press.
- [70] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D.B. Gannon, C.A. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.
- [71] C.A. Goble. Position statement: Musings on provenance, workflow and (semantic web) annotations for bioinformatics. In Y. Zhao, editor, *Proceedings of the Workshop on Data Derivation and Provenance*, pages 1–6, Chicago, USA, 2002.

- [72] C.A. Goble and D. De Roure. myExperiment: Social networking for workflow-using e-scientists. In E. Deelman and I.J. Taylor, editors, *WORKS'07*, pages 1–2, Monterey, California, USA., 2007.
- [73] C.A. Goble and D. De Roure. Curating scientific web services and workflows. *Educause Review*, 43(5):1–4, 2008.
- [74] C.A. Goble, R. Stevens, K. Glover, C. Greenhalgh, C. Jennings, S. Pearce, P. Li, M. Radenkovic, and A. Wipat. The myGrid project: services, architectures and demonstrator. In S.J. Cox, editor, *UK e-Science All Hands Meeting 2003*, pages 595–602, Nottingham, UK, 2003.
- [75] C.A. Goble, R. Stevens, D. Hull, K. Wolstencroft, and R. Lopez. Data curation + process curation=data integration + science. *Briefings in bioinformatics*, 9(6):506–517, 2008.
- [76] A. Goderis, D. De Roure, C.A. Goble, J. Bhagat, D. Cruickshank, P. Fisher, D. Michaelides, and F. Tanoh. Discovering scientific workflows: The myExperiment benchmarks. *IEEE Transactions on Automation Science and Engineering*, Submitted, 2008.
- [77] A. Goderis, P. Li, and C.A. Goble. Workflow discovery: the problem, a case study from e-science and a graph-based solution. In F. Leymann and L.J. Zhang, editors, *International Conference on Web Services, 2006 (ICWS '06)*, pages 312–319, Chicago, Illinois, USA, 2006.
- [78] A. Goderis, U. Sattler, P.W. Lord, and C.A. Goble. Seven bottlenecks to workflow reuse and repurposing. In Y. Gil, E. Motta, V.R. Benjamins, and M.A. Musen, editors, *The Semantic Web - ISWC 2005*, pages 323–337, Galway, Ireland, 2005.
- [79] P.M.K. Gordon and C.W. Sensen. A pilot study into the usability of a scientific workflow construction tool. Technical Report 2007-874-26, Sun Center of Excellence for Visual Genomics, 2007.
- [80] P.M.K. Gordon and C.W. Sensen. Seahawk: moving beyond HTML in web-based bioinformatics analysis. *BMC bioinformatics*, 8(208):1–13, 2007.
- [81] P.M.K. Gordon, Q. Trinh, and C.W. Sensen. Semantic web service provision: a realistic framework for bioinformatics programmers. *Bioinformatics*, 23(9):1178–1180, 2007.
- [82] S. Graham, A. Karmarkar, J. Mischkinisky, I. Robinson, and I. Sedukhin. The WS-Resource framework. Technical report, OASIS, 2006.
- [83] T. R. G. Green and M. Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*, 7(2):131–174, 1996.
- [84] M. Greenwood, C.A. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau, and T.M. Oinn. Provenance of e-Science experiments - experience from bioinformatics. In S.J. Cox, editor, *Proceedings of UK e-Science All Hands Meeting 2003*, pages 223–226, Nottingham, UK, 2003.
- [85] M. Greenwood, C. Wroe, R. Stevens, and C.A. Goble. Are bioinformaticians doing e-Business? In B. Matthews, B. Hopgood, and M. Wilson, editors, *Euroweb 2002*, pages 3–26, Oxford, UK Hopgood, B. Wilson, M.23-26, 2002.
- [86] P. Groth and Y. Gil. Analyzing the gap between workflows and their natural language descriptions. In L.J. Zhang, editor, *SERVICES 2009 (Part I)*, pages 299–305, Los Angeles, USA, 2009.

- [87] P. Groth, M. Luck, and L. Moreau. A protocol for recording provenance in service-oriented. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *8th International Conference On Principles of Distributed Systems (OPODIS 2004)*, pages 124–139, Grenoble, France, 2004.
- [88] P. Groth, S. Miles, and L. Moreau. PReServ: Provenance Recording for Service. In S.J. Cox, editor, *UK e-Science All Hands Meeting 2005*, pages 1–8, Nottingham, UK, 2005.
- [89] M. Hartung, T. Kirsten, and E. Rahm. Analyzing the evolution of life science ontologies and mappings. In A. Bairoch, S. Cohen-Boulakia, and C. Froidevaux, editors, *5th International Workshop on Data Integration in the Life Sciences*, pages 11–27, Evry, France, 2008. Springer Verlag.
- [90] Z. He, L. Wu, X. Li, M.W. Fields, and J. Zhou. Empirical establishment of oligonucleotide probe design criteria. *Applied and Environmental Microbiology*, 71(7):3753–3760., 2005.
- [91] M.L. Hekkelman and G. Vriend. MRS: a fast and compact retrieval system for biological data. *Nucleic Acids Research*, 33(Web Server Issue):766–769, 2005.
- [92] K. Henrick, Z. Feng, W.F. Bluhm, D. Dimitropoulos, J.F. Doreleijers, S. Dutta, J.L. Flippen-Anderson, J. Ionides, C. Kamada, E. Krissinel, C.L. Lawson, J.L. Markley, H. Nakamura, R. Newman, Y. Shimizu, J. Swaminathan, S. Velankar, J. Ory, E.L. Ulrich, W. Vranken, J. Westbrook, R. Yamashita, H. Yang, J. Young, M. Yousufuddin, and H.M. Berman. Remediation of the protein data bank archive. *Nucleic Acids Research*, 36(Database issue):D426–D433, 2008.
- [93] T. Hey and A.E. Trefethen. The UK e-Science core programme and the grid. *Future Generation Computer Systems*, 18(8):1017–1031, 2002.
- [94] T. Hey and A.E. Trefethen. e-Science and its implications. *Philosophical Transactions of the Royal Society*, 361(1809):1809–1825, 2003.
- [95] T.C. Hudson, A.E. Stapleton, and J.L. Brown. Codifying bioinformatics processes without programming. *Drug Discovery Today: Biosilico*, 2(4):164–169, 2004.
- [96] D. Hull, R. Stevens, P.W. Lord, C. Wroe, and C.A. Goble. Treating shimantic web syndrome with ontologies. In J. Domingue, L. Cabral, and E. Motta, editors, *First AKT workshop on Semantic Web Services (AKT-SWS04) KMi*, pages 1–4, The Open University, Milton Keynes, UK., 2004. Workshop proceedings CEUR-WS.org ISSN:1613-0073.
- [97] C.D. Hundhausen and J. Lee Brown. 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.
- [98] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.
- [99] S. Jablonski and C. Bussler. *Workflow management. Modeling concepts, architecture and implementation*. Thomson, London, UK, 1996.
- [100] B. Jayashree, P.T. Reddy, Y. Leeladevi, V. Crouch, J.H. and Mahalakshmi, H.K. Buhariwalla, K.E. Eshwar, E. Mace, R. Folksterma, S. Senthilvel, K. Varshney, R.K. and Seetha, R. Rajalakshmi, V.P. Prasanth, S. Chandra, L. Swarupa, P. SriKalyani, and D.A. Hoisington. Laboratory information management software for genotyping workflows: applications in high throughput crop genotyping. *BMC Bioinformatics*, 7(383):1–6, 2006.

- [101] K. Jensen. *Coloured Petri Nets (3 vols.)*. Springer Verlag, Berlin, DE, 1995 1992.
- [102] H. Jong de and A. Rip. The computer revolution in science: steps toward the realization of computer-supported discovery environments. *Artificial Intelligence*, 91(2):225–256., 1997.
- [103] G. Kahn and D.B. MacQueen. Coroutines and networks of parallel processes. In B. Gilchrist, editor, *Information Processing 77*, pages 993–998, Toronto, Canada, 1977.
- [104] E.C. Kaletas, H. Afsarmanesh, and L.O. Hertzberger. Modelling multi-disciplinary scientific experiments and information. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Computer and Information Sciences - ISCIS 2003*, pages 75–82, Antalya, Turkey, 2003.
- [105] M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic Acids Research*, 36(Database issue):D480–D484, 2008.
- [106] M.A. Kappler. Software for rapid prototyping in the pharmaceutical and biotechnology industries. *Current Opinion in Drug Discovery & Development*, 11(3):389–392, 2008.
- [107] E. Kawas, M. Senger, and M.D. Wilkinson. BioMoby extensions to the Taverna workflow management and enactment software. *BMC Bioinformatics*, 7(523):1–13, 2006.
- [108] D.B. Kell and S.G. Oliver. Here is the evidence, now what is the hypothesis? *BioEssays*, 26(1):99–105, 2004.
- [109] G. Keller, M. Nüttgens, and A.W. Scheer. Processmodellierung auf der grundlage ereignisgesteuerter processketten. Technical Report 89, Instituts für Wirtschaftsinformatik, University of Saarland, Saarbrücken, Germany, 1992.
- [110] W.J. Kent. BLAT—the BLAST-like alignment tool. *Genome Research*, 12(4):656–664, Apr 2002.
- [111] K. Kochut, J. Arnold, A.P. Sheth, J. Miller, E. Kraemer, B. Arpinar, and J. Cardoso. IntelliGEN: A distributed workflow system for discovering protein-protein interactions. *Distributed and Parallel Databases*, 13(1):43–72, 2003.
- [112] E.L. Koua and M.J. Kraak. A usability framework for the design and evaluation of an exploratory geovisualization environment. In E. Banissi, K. Börner, C. Chen, M. Dastbaz, G. Clapworthy, A. Failoa, E. Izquierdo, C. Maple, J. Roberts, C. Moore, A. Ursyn, and J. Zhang, editors, *Eighth International Conference on Information Visualisation (IV'04)*, pages 153–158, Parma, Italy, 2004. IEEE Computer Society Press.
- [113] N. Krishnakumar and A.P. Sheth. Managing heterogeneous multi-system tasks to support enterprise-wide operations. *Distributed and Parallel Databases*, 3(2):155–186, 1995.
- [114] O. Kulyk, R. Kosara, J. Urquiza-Fuentes, and I. Wassink. Human-centered aspects. In *Human-Centered Visualization Environments*, volume 4417 of *Lecture Notes in Computer Science*, chapter Human-Centered Aspects, pages 13–75. Springer, Human-centered aspects, 2007.

- [115] O. Kulyk, G.C. van der Veer, and E.M.A.G. van Dijk. Situational awareness support to enhance team-work in collaborative environments. In J. Abascal, I. Fajardo, and I. Oakley, editors, *ECCE '08: Proceedings of the 15th European conference on Cognitive ergonomics*, pages 1–5, New York, NY, USA, 2008. ACM.
- [116] O. Kulyk and I. Wassink. Getting to know bioinformaticians: Results of an exploratory user study. In E. Zudilova-Seinstra and T. Adriaansen, editors, *HCI 2006 Engage, Combining Visualisation and Interaction to Facilitate Scientific Exploration and Discovery*, pages 30–37, London, UK, 2006.
- [117] O. Kulyk, I. Wassink, P.E. van der Vet, G.C. van der Veer, and E.M.A.G. van Dijk. 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.
- [118] N. Kwasnikowska and J. Bussche. Mapping the NRC dataflow model to the Open Provenance Model. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Second International Provenance and Annotation Workshop (IPAW 2008)*, pages 3–16, Salt Lake City, UT, USA, 2008. Springer-Verlag.
- [119] D. Laforenza, R. Lombardo, M. Scarpellini, M. Serrano, F. Silvestri, and P. Faccioli. Biological experiments on the grid: A novel workflow management platform. In P. Kokol, V. Podgorelec, D. Mičetić-Turk, M. Zorman, and M. Verlič, editors, *CBMS '07: Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems*, pages 489–494, Washington, DC, USA, 2007. IEEE Computer Society Press.
- [120] A. Lanzén and T.M. Oinn. The Taverna interaction service: enabling manual interaction in workflows. *Oxford Bioinformatics*, 24(8):1118–1120, 2008.
- [121] M.A. Larkin, G. Blackshields, N.P. Brown, R. Chenna, P.A. Mcgettigan, H. McWilliam, F. Valentin, I.M. Wallace, A. Wilm, R. Lopez, J.D. Thompson, T.J. Gibson, and D.G. Higgins. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.
- [122] T.A. Lau and D.S. Weld. Programming by demonstration: an inductive learning formulation. In M. Maybury, P. Szekely, and C.G. Thomas, editors, *IUI '99: Proceedings of the 4th international conference on Intelligent user interfaces*, pages 145–152, New York, NY, USA, 1999. ACM Press.
- [123] P.A. Lawrence. *Workflow Handbook 1997*. Wiley, Chichester, 1997.
- [124] A.M. Lesk. *Introduction to Bioinformatics*. Oxford University Press, Oxford, UK, 2nd edition edition, 2005.
- [125] P. Li, J.I. Castrillo, G. Velarde, I. Wassink, S. Soiland-Reyes, S. Owen, D. Withers, T.M. Oinn, M.R. Pocock, C.A. Goble, S.G. Oliver, and D.B. Kell. Performing statistical analyses on quantitative data in Taverna workflows: an example using R and maxdBrowse to identify differentially-expressed genes from microarray data. *BMC Bioinformatics*, 9(334):1–38, 2008.
- [126] X. Liu, Y. Xiong, and E.A. Lee. The Ptolemy II framework for visual languages. In S. Levialdi, editor, *IEEE 2001 Symposium on Human Centric Computing Languages and Environments (HCC'01)*, pages 50–51, Tresa, Italy, 2001.
- [127] P.W. Lord, P. Alper, C. Wroe, and C.A. Goble. Feta: A light-weight architecture for user oriented semantic service discovery. In A. Gómez-Pérez and J. Euzenat, editors, *The Semantic Web: Research and Applications*, pages 17–31, Crete, Greece, 2005.

- [128] P.W. Lord, S. Bechhofer, M.D. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C.A. Goble, and L. Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In S.A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *3rd International Semantic Web Conference*, volume 3298, pages 350–364, Hiroshima, Japan, 2004.
- [129] B. Ludäscher, I. Altintas, C.H.D.J.E. Berkley, M. Jones, E.A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.
- [130] C. Macaulay, D. Sloan, X. Jiang, P. Forbes, S. Loynton, J.R. Swedlow, and P. Gregor. Usability and user-centered design in scientific software development. *IEEE Software*, 26(1):96–102, 2009.
- [131] W.J. MacMullen and S.O. Denn. Information problems in molecular biology and bioinformatics: Research articles. *Journal of the American Society for Information Science and Technology*, 56(5):447–456, 2005.
- [132] R.R. Mahaffey. *LIMS: Applied Information Technology for the Laboratory*. Van Nostrand Reinhold, New York, NY, USA, 1990.
- [133] M. Mahoui, L. Lu, N. Gao, N. Li, J. Chen, O. Bukhres, and Z.B. Miled. A dynamic workflow approach for the integration of bioinformatics services. *Cluster Computing*, 8(4):279–291, 2005.
- [134] S. Majithia, M.S. Shields, I.J. Taylor, and I. Wang. Triana: A graphical web service composition and execution toolkit. In H. Jain and L. Liu, editors, *IEEE International Conference on Web Services (ICWS'04)*, pages 514–521, San Diego, California, USA, 2004. IEEE Computer Society Press.
- [135] C.B. Medeiros, G. Vossen, and M. Weske. WASA: A workflow-based architecture to support scientific database applications. In N. Revell and A.M. Tjoa, editors, *Proceedings of the 6th Database and Expert Systems Applications*, pages 574–583, London, United Kingdom, 1995.
- [136] K. Michalickova, G.D. Bader, M. Dumontier, H. Lieu, D. Betel, R. Isserlin, and C.W.V. Hogue. SeqHound: biological sequence and structure database as a platform for bioinformatics research. *BMC Bioinformatics*, 3(32):1–13, 2002.
- [137] S. Miles, S.C. Wong, W.C. Fang, P. Groth, KP. Zauner, and L. Moreau. Provenance-based validation of e-Science experiments. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):28–38, 2007.
- [138] L. Moreau, J. Freire, J. Futrelle, R. Mcgrath, J. Myers, and P. Paulson. The Open Provenance Model: An overview. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Second International Provenance and Annotation Workshop (IPAW 2008)*, pages 323–326, Salt Lake City, UT, USA, 2008.
- [139] L. Moreau, B. Plale, S. Miles, C.A. Goble, P. Missier, R.S. Barga, Y.L. Simmhan, J. Futrelle, R. McGrath, J. Myers, P. Paulson, S. Bowers, B. Ludäscher, N. Kwasnikowska, J. van den Bussche, T. Ellkvist, J. Freire, and P. Groth. The Open Provenance Model (v1.01). Technical report, University of Southampton, 2008.
- [140] H. Morisawa, M. Hirota, and T. Toda. Development of an open source laboratory information management system for 2-D gel electrophoresis-based proteomics workflow. *BMC Bioinformatics*, 7(430):1–11, 2006.

- [141] P.B.T. Neerincx. *Web services for Transcriptomics*. PhD thesis, University of Wageningen, 2009.
- [142] P.B.T. Neerincx and J.A.M. Leunissen. Evolution of web services in bioinformatics. *Briefings in Bioinformatics*, 6(2):178–188, 2005.
- [143] P.B.T. Neerincx, H. Rauwerda, H. Nie, M.A.M. Groenen, T.M. Breit, and J.A.M. Leunissen. OligoRAP - an Oligo Re-Annotation Pipeline to improve annotation and estimate target specificity. *BMC Proceedings*, 3(Suppl 4):S4, 2009.
- [144] A. Ngu, S. Bowers, N. Haasch, T.M. McPhillips, and T. Critchlow. Flexible scientific workflow modeling using frames, templates, and dynamic embedding. In *Scientific and Statistical Database Management*, pages 566–572. Springer Verlag, 2008.
- [145] T.M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, C.A. Goble, A. Wipat, P. Li, and T. Carver. Delivering web service coordination capability to users. In S. Feldman and M. Uretsky, editors, *Proceedings of the 13th international conference on World Wide Web*, pages 438–439, New York, NY, USA, 2004. ACM Press.
- [146] T.M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Oxford Bioinformatics*, 20(17):3045–3054, 2004.
- [147] T.M. Oinn, M. Greenwood, M. Addis, M.N. Alpdemir, J. Ferris, K. Glover, C.A. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P.W. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat, , and C. Wroe. Taverna: Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience Grid Workflow*, 18(10):1067–1100, 2006.
- [148] T.M. Oinn, P. Li, D.B. Kell, C.A. Goble, A. Goderis, M. Greenwood, D. Hull, R. Stevens, D. Turi, and J. Zhao. Taverna/myGrid: Aligning a workflow system with the life sciences community. In *Workflows for e-Science*, chapter Taverna/myGrid: Aligning a Workflow System with the Life Sciences Community, pages 300–319. Springer Verlag, Berlin, DE, 2007.
- [149] C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Science of Computer Programming archive*, 67(2-3):162–198, 2007.
- [150] C. Pautasso. JOpera: Visual composition of grid services. *ERCIM News*, pages 46–47, 2004.
- [151] C. Pautasso and G. Alonso. The JOpera visual composition language. *Journal of Visual Languages and Computing*, 16(1–2):119–152, 2005.
- [152] George Pavlou. On the evolution of management approaches, frameworks and protocols: A historical perspective. *Journal of Network and System Management*, 15(4):425–445, 2007.
- [153] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, 1962.
- [154] S. Pettifer, D. Thorne, P. McDermott, T. Attwood, J. Baran, J. C. Bryne, T. Hupponen, D. Mowbray, and G. Vriend. An active registry for bioinformatics web services. *Bioinformatics*, 25(16):2090–2091, 2009.

- [155] S. Pillai, V. Silventoinen, K. Kallio, M. Senger, S. Sobhany, J. Tate, S. Velankar, A. Golovin, K. Henrick, P. Rice, P. Stoehr, and R. Lopez. SOAP-based services provided by the European Bioinformatics Institute. *Nucleic Acids Research*, 33(Web Server issue):25–28, 2005.
- [156] C. Prior. Workflow and process management. In L. Fischer, editor, *The Workflow Handbook 2003*, pages 17–25. Future Strategies Inc., Alameda, CA, 2003.
- [157] M.C. Puerta Melguizo, C. Chisalita, and G.C. Van der Veer. Assessing users mental models in designing complex systems. In A. El Kamel, K. Mellouli, and P. Borne, editors, *IEEE International Conference on Systems, Man and Cybernetics*, pages 1–6, Yasmine Hammamet, Tunisia, 2002.
- [158] H. Rauwerda, M. Roos, L.O. Hertzberger, and T.M. Breit. The promise of a virtual lab in drug discovery. *Drug Discovery Today*, 11(5-6):228–236, 2006.
- [159] M. Reichert and P. Dadam. Enabling adaptive process-aware information systems with ADEPT2. In *Research on Business Process Modeling*, pages 173–203. Information Science Reference, 2009.
- [160] W. Reisig. *A Primer in Petri Net Design*. Springer Verlag, Berlin, DE, 1992.
- [161] T. Reuss, G. Vossen, and M. Weske. Modeling samples processing in laboratory environments as scientific workflows. In R.R. Wagner, editor, *8th International Workshop on Database and Expert Systems Applications (DEXA '97)*, pages 49–54, Washington, DC, USA, 1997. IEEE Computer Society Press.
- [162] R. Robbes and M. Lanza. How program history can improve code completion. In P. Inverardi, A. Ireland, and W. Visser, editors, *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008)*, pages 317–326, Aquila, Italy, 2008.
- [163] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and F.J. Fidge. Workflow simulation for operational decision support. *Data and Knowledge Engineering*, 68(9):834–850, 2009.
- [164] M. Rusinkiewicz and A.P. Sheth. Specification and execution of transactional workflows. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond*, chapter Specification and execution of transactional workflows, pages 592–620. ACM Press/Addison-Wesley Publishing Co., Boston, USA, 1995.
- [165] L.H. Saal, C. Troein, J. Vallon-Christersson, S. Gruvberger, A. Borg, and C. Peterson. Bioarray software environment (BASE): a platform for comprehensive management and analysis of microarray data. *Genome biology*, 3(8):software 1–6, 2002.
- [166] P. Saraiya, C. North, and K. Duca. An evaluation of microarray visualization tools for biological insight. In M. Ward and T. Munzer, editors, *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 1–8, Washington, DC, USA, 2004. IEEE Computer Society Press.
- [167] M.H. Schonenberg, R.S. Mans, N.C. Russell, N.A. Mulyar, and W.M.P. van der Aalst. Towards a taxonomy of process flexibility (extended version). Technical Report BPM-07-11, BPM Center, 2007.
- [168] P.N. Seibel, J. Krüger, S. Hartmeier, K. Schwarzer, K. Löwenthal, H. Mersch, T. Dandekar, and R. Giegerich. XML schemas for common bioinformatic data types and their application in workflow systems. *BMC Bioinformatics*, 7(490):1–11, 2006.

- [169] S. Seltzsam, R. Holzhauser, and A. Kemper. Semantic caching for web services. In B. Benatallah, F. Casati, and P. Traverso, editors, *Service-Oriented Computing, ICSOC 2005*, volume 3826 of *LNCS*, pages 324–340, Amsterdam, The Netherlands, 2005. Springer.
- [170] M. Senger, P. Rice, and T.M. Oinn. SoapLab - a unified sesame door to analysis tools proceedings. In S.J. Cox, editor, *Proceedings of the UK e-Science- All Hands Meeting 2003*, pages 509–513, Nottingham, UK, 2003.
- [171] Q. Shao, P. Sun, and Y. Chen. Efficiently discovering critical workflows in scientific explorations. *Future Generation Computer Systems*, 25(5):577–585, 2009.
- [172] X. Shi. Semantic web services: An unfulfilled promise. *IT Professional*, 9(4):42–45, 2007.
- [173] M.S. Shields. Control- versus data-driven workflows. In I.J. Taylor, E. Deelman, D.B. Gannon, and M.S. Shields, editors, *Workflows for e-Science*, pages 258–275. Springer Verlag, Berlin, DE, 2007.
- [174] J. Shon, H. Ohkawa, and J. Hammer. Scientific workflows as productivity tools for drug discovery. *Current Opinion in Drug Discovery & Development*, 11(3):381–388, 2008.
- [175] Y.L. Simmhan, B. Plale, and D.B. Gannon. A survey of data provenance in e-Science. *ACM SIGMOD Record*, 34(3):31–36, 2005.
- [176] L.N. Soldatova and R.D. King. Are the current ontologies in biology good ontologies? *Nature Biotechnology*, 23(9):1095–1098, 2005.
- [177] Y.C. Song, E. Kawas, B.M. Good, M.D. Wilkinson, and S.J. Tebbutt. DataBiNS: a BioMoby-based data-mining workflow for biological pathways and non-synonymous SNPs. *Oxford Bioinformatics*, 23(6):780–782, 2007.
- [178] J.F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, 1999.
- [179] L. Stein. Creating a bioinformatics nation. *Nature*, 417(6885):119–120, 2002.
- [180] C. Steinbeck, Y. Q. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E.L. Willighagen. The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *Journal of Chemical Information and Computer Sciences*, 43(2):493–500, 2003.
- [181] C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha, and E.L. Willighagen. Recent developments of the Chemistry Development Kit (CDK) - an open-source Java library for chemo- and bioinformatics. *Current pharmaceutical design*, 12(17):2111–2120, 2006.
- [182] D. Stekel. *Microarray Bioinformatics*. Cambridge University Press, Cambridge, USA, 2003.
- [183] R. Stevens, C.A. Goble, P.G. Baker, and A. Brass. A classification of tasks in bioinformatics. *Bioinformatics*, 17(2):180–188, 2001.
- [184] R. Stevens, C.A. Goble, and S. Bechhofer. Ontology-based knowledge representation for bioinformatics. *Briefings in Bioinformatics*, 1(4):398–414, 2000.

-
- [185] H. Sundholm, H. Artman, and R. Ramberg. Backdoor creativity - collaborative creativity in technology supported teams. In *Cooperative systems design: Scenario-based design of collaborative systems*, pages 99–114. IOS press, 2004.
- [186] M.A. Swertz and R.C. Jansen. Beyond standardization: dynamic software infrastructures for systems biology. *Nature R*, 8(3):235–243, 2007.
- [187] I. Taylor, M.S. Shields, I. Wang, and A. Harrison. The Triana workflow environment: Architecture and applications. In I.J. Taylor, E. Deelman, D.B. Gannon, and M.S. Shields, editors, *Workflows for e-Science*, pages 320–339. Springer Verlag, Berlin, DE, 2007.
- [188] The BioMoby Consortium. Interoperability with Moby 1.0–It’s better than sharing your toothbrush! *Briefings in Bioinformatics*, 9(3):220–231, 2008.
- [189] The Gene Ontology Consortium. Creating the Gene Ontology Resource: Design and Implementation. *Genome Research*, 11(8):1425–1433, 2001.
- [190] The UniProt Consortium. The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 36(Database issue):D190–D195, 2008.
- [191] The Workflow Management Coalition. Workflow process definition interface – XML process definition language. Technical Report WPMC-TC-1025, The Workflow Management Coalition, 2002.
- [192] S. Thew, A.G. Sutcliffe, R. Procter, O. de Bruijn, J. McNaught, C.C. Venters, and I. Buchan. Requirements engineering for e-Science: Experiences in epidemiology. *IEEE Software*, 26(1):80–87, 2009.
- [193] J.D. Thompson, T.J. Gibson, F. Plewniak, F. Jeanmougin, and D.G. Higgins. The CLUSTAL X windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, 25(24):4876–4882, 1997.
- [194] J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.
- [195] A. Tiwari and A.K.T. Sekhar. Workflow based framework for life science informatics. *Computational Biology and Chemistry*, 2007(31):305–319, 2007.
- [196] A. Tomasic, R.M. McGuire, and B. Myers. Workflow by example: Automating database interactions via induction. Technical Report CMU-ISRI-06-103, Carnegie Mellon University, 2006.
- [197] D. Turi, P. Missier, C.A. Goble, D. De Roure, and T.M. Oinn. Taverna workflows: Syntax and semantics. In G. Fox, K. Chiu, and R. Buyya, editors, *IEEE International Conference on e-Science and Grid Computing*, pages 441–448, Bangalore, India, 2007.
- [198] S. Urbanek. Rserve – a fast way to provide R functionality to applications. In K. Hornik, F. Leisch, and A. Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, pages 20–22, Vienna, Austria, 2003.
- [199] W.M.P. van der Aalst. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

- [200] W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and implementation of the YAWL system. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *16th International Conference on Advanced Information Systems Engineering (CAiSE'04)*, pages 142–159, Riga, Latvia, 2004. Springer-Verlag.
- [201] W.M.P. van der Aalst and A.H.M. ter Hofstede. Workflow patterns: On the expressive power of (Petri-net-based) workflow languages. In K. Jensen, editor, *Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002)*, pages 1–20, Aarhus, Denmark, 2002. DAIMI.
- [202] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
- [203] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods and Systems*. The MIT Press, 2002.
- [204] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods and Systems*. The MIT Press, 2004.
- [205] P.E. van der Vet, O. Kulyk, I. Wassink, F.W. Fikkert, H. Rauwerda, E.M.A.G. van Dijk, G.C. van der Veer, T.M. Breit, and A. Nijholt. Smart environments for collaborative design, implementation, and interpretation of scientific experiments. In T. Huang, A. Nijholt, M. Pantic, and A. Pentland, editors, *Workshop on AI for Human Computing (AI4HC)*, pages 79–86, Hyderabad, India, 2007.
- [206] K.M. van Hee, A. Serebrenik, N. Sidorova, and M. Voorhoeve. Soundness of resource-constrained workflow nets. In G. Ciardo and P. Darondeau, editors, *Applications and theory of Petri nets 2005*, pages 250–267, Miami, USA, 2005. Springer.
- [207] K.M. van Hee, N. Sidorova¹, and M. Voorhoeve. Resource-constrained workflow nets. *Fundamenta Informaticae*, 71(2–3):243–257, 2006.
- [208] B. P. Vandervalk, E.L. McCarthy, and M.D. Wilkinson. Moby and Moby 2: Creatures of the Deep (Web). *Briefings in Bioinformatics*, 10(2):114–128, 2009.
- [209] I.P. Vélez and B. Vélez. Lynx: an open architecture for catalyzing the deployment of interactive digital government workflow-based systems. In J.A. B. Fortes and A. Macintosh, editors, *DGO'06. Proceedings of the 2006 International Conference on Digital Government Research*, pages 309–318, New York, NY, USA, 2006. ACM.
- [210] K.K. Verdi, H.J.C. Ellis, and M.R. Gryk. Conceptual-level workflow modeling of scientific experiments using NMR as a case study. *BMC Bioinformatics*, 8(31):1–12, 2007.
- [211] K. Verma and A.P. Sheth. Using SAWSDL for semantic service interoperability. World Wide Web electronic publication, 2007.
- [212] J. Wainer, M. Weske, G. Vossen, and C.B. Bauzer Medeiros. Scientific workflow systems. In *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems*, pages 1–5, Athens, Georgia, 1997.
- [213] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, San Francisco, USA, 1999.

- [214] C. Ware, D. Christopher, and J.G. Hollands. *Information Visualization: Perception for Design*, volume 2. Morgan Kaufmann, San Fransisco, CA, USA, 2004.
- [215] I. Wassink, O. Kulyk, E.M.A.G. van Dijk, G.C. van der Veer, and P.E. van der Vet. Applying a user-centred approach to interactive visualization design. In *Trends in Interactive Visualization*. Springer Verlag, 2008.
- [216] I. Wassink, M.J. Ooms, and P.E. van der Vet. Designing workflows on the fly using e-BioFlow. In L. Baresi, CH. Chi, and J. Suzuki, editors, *International Conference on Service-Oriented Computing (ICSOC-ServiceWave 2009)*, pages 395–409, Stockholm, Sweden, 2009.
- [217] I. Wassink, H. Rauwerda, P.B.T. Neerincx, P.E. van der Vet, T.M. Breit, J.A.M. Leunissen, and A. Nijholt. Using R in Taverna: RShell v1.2. *BMC Research Notes*, 2(138):1–8, 2009.
- [218] I. Wassink, H. Rauwerda, P.E. van der Vet, T.M. Breit, and A. Nijholt. e-BioFlow: Different perspectives on scientific workflows. In M. Elloumi, Josef Küng, Michal Linial, Robert F. Murphy, Kristan Schneider, and Christian Toma, editors, *2nd International Conference on Bioinformatics Research and Development (BIRD)*, pages 243–257, Vienna, Austria, 2008.
- [219] I. Wassink, P.E. van der Vet, E.M.A.G. van Dijk, G.C. Veer, and M. Roos. New Interactions with Workflow Systems. In L. Norros, H. Koskinen, L. Salo, and P. Savioja, editors, *European Conference on Cognitive Ergonomics 2009 (ECCE 2009)*, pages 349–352, Helsinki, Finland, 2009.
- [220] I. Wassink, P.E. van der Vet, K. Wolstencroft, P.B.T. Neerincx, M. Roos, H. Rauwerda, and T.M. Breit. Analysing scientific workflows: why workflows not only connect web services. In L.J. Zhang, editor, *SERVICES 2009 (Part I)*, pages 314–321, Los Angeles, USA, 2009.
- [221] I. Wassink, E.M.A.G. van Dijk, J. Zwiers, A. Nijholt, J. Kuipers, and A.O. Brugman. Bringing hollywood to the driving school: Dynamic scenario generation in simulations and games. In O. Stock, W. Wolfgang, and M. Maybury, editors, *INTETAIN 2005*, volume 3814/2005, pages 288–292, Madonna di Campiglio, Italy, 2005. Springer-Verlag GmbHV.
- [222] I. Wassink, E.M.A.G. van Dijk, J. Zwiers, A. Nijholt, J. Kuipers, and A.O. Brugman. In the truman show: Generating dynamic scenarios in a driving simulator. *IEEE Intelligent Systems*, 21(5):28–32, 2006.
- [223] S.C. Webb, A. Attwood, T. Brooks, T. Freeman, P. Gardner, C. Pritchard, D. Williams, P. Underhill, M. Strivens, A. Greenfield, and E. Pilicheva. LIMaS: the JAVA-based application and database for microarray experiment tracking. *Mammalian Genome*, 15(9):740–747, 2004.
- [224] P. Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91, 1997.
- [225] D.L. Wheeler, T. Barrett, D.A. Benson, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church, M. Dicuccio, R. Edgar, S. Federhen, M. Feolo, L.Y. Geer, W. Helmberg, Y. Kapustin, O. Khovayko, D. Landsman, D.J. Lipman, T.L. Madden, D.R. Maglott, V. Miller, J. Ostell, K.D. Pruitt, G.D. Schuler, M. Shumway, E. Sequeira, S.T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R.L. Tatusov, T.A. Tatusova, L. Wagner, and E. Yaschenko. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 36(Supplement 1):D13–D21, 2008.

- [226] P. Whetzel, H. Parkinson, H.C. Causton, L. Fan, J. Fostel, G. Fragoso, L. Game, M. Heiskanen, P. Morrison, N. Rocca-Serra, S.A. Sansone, C. Taylor, J. White, and C. Stoeckert. The MGED ontology: a resource for semantics-based description of microarray experiments. *Bioinformatics*, 22(7):866–873, 2006.
- [227] T.E. White and L. Fischer. *New tools for new times : the workflow paradigm : the impact of information technology on business process reengineering*. Future Strategies, Alameda, CA, 1994.
- [228] K.N. Whitley. Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages & Computing*, 8(1):109–142, 1997.
- [229] A. Wibisono, V. Korkhov, D. Vasunin, V. Guevara-Masis, A.S.Z. Belloum, C.T.A.M. de Laat, P. Adriaans, and L.O. Hertzberger. WS-VLAM: Towards a scalable workflow system on the grid. In E. Deelman and I.J. Taylor, editors, *Proceeding of the 16th IEEE International Symposium on High Performance Distributed Computing*, pages 63–68, Monterey Bay, California, USA, 2007.
- [230] A. Wibisono, D. Vasyunin, V. Korkhov, Z. Zhao, A.S.Z. Belloum, C.T.A.M. de Laat, P. Adriaans, and L.O. Hertzberger. WS-VLAM: A GT4 based workflow management system. In Y. Shi, G.D. van Albada, J. Dongarra, and P.M.A. Sloot, editors, *Proceedings of the 2nd International Workshop on Workflow systems in e-Science (WSES07)*, pages 191–198, Beijing, China, 2007.
- [231] A. Wiggins, J. Howison, and K. Crowston. Social dynamics of floss team communication across channels. In *Open Source Development, Communities and Quality*, pages 131–142. Springer, 2008.
- [232] M.D. Wilkinson. Gbrowse Moby: a web-based browser for biomoby services. *Source Code for Biology and Medicine*, 1(4):1–8, 2006.
- [233] M.D. Wilkinson, D. Gessler, A. Farmer, and L. Stein. The BioMOBY project explores open-source, simple, extensible protocols for enabling biological database interoperability. In *Proceedings of the Virtual Conference on Genomics and Bioinformatics*, pages 16–26, 2003.
- [234] M.D. Wilkinson and M. Links. BioMOBY: an open source biological web services proposal. *Briefings in Bioinformatics*, 3(4):331–341, December 2002.
- [235] M.D. Wilkinson, H. Schoof, R. Ernst, and D. Haase. BioMOBY successfully integrates distributed heterogeneous bioinformatics web services. the PlaNet exemplar case. *Plant Physiol*, 138(1):5–17, 2005.
- [236] L.G. Wilming, J.G.R. Gilbert, K. Howe, S. Trevanion, T. Hubbard, and J.L. Harrow. The Vertebrate Genome Annotation (Vega) database. *Nucleic Acid Res*, 33(Database issue):D459–D465, 2008.
- [237] S.C. Wong, S. Miles, W.C. Fang, P. Groth, and L. Moreau. Provenance-based validation of e-Science experiments. In Y. Gil, E. Motta, V.R. Benjamins, and M.A. Musen, editors, *4th International Semantic Web Conference, ISWC*, volume 3729 of LNCS, pages 801–815, Galway, Ireland, 2005. Springer.
- [238] C. Wroe, C.A. Goble, A. Goderis, P.W. Lord, S. Miles, J. Papay, P. Alper, and L. Moreau. Recycling workflows and services through discovery and reuse. *Concurrency and Computation: Practice and Experience*, 19(2):181–194, 2007.
- [239] C. Wroe, C.A. Goble, M. Greenwood, P.W. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau. Automating experiments using semantic data on a bioinformatics grid. *IEEE Intelligent Systems*, 19(1):48–55, 2004.

- [240] X. Xiang and G. Madey. Improving the reuse of scientific workflows and their by-products. In LJ. Zhang, K.P. Birman, and J. Zhang, editors, *IEEE International Conference on Web Services (ICWS 2007)*, pages 792–799, Salt Lake City, USA, 2007.
- [241] J. Zhao, C.A. Goble, and R. Stevens. Semantically linking and browsing provenance logs for e-Science. In M. Bouzeghoub, C.A. Goble, V. Kashyap, and S. Spaccapietra, editors, *1st International Conference on Semantics of a Networked World*, pages 158–176, Paris, France, 2004. Springer.
- [242] J. Zhao, C.A. Goble, R. Stevens, and D. Turi. Mining Taverna’s semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 20(5):463–472, 2008.
- [243] J. Zhao, C. Wroe, C.A. Goble, R. Stevens, D. Quan, and M. Greenwood. Using semantic web technologies for representing e-Science provenance. In S.A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Third International Semantic Web Conference on The Semantic Web (ISWC 2004)*, volume 3298s, pages 92–106, Hiroshima, Japan, 2004.
- [244] Y. Zhao, M. Wilde, and I. Foster. Virtual data language: A typed workflow notation for diversely structured scientific data. In *Workflows for e-Science*, pages 258–275. Springer Verlag, Berlin, 2007.
- [245] Z. Zhao, A.S.Z. Belloum, A. Wibisono, F. Terpstra, P.T. de Boer, P.M.A. Sloot, and L.O. Herzberger. Scientific workflow management: between generality and applicability. In KY. Cai, A. Ohnishi, and M.F. Lau, editors, *International Workshop on Grid and Peer-to-Peer based Workflows in conjunction with the 5th International Conference on Quality Software*, pages 357–364, Melbourne, Australia, 2005.

Abstract

The introduction of computer science technology in the life science domain has resulted in a new life science discipline called bioinformatics. Bioinformaticians are biologists who know how to apply computer science technology to perform computer based experiments, also known as in-silico or dry lab experiments. Various tools, such as databases, web applications and scripting languages, are used to design and run in-silico experiments. As the size and complexity of these experiments grow, new types of tools are required to design and execute the experiments and to analyse the results. Workflow systems promise to fulfill this role. The bioinformatician composes an experiment by using tools and web services as building blocks, and connecting them, often through a graphical user interface. Workflow systems, such as Taverna, provide access to up to a few thousand resources in a uniform way. Although workflow systems are intended to make the bioinformaticians' work easier, bioinformaticians experience difficulties in using them. This thesis is devoted to find out which problems bioinformaticians experience using workflow systems and to provide solutions for these problems.

This thesis consists of three parts. The first part discusses the daily working practices of bioinformaticians and the infrastructure they use to perform their computer based experiments. Within a single in-silico experiment, often scientists from different disciplines and organisations are involved. The collaboration takes place in different forms, ranging from working at the same working place to sharing knowledge by means of publications as well as tools and data. The collaborating scientists have different experience levels in using computer tools. The bioinformatician is the expert. She knows how to use life science tools, how to program and how to connect different tools.

The multidisciplinary collaborative work is reflected by the life science infrastructure. Bioinformaticians construct and share databases and tools and reuse those created by

others. Many tools are currently available as web services. The bioinformatician generates scripts to access and connect such web services.

In the second part of this thesis, the difficulties bioinformaticians have using workflow systems are analysed and discussed. By using a workflow system, the bioinformatician should be able to easily construct the experiment without programming. This is, however, an idealistic view on workflow systems. Workflow systems do not support the explorative research approach the bioinformatician normally uses. Data are sources of inspiration for the bioinformatician and are used to determine the next steps in the experiment. In traditional workflow systems, the bioinformatician needs to design the entire workflow in advance before it can be run. She therefore has to make many of the design decisions without appropriate data.

Another issue many workflow designers face is solving data incompatibility problems. Different organisations often use different data structures in their services, even to represent the same information. This results in a situation where about 30% of the tasks in the Taverna workflows stored at myExperiment represent data transformations. Standardising on data formats will be the best solution, but is infeasible, because if people are free to use their own data format they will. It would be better if a workflow system provides the means to handle data transformations. It should support scripting tasks for various programming languages, to enable the bioinformatician to program in the language he is familiar with. Additionally, a workflow can provide tasks to automatically compose and decompose complex data structures. Furthermore, the workflow system can suggest tasks that produce compatible output or that can consume the data available.

Once finished, the workflow model is a knowledge representation of an experiment, that can easily be shared with peers, for validation or to construct similar experiments. Due to portals such as myExperiment, workflow sharing has become popular. These portals, however, have introduced a new type of problem bioinformaticians have to deal with. The services used in a workflow can become unavailable. Services may be down or moved to another location or may have a changed interface. The workflow (re)user has no influence on the existence of services, because the services are often hosted by other organisations. As a result, at the time of writing approximately one out of ten of the Taverna workflows at myExperiment are broken. In order to reuse these workflows, the tasks representing these dead services need to be replaced. Workflow systems that support late binding could solve these problems, in case the service is moved to another location. The bioinformatician will not even notice the service has been moved. In all other situations, she has to replace the broken service with an alternative.

In the third part, we discuss our design solutions realised in our workflow system e-

BioFlow. The system we propose supports the explorative working approach of the bioinformatician. It combines aspects from both a data flow and control flow system and therefore is what is called a hybrid workflow system. It supports more control flow patterns than many existing workflow systems for bioinformatics. Examples of these patterns are conditional branching and iteration through loops.

Additionally, it supports late binding: the workflow designer can abstract from real resources. The engine performs the actual task resource binding at enactment time. This way, workflows designed in e-BioFlow are independent of the location of the resources at design time. e-BioFlow's provenance system interacts with the engine. It automatically captures the data produced during a workflow run and saves it in an Open Provenance Model compatible file format. Using this open standard, scientists can easily share their experiment results with peers for inspection and validation. The provenance archive can also be used as cache to speed up future executions of tasks. The provenance system provides an interactive provenance browser and a query interface to explore and query provenance data.

e-BioFlow provides an ad-hoc workflow editor to enable the bioinformatician to design and execute unfinished workflows. The main advantages are: data are explicitly present in the workflow and the workflows can be constructed step by step. The workflow editor helps the workflow designer by suggesting compatible tasks, not only to prevent data incompatibility problems, but also as a source of inspiration. In this interface, the workflow designer can use the explorative research approach. By means of a mockup implementation, we have presented our design ideas to 50 life scientists in an early design stage. Most participants were enthusiastic about the new interface and expected it to be much easier to use than traditional workflow editor interfaces.

Workflow systems can fit in the explorative research of bioinformaticians. These systems can help bioinformaticians to design and run their experiments and to automatically capture and store the data generated at runtime. A next challenge will be an interface that brings workflow design closer to the conceptual model bioinformaticians have of an experiment. Bioinformaticians do not think in terms of web services, but in terms of actions they want to perform on the data. The workflow system is responsible for mapping these higher level actions to the services available. Such a workflow system will be much easier in use and will better suit the bioinformaticians' needs.

Samenvatting

De introductie van informatie- en communicatietechnologie in de levenswetenschappen heeft geleid tot een nieuwe discipline binnen de levenswetenschappen genaamd bioinformatica. De bioinformaticus is een bioloog die weet hoe hij computertechnieken moet gebruiken om digitaal biologische experimenten uit te voeren. Deze experimenten worden ook wel in-silico experimenten genoemd. Verschillende gereedschappen, zoals gegevensbanken, webapplicaties en scripttalen, worden gebruikt om deze in-silico experimenten te ontwerpen en uit te voeren. Doordat de omvang en complexiteit van deze experimenten groeien, zijn er nieuwe gereedschappen nodig bij het ontwerpen en uitvoeren van deze experimenten, en bij het analyseren van de resultaten. Workflowsystemen beloven deze rol te vervullen. De bioinformaticus kan een experiment ontwerpen door gereedschappen en webdiensten te gebruiken als bouwblokken en deze te verbinden, vaak met behulp van een grafische gebruikersinterface. Workflowsystemen, zoals Taverna, bieden toegang tot duizenden bronnen op een uniforme manier. Alhoewel workflowsystemen bedoeld zijn om het werk van de bioinformatici te vereenvoudigen, ondervinden deze bioinformatici moeilijkheden in het gebruik ervan. Dit proefschrift richt zich op de vraag waarom bioinformatici deze problemen ervaren, en op het vinden van oplossingen hiervoor.

Dit proefschrift bestaat uit drie delen. Het eerste deel behandelt de dagelijkse werkzaamheden van bioinformatici en de infrastructuur die ze gebruiken bij het uitvoeren van hun computergebaseerde experimenten. Binnen een in-silico experiment zijn vaak wetenschappers van verschillende organisaties betrokken. De samenwerking kan op verschillende manieren plaatsvinden, van werken in dezelfde werkruimte tot het delen van kennis door middel van publicaties, maar ook het delen van gereedschappen en gegevens. Deze wetenschappers verschillen in ervaring met het gebruik van bioinformatica-

gereedschappen. De bioinformaticus is de expert. Hij weet hoe hij deze gereedschappen moet gebruiken. Daarnaast kan hij programmeren en weet hij hoe hij verschillende gereedschappen kan verbinden.

De multidisciplinaire samenwerking is terug te vinden in de infrastructuur. Bioinformatici bouwen en delen gegevensbanken en gereedschappen, en gebruiken die van anderen. Veel van deze gereedschappen zijn tegenwoordig beschikbaar als webdiensten. De bioinformaticus gebruikt scripts om deze webdiensten aan te spreken en te verbinden.

In het tweede deel worden de problemen behandeld die bioinformatici ondervinden bij het gebruik van workflowsystemen. Een workflowsysteem zou de bioinformaticus in staat moeten stellen om eenvoudig experimenten te ontwerpen zonder te programmeren. Dit is echter een idealistische kijk op workflowsystemen. Deze systemen ondersteunen niet de exploratieve werkwijze die bioinformatici normaliter toepassen. Gegevens vormen bronnen van inspiratie en worden normaliter gebruikt bij het bepalen van de volgende stap in een experiment. In traditionele workflowsystemen moet de bioinformaticus echter de hele workflow vooraf definiëren. Pas dan kan hij de workflow laten uitvoeren. Hierdoor moet hij veel ontwerpbeslissingen nemen zonder de gegevens tot zijn beschikking te hebben.

Een ander probleem waar workflowontwerpers mee te maken hebben is gegevensincompatibiliteitsproblemen. Verschillende organisaties gebruiken vaak verschillende gegevensstructuren in hun webdiensten, zelfs om dezelfde informatie te representeren. Dit heeft geleid tot een situatie waarin ongeveer 30% van de taken in de Taverna-workflows op myExperiment gegevenstransformaties representeren. Standaardiseren op gegevensformaten zou de beste oplossing zijn, maar is niet haalbaar, want als mensen vrij zijn om hun eigen gegevensformaten te gebruiken, dan gebeurt dat ook. Een betere oplossing is een workflowsysteem dat middelen aanbiedt voor het uitvoeren van deze gegevenstransformaties. Zo zou een workflowsysteem scripttaken moeten ondersteunen. Het liefst voor verschillende talen, zodat de bioinformaticus taken kan programmeren in de taal die hij kent. Verder zou een workflowsysteem taken kunnen aanbieden die automatisch complexe gegevensstructuren kunnen bouwen en ontleden. Tenslotte kunnen workflowsystemen taken voorstellen die compatibele uitvoer produceren of die de beschikbare gegevens kunnen consumeren.

Als een workflowmodel ontworpen is, is het een kennisrepresentatie van een experiment dat eenvoudig gedeeld kan worden, voor validatie of voor een gelijksoortig experiment. Dankzij webpagina's als myExperiment is het delen van workflows populair geworden. Deze webpagina's hebben echter ook een nieuw probleem geïntroduceerd. De webdiensten die in workflows gebruikt worden kunnen ontoegankelijk worden. Web-

diensten kunnen verdwenen of verplaatst zijn of een nieuwe interface hebben gekregen. De (her)gebruiker van een workflow heeft vaak geen invloed op het bestaan van webdiensten, omdat deze vaak door andere organisaties aangeboden worden. Het resultaat is dat op het moment van schrijven één op de tien Taverna-workflows op myExperiment defect is. Om deze workflows te hergebruiken moeten de taken die ontoegankelijke webdiensten representeren vervangen worden. Workflowsystemen die *late binding* ondersteunen kunnen een oplossing bieden in het geval dat webdiensten verplaatst zijn. De gebruiker zou niet eens hoeven te merken dat een webdienst verplaatst is. In alle andere situaties moet hij nog steeds de webdienst vervangen door een alternatief.

In het derde deel behandelen we onze ontwerp oplossingen die we gerealiseerd hebben in ons workflowsysteem e-BioFlow. Het systeem dat wij voorstellen ondersteunt de exploratieve werkwijze van de bioinformaticus. Het combineert aspecten van zowel een dataflow- als een controlflowsysteem en is daardoor een zogenaamde hybride workflowsysteem. Het systeem ondersteunt meer controlflowpatronen dan veel bestaande workflowsystemen in de bioinformatica. Voorbeelden van deze controlflowpatronen zijn conditionele vertakkingen en iteratie door middel van lussen.

Verder ondersteunt e-BioFlow *late binding*: de workflowontwerper kan abstraheren van de te gebruiken bronnen. De workflowengine, oftewel het onderdeel dat workflows instantieert en draait, voert de daadwerkelijke koppeling tussen taken en bronnen uit. Op deze manier zijn de workflows ontworpen in e-BioFlow onafhankelijk van de locaties die de bronnen op het moment van ontwerpen hebben. e-BioFlow's provenancesysteem interacteert met de workflowengine voor het bijhouden van *provenance*. De gegevens, die gegenereerd worden tijdens de uitvoeren van een workflow, worden automatisch opgeslagen in het provenance-archief. Voor het opslaan gebruikt het provenancesysteem een bestandsformaat dat compatibel is met het Open Provenance Model. Door het gebruik van deze standaard kan men eenvoudig resultaten van experimenten delen met anderen voor inspectie en validatie. Het provenance-archief kan in e-BioFlow ook gebruikt worden als tijdelijke opslagplaats (cache) voor het versnellen van toekomstige uitvoering van taken. Het provenancesysteem heeft een interactieve provenanceverkenner en een zoek-interface. Deze kunnen gebruikt worden om de provenancegegevens in het provenance-archief te exploreren en te doorzoeken.

e-BioFlow beschikt over een ad-hoc workflowontwerpapplicatie, waarmee de bioinformaticus onvolledige workflows kan ontwerpen en uitvoeren. De grootste voordelen hiervan zijn: gegevens zijn expliciet aanwezig in de workflow en de workflow kan stap voor stap ontworpen worden. Deze workflowontwerpapplicatie helpt de workflowontwerper door compatibele taken voor te stellen. Dit helpt niet alleen om gegevensincompatibiliteitsproblemen te voorkomen, maar vormt tegelijkertijd een bron van inspi-

ratie voor het ontdekken van volgende stappen voor de workflow. De workflowontwerper kan door middel van deze nieuwe interface zijn exploratieve werkwijze toepassen in een workflowomgeving. We hebben onze ontwerpideeën in een vroeg stadium aan 50 levenswetenschappers voorgelegd door middel van een modelimplementatie (mockup). De meeste deelnemers waren enthousiast over de nieuwe interface en verwachtten dat het veel eenvoudiger te gebruiken is dan traditionele workflowinterfaces.

Workflowsystemen kunnen dus van dienst zijn in de exploratieve werkwijze van bio-informatici. Ze helpen bioinformatici bij het ontwerpen en het uitvoeren van hun experimenten en het automatisch binnenhalen en opslaan van de gegenereerde gegevens. Een volgende uitdaging is een interface te ontwerpen waarin een workflowmodel dicht bij het conceptuele model staat dat een bioinformaticus heeft van een experiment. Bioinformatici denken namelijk niet in termen van webdiensten, maar in termen van acties die ze willen uitvoeren. Een workflowsysteem zou in staat moeten zijn om deze hogere-orde acties te kunnen vertalen naar de beschikbare webdiensten. Zo'n workflowsysteem zal eenvoudiger te gebruiken zijn dan huidige workflowsystemen en zal nog beter passen bij de werkwijze van bioinformatici.

SIKS dissertation series

Since 1998, all dissertations written by Ph.D. students who have conducted their research under auspices of a senior research fellow of the SIKS research school are published in the SIKS Dissertation Series. This thesis is the 245st in the series.

- 2010-01** Matthijs van Leeuwen (UU), *Patterns that Matter*
- 2009-45** Jilles Vreeken (UU), *Making Pattern Mining Useful*
- 2009-44** Roberto Santana Tapia (UT), *Assessing Business-IT Alignment in Networked Organizations*
- 2009-43** Virginia Nunes Leal Franqueira (UT), *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*
- 2009-42** Toine Bogers (UvT), *Recommender Systems for Social Bookmarking*
- 2009-41** Igor Bereznyy (UvT), *Digital Analysis of Paintings*
- 2009-40** Stephan Raaijmakers (UvT), *Multinomial Language Learning: Investigations into the Geometry of Language*
- 2009-39** Christian Stahl (TUE, Humboldt-Universitaet zu Berlin), *Service Substitution – A Behavioral Approach Based on Petri Nets*
- 2009-38** Riina Vuorikari (OUN), *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*
- 2009-37** Hendrik Drachler (OUN), *Navigation Support for Learners in Informal Learning Networks*
- 2009-36** Marco Kalz (OUN), *Placement Support for Learners in Learning Networks*
- 2009-35** Wouter Koelewijn (UL), *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*
- 2009-34** Inge van de Weerd (UU), *Advancing in Software Product Management: An Incremental Method Engineering Approach*
- 2009-33** Khiet Truong (UT), *How Does Real Affect Affect Affect Recognition In Speech?*
- 2009-32** Rik Farenhorst (VU) and Remco de Boer (VU), *Architectural Knowledge Management: Supporting Architects and Auditors*
- 2009-31** Sofiya Katrenko (UVA), *A Closer Look at Learning Relations from Text*
- 2009-30** Marcin Zukowski (CWI), *Balancing vectorized query execution with bandwidth-optimized storage*
- 2009-29** Stanislav Pokraev (UT), *Model-Driven Semantic Integration of Service-Oriented Applications*
- 2009-27** Christian Glahn (OUN), *Contextual Support of social Engagement and Reflection on the Web*
- 2009-26** Fernando Koch (UU), *An Agent-Based Model for the Development of Intelligent Mobile Services*
- 2009-25** Alex van Ballegooij (CWI), *RAM: Array Database Management through Relational Mapping*
- 2009-24** Annerieke Heuvelink (VUA), *Cognitive Models for Training Simulations*
- 2009-23** Peter Hofgesang (VU), *Modelling Web Usage in a Changing Environment*
- 2009-22** Pavel Serdyukov (UT), *Search For Expertise: Going beyond direct evidence*
- 2009-21** Stijn Vanderlooy (UM), *Ranking and Reliable Classification*
- 2009-20** Bob van der Vecht (UU), *Adjustable Autonomy: Controlling Influences on Decision Making*
- 2009-19** Valentin Robu (CWI), *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*

- 2009-18** Fabian Groffen (CWI), *Armada, An Evolving Database System*
- 2009-17** Laurens van der Maaten (UvT), *Feature Extraction from Visual Data*
- 2009-16** Fritz Reul (UvT), *New Architectures in Computer Chess*
- 2009-15** Rinke Hoekstra (UVA), *Ontology Representation - Design Patterns and Ontologies that Make Sense*
- 2009-14** Maksym Korotkiy (VU), *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
- 2009-13** Steven de Jong (UM), *Fairness in Multi-Agent Systems*
- 2009-12** Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin), *Operating Guidelines for Services*
- 2009-11** Alexander Boer (UVA), *Legal Theory, Sources of Law & the Semantic Web*
- 2009-10** Jan Wielemaker (UVA), *Logic programming for knowledge-intensive interactive applications*
- 2009-09** Benjamin Kanagwa (RUN), *Design, Discovery and Construction of Service-oriented Systems*
- 2009-08** Volker Nannen (VU), *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
- 2009-07** Ronald Poppe (UT), *Discriminative Vision-Based Recovery and Recognition of Human Motion*
- 2009-06** Muhammad Subianto (UU), *Understanding Classification*
- 2009-05** Sietse Overbeek (RUN), *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*
- 2009-04** Josephine Nabukenya (RUN), *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
- 2009-03** Hans Stol (UvT), *A Framework for Evidence-based Policy Making Using IT*
- 2009-02** Willem Robert van Hage (VU), *Evaluating Ontology-Alignment Techniques*
- 2009-01** Rasa Jurgelenaite (RUN), *Symmetric Causal Independence Models*
- 2008-35** Ben Torben Nielsen (UvT), *Dendritic morphologies: function shapes structure*
- 2008-34** Jeroen de Knijf (UU), *Studies in Frequent Tree Mining*
- 2008-33** Frank Terpstra (UVA), *Scientific Workflow Design; theoretical and practical issues*
- 2008-32** Trung H. Bui (UT), *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes*
- 2008-31** Loes Braun (UM), *Pro-Active Medical Information Retrieval*
- 2008-30** Wouter van Atteveldt (VU), *Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content*
- 2008-29** Dennis Reidsma (UT), *Annotations and Subjective Machines – Of Annotators, Embodied Agents, Users, and Other Humans*
- 2008-28** Ildiko Flesch (RUN), *On the Use of Independence Relations in Bayesian Networks*
- 2008-27** Hubert Vogten (OUN), *Design and Implementation Strategies for IMS Learning Design*
- 2008-26** Marijn Huijbregts (UT), *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*
- 2008-25** Geert Jonker (UU), *Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency*
- 2008-24** Zharko Aleksovski (VU), *Using background knowledge in ontology matching*
- 2008-23** Stefan Visscher (UU), *Bayesian network models for the management of ventilator-associated pneumonia*
- 2008-22** Henk Koning (UU), *Communication of IT-Architecture*
- 2008-21** Krisztian Balog (UVA), *People Search in the Enterprise*
- 2008-20** Rex Arendsen (UVA), *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven.*
- 2008-19** Henning Rode (UT), *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*
- 2008-18** Guido de Croon (UM), *Adaptive Active Vision*
- 2008-17** Martin Op 't Land (TUD), *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
- 2008-16** Henriëtte van Vugt (VU), *Embodied agents from a user's perspective*
- 2008-15** Martijn van Otterlo (UT), *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.*
- 2008-14** Arthur van Bunningen (UT), *Context-Aware Querying; Better Answers with Less Effort*
- 2008-13** Caterina Carraciolo (UVA), *Topic Driven Access to Scientific Handbooks*
- 2008-12** József Farkas (RUN), *A Semiotically Oriented Cognitive Model of Knowledge Representation*
- 2008-11** Vera Kartseva (VU), *Designing Controls for Network Organizations: A Value-Based Approach*
- 2008-10** Wauter Bosma (UT), *Discourse oriented summarization*
- 2008-09** Christof van Nimwegen (UU), *The paradox of the guided user: assistance can be counter-effective*
- 2008-08** Janneke Bolt (UU), *Bayesian Networks: Aspects of Approximate Inference*
- 2008-07** Peter van Rosmalen (OUN), *Supporting the tutor in the design and support of adaptive e-learning*

- 2008-06** Arjen Hommersom (RUN), *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
- 2008-05** Bela Mutschler (UT), *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
- 2008-04** Ander de Keijzer (UT), *Management of Uncertain Data – towards unattended integration*
- 2008-03** Vera Hollink (UVA), *Optimizing hierarchical menus: a usage-based approach*
- 2008-02** Alexei Sharpanskykh (VU), *On Computer-Aided Methods for Modeling and Analysis of Organizations*
- 2008-01** Katalin Boer-Sorbán (EUR), *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
- 2007-25** Joost Schalken (VU), *Empirical Investigations in Software Process Improvement*
- 2007-24** Georgina Ramírez Camps (CWI), *Structural Features in XML Retrieval*
- 2007-23** Peter Barna (TUE), *Specification of Application Logic in Web Information Systems*
- 2007-22** Zlatko Zlatev (UT), *Goal-oriented design of value and process models from patterns*
- 2007-21** Karianne Vermaas (UU), *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
- 2007-20** Slinger Jansen (UU), *Customer Configuration Updating in a Software Supply Network*
- 2007-19** David Levy (UM), *Intimate relationships with artificial partners*
- 2007-18** Bart Orriëns (UvT), *On the development an management of adaptive business collaborations*
- 2007-17** Theodore Charitos (UU), *Reasoning with Dynamic Networks in Practice*
- 2007-16** Davide Grossi (UU), *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
- 2007-15** Joyca Lacroix (UM), *NIM: a Situated Computational Memory Model*
- 2007-14** Niek Bergboer (UM), *Context-Based Image Analysis*
- 2007-13** Rutger Rienks (UT), *Meetings in Smart Environments; Implications of Progressing Technology*
- 2007-12** Marcel van Gerven (RUN), *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
- 2007-11** Natalia Stash (TUE), *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
- 2007-10** Huib Aldewereld (UU), *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
- 2007-09** David Mobach (VU), *Agent-Based Mediated Service Negotiation*
- 2007-08** Mark Hoogendoorn (VU), *Modeling of Change in Multi-Agent Organizations*
- 2007-07** Nataša Jovanović (UT), *To Whom It May Concern – Addressee Identification in Face-to-Face Meetings*
- 2007-06** Gilad Mishne (UVA), *Applied Text Analytics for Blogs*
- 2007-05** Bart Schermer (UL), *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
- 2007-04** Jurriaan van Diggelen (UU), *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
- 2007-03** Peter Mika (VU), *Social Networks and the Semantic Web*
- 2007-02** Wouter Teepe (RUG), *Reconciling Information Exchange and Confidentiality: A Formal Approach*
- 2007-01** Kees Leune (UvT), *Access Control and Service-Oriented Architectures*
- 2006-28** Börkur Sigurbjörnsson (UVA), *Focused Information Access using XML Element Retrieval*
- 2006-27** Stefano Bocconi (CWI), *Vox Populi: generating video documentaries from semantically annotated media repositories*
- 2006-26** Vojkan Mihajlović (UT), *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
- 2006-25** Madalina Drugan (UU), *Conditional log-likelihood MDL and Evolutionary MCMC*
- 2006-24** Laura Hollink (VU), *Semantic Annotation for Retrieval of Visual Resources*
- 2006-23** Ion Juvina (UU), *Development of Cognitive Model for Navigating on the Web*
- 2006-22** Paul de Vrieze (RUN), *Fundamentals of Adaptive Personalisation*
- 2006-21** Bas van Gils (RUN), *Aptness on the Web*
- 2006-20** Marina Velikova (UvT), *Monotone models for prediction in data mining*
- 2006-19** Birna van Riemsdijk (UU), *Cognitive Agent Programming: A Semantic Approach*
- 2006-18** Valentin Zhizhkun (UVA), *Graph transformation for Natural Language Processing*
- 2006-17** Stacey Nagata (UU), *User Assistance for Multitasking with Interruptions on a Mobile Device*
- 2006-16** Carsten Riggelsen (UU), *Approximation Methods for Efficient Learning of Bayesian Networks*
- 2006-15** Rainer Malik (UU), *CONAN: Text Mining in the Biomedical Domain*
- 2006-14** Johan Hoorn (VU), *Software Requirements: Update, Upgrade, Redesign – towards a Theory of Requirements Change*
- 2006-13** Henk-Jan Lebbink (UU), *Dialogue and Decision Games for Information Exchanging Agents*
- 2006-12** Bert Bongers (VU), *Interactivation – Towards an e-cology of people, our technological environment, and the arts*

- 2006-11** Joeri van Ruth (UT), *Flattening Queries over Nested Data Types*
- 2006-10** Ronny Siebes (VU), *Semantic Routing in Peer-to-Peer Systems*
- 2006-09** Mohamed Wahdan (UM), *Automatic Formulation of the Auditor's Opinion*
- 2006-08** Eelco Herder (UT), *Forward, Back and Home Again – Analyzing User Behavior on the Web*
- 2006-07** Marko Smiljanic (UT), *XML schema matching – balancing efficiency and effectiveness by means of clustering*
- 2006-06** Ziv Baida (VU), *Software-aided Service Bundling – Intelligent Methods & Tools for Graphical Service Modeling*
- 2006-05** Cees Pierik (UU), *Validation Techniques for Object-Oriented Proof Outlines*
- 2006-04** Marta Sabou (VU), *Building Web Service Ontologies*
- 2006-03** Noor Christoph (UVA), *The role of metacognitive skills in learning to solve problems*
- 2006-02** Cristina Chisalita (VU), *Contextual issues in the design and use of information technology in organizations*
- 2006-01** Samuil Angelov (TUE), *Foundations of B2B Electronic Contracting*
- 2005-21** Wijnand Derks (UT), *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*
- 2005-20** Cristina Coteanu (UL), *Cyber Consumer Law, State of the Art and Perspectives*
- 2005-19** Michel van Dartel (UM), *Situated Representation*
- 2005-18** Danielle Sent (UU), *Test-selection strategies for probabilistic networks*
- 2005-17** Boris Shishkov (TUD), *Software Specification Based on Re-usable Business Components*
- 2005-16** Joris Graaumans (UU), *Usability of XML Query Languages*
- 2005-15** Tibor Bosse (VU), *Analysis of the Dynamics of Cognitive Processes*
- 2005-14** Borys Omelayenko (VU), *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*
- 2005-13** Fred Hamburg (UL), *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
- 2005-12** Csaba Boer (EUR), *Distributed Simulation in Industry*
- 2005-11** Elth Ogston (VU), *Agent Based Matchmaking and Clustering – A Decentralized Approach to Search*
- 2005-10** Anders Bouwer (UVA), *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
- 2005-09** Jeen Broekstra (VU), *Storage, Querying and Inferring for Semantic Web Languages*
- 2005-08** Richard Vdovjak (TUE), *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
- 2005-07** Flavius Frasinca (TUE), *Hypermedia Presentation Generation for Semantic Web Information Systems*
- 2005-06** Pieter Spronck (UM), *Adaptive Game AI*
- 2005-05** Gabriel Infante-Lopez (UVA), *Two-Level Probabilistic Grammars for Natural Language Parsing*
- 2005-04** Nirvana Meratnia (UT), *Towards Database Support for Moving Object data*
- 2005-03** Franc Grootjen (RUN), *A Pragmatic Approach to the Conceptualisation of Language*
- 2005-02** Erik van der Werf (UM), *AI techniques for the game of Go*
- 2005-01** Floor Verdenius (UVA), *Methodological Aspects of Designing Induction-Based Applications*
- 2004-20** Madelon Evers (Nyenrode), *Learning from Design: facilitating multidisciplinary design teams*
- 2004-19** Thijs Westerveld (UT), *Using generative probabilistic models for multimedia retrieval*
- 2004-18** Vania Bessa Machado (UvA), *Supporting the Construction of Qualitative Knowledge Models*
- 2004-17** Mark Winands (UM), *Informed Search in Complex Games*
- 2004-16** Federico Divina (VU), *Hybrid Genetic Relational Search for Inductive Learning*
- 2004-15** Arno Knobbe (UU), *Multi-Relational Data Mining*
- 2004-14** Paul Harrenstein (UU), *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
- 2004-13** Wojciech Jamroga (UT), *Using Multiple Models of Reality: On Agents who Know how to Play*
- 2004-12** The Duy Bui (UT), *Creating emotions and facial expressions for embodied agents*
- 2004-11** Michel Klein (VU), *Change Management for Distributed Ontologies*
- 2004-10** Suzanne Kabel (UVA), *Knowledge-rich indexing of learning-objects*
- 2004-09** Martin Caminada (VU), *For the Sake of the Argument; explorations into argument-based reasoning*
- 2004-08** Joop Verbeek (UM), *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politie gegevens-uitwisseling en digitale expertise*
- 2004-07** Elise Boltjes (UM), *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*
- 2004-06** Bart-Jan Hommes (TUD), *The Evaluation of Business Process Modeling Techniques*
- 2004-05** Viara Popova (EUR), *Knowledge discovery and monotonicity*
- 2004-04** Chris van Aart (UVA), *Organizational Principles for Multi-Agent Architectures*
- 2004-03** Perry Groot (VU), *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
- 2004-02** Lai Xu (UvT), *Monitoring Multi-party Contracts for E-business*

- 2004-01** Virginia Dignum (UU), *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
- 2003-18** Levente Kocsis (UM), *Learning Search Decisions*
- 2003-17** David Jansen (UT), *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
- 2003-16** Menzo Windhouwer (CWI), *Feature Grammar Systems – Incremental Maintenance of Indexes to Digital Media Warehouses*
- 2003-15** Mathijs de Weerd (TUD), *Plan Merging in Multi-Agent Systems*
- 2003-14** Stijn Hoppenbrouwers (KUN), *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
- 2003-13** Jeroen Donkers (UM), *Nosce Hostem – Searching with Opponent Models*
- 2003-12** Roeland Ordelman (UT), *Dutch speech recognition in multimedia information retrieval*
- 2003-11** Simon Keizer (UT), *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
- 2003-10** Andreas Lincke (UvT), *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*
- 2003-09** Rens Kortmann (UM), *The resolution of visually guided behaviour*
- 2003-08** Yongping Ran (UM), *Repair Based Scheduling*
- 2003-07** Machiel Jansen (UvA), *Formal Explorations of Knowledge Intensive Tasks*
- 2003-06** Boris van Schooten (UT), *Development and specification of virtual environments*
- 2003-05** Jos Lehmann (UVA), *Causation in Artificial Intelligence and Law – A modelling approach*
- 2003-04** Milan Petković (UT), *Content-Based Video Retrieval Supported by Database Technology*
- 2003-03** Martijn Schuemie (TUD), *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
- 2003-02** Jan Broersen (VU), *Modal Action Logics for Reasoning About Reactive Systems*
- 2003-01** Heiner Stuckenschmidt (VU), *Ontology-Based Information Sharing in Weakly Structured Environments*
- 2002-17** Stefan Manegold (UVA), *Understanding, Modeling, and Improving Main-Memory Database Performance*
- 2002-16** Pieter van Langen (VU), *The Anatomy of Design: Foundations, Models and Applications*
- 2002-15** Rik Eshuis (UT), *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
- 2002-14** Wieke de Vries (UU), *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
- 2002-13** Hongjing Wu (TUE), *A Reference Architecture for Adaptive Hypermedia Applications*
- 2002-12** Albrecht Schmidt (Uva), *Processing XML in Database Systems*
- 2002-11** Wouter C.A. Wijngaards (VU), *Agent Based Modelling of Dynamics: Biological and Organisational Applications*
- 2002-10** Brian Sheppard (UM), *Towards Perfect Play of Scrabble*
- 2002-09** Willem-Jan van den Heuvel (KUB), *Integrating Modern Business Applications with Objectified Legacy Systems*
- 2002-08** Jaap Gordijn (VU), *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*
- 2002-07** Peter Boncz (CWI), *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*
- 2002-06** Laurens Mommers (UL), *Applied legal epistemology: Building a knowledge-based ontology of the legal domain*
- 2002-05** Radu Serban (VU), *The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents*
- 2002-04** Juan Roberto Castelo Valdeuza (UU), *The Discrete Acyclic Digraph Markov Model in Data Mining*
- 2002-03** Henk Ernst Blok (UT), *Database Optimization Aspects for Information Retrieval*
- 2002-02** Roelof van Zwol (UT), *Modelling and searching web-based document collections*
- 2002-01** Nico Lassing (VU), *Architecture-Level Modifiability Analysis*
- 2001-11** Tom M. van Engers (VUA), *Knowledge Management: The Role of Mental Models in Business Systems Design*
- 2001-10** Maarten Sierhuis (UvA), *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design*
- 2001-09** Pieter Jan 't Hoen (RUL), *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*
- 2001-08** Pascal van Eck (VU), *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*
- 2001-07** Bastiaan Schonhage (VU), *Divya: Architectural Perspectives on Information Visualization*
- 2001-06** Martijn van Welie (VU), *Task-based User Interface Design*
- 2001-05** Jacco van Ossenbruggen (VU), *Processing Structured Hypermedia: A Matter of Style*
- 2001-04** Evgueni Smirnov (UM), *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
- 2001-03** Maarten van Someren (UvA), *Learning as problem solving*
- 2001-02** Koen Hindriks (UU), *Agent Programming Languages: Programming with Mental Models*
- 2001-01** Silja Renooij (UU), *Qualitative Approaches to Quantifying Probabilistic Networks*
- 2000-11** Jonas Karlsson (CWI), *Scalable Distributed Data Structures for Database Management*
- 2000-10** Niels Nes (CWI), *Image Database Management System Design Considerations, Algorithms and Architecture*

- 2000-09** Florian Waas (CWI), *Principles of Probabilistic Query Optimization*
- 2000-08** Veerle Coupé (EUR), *Sensitivity Analysis of Decision-Theoretic Networks*
- 2000-07** Niels Peek (UU), *Decision-theoretic Planning of Clinical Patient Management*
- 2000-06** Rogier van Eijk (UU), *Programming Languages for Agent Communication*
- 2000-05** Ruud van der Pol (UM), *Knowledge-based Query Formulation in Information Retrieval*
- 2000-04** Geert de Haan (VU), *ETAG, A Formal Model of Competence Knowledge for User Interface Design*
- 2000-03** Carolien M.T. Metselaar (UVA), *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief*
- 2000-02** Koen Holtman (TUE), *Prototyping of CMS Storage Management*
- 2000-01** Frank Niessink (VU), *Perspectives on Improving Software Maintenance*
- 1999-08** Jacques H.J. Lenting (UM), *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*
- 1999-07** David Spelt (UT), *Verification support for object database design*
- 1999-06** Niek J.E. Wijngaards (VU), *Re-design of compositional systems*
- 1999-05** Aldo de Moor (KUB), *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*
- 1999-04** Jacques Penders (UM), *The practical Art of Moving Physical Objects*
- 1999-03** Don Beal (UM), *The Nature of Minimax Search*
- 1999-02** Rob Potharst (EUR), *Classification using decision trees and neural nets*
- 1999-01** Mark Sloof (VU), *Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products*
- 1998-05** E.W. Oskamp (RUL), *Computerondersteuning bij Straftoemeting*
- 1998-04** Dennis Breuker (UM), *Memory versus Search in Games*
- 1998-03** Ans Steuten (TUD), *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*
- 1998-02** Floris Wiesman (UM), *Information Retrieval by Graphically Browsing Meta-Information*
- 1998-01** Johan van den Akker (CWI), *DEGAS – An Active, Temporal Database of Autonomous Objects*