Scientific workflow design : theoretical and practical issues
Terpstra, F.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Promotiecommissie

promotor: Prof. dr. P.W. Adriaans

coopromotor: Dr. G.R. Meijer

overige leden: Prof. dr. M.T. Bubak
                 Prof. dr. C.A. Goble
                 Prof. dr. P. van Emde Boas
                 Prof. dr. L.O. Hertzberger
                 Prof. dr. ir. W. Bouten

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Coverdesign by Justus Tomlow

SIKS Dissertation Series No. 2008-33
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.
## Contents

1 Introduction  
   1.0.1 Grid ............................................. 1  
   1.1 Virtual Laboratory ................................. 3  
   4  
   1.3 Sharing Resources ................................. 5  

2 Methodology ........................................... 9  
   2.1 Introduction ....................................... 9  
   2.2 Methodology of Science ............................ 9  
      2.2.1 Philosophical issues ........................... 11  
   2.3 Methodology of e-Science .......................... 13  
      2.3.1 Definition of e-Science ......................... 13  
      2.3.2 Empirical Cycle for e-Science ................. 15  
   2.4 Differences Science and e-Science ............... 19  
   2.5 Future Scenario e-Science ......................... 22  
   2.6 Conclusion ........................................ 23  

3 Workflow Design Space  ............................... 25  
   3.1 Introduction ....................................... 25  
   3.2 Related Work ...................................... 25  
   3.3 Workflow Design .................................... 26  
      3.3.1 Workflow design using abstraction .......... 27  
   3.4 Theoretical limits of Workflow design ............. 30  
      3.4.1 Building blocks ............................... 30  
      3.4.2 Workflow construction ......................... 31  
      3.4.3 Complex workflow construction ............... 32  
      3.4.4 Workflow design limits ....................... 35  
   3.5 Discussion ......................................... 38  
   3.6 Conclusions ....................................... 40  

4 Workflow formalisms ................................. 41  
   4.1 Introduction ....................................... 41  
   4.2 Problem domains ................................... 42
| CONTENTS |
|----------------------|-------------------|
| 4.3 Formalisms       | 43                |
| 4.3.1 Overview       | 45                |
| 4.4 Discussion       | 49                |
| 4.5 Conclusions & future work | 50            |
| 5 Workflow Systems Analysis | 53            |
| 5.1 Introduction     | 53                |
| 5.2 Scientific Workflow Management Systems | 54          |
| 5.2.1 Workflow lifecycle | 54            |
| 5.2.2 Workflow model  | 55                |
| 5.2.3 Workflow engine | 56                |
| 5.2.4 User support   | 57                |
| 5.3 State of the art | 61                |
| 5.4 Towards a shared software resource | 62          |
| 6 Data Assimilation  | 65                |
| 6.1 Introduction     | 65                |
| 6.2 Weather Prediction | 65            |
| 6.3 Data Assimilation Algorithms | 67        |
| 6.3.1 Observation Data | 68              |
| 6.3.2 Computational Model | 69          |
| 6.3.3 State Estimate | 69                |
| 6.3.4 Prediction     | 69                |
| 6.3.5 Estimator      | 69                |
| 6.3.6 Use of ensembles | 70             |
| 6.4 Data assimilation toolkits | 70        |
| 6.4.1 Overview of Toolkits | 71          |
| 6.4.2 Toolkits in detail | 71            |
| 6.4.3 Grid use       | 73                |
| 6.5 Conclusion       | 74                |
| 7 Data Assimilation Case Studies | 75          |
| 7.1 Introduction     | 75                |
| 7.2 Bird migration model | 75           |
| 7.2.1 Data           | 76                |
| 7.2.2 Model          | 76                |
| 7.2.3 Estimator      | 77                |
| 7.2.4 Experiment     | 77                |
| 7.2.5 Conclusions bird migration | 81          |
| 7.3 Traffic Forecasting | 84              |
| 7.3.1 Intelligent Transport Systems | 84        |
| 7.3.2 Prediction for ITS | 85             |
| 7.3.3 Current solution | 88              |
| 7.3.4 Discussion and Requirements | 102        |
## CONTENTS

7.3.5 Conclusions for traffic prediction .................................. 103
7.4 Conclusion .......................................................................... 103

### 8 Ideal Workflow for Data Assimilation 105

8.1 Introduction ................................................................. 105
8.2 Workflow representation .................................................. 105
8.3 Workflow composition ...................................................... 107
  8.3.1 Defining data ......................................................... 107
  8.3.2 Defining resources .................................................. 108
  8.3.3 Defining goals ....................................................... 109
  8.3.4 Provenance ........................................................... 109
  8.3.5 Partial workflows ................................................... 110
  8.3.6 Dissemination ......................................................... 110
  8.3.7 Meta workflows ...................................................... 111
8.4 Workflow Design methodology for Data Assimilation .......... 111
  8.4.1 Shared software resource ........................................ 113
  8.4.2 Methodology ........................................................ 113
  8.4.3 Data preparation ..................................................... 114
  8.4.4 State Estimation ..................................................... 117
  8.4.5 Model ................................................................. 118
  8.4.6 Workflow Patterns ................................................ 120
8.5 Optimization .................................................................. 122
8.6 Requirements for Scientific Workflow Management Systems 122
  8.6.1 Meta-data .............................................................. 124
  8.6.2 Expressivity .......................................................... 124
  8.6.3 Composition ........................................................ 125
  8.6.4 Grid support ......................................................... 125
8.7 Overview of features in existing SWMS ............................ 126
8.8 Discussion & conclusion ................................................... 128

### 9 Conclusions & Future work 131

9.1 Introduction ................................................................. 131
9.2 Work Performed ........................................................... 131
9.3 Role of Workflow in e-Science ....................................... 132
  9.3.1 Resource Sharing ................................................... 132
  9.3.2 Dissemination and publishing .................................. 133
  9.3.3 Reproducibility ..................................................... 133
  9.3.4 Workflow design ................................................... 133
9.4 Current state of Workflow in e-Science ......................... 134
  9.4.1 Workflow design ................................................... 134
  9.4.2 Formalisms for Workflow .................................... 134
  9.4.3 Scientific Workflow Management Systems ................ 134
  9.4.4 Sharing of resources ............................................. 135
  9.4.5 When to use an e-Science approach ....................... 136
CONTENTS

9.5 Future Work ................................................. 136
  9.5.1 Standardization ........................................ 136
  9.5.2 Connectivity ........................................... 137
  9.5.3 Data assimilation in SWMS ........................... 137
  9.5.4 Formalisms ............................................. 138

A List of abbreviations ........................................... 139

B Turing Completeness I/O Automata ........................... 143

Acknowledgements ................................................. 155

Summary .................................................................. 157

Samenvatting .......................................................... 161
Chapter 1

Introduction

Over the last few years grid computing has emerged as a concept for exploiting massive distributed computing power and managing massive distributed data. Closely related is the concept of enhanced Science or e-Science which aims to support scientific experiments through the use of grid computing and associated tools. The research presented in this thesis was conducted within the Virtual Laboratory for e-Science project (VL-e)[21] which states as its aim:

The aim of the “Virtual Laboratory for e-Science” project is to bridge the gap between the technology push of the high performance networking and the Grid and the application pull of a wide range of scientific experimental applications. It will provide generic functionalities that support a wide class of specific e-Science application environments and set up an experimental infrastructure for the evaluation of the ideas.

The introduction will give a brief explanation of the grid, what functionalities and tools constitute a virtual laboratory, and go into the basics of workflow tools and workflow design, the focus of this thesis.

1.0.1 Grid

In 1997 the most widely used basis for grid computing, the Globus Toolkit [10], started out as a way of linking parallel computing clusters as well as high speed research networks into one virtual system. It is thus no surprise that the first e-Science applications originated in projects which consume a lot of supercomputer power such as the Large Hadron Collider at CERN1. Over the last few years both e-Science and grid computing have evolved to deal with ever more heterogeneous resources and to solve problems other than massive data and large scale computations.

1http://lhc.web.cern.ch/lhc/
CHAPTER 1. INTRODUCTION

One of the early benefits of grid computing was sharing computational, data storage and experiment related resources. Currently the list has expanded it now includes:

- computational resources, cluster computers or idle time on desktop pc’s
- storage resources, raid disk arrays or harddisks in a desktop
- software resources, any software which can run on the grid in some way.
- data, that has relevance to many people, for instance genome data
- measuring equipment, such as microscopes and telescopes
- human experts, who offer a service via grid infrastructure

These resources are available for sharing through grid middleware. People with grid access can schedule jobs including shared software resources onto computational resources. Shared data can be stored in a transparent distributed fashion on storage resources. Measurement equipment can be remotely controlled as well as store its output data on storage resources through the use of grid middleware. In exceptional cases where analysis or other tasks can only be done by human experts, interaction with these experts can be offered through grid infrastructure.

This expansion of the types of shared resources was enabled by the current trend of moving to a Service Oriented Architecture (SOA), where resources are shared using web or grid services. Most notably the Open Grid Services Architecture (OGSA) [64] builds on the Globus Toolkit to allow the implementation of a Service Oriented Architecture on top of the grid.

Grid computing and e-Science have now been adopted by many more and diverse fields of science and are still growing. Grid computing has also started to branch out into the commercial domain with companies such as IBM2, SUN3, HP4 and Microsoft5 actively supporting e-Science projects. Their aim is not only to support science but to pave the way for large scale commercial use of grid computing and associated concepts. What attracts them is the concept of virtual organizations, defined in grid context as “flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources” [65]. However until issues such as security, protection of intellectual property, accounting and access are fully developed commercial adoption will be limited. Within the scientific

2http://www-1.ibm.com/grid/
3http://www.network.com/
4http://www.hp.com/go/grid
5http://research.microsoft.com/ur/us/escience/
1.1. VIRTUAL LABORATORY

In the VL-e project a virtual laboratory consists of

- Storage and computational resources that are grid enabled, that means accessible through grid middleware.
- Programming tools for creating (high performance distributed) applications that run on the grid
- Problem solving environments for running existing application specific (legacy) code on the grid
- Visualization and virtual reality interfaces
- Information management tools for dealing with distributed data
- Scientific Workflow Management Systems (SWMS) enabling sharing of (software) resources and cooperation

The VL-e aims to be generic, thus e-Science applications from diverse research areas are supported. The VL-e is of interest to these diverse research areas because it offers a sharing of the burden to gain access to the computational power and storage capacity of the grid. As an additional benefit scientific (software) resources and the knowledge on how to use them can more easily shared when this is done in a uniform environment. In this thesis we are interested in Workflow design and the way it can help the sharing of resources, so we will now introduce the concepts of workflow and SWMS.
CHAPTER 1. INTRODUCTION

Figure 1.1: Overview of the basic architecture of Scientific Workflow Management Systems and the place different resources have within this architecture and Grid.

1.2 Workflow and Scientific Workflow Management Systems

Resources in e-Science are never used on their own. Experiments in e-Science are therefore often performed with the help of Scientific Workflow Management Systems (SWMS), where workflows are used to define all connections and parameters of resources involved. While for the first e-Science applications workflows often took the form of batch jobs manually programmed in the users favorite text editor, currently most use a graphical workflow representation. The user defines a diagram with blocks representing resources and arrows indicating connections in a manner similar to programming in a visual programming language such as visual basic. A SWMS represents resources as workflow components, the links between workflow components represent data connections, the combination of workflow components and connections is called a workflow topology. A workflow engine takes care of the execution of the workflow which can be divided in two tasks. First, there is the orchestration of the workflow, which components are allowed to execute and which links transmit data at any particular time. Second, there is scheduling, that schedules each execution task onto available computational (grid) resources. Although SWMS can share measuring equipment, data and human experts, the main focus is on sharing software resources. Grid technology such as the globus toolkit takes care of sharing computational and storage resources. The place of resources in SWMS and the grid is illustrated in Figure 1.1.

When sharing knowledge on how to use a resource, workflows offer an opportunity to include knowledge on the context in which resources are used. This type of knowledge is commonly referred to as Provenance, which
can be divided into multiple types. Data Provenance is a record of the transformation and aggregation of data by workflow components which is stored as meta-data. Similarly the provenance of workflow components is stored in the collection of workflows in which it occurs.

1.3 Sharing Resources

The usability of workflows and the ability to express knowledge about an experiment through workflows is very important for e-Science. Workflows enable e-Science and grid computing to fully exploit the possibilities of sharing resources as well as offering greater usability in general.

This sharing of resources can lead to a major shift in how science is conducted, if indeed it becomes as useable as envisioned by its proponents[81]. The first step in this thesis will therefore be to explore the differences between e-Science and science in general and learn whether e-Science is or can become a paradigm shift for the way science is conducted. This look at e-Science will be concluded with a scenario of what a future scientist would do in a fully developed virtual laboratory for e-Science. This scenario will form a reference to which we can compare currently available tools and methods for sharing resources. In this way we can determine to what extent these tools and methods are already sufficient, and what still needs to happen to achieve this future scenario. Workflows are a central tool to enable the sharing of resources, thus we concentrate on a methodology for designing workflow components that can be shared.

In workflow design many issues exist relating to both the control of processes and data flow, that have long been present in parallel computing. Yet the formal approach to software design that has existed for a long time for parallel computation has not been adopted for workflow. Workflow is different in nature from parallel computing in general in that development of workflows happens at multiple levels of abstraction, there is a strict separation between the individual workflow components and the coordination that happens through the connections of these components. As the large majority of resources shared in SWMS are workflow components representing software resources, this leads us to an important question:

**What is a proper design methodology for both workflow components and workflow topologies that supports the sharing of software resources?**

There are several stages to finding a proper answer to this question and it incorporates both a practical and a theoretical side.

The first side is studied in two different uses of one type of software resource; data assimilation tools. Data assimilation is a technique for mini-
mizing errors both in observations as well as in the model involved in doing predictions. It finds its origins in climatological and geographical research such as weather prediction and oceanography. This type of error minimization occurs frequently in scientific experiments therefore data assimilation tools have great potential for reuse in areas other than weather prediction and oceanography. In the two cases which are studied in this thesis it is used for the prediction of bird migration and the long term prediction of road traffic. This look at the practical side will try to answer the following questions, what data assimilation tools are currently available and to what extent do they support incorporation in a virtual laboratory right now? Through employing data assimilation in two unrelated fields, what are the areas where a workflow design methodology for data assimilation can support the construction of an experiment?

The theoretical side is approached by looking at the theoretical limits of a what constitutes a workflow design space and which formalisms are most suitable to reason about workflows inside of this design space. Design space and design methodologies have been the subject in many related fields such as embedded systems design\cite{107, 108} as well as parallel computing\cite{55}. While formal approaches to workflows exist they focus mainly on the expressiveness of workflow design\cite{24} or the automatic construction of workflows\cite{71}. A design methodology for going from initial experiment hypothesis to concrete executable workflow has not yet been documented. The theoretical side will try to answer the following questions: How does the methodology for e-Science relate to scientific methodology in general and what is the importance and place of a design methodology for workflows and workflow components within the e-Science methodology in general? What is the workflow design space and what are its limits; what can and cannot be achieved in workflow design? What is a suitable formalism to reason about workflows in such a design space?

On the basis of a careful study of both these practical and theoretical issues, a method for sharing data assimilation resources is proposed. The conclusion of this thesis will try to answer the following:

- To what extend a methodology for sharing data assimilation as a software resource can provide answers to the main research question, the proper design methodology for shared software resources in general?
- Do current tools lend themselves to sharing resources, and what are the areas that need improvement to realize the full potential of e-Science.

The thesis starts with describing how e-Science methodology relates to general scientific methodology and what future use of a fully developed virtual laboratory could look like in chapter 2. In the following chapters the theoretical issues are addressed, both workflow design space in chapter 3 and suitable workflow formalisms in chapter 4. The practical part starts
1.3. **SHARING RESOURCES**

with an overview of what existing Scientific Workflow Management Systems have to offer in chapter 5, followed in chapter 6 by an introduction into data assimilation. The details of both bird migration and traffic prediction case studies are presented in chapter 7. Lessons learned from the investigation into both the practical and theoretical issues are applied in chapter 8 with a method for sharing data assimilation resources in a workflow environment. This method is presented as an ideal workflow for data assimilation. Finally other issues encountered during research are discussed and conclusions are reached in the final chapter.
Chapter 2

Methodology

2.1 Introduction

The research described in this thesis concerns experiments within the context of a Virtual Laboratory also known as e-Science. Scientific methodology is fundamental to conducting experiments. In this chapter we describe how the scientific methodology can be applied in a Virtual Laboratory, and what aspects in particular are affected by the Virtual Laboratory environment. The Virtual Laboratory environment is, for the scientific user, dominated by the workflow environment making up the frontend. This leads to workflow playing a pivotal role in e-Science methodology. We will compare the method of science in general to that of e-Science and discuss if the new assumptions and methods e-Science bring constitute a paradigm shift. We will start by looking at the methodology for science in general and what exactly a paradigm shift is. We continue by looking at what exactly what different people understand e-Science to be and what its defining features are. A short overview is given of the different roles that exist for people that together conduct e-Science in a virtual laboratory. Then we show how the scientific method in e-Science differs from that of science in general and discuss whether this constitutes a paradigm shift. Finally we give a scenario of what the work of a scientist working in a virtual laboratory will look like if all the aims of e-Science have been realized. In this way this chapter sets up a reference against which we can compare whether systems and techniques presented in the rest of this thesis can realize the full potential of e-Science.

2.2 Methodology of Science

To give a better understanding of what aspects of scientific methodology are important in e-Science we will first take a look at scientific methodology in general. This will by no means be an all inclusive overview but it will touch upon the essential problems within the philosophy of science which
are relevant to e-Science.

The foundations for our current scientific methodology were laid in the renaissance when experimental science started. This methodology for experimental science is dominated by the empirical cycle. It is often depicted in the form used here in Figure 2.1 for instance by Adriaans and Zantinge in [26], and based upon the work by scientific philosophers such as Popper, Lakatos and Stegmuller [110, 89, 115]. It is noteworthy though, that there does not seem to be one firm definition of the empirical cycle, upon which everyone in the scientific community agrees. The empirical cycle describes an incremental process for increasing knowledge. In different incarnations it contains the following steps: Define research question, characterization, hypothesis, theory, model, prediction, experiment, observation, analysis, publish results, dissemination, reproduce result, verification. The empirical cycle will be described here in terms of four main elements:

- Theory
- Prediction
- Observation
- Analysis

We start with theory in which all relevant assumptions are defined, these include the research question and methods of observation and measurements as well as existing theories which are relevant for the experiment(s) that are about to be performed. A model would also fall under the theory step. The Prediction step is a logical deduction from the theory defined in the first step, one could also call a prediction a hypothesis. It is defined in such a way that it can be directly compared to observations in the next step. The analysis step performs this comparison, an explanation is sought which explains the differences or similarity between observation and prediction. This can be
done by the scientists themselves or by their peers through publication in the scientific community. Based upon the analysis step multiple actions are possible all of which let the cycle continue for another iteration. The reproducibility can be tested by performing the whole cycle again. This can be done by scientists themselves or by their peers who are reviewing the work based upon a publication. Assumptions in the theory step can be adjusted to find extra evidence either verifying or falsifying the theory. Finally the theory itself can be adjusted based upon the analysis. The meaning of experiment is strongly associated with the actual measurements performed. We take a wider view of what an experiment is and view it as preforming one or more iterations of the empirical cycle.

To illustrate the empirical cycle we will look at an historical example. According to Aristotle heavier objects fell more quickly than lighter objects. The underlying theory being that of the five elements, earth, water, air, fire and aether. The properties of each object are determined by its likeness to four of the elements, earth, water, air and fire, while aether acts as a conduit attracting each object to its likeness. According to this theory a heavy object like a stone had most in common with earth and wanted to move there more quickly than a lighter object. A feather on the contrary had something in common with air and would move to the ground less quickly. Based on this theory Aristotle could make the prediction that a stone would fall more quickly than a feather. Indeed performing this experiment the feather could be observed falling much more slowly than the stone. Other observations could be made as well, the feather does not fall in a straight line. In the analysis this could be explained as the feather showing properties of air, thereby confirming the theory. As we now know Aristotle was wrong. We will return to this example later to show how he was wrong and more importantly illustrate other issues in the philosophy of science.

2.2.1 Philosophical issues

The validity of the scientific method is the core discussion within the philosophy of science. One problem first raised by Hume[78] is the question whether induction, abstracting general knowledge from specific examples is valid. This is very relevant to experimental science as it is this reasoning method that is used to derive knowledge from experimental results. Predictions made, based on a theory that has often been verified, can still not be proven to be the absolute truth. This “problem of induction” is also referred to as Hume’s puzzle, as although induction can not be proven to be valid most of our knowledge is based upon it. Another important problem in the scientific method is the aim to discover objective truths. The question is whether it is possible for humans to perform unbiased objective measurements or to postulate objective hypotheses. Popper argues that all observation is theory laden [110]. Theory guides us to determine which
observations are significant, and in what way they are significant. To put it another way, the scope of our search space for observations is directed by our existing theories. This does not mean we try to find the answers we were expecting and are not open to unexpected, strange, surprising or serendipitous results. In an infinite universe we can only make a finite number of observations, thus we have to restrict our search space based upon our existing theories. According to both Feyerabend and Popper[63, 110] descriptions of observation are necessarily theory laden. We cannot claim that the data we use is purely objective, there is always some theory used to translate our experience, or those of measuring instruments into data.

An often heard term in the philosophy of science is the “paradigm shift”. It was introduced by Thomas Kuhn[88] to counter the idea of gradual and cumulative growth of scientific knowledge, a position taken among others by Popper in his essay “the aim of science” [109]. The basic premise is that instead of gradual evolution there are periodic revolutions in scientific understanding when the scientific community adopts a new theory in favor of an older one. Kuhn argues that to adopt a new theory one also has to change many of the basic assumptions on which a theory is based. Thus it is not a cumulative growth of adding the new theory to existing understanding, but rather a shift of theory and basic assumptions in the area the new theory covers: a paradigm shift. Multiple competing paradigms can exist at the same time. During a shift dominance shifts from one to the other. This, according to Kuhn, is accompanied by fierce arguments of proponents of the different paradigms arguing the correctness of their theories, until results clearly point the community in the direction of the new paradigm.

Now we return to our example: the theory of Aristotle. It was disproved by Galileo. He showed that two balls of different weight, a cannon-ball and a musket ball fell at the same speed despite their weight difference. Galileo came to this experiment because his underlying theory of motion had changed. According to Galileo the distance covered by an accelerating object (for instance through falling) from rest is proportional to the time squared. There is therefore no relation of weight to the speed with which an object falls. Furthermore an object maintains its velocity unless a force acts upon it. The feathers slower fall is due to the force of air resistance acting upon it. This example is a good demonstration of a paradigm shift. The literal meaning of paradigm is example. In this case Galileo’s experiment is an example of a shift in paradigm in the most literal sense. Galileo was not the first to perform this experiment or disprove Aristotle. In fact there is doubt whether he actually ever performed it as is commonly told on the tower of Pisa. Yet this example stuck, because Galileo backed it up with a new theory. The example by which the theory of motion was illustrated shifted to Galileo’s experiment. More interesting though is the underlying shift in theory and assumptions which is most often intended when the term paradigm shift is used.
2.3 Methodology of e-Science

After a brief introduction into scientific methodology in the previous section the focus is now on what differentiates e-Science from science in general and what is important in a methodology for e-Science. To answer this a definition of e-Science is needed first.

2.3.1 Definition of e-Science

There is no single clear definition for the term e-Science. However looking at the collection of definitions presented below a common picture emerges.

"Science increasingly done through distributed global collaborations enabled by the Internet, using very large data collections, tera-scale computing resources and high performance visualization."\(^1\)

"E-Science is about global collaboration in key areas of science and the next generation of infrastructure that will enable it."\(^2\)

"E-Science refers to science that is enabled by the routine use of distributed computing resources by end-user scientists. E-Science is often most effective when the resources are accessed transparently and pervasively, and when the underlying infrastructure is resilient and capable of providing a high quality of service. Thus, grid computing and autonomic computing can serve as a good basis for e-Science. E-Science often involves collaboration between scientists who share resources, and thus act as a virtual organization."\(^3\)

"The next generation of scientific research and experiments will be carried out by communities of researchers from organizations that span national boundaries. These activities will involve geographically distributed and heterogeneous resources such as computational systems, scientific instruments, databases, sensors, software components, networks, and people. Such large-scale and enhanced scientific endeavors, popularly termed as e-Science, are carried out via collaborations on a global scale."\(^4\)

"The e-Science vision is a future research environment based on virtual organizations of people and agents highly dynamic and with large-scale computation, data, and collaboration."\(^5\)

---

\(^1\)U.K. Government

\(^2\)Dr John Taylor, Director General of Research Councils, 2000 http://www.rcuk.ac.uk/escience/

\(^3\)e-Science Gap Analysis june 2003 UK e-Science Grid

\(^4\)e-Science 2005 conference Australia

\(^5\)e-Science: The Grid and the Semantic Web IEEE intelligent systems vol 19 no1, January/February 2004
Important factors in e-Science can be brought together in a few distinct clusters

- **massive computing**: grid computing, distributed computing resources, large scale computation
- **massive data handling**: data management, very large data collections, large scale data, distributed databases
- **virtual organizations**: large scale/global collaboration
- **Resource sharing**: computational, storage, software, data, measuring equipment, human experts

The development of e-Science environments is being driven by applications such as ATLAS[37] from the field of high energy physics, which place a big demand on all four of the previously mentioned factors. E-Science environments aim for the sharing of as many resources as possible of all types of resources. This allows other scientific applications, which are not the primary driving force behind e-Science development, to benefit from the available shared resources. Within an e-Science environment scientists will have access to resources which are outside of their own expertise. These can be resources within a large interdisciplinary project or resources shared with other projects in a common e-Science environment. Within e-Science there are different roles that perform the various tasks in a functioning e-Science environment. In his thesis[82] Ersin Kaletas identifies four types of users which are briefly explained here:

- **Scientist**, uses resources in an e-Science environment, but does not necessarily have specialist knowledge about these resources.

- **Domain Expert**, someone who has specialized knowledge about one particular domain and the resources involved. Tries to make resources available in a generic and easily reusable form.

- **Tool Developer**, a (scientific) programmer who develops the core functionality of resources. Access to scientific instruments data processing software tools etc.

- **Administrator**, performs user- infrastructure and resource management. Keeps a Virtual laboratory in proper working order.

Scientific methodology should only be relevant for the scientist and domain expert, whose responsibility it is to ensure the scientific validity of experiments. There is a big burden on the domain expert in developing a shared resource. Not only should he himself be able to use it in a scientifically sound way, others - e.g. non-experts - should be able to do this
as well without his direct supervision. The scientist using shared resources should also be very much aware that he is relying on the developers of the shared resource for the scientific soundness of his experiment. Due to the importance of sharing and reuse within e-Science, performing an experiment should be interpreted in the wider sense as defined in the previous section. An e-Science experiment includes the definition of all assumptions as well as dissemination of results, be they resources which can be reused by others or just data. e-Science environments are setup in such a way that they allow for easy reproducibility. Repeating experiments can be made easy through the use of a workflow environment. In such an environment all steps of an experiment are defined and can be easily repeated many times without much extra administrative burden. Recreating an experiment performed by another scientist is also relatively easy through the use of shared resources when the original experiment and the reproduction are both performed in similar e-Science environments. This does bring with it the danger that errors inherent in either the e-Science environment in general or in the shared resources are repeated unnoticed in the reproduction of an experiment.

2.3.2 Empirical Cycle for e-Science

Now that the definition of e-Science is understood and we know the roles of the different people involved a clear methodology is needed. This is needed in particular for both the domain expert making his work available as a shared resource and for the non-expert scientist using the e-Science environment. First we define what we consider to be the empirical cycle for e-Science (see figure 2.2) and compare each step of this cycle to the one for science in general that was defined earlier in this chapter. Then we discuss how these differences differentiate e-Science from science in general and if these differences can be considered a paradigm shift.

Theory: e-Science differentiates itself from science in general in the way experiments are performed. Thus the theory step of the cycle remains more or less unchanged for e-Science.

Design Workflow: The workflow design process is analogous to formulating a Hypothesis. In its most abstract view a workflow consists of available input and desired output. This is similar to the research question which is used as the starting point for formulating a hypothesis. To move from the most abstract view of a workflow, the input and output requirements, to a concrete workflow is a process of refinement into multiple workflow steps. The assumptions made in this refinement process are clearly defined until an executable workflow is produced. This is similar to arriving at a testable hypothesis based upon the original research question. Defining workflow elements and resources as well as defining data are all part of explicitly defining assumptions in e-Science. This can be limited to data types (i.e. integer) and resource location (i.e. ip-nr 145.50.10). It can go further by providing meta
data such as temperature measured in Celsius with -273.15 lower boundary or MySQL data base located at Foo maintained by John Doe. The advantage of using meta data, based on an ontology, is that it makes the theory laden nature of data more explicit: the assumptions made for representing data are clearly defined. This can potentially increase standardization and reusability of results. While not every workflow is a hypothesis in e-Science, a hypothesis can be expressed in a workflow. In fact a workflow can be as valid a method of expressing a hypothesis as a mathematical function is. This does require that both the workflow language as well as the processes making up each workflow step are formally described. In this way a workflow effectively expresses the hypothesis in a formal language. Insight into how this can be achieved will be provided in later chapters.

Perform Workflow: Actually performing a workflow is similar to doing observations in science in general. Not only are observations done within the workflow, the fact whether the workflow itself is actually executable and delivers the expected type of data is also verified. To ensure reproducibility, but also to aid the analysis of the workflow execution, intermediate data for each of the workflow steps is stored. A record is kept of the provenance of data, the steps that led to their creation. Similarly the provenance of the workflow steps can be kept: in which workflows these components were employed.

Analysis / Share Resources: Furthermore workflow patterns, common topologies of resources in a workflow, can be stored as abstract workflows. All of this helps the scientist trying to reproduce his own work or that of his peers. Very much in the same way that a detailed written description and laboratory logbook can help in traditional science. It also helps the scientist to analyze the data produced by the workflow, possibly leading to a change in the workflow, or even in the underlying theory. Just like in science in general, multiple experiments are usually performed before the underlying theory is changed or considered to be confirmed, in e-Science multiple executions of different workflows are performed.

Roles in Empirical Cycle for e-Science

The scientific process in e-Science is divided over multiple roles, the scientist and the domain expert. The Scientist is mainly concerned with the empirical cycle, whilst the domain expert is involved in the dissemination of resources and associated knowledge, as well as ensuring reproducible behavior for the resources he makes available. The scientist who performs experiments uses the e-Science specific version of the empirical cycle. Within this cycle, as presented in figure 2.2, the domain expert can also have a small task in defining workflow resources. This occurs when the scientist uses shared resources. Furthermore when previously defined abstract workflows are used to compose the workflow for an experiment. The domain expert, which pro-
2.3. METHODOLOGY OF E-SCIENCE

![Diagram depicting the empirical cycle within an e-Science context.]

Figure 2.2: The empirical cycle within an e-Science context, as defined in this thesis.

The empirical cycle within an e-Science context, as defined in this thesis, consists of the following steps: Design Workflow, Perform Workflow, Analysis/Share Resources, and Theory.

Provided these abstract workflows, has taken part in the workflow definition phase as well. The main methodology for the domain expert concerns itself with dissemination and to a lesser extent reproducibility. In practice this means creating shared resources and associated methodologies. Methodologies can consist of both workflow patterns and documentation on how to use resources. In many ways this is analogous to publishing results in scientific publications, because by making resources available you invite other scientists to test their validity as well as reproducing results from previous experiments. It can also augment scientific publications by offering a workflow interface to the scientists work which a reviewer can access and review remotely. Clearly validation of the shared resources is a very important part in making shared resources available in a scientific context.

Sharing resources in e-Science

The methodology as presented in figure 2.3 deals with sharing resources. It is a simple abstraction from the concrete executable resource. The creation of a shared resource from an existing dedicated resource, whether this is a small software component or something fundamental like grid access, is guided by four main criteria that have to be satisfied:

- **Consistency** A workflow component must always be able to reach a final finished state and the content of its output must always remain consistent with the associated data definitions.

- **Generality** A workflow component should minimize its dependence on: specific resources on which to run as well as specific types of data required for either input or output.

- **Simplicity** The number of inputs/outputs parameters needed for the correct functioning of the component needs to be limited. The com-
CHAPTER 2. METHODOLOGY

Figure 2.3: Methodology for creating shared resources

Communication and computation performed should happen as efficiently as possible with a minimum of overhead.

- **Useability** The design time of a workflow employing a shared resource should be minimized, a shared resource should be quick and easy to implement.

It is obvious that there will be a trade-off to some extent between simplicity and generality on the one hand and useability on the other. Below we describe a three step process to derive a shared resource from an existing dedicated resource.

- First all experiment specific parts of a resource have to be reviewed. A decision needs to be made on how far to abstract. The higher the abstraction the broader the applicability of the shared resource. At the same time higher abstraction means a longer route to implementation when a resource is used in a different context. With higher abstraction usability goes down. Abstraction can thus be as simple as removing instantiations of resource parameters, usually though it will be a more complex task. Every aspect of a resource that in some way interacts with other resources will have to be reviewed to see if it is general enough. This includes: parameter defaults and boundaries, types of data input which are accepted as well as the form of data output.

- Knowledge associated with using a resource has to be made explicit i.e. documentation help, links to proper successful uses of the resource, links to abstract workflows and ontologies of associated expert knowledge.
Finally a validation has to take place in which the generalized component is tested by scientists outside of the domain of the domain expert who is generalizing his resource.

In 2.2.1 the theory laden nature of both observation and hypothesis was touched upon. In e-Science these issues manifest themselves among other things as a bias in the representation of data and the implementation of resources. It should be the aim to minimize these biases when creating a shared resource.

The validation part in practice never stops, a workflow should be seen as a form of hypothesis which is falsifiable. If any of the assumptions used in defining a scientific theory turns out to be false, the theory itself can be falsified. Similarly if any of the assumptions made in the construction of a workflow is incorrect the whole workflow can be incorrect. Assumptions here include assuming the work of others is correct. Using any shared resource in a workflow should be done with appropriate skepticism to its correctness. Anyone offering a resource or using it should be aware to what extent it has been validated before giving too much value to results attained using this resource.

The scientist constructing an experiment using shared resources should carefully consider for each resource whether its use is appropriate and after running the experiment evaluate if the shared resource performed its task properly. When this is not the case he should consider a different shared resource, or build his own.

2.4 Differences Science and e-Science

Do the features of e-Science that distinguish it from regular science like massive data, massive computing power, virtual organizations and sharing of resources truly enhance science as we know it? What improvements do they bring to classical scientific problems? First the ability of e-Science to handle massive amounts of data and have massive computing power allows the scientist a bigger search space in his quest for knowledge. As the amount of observations and associated hypotheses grows larger, the scientist can put less constraints based on prior assumptions on his search for knowledge, thereby potentially increasing the objectivity of his work. Similarly virtual organizations and the sharing of resources allow more people to work on one potentially larger problem. Projects such as the Large Hadron Collider[37] would not be possible without some form of e-Science infrastructure. Sharing of resources also allows scientists to build on each others work in a more efficient way than before. There are also potential problems in e-Science. Working towards sharing resources encourages standardization especially in data representation. The prospect of interoperability can encourage the use of a standard representation which is less suitable to the
problem but works well with other parts of the workflow. Interoperability therefore discourages scientists to experiment with different representations based on alternative assumptions, thus potentially decreasing the objectivity of data and hypotheses.

The term paradigm shift is used frequently in connection with e-Science. It can refer to any number of distinct paradigm shifts:

- A paradigm shift for scientific disciplines who have become dependent upon computational resources.
- It can be used to denote a shift from Object Oriented Computing towards Service Oriented Computing.
- The virtualization of computational and data resources.
- A paradigm shift for the methodology of science, from "conventional science" to "enhanced science"

Can it be argued that e-Science is a paradigm shift in the specific way that Kuhn defines it, or are we dealing here with a more general use of the term? And is e-Science a paradigm shift for all of science, for computer science concepts or for the scientific disciplines newly introduced to the concept. Kuhn describes a paradigm shift as being about the acceptance of scientific theory and the assumptions which go with this theory. One classic example of a paradigm shift is the proof for the four color theorem in 1977 by Appel and Haken[36]. The four color theorem states that any map or indeed any plane divided into regions can be colored using four colors in such a way that every border between two regions has different colors on each side. The proof by Appel and Haken was unique in that it was generated using a computer. Another example which seems relevant here is the use of the telescope and microscope for observations. Before their introduction direct observation by the human senses was the only method considered in science. The arrival of instruments such as the telescope and microscope brought about a mechanization of scientific observation. In both of these examples the assumptions changed, in case of the four color problem the assumption that only humans can create proofs was changed. It was now clear that computers could generate proofs no human could realistically generate. Similarly the telescope and microscope changed the assumption that observation was done only directly by the naked eye. If e-Science is a paradigm shift, it needs to change some important assumption in the scientific process.

First let us turn our attention to two types of paradigm shift mentioned earlier. For scientific disciplines such as Biology and Biochemistry that have recently started to explore the massive amounts of data contained in the genetic code, e-Science has brought about a new type of experimentation. Where before studies into relevant related work could be done by hand, now
2.4. DIFFERENCES SCIENCE AND E-SCIENCE

databases all over the world need to be consulted in an automated fashion. One can argue that this is a paradigm shift. The assumption of what the maximum size of the search space is and what can be found in this search space has dramatically changed. Similarly in computer science the shift towards service oriented architecture (SOA) which is employed in e-Science can be seen as a paradigm shift. SOA can be seen as shift in paradigm where previously there were only the concepts of client server architecture and object oriented computing. On the other hand one could argue SOA is an additional concept since it has not replaced nor made obsolete concepts such as client server architectures and object oriented computing.

The infrastructure used in e-Science aims to virtualize data and computational resources through grid technology, while human resources are virtualized as virtual organizations. We need to know whether this virtualization is just an addition of another tool that can be employed for scientific experiments or whether it brings with it a change in our understanding of the basic concepts of science. e-Science as a scientific method, one could say is a theory about what the best method for doing certain types of science is.

There are different assumptions of what the basic concepts are in this scientific method as compared to traditional science. For instance an observation is not just a numerical representation as it is often in traditional science. An observation also has meta data detailing its provenance: where was this data generated, where has it been used in the past, what is its relation to other data and so on. This is information which is usually kept implicit or at least separate from observations. Yet all information that can be kept in meta data associated with an observation could have been recorded in a traditional scientific methodology. Virtualization demands that it is made more explicit but the concept itself is not a new one, scientists have been recording provenance in lab logbooks for a very long time. Similarly one can say that in e-Science a hypothesis is an abstract workflow. This is a new formal method of notation, but just as with observations there is no fundamental change in the semantics of the concept. Neither concept of observation nor of hypothesis changes in a way which makes it incompatible with previous scientific methodologies.

What does change - as mentioned at the beginning of the discussion - is the size of the search space and with that the assumption on what can potentially be found in scientific experiments. Just as the search space increases for bio-informatics, it can increase for many other scientific disciplines. It is important to note that not all scientific disciplines are as data dependent as bio-informatics. Thus the sharing of other resources, particularly scientific software, will need to increase dramatically in order for the same type of shift to occur for other disciplines. This is one of the aims of e-Science but in case of software it is inherently more complex than sharing data. The sharing of software on the scale that bio-informatics share data has not occurred yet.
CHAPTER 2. METHODOLOGY

Another property of a paradigm shift according to Kuhn is that it is accompanied by vigorous arguments of the proponents and detractors of the new theory. Thus for e-Science to be a paradigm shift according to Kuhn we should see heated debate, and scientific papers both arguing for, but also against e-Science and perhaps in favor of a different methodology. The discussion that perhaps needs to occur most for e-Science is whether the increase in search space is worth the loss of control and understanding of the resources used; whether shared resources under the control of others can be trusted enough to produce accurate results.

In its current state e-Science is clearly not a paradigm shift for science as a whole. There has not been the widespread reuse of resources within in all disciplines that employ e-Science, that would constitute a dramatic change in the way science is performed. Nor has there been a heated debate or even a vigorous discussion on the merits of sharing resources. For the moment it can only be a contributing factor in the paradigm shifts for a number of specific disciplines such as bio-informatics. In the following section we take a look at what it would take for e-Science to cause a paradigm shift for science in general.

2.5 Future Scenario e-Science

To get a better picture of what e-Science looks like when it does constitute a paradigm shift for science in general we present a future scenario for e-Science. As described in the previous section such a paradigm shift would be achieved if many resources can be shared. By many we mean not only the storage and computational resources that can currently be shared through grid technology. We also mean data resources such as the databases of gene information as used by bio-informatics or scientific publications accessible through webservies such as the medline database. These are all currently available for sharing. Most work is needed in facilitating the sharing of software resources used for performing experiments, scientific instruments as well as expert knowledge.

Problems that need to be addressed by a future e-Scientist when building an experiment with a large (and diverse) number of shared resources at his disposal are:

- Connectivity, do different software resources connect and do they have the same semantic understanding of the data they communicate.

- What workflow model should be employed for each stage of the experiment, and if multiple models are to be employed how do they relate to each other.

  - Hierarchically where a workflow using one model has to execute inside a workflow using another model.
2.6 Conclusion

In this chapter a very high level methodology for e-Science was derived from general scientific methodology. Furthermore the merits of e-Science as a paradigm shift were discussed. It is clear from the methodology that workflows and workflow components play a central role in e-Science. The next chapters will go into more detail concerning the formal grounding for workflow design. We look at the current state of the art in many of the areas mentioned in the future scenario and what is needed to get closer to this future scenario.
Chapter 3
Workflow Design Space

3.1 Introduction

When designing a workflow it is important to know what the design space is in which one is operating. In this chapter we first look at design approaches in general and then investigate the limits of the design space. We abstract from a lot of detail. One could see the workflow design problem as the task of constructing a complex workflow that is equivalent to a certain predefined computable function out of a set of standard components. As we will see this problem is in general unsolvable due to Rice’s theorem. Any formalism that tries to model automatic workflow composition in any generality is bound to fail. The work in this chapter is based on two previous publications by the author[120, 121] of this thesis. Suitable formalisms for reasoning about specific workflows is the subject of the next chapter. At the end of this chapter there is a discussion on the practical implications of the limits on the design space and what will be needed in a formal approach to specific workflows.

3.2 Related Work

A workflow is in essence a concurrent computational process. A number of formal approaches to the study of concurrent systems exist, amongst others: process algebras[74], guarded command languages[59] and actor theory[75]. Since most of these approaches commit themselves to certain philosophical preconceptions about the modeling of concurrent processes, they lack the flexibility to express the design issues we want to study. A well known dichotomy in this context is the division between *shared memory communication* and *message passing communication*. A choice for actor theory for instance would imply an undesirable restriction to message passing systems. Much practical work on design methods and in particular the use of different types of execution control has been done within the Ptolemy[79] project.
Although this project is aimed at embedded systems research, the fact that the Kepler\cite{31} scientific workflow management system is based on Ptolemy shows its applicability to workflow design.

Workflows and related tools such as workflow engines which coordinate the execution of a workflow are certainly not limited to e-Science. Within the Business Process community much engineering as well as research has been done \cite{24}. In fact the SOA paradigm\cite{64} which has been widely adopted within the e-Science field, originated in the Business Process field. This can be seen in the WSDL language being adopted for describing services, as well as the use of the SOAP protocol for communicating with webservices. Although the gridservice architecture defined in \cite{64} leaves open the possibility of another implementation, no complete alternative to SOAP in gridservices has been implemented. Another area where the Business Process community is influencing e-Science is workflow description languages. The most widely used workflow description language in the Business Process field is the Business Process Execution Language (BPEL), in \cite{61} it is shown that the BPEL for Web Services (BPEL4WS)\cite{35} is suited for use in e-Science Applications. BPEL4WS is a very expressive language, as can be seen in the comparison\cite{24} between different workflow definition languages from the Business Process domain. While within the BP field being able to express every form of business process and being able to reliably execute workflows are the main priorities, e-Science has additional priorities which sometimes take precedent over the BP priorities. First of all e-Science workflows have to deal with massive amounts of data and massive (parallel) computation. Furthermore sharing knowledge is important within e-Science, which places extra emphasis on knowledge transfer associated with workflows. Due to the knowledge transfer task of workflows in e-Science, proper representation plays a more important part in the design of these workflows.

### 3.3 Workflow Design

Within e-Science different approaches exist to compose a workflow:

- **Concrete**: the manual combination of a set of elementary workflow components into a workflow.

- **Abstract to Concrete Design**: given an abstract high level description of a computational task and a number of elementary workflow components, design a workflow that is equivalent.

- **Abstract to Concrete Construction**: given an abstract high level description of a computational task and a number of elementary workflow components let an automatic design process generate a workflow that is equivalent.
Abstract workflows can be a mechanism to share not only workflow components, but also common e-Science workflow patterns such as running processes in parallel. They can also be used to share knowledge on the design pattern associated with a very particular technique, for instance data assimilation which is used as a case study in chapter 7. Another technique employed in workflow construction is the composite workflow. In a composite workflow one workflow element can be an interface to another complete sub workflow. This helps in keeping workflows understandable. Formally this implies workflow components need to be compositional. The composite workflow should be equivalent to a workflow where hierarchy is removed as illustrated in figure 3.1. Most workflow systems only allow for computational workflow elements, however some[27] allow human activity to be represented in a workflow as well.

### 3.3.1 Workflow design using abstraction

In section 3.2 it was already mentioned that formal methods are applied most often in workflow research for reasoning about expressivity. They are used to determine:

- What control flow constructs are needed.
- The execution model and workflow language that enable these control flow constructs.

We study reasoning about representation as a hierarchical problem. In abstract to concrete design methods one has to be able to represent a workflow at different levels of abstraction. If one wants to reason about this in a formal way it puts different demands on a formalism then when one is considering
the expressivity of workflows in a concrete workflow design method. In a hierarchical representation abstract descriptions should be computationally equivalent to detailed low level descriptions of workflows.

Now we will define a number of important concepts in workflow design using abstraction.

**Problem definition:** This is the most abstract representation of a workflow. It consists of a definition of desired input, as well as the desired output. These can either be defined in terms of data or processes. In e-science context it can be viewed as the research question associated with an experiment.

**Atomic workflow component:** Represents a computational process for which there is a direct mapping to an actual implementation. This can be a deterministic process like adding two variables, or it can be a non deterministic process: a user entering a value based on the input he sees. In practice this usually is a web service, but it can also be an action performed by a human. For the rest of this chapter we assume an atomic workflow component to be computational. Furthermore as mentioned in the previous chapter, we assume it to be consistent. This means a workflow component must always be able to reach a final finished state and the content of its output must always remain consistent with the associated data definitions.

**Control Flow:** In a workflow data moves from one workflow component to another. The rules by which this data movement happens is called control flow. An example of such a rule is allowing or disallowing loops in a workflow.

**Execution model:** Not only does data need to be moved but execution of workflow components has to be started and stopped. The execution model takes care of orchestrating this execution and ensures only allowed
control flow constructs are used. Furthermore the execution model can be used to ensure workflow components are only connected when both components use the same data type.

**Compositionality:** is a property which can hold for workflow components, data connectors and a combination of both. Two or more atomic workflow components can be composed together to form a single composite workflow component. The behavior of this composite component can be explained in terms of its parts, by composing the parts together no new behavior emerges. Through composition internal actions are hidden from direct observation, the composite component presents itself to the outside only in terms of its inputs and outputs. Through compositionality complicated control flow patterns can be represented as one composite data connector. An entire workflow can also be represented as one composite workflow component.

**Workflow composition:** Workflow composition is achieved by connecting output and input of the workflow components through data connectors. These data connectors can be atomic communication channels, composite channels representing control flow patterns such as split or merge or in its most abstract form all interaction of a workflow can be one composite object with links to all workflow components. The composition of a workflow representation starts with a problem definition as well as a set of available atomic workflow components. The representation of a workflow needs to strike a balance between generality and specificity, resulting in an ideal workflow which is neither too abstract nor too detailed. This idea is illustrated in figure 3.2. Workflow design can be bottom up, top down, automatic or done by hand. To efficiently find a grounded design which satisfies the problem definition, the design space needs to be constrained. By grounded design is meant: a design that only consists of existing implemented workflow components, when represented in its most atomic form. In bottom up design the problem definition constrains the atomic workflow components that can be used in the first and last step of the workflow due to the fact that the inputs of the first step and outputs of the last step must match those of the problem definition. The outputs of the first step and inputs of the last step then form the constraints for the steps which can be used in between.

Within such a design process one should know what the design space looks like and what its limits are. We will look at this design space formally. For this we will use formal definitions of process and data set, of which workflow component and data connectors are respective instances. We show how a design space can be set up using composition of both processes and data sets. Note that this form of composition is more specific than that which happens in current workflow systems, where a combination of workflow components and connectors is composed together. In the design space that will be represented here both connectors and components can be composed separately.
3.4 Theoretical limits of Workflow design

In this section we give an informal treatment of the workflow design problem. We abstract from a lot of detail. As mentioned in the previous chapter when designing a workflow the goal is to answer a research question. In our treatment of the workflow design problem the research question is equivalent to a predefined computable function. The end result of the design process is a complex workflow, constructed from standard components, that is computationally equivalent to this function. As we will see this problem is in general unsolvable due to Rice’s theorem. Any formalism that tries to model automatic workflow composition in any generality is bound to fail. Yet the following definitions give a feel for the issues that are at stake.

3.4.1 Building blocks

We conceptualize a complex workflow as a collection of Turing complete processes which share data sets. The shared data sets may be thought of as memory locations, databases or variables:

Definition 3.4.1 (The class of data types) A class of recursive (effectively decidable) types $\tau$ defined on the class of binary strings $\Sigma = \{0, 1\}^*$ and closed under boolean operations.

Definition 3.4.2 (The class of data sets) A countable class $\Delta$ of typed data sets $d^\tau(t \in \tau)$ defined as memory locations with unlimited storage capacity.

Definition 3.4.3 A process is a deterministic computational function that has been defined in a Turing complete computational system. A process uses at least one dataset, to read its inputs and write its outputs.

As mentioned in definition and illustrated in figure 3.3 a process has at least one dataset on which to operate. One process is however allowed to manipulate more than one distinct dataset. This principle is illustrated in figure 3.4. This is needed because datasets will have to function as data connections in a workflow. Thus one process should be able to connect
3.4. THEORETICAL LIMITS OF WORKFLOW DESIGN

![Figure 3.4: Illustration of a single process with multiple datasets](image)

![Figure 3.5: Illustration of an elementary workflow component](image)

to multiple other processes. The connections in workflows are typed as is defined in the definition of datasets. A dataset can potentially accept multiple types. These complex data types are not trivial to implement, however using prefix-free or fixed length data types this is possible within the given definition of datasets. The possibility of using complex data types is needed later on, when the merging of multiple datasets into one abstract representation will be explored.

### 3.4.2 Workflow construction

Now that the basic building blocks have been defined we move to the construction of workflows starting with workflow components.

**Definition 3.4.4 (The class of elementary workflow components)**

An elementary workflow component $E = (I, O, D, P)$ consists of a computable function $P$ that takes the data set $I$ as input (read) and writes the result of the computation to the output data set $O$, where $i, j \in \tau$. We have $D = (I \cup O \cup D_{\text{int}})$ where $D_{\text{int}}$ are the internal data sets used by $P$.

Such an elementary workflow component is our formal equivalent to real world workflow components such as Kepler\[31\] actors, units in Triana\[96\] and processors in Taverna\[104\].

These workflow components are combined to form workflows. The way in which workflow components are combined has to be restricted in order to reflect the properties of actual workflow systems. That is why for our exploration of workflow design we are only interested in the class of consistent workflows, which is defined as follows:

**Definition 3.4.5 (The class of consistent workflows)**

A workflow is a set of elementary workflow components $W$. A set of elementary workflow components $W$ is consistent if for each $E = (I, O, D, P) \in W$: 

CHAPTER 3. WORKFLOW DESIGN SPACE

Figure 3.6: Illustration of example 3.4.7 an elementary workflow consisting of two elements connected through a shared dataset

- $I$ and $O$ are datasets of $P_i$ contained in $D$

- $P_i$ has an accepting computation for each possible variable assignment to members of $I$

- each computation ends in an assignment of values to members of $O$ that is consistent with their data-types.

As an example we show how the simplest possible elementary workflow component forms the identity workflow. The most basic workflow possible.

**Example 3.4.6 (identity workflow)** We take workflow component $E_i = (IO, IO, IOP_i)$ where dataset $IO$ is the only dataset. It is used for both input and output. Process $P_i$ has just one state which is both its start state and accepting state. It does nothing with the data, resulting in a workflow that passes its input data unchanged to its output.

It should be clear that this simplest of workflows is consistent. For every possible allowed input there is an accepting state and output that is consistent with its data type. Workflows usually consist of more than one element, so next we show how one can sequence two elementary workflow components in a consistent workflow.

**Example 3.4.7 (sequencing)** Suppose we have three different datasets $d_1, d_2, d_3$ and two processes $p_1, p_2$. With these we can construct two elementary workflow components $e_1 = (d_1, d_2, \{d_1, d_2\}, p_1), e_2 = (d_2, d_3, \{d_2, d_3\}, p_2)$. The output of $e_1$ is the input of $e_2$. If we define the datasets and processes to perform in the same way as example 3.4.6 we have a consistent workflow consisting of a sequence of two elementary workflow components.

3.4.3 Complex workflow construction

In the same way as the simple example of sequencing, more complex data connections can be set-up. It is not the purpose of this formalization to look at this type of expressivity as it has been shown extensively using other methods [23]. However we do want to look at the implications of employing more complex workflow patterns. The collection of all basic building blocks for the design of both simple and complex workflows is defined as a workflow repository:
Figure 3.7: Illustration of encapsulation of processes, an elementary workflow can be simulated on a computationally equivalent single process with multiple datasets

**Definition 3.4.8 (Workflow repository)** A workflow repository $R$ is a finite set of elementary workflow components $E_i = (I_i, O_i, D_i, P_i)$ with disjoint data sets $D_i$.

The construction of complex workflows out of such a workflow repository involves two highly non-trivial operations: merging of data sets and encapsulation of processes. We briefly discuss both.

**Encapsulation of processes**

Suppose we have two workflows $E_i = (I_i, O_i, D_i, P_i)$ and $E_j = (I_j, O_j, D_j, P_j)$. We define a new encapsulated workflow that is the result of a merge of the two workflows and that is intended to be computationally equivalent to the combination of the two original workflows:

$$E_{i||j} = (I_i \cup I_j, O_i \cup O_j, D_i \cup D_j, P_i||P_j)$$

Two types of computational flow are possible, consecutive where one component in $E_{i||j}$ starts execution after the previous one has reached an accepting state, or parallel where more than one component is executing at the same time. Consecutive execution of two or more $p \in P_i, j$ can be simulated by $P_i||P_j$ through appending the transition function of $p \in P_i||P_j$. Parallel execution can be simulated on $P_i||P_j$ through dovetailing transition functions of elements of $P_i$ and $P_j$ involved. The created transition function for $P_i||P_j$ emulates the transitions in the set of processes $P_i \cup P_j$ using the datasets $D_i, j$.

This operation is non trivial. It might be impossible to construct adequate scheduling for the merged routines and to prove the equivalence and in relation to the merge operation on data sets some of the data sets in $O_i, j$ and $I_i, j$ might have to be added to $D_i, j$. 
Merging of data sets

We can define a merge operation for data sets with the following redefinition of the signature:

\[ d_k \otimes d_l = D_{\{k\} \cup \{l\}} \]

This means we take the conjunction of the tests for the types and the union of the indices. A related merging operation can be defined for classes of data sets.

The merging of data sets will not result in an unacceptable rise in the computational complexity \( O \) of workflow components:

**Theorem 3.4.9** The time taken by a workflow component consisting of one dataset and one process to simulate \( n \) steps of a workflow component consisting of multiple datasets and one process is \( O(n^2) \).

For a proof, see [77] where the complexity of emulating \( n \) steps of a multi-tape Turing machine on a single-tape Turing machine is shown to be \( O(n^2) \).

**Lemma 3.4.10** A consistent elementary workflow can always be emulated on a workflow component consisting of one process and one dataset, where \( n \) steps of the elementary workflow take no more than \( O(n^2) \) steps of this workflow component.

This is a consequence of definition 3.4.3 and theorem 3.4.9.

Due to the non trivial nature of the encapsulation operation employed the workflow component in lemma 3.4.10 is not necessarily consistent. The merge operation is also non trivial. By means of merging of data sets we create links between elementary work flows. Even if the types of the data sets are the same, timing issues might cause the data to be corrupt or overwritten.

**Execution model**

Generally speaking in order to construct a complex workflow on the basis of a workflow repository a number of issues have to be dealt with:
3.4. THEORETICAL LIMITS OF WORKFLOW DESIGN

- The creation of a workflow topology by means of merging input and/or output data sets (possibly of different signature) and the encapsulation of processes.
- The selection of an execution model for the topology.

The execution model deals with the timing and scheduling issues mentioned in the previous sections on encapsulation and merging operations. In general it is the equivalent of a director in the Kepler workflow system or the data flow mechanism which is implicit in most other workflow management systems. In the rest of this chapter we will concentrate on topologies as they are the part a workflow designer manipulates. Execution models are also important as they deal with the timing and scheduling issues mentioned previously. The workflow designer has to select a suitable execution model, however the construction of an execution model is not his domain. Therefore, it is assumed that for the class of consistent workflows at least one consistent execution model exists for each workflow. With this assumption in place additional limitations to workflow design still remain.

3.4.4 Workflow design limits

Any concurrent computational process that is associated with a workflow can exist in at least three extreme guises:

- A consistent elementary workflow $W$ based on a multitude of processes combined in a complex topology.
- A workflow component consisting of one process and multiple datasets
- A workflow component consisting of one process and one dataset.

This analysis gives us the coordinates of a design space for workflow components. Although it must be noted that this domain does not allow the construction of composite workflows, it allows us to study a number of design issues in a formal context. The workflows can be ordered along two dimensions: data-complexity and process-complexity. The four extreme corners of these dimensions are:

- Multi Process, Multi Data (MPMD) workflow: the consistent elementary workflow $W$ based on a multitude of processes using a multitude of data sets.
- Single Process, Multi Data (SPMD) workflow: A multi dataset process. This is often the start of an abstract-to-concrete design process: the application is conceived as a single complex process working on a multitude of data sets.
CHAPTER 3. WORKFLOW DESIGN SPACE

Figure 3.9: Lattice of all possible representations for a computationally invariant workflow

- Single Process, Single Data (SPSD) workflow: A single dataset process. Conceptually this might be interpreted as a classical compiled application working on a dedicated database.

- Multi Process, Single Data (MPSD) workflow: this is a degenerate case in which a multitude of processes use one dataset. One might think of a collection of agents managing a single database.

This domain describes implementation variants for routines that are computationally equivalent. Both dimensions generate a lattice-like structure. If one starts with a MPMD workflow one can gradually combine either processes or datasets to create a SPSD workflow in the limit. This is illustrated in figure 3.9 where each arrow indicates either a combination (or split) of a dataset or process. The combination of maximum generality and simplicity is to be found within an SPSD workflow. All different workflows are computationally invariant, i.e. they compute exactly the same function and are of comparable complexity (according to theorem 3.4.10). The only reason to favor one version in the domain over another is a matter of desirable design qualities.

It must be noted that this domain is far too complex to be characterized by any finite analysis. Thus instead of a precise analysis of all design issues, we concentrate on some important issues.

We can study the influence of parameterization in the context of this domain. Each combination of two processes into a single more abstract process representation leads to extra parameters being created in this more
3.4. THEORETICAL LIMITS OF WORKFLOW DESIGN

abstract representation. Each time multiple datasets are replaced by a single dataset, this single dataset has to accept additional types. This leads to more complex data types. In the ultimate SPSD form we end up with something that can take the description of any process with any data and execute it. In other words a universal Turing machine. Such a universal Turing machine \( T_U \) is the ultimate parameterized routine.

That there are limits to generalization by means of parameterization is clear on the basis of the Halting set. There is no recursive routine that in general will decide whether a Turing machine will stop on a certain input.

This is bad news for workflow designers since the universal Turing machine is the ultimate SPSD workflow. It gives rise to the following claim:

**Claim 3.4.11** In workflow design simplicity, generality and consistency are mutually exclusive.

In other words a workflow that is both general and simple will be inconsistent. Another way of picturing the situation is the following: suppose one starts the design of a workflow system on the basis of an MPMD representation. By way of merging datasets (thus creating the necessity for complex data types) and merging processes (by means of adding parameters to the data types) one can create more general workflows using less datasets and less processes. The price one has to pay is computational complexity and dataset complexity: the ultimate SPSD workflow is undecidable and thus inconsistent. In other words: simplicity and generality imply complexity of both the data and the computational processes.

We can now define the general abstract to concrete workflow design problem: given the description of a computational process and a number of elementary workflow components can we construct a workflow with a topology that is computationally equivalent? In other words: given a SPMD workflow, can we automatically construct an equivalent MPMD workflow? Formally:

**Definition 3.4.12 (General workflow design problem)** Given a workflow component consisting of a single process and multiple datasets \( E_i = (I, O, D, p_i) \), where \( D \) is a collection of multiple datasets and given a repository \( R \) of elementary workflow components employing a set of processes \( P \) using \( D \) as datasets can we construct a workflow \( W = (I, O, D, P) \) that is computationally equivalent to \( p_i \)?

Due to Rice’s theorem[111] this problem is undecidable. Rice’s theorem states that for every non-trivial property of any partial function it is undecidable whether any other partial function computes this property. The computational equivalence of two workflow representations is non-trivial in almost any case, the behavior of a workflow does not hold for all partial functions nor for none. Thus for the general case of workflow representation
this means automatic workflow composition is not possible. Only when serious constraints are placed on the properties of workflow components and data connectors does this become possible.

3.5 Discussion

Automatic workflow composition in the general case is not possible. Under certain restrictions however, there can be a role for automatic workflow composition. From a software engineering standpoint this seems attractive (it can drastically reduce design time). On the other hand its role will always be a limited one, since it cannot guarantee the correctness of all aspects of a workflow. From a methodological point of view automatic composition is not always desirable. Within e-Science it is still the responsibility of the scientist performing an experiment to make sure that it is scientifically sound. Ensuring user transparency becomes increasingly important when automatic composition is employed. Thus in practice it is not always desirable to have an abstract workflow which is specified to such a degree that a consistent concrete workflow can be automatically derived from it. Letting the user interactively add information to the abstract workflow specification allows for a more general applicability of an abstract workflow and also ensures that the user knows what is happening. This would be especially true when the abstract workflow takes the form of a design pattern such as parameter sweep or data assimilation which can be employed in many different domains.

To formally verify whether a workflow satisfies the problem definition and behaves correctly, all atomic workflow components and their connectors have to be taken into account. This is to be expected when no part of such a workflow has been constructed before. However, when there is the possibility of reusing previous designs, formally verifying the reused parts, which were already verified previously, is inefficient. The workflow design lattice introduced earlier in this section offers a solution. By using the property of compositionality previously created workflows and connectors can be abstracted to single components. Compositionality can be used as the mechanism to abstract both data and process in the workflow design lattice. If these abstracted components have been verified, their internal actions will not have to be verified again when they are reused in a new design. They have been shown to satisfy their problem definition, thus this can be used for defining the composite component, leaving out all the details of what happens internally. The big problem however, as already identified for automatic workflow composition, is that the correctness of an entire workflow is undecidable for the general case. In the workflow design lattice proving one representation is computationally equivalent to another is impossible due to Rice’s theorem in the same way that it is impossible to prove a specific
Figure 3.10: checking whether sub workflow with different execution model implements sub problem description

workflow satisfies the requirements when performing automatic workflow composition. Thus when using any formal approach to workflow verification, a careful approach has to be taken in which constraints are placed on the expressiveness of the formal model of a workflow. In the next chapter we shall see that different formalisms take different approaches in representing workflows. Each with benefits and limitations when trying to verify different aspects of a workflow design.

The issue of execution models was skipped in the formal approach represented here. For setting up the lattice representing the workflow design space it was not necessary to go into much detail concerning execution models. However any scientist who actually needs to formally check the validity of his workflow but also wants to use the hierarchical abstract to concrete design method presented, will need a formalism which can handle execution models in a practical way. A very good example where both abstract to concrete design and execution models come into play is the use of multiple execution models. When creating a representation of a workflow with multiple models of execution (illustrated in Figure 3.10) one can first create the workflow using execution model A and leave the sub problem, which can only be solved under execution model B, as a problem definition. In other words the sub problem is left as a stub to be resolved in a different
workflow system with a different model of execution. One often used model of execution is data-flow which allows for high data throughput, but puts limits on expressivity. Sometimes both expressivity and high throughput are needed in one experiment. As an example: let A be a data-flow based system and B a less constraint execution model which allows a user to steer execution. By allowing this, system B possibly violates the constraints of system A. If this is the case then the composition of WFA and WFB is not valid. This is clearly an undesirable situation, for which the simple solution is not using system B within A. In practice however it can often be the case that execution model B could violate the constraints, but that the actual sub workflow that was created under B does not. It can therefore be useful to test for this case.

The definitions of process and dataset presented in this chapter are not the most practical way of formally reasoning about actual instances of workflows as used by an e-Scientist. That is why in the next chapter there is a review of formalisms which can be used in the workflow design process itself. Such a formalism needs to have certain properties. First of all it needs to be able to abstract both details of process and communication through composition. It also needs to be expressive enough to express all possible workflow patterns, but also all possible processes which can be workflow components. Practical tools such as simulators, model checkers and theorem provers have to be available for such a formalism. Preferably these tools are already tailored towards workflows.

3.6 Conclusions

In this chapter we presented a number of evaluation criteria for workflow components and we constructed a domain of computationally equivalent workflows with different process and data complexities. We have shown that computationally equivalent workflows can be evaluated in terms of two dimensions: data complexity and process complexity. Using this formal framework we have proved that maximal simplicity, generality and consistency are mutually exclusive. We have defined a formal version of the General Workflow Design Problem and have shown that this problem is undecidable in the general case thus putting limitations on formal verification of workflows. We have shown that this limits the applicability of automatic workflow composition, but also calls for careful selection of formalisms used in a abstract to concrete design methodology.
Chapter 4

Workflow formalisms

4.1 Introduction

Within e-science research, workflow environments are often viewed as a practical tool for building experiments. Right from the inception of these systems, formal foundations for workflows have been constructed [31, 79]. One of these first formal foundations provided the underpinnings for data flow based execution models and were based on Kahn Process Networks[79]. For cases where more expressiveness was needed, for instance in hierarchical composition or more advanced control flow, scientific workflow management systems have turned to Petri Nets[105], see e.g. [29]. Similarly in the business workflow community[24] where there has been more emphasis on expressiveness from the outset, the most commonly used formalism is Petri Nets.

With the increasing complexity of workflows and the increasing abundance of available services to be used in their construction, new issues have arrived which need a formal basis as well. As already introduced in chapter 2, there are two very important criteria in scientific experiments, correctness and reproducibility. The formal foundations of a workflow environment should thus be able to prove that these criteria are met in an experiment. The ever expanding amount of web-services means that determining whether a web-service is suitable and correct for use in an e-Science experiment should be automated as much as possible. Not only is checking the type and semantics of a service’s connections needed, the runtime behavior also has to be correct within the workflow where it is employed. The increasing complexity of workflows, for instance using multiple execution models within one experiment [93, 135] also increases the complexity of assuring reproducibility. Questions that need to be answered include: how much provenance data needs to be recorded and how can the interaction between different workflow systems be coordinated? In order to facilitate such interactions a formal description of the workings of workflow systems
CHAPTER 4. WORKFLOW FORMALISMS

such as offered by Kepler[47] and Taverna[102] is helpful. The new problems mentioned above involve more than just the expressivity allowed by the execution model, and call for their own formalized description. This may involve additional and different formalisms than the formalisms now common in workflow research. Reasons for choosing a formalism for these problems are not just the ability to express these problems, but also the availability of tools to support reasoning about these problems in their desired formal representation. In this chapter we investigate existing formalisms both from workflow research as well as formalisms originating from parallel computing and software engineering. This chapter is based on a previous publication by the author of this thesis[122].

4.2 Problem domains

In e-Science, workflows are not just describing existing experiment processes, as business workflows describe existing business processes. They are meant to explore new experiment ideas. The design of these experiment workflows can be top down, bottom up or a combination of both. In a top down approach one starts with a very abstract design definition and gradually through refinement adds more details. With a bottom up approach one starts with the atomic building blocks and gradually creates a more abstract design by merging building blocks into compositions. During this design process abstraction and refinement of the workflow design play a crucial role. Within workflow systems such as ICENI[72] or Chimera[66] automated abstract to concrete workflow design is possible. Many workflow systems offer hierarchical composition where an entire workflow can be a workflow component, or sub-workflow, of another workflow. This concept becomes more interesting when a sub-workflow uses a different model of computation compared to that of the main workflow. It can be taken one step further by having a workflow component which is actually a workflow in a different workflow management system. The workflow-bus that is also being developed within the VL-e project, is an example of this[135]. Both Triana[96] and shortly Taverna[104], offer the option of exposing a workflow as a web-service. This allows a different workflow management system to use this web-service as a workflow component in its own workflow. Another problem which can be analyzed using formal methods is that of connectivity. The goal here is to determine which workflow components can be connected to form a valid workflow. This problem has two parts, first both data type and data semantics must match. Secondly there is the runtime aspect: are two components compatible at the connection level when they are executing? This second part comes to the foreground when dealing with multiple models of computation within one workflow. It needs to be verified whether runtime behavior which is allowed under one model of computation
is acceptable to a component operating under another model of computation. This is also important in the reuse of workflow components. The issue in this case is to test if a workflow, that has been used for one experiment with certain data, can be reused in another experiment which tries to find answers to a different problem with different data. Finally, within science and therefore within e-science reproducibility of an experiment is very important. To ensure the reproducibility of an experiment, provenance data is recorded during the execution of an experiment. When dealing with an experiment using multiple workflow management systems, the question is what provenance data needs to be shared between systems to maintain reproducibility.

4.3 Formalisms

In this section we will introduce the formalisms and give a comparison of their features. In the discussion we will address the suitability of the formalisms to the problems introduced in the previous section.

**Petri Nets** are based on the work described in the Phd thesis of Carl Adam Petri in 1962[106]. His thesis deals with the asynchronous communication between components of a computer system. In its basic form Petri Nets are a graph based formalism, consisting of places, transitions as well as input and output functions. This is the starting point for numerous extensions which give Petri Nets additional properties such as Turing completeness through the addition of a zero test arc[105], explicit data through “colored” Petri Nets[105], as well as hierarchy[90, 129, 30]. Petri Nets have been used frequently as a formalism in workflow research[24, 76].

**I/O Automata** were first introduced by Lynch and Tuttle [95], and have been used for the study of concurrent computing problems. They form a labeled state transition system consisting of a set of states, a set of actions divided into input- internal- and output actions (as illustrated in figure 4.1) and a set of transitions which consists of triples of state, action and state. This allows us to study the inherently concurrent nature of workflow systems. One of the characterizing properties of I/O Automata is that input actions are "input enabled", they have to accept and act upon any input. Figure 4.2 illustrates both that the "input enabled" property defines connections between automata as well as one I/O automaton being computationally equivalent to a composition of several automata. In previous work we have argued that I/O Automata are suitable as a workflow formalism [120].

**Turing Machines** were introduced by Alan Turing in 1936 [124]. They consist of a tape for storing data, a head for reading and writing, a table of instructions and a (finite) state register which stores the state of the table. Turing machines are a very general formalism which was used previously [121] to reason about the workflow design space.
CHAPTER 4. WORKFLOW FORMALISMS

Figure 4.1: Visual representation of an I/O Automaton

Figure 4.2: Illustration of compositionality principle using I/O Automata
4.3. **FORMALISMS**

**Constraint Automata** were introduced in 2003 by Arbab, Sirjani, Rutten and Baier[39], as a means of providing formal semantics and analysis of component connectors. Component connectors are a means of connecting software components. This work is very similar to workflow design. Constraint Automata are similar to I/O Automata but with some important differences. For instance Constraint Automata are not input enabled and do not follow a strictly time synchronous approach. These differences will be shown in more detail in the upcoming comparison.

**π Calculus** was introduced by Robert Milner in 1989[101]. It belongs to the family of process calculi, just as λ Calculus[85] and CCS[100] (also created by Milner). The specific task it tries to address is the description of concurrent processes whose configuration can change during computation. It has been used for describing business workflows [114]. There has been some discussion about its suitability particularly in comparison to Petri Nets[126].

### 4.3.1 Overview

**Turing completeness**

A formalism is Turing complete when it is able to represent all possible computational processes: Turing machines and λ Calculus are the classic examples. As there was no reference in literature for the Turing completeness of I/O Automata we constructed our own, it is detailed in appendix B. Petri Nets are only Turing equivalent when they are extended to include a zero test arc [105]. π Calculus is Turing complete as long as recursion is allowed [57]

**Graphical representation**

For purposes of dissemination it is nice to have a standard graphical representation. Petri Nets have graphical representations which are commonly used and are well suited to representing workflows. For Turing machines standard representations do exist visualizing the tape and write head, however this is not very well suited as a workflow representation. I/O Automata have no generally used visual representation. For both π Calculus and Constraint Automata visual representations suitable for workflows are available.

**Data**

Whereas in Turing machines data is written explicitly to the tape, in both I/O Automata and Petri Nets data is implicit in the state. Unless for Petri Nets an extension is used, such as colored Petri Nets[105], which explicitly model data. In Constraint automata data is explicitly driving the execution through the use of Timed Data Streams. Within π Calculus data is transferred explicitly over named channels.

**Process**

Processes are modeled explicitly in Turing machines through explicit instructions for reading and writing. In both Petri Nets and I/O Automata processes are modeled by explicit transitions, while in π Calculus and Con-
<table>
<thead>
<tr>
<th>Feature</th>
<th>Turing Machine</th>
<th>I/O Automata</th>
<th>Petri Nets</th>
<th>Constraint Automata</th>
<th>π Calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing completeness</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>only with zero test</td>
<td>yes</td>
</tr>
<tr>
<td>Graphical representation</td>
<td>not suited to WF</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Data</td>
<td>explicit</td>
<td>implicit</td>
<td>implicit</td>
<td>explicit</td>
<td>explicit</td>
</tr>
<tr>
<td>Process</td>
<td>explicit</td>
<td>explicit</td>
<td>explicit</td>
<td>explicit</td>
<td>explicit</td>
</tr>
<tr>
<td>Orientation</td>
<td>process</td>
<td>process</td>
<td>process</td>
<td>data interaction</td>
<td>process</td>
</tr>
<tr>
<td>Non-determinism</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
</tr>
<tr>
<td>Implicit multiple instances</td>
<td>not possible</td>
<td>not possible</td>
<td>possible</td>
<td>not possible</td>
<td>not possible</td>
</tr>
<tr>
<td>Compositional</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
</tr>
<tr>
<td>Simulator</td>
<td>available</td>
<td>available</td>
<td>wf specific</td>
<td>wf specific</td>
<td>available</td>
</tr>
<tr>
<td>Model checker</td>
<td>available</td>
<td>available</td>
<td>wf specific</td>
<td>wf specific</td>
<td>available</td>
</tr>
<tr>
<td>Theorem prover</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Workflow patterns</td>
<td>not yet shown</td>
<td>not yet shown</td>
<td>many shown</td>
<td>many shown</td>
<td>many shown</td>
</tr>
</tbody>
</table>

Table 4.1: Features of formal workflow models compared
4.3. FORMALISMS

Turing machines are deterministic by definition and can therefore not directly represent non-deterministic processes. A Turing machine can always simulate a non-deterministic process. A Turing equivalent variation exists, called non-deterministic Turing machines, that can represent non-deterministic processes. All other formalisms can directly model non-determinism.

**Implicit multiple instances**
The execution of a petri net can exist of multiple instances of the same petri net, this can occur implicitly through the structure of the petri net, while in other formalisms this has to be specified explicitly.

**Compositional**
Compositionality is a property which can hold for workflow components, data connectors and a combination of both. Two or more atomic workflow components can be composed together to form a single composite workflow component. The behavior of this composite component is exactly that of its parts. By composing the parts together no new behavior emerges. Through composition internal actions are hidden from direct observation. The composite component presents itself to the outside only in terms of its inputs and outputs. Through compositionality complicated control flow patterns can be represented as one composite data connector. An entire workflow can also be represented as one composite workflow component. This definition was already introduced in chapter 2. In the same chapter it was also shown how Turing machines support this. Many different ways of performing hierarchical compositions exist as extensions to Petri Nets (for instance [90, 129]). Compositionality is supported by I/O Automata with some restrictions on which elements can be composed together, this in order to avoid ambiguity during execution. Constraint Automata are fully compositional. There are no limits on the composition of either processes or communication. In π Calculus processes are compositional, however there is no way to explicitly provide composition for interaction between processes.

**Refinement**
For I/O Automata it is always possible to more precisely define the actions, states and transitions of a particular automaton without changing the basic topology of a composition. In Petri Nets it is possible to refine data types by using colored Petri Nets, an extension of Petri Nets. Refining places and transitions in Petri Nets can be done with another extension [129, 90].

**Simulator**
Using a simulator and a formal description of a workflow one can check for:

- Reachability: whether a certain state in a workflow component can be reached.
• Safety: there are no undesirable terminations possible.

• Deadlock: test whether the workflow is free of potential deadlock situations.

• Bottlenecks: do certain transitions take a lot of time.

The simulator for I/O Automata[16] can perform all tests except the last one as it has not got an explicit notion of time. Constraint Automata are a formal model for connectors in an environment called Reo[39, 19]. This environment can potentially be used not only for simulation but also as an actual workflow engine. For π Calculus there are several simulators developed for Biochemical processes ¹ but they are not suited to workflow modelling. A company called Intalio is developing Business process workflow tools based on π Calculus, which will include a simulator at some point in the future. Many different simulators exist both for Petri Nets and Turing machines, including ones for timed Petri Nets that can deal with bottlenecks.

Model checker

Model checking tools are available for I/O Automata[16], Petri Nets[127], Constraint Automata[86], and π Calculus[131]. The model checker for I/O Automata does so in a compositional way, where as the Petri net model checker does not do this as of yet[127].

Theorem prover

Theorem provers are currently available for I/O Automata[103]. Several theorem provers for Pi Calculus also exist, such as [97]. They are however still work in progress. Work is in progress² on a theorem prover for use with Petri Nets specifically aimed at business workflows.

Workflow patterns

Workflow patterns that demonstrate the ability of a formalism to succinctly express existing patterns in (business)workflows have been demonstrated in great detail for:

• Petri Nets: demonstrated in great detail at the workflow patterns webpage³

• π-calculus: with slightly fewer patterns on the pi workflow homepage⁴

• Constraint Automata: some patterns available in the related Reo format⁵

Constraint Automata can more easily express some of the more complicated patterns due to the fact that they can represent asynchronous and synchronous interaction simultaneously. For other formalisms these patterns

¹http://www.wisdom.weizmann.ac.il/~biopsi/psi.htm
²http://wwwis.win.tue.nl/~movebp/description.html
³http://www.workflowpatterns.com
⁴http://www.pi-workflow.org/
⁵http://homepages.cwi.nl/~proenca/webreo/
4.4 Discussion

In this chapter we have given an overview of formalisms that can provide the formal analysis for workflow design issues given in the previous chapter. The question now is which formalism is best suited to these problems. Let us first consider Turing machines: from the overview it is clear that they are expressive enough in theory yet they are not a practical solution for modeling real workflows. There are no tools written for modeling workflows as Turing machines. The composition of process nor that of data is practical when modeling real problems. It is theoretically always possible to merge tapes or simulate multiple Turing machines on one machine, however this is not a practical solution when Turing machines accurately model real data or processes. Turing machines have their place in purely theoretical proofs, but not in a design approach that results in real workflows. In the overview of formalisms Constraint Automata are oriented to describing interaction between processes, while the other formalisms are oriented towards describing the process. The consequence of this is that the problem definition, the most abstract representation of a workflow, is defined most easily in terms of data interaction for Constraint Automata. For the other formalisms a process description is easier.

We have argued that compositionality can be a very useful property when analyzing workflow problems. All formalisms support composition in some form: in Petri Nets it is an extension for which multiple solutions exist, in I/O Automata and Constraint Automata it was included from the outset. The advantage of this is that many of the tools associated with these formalisms also work in a compositional way while for Petri Nets this is not always the case. A Petri Net model checker\cite{127} aimed at workflow analysis, for instance does not yet work in a compositional way.

Defining data connectors in a compositional way using a process oriented formalism is more difficult, as data connections need to be defined in terms of processes. This is especially true when data is already explicit which is the case for $\pi$ Calculus. In Petri Nets a composition of only places, or only processes is not possible. In I/O Automata it is possible to abstract data connectors and processes separately if one defines all data connectors in terms of processes. Even then there are still some limitations on the compositionality of I/O Automata: not all automata are allowed in the same composition to avoid ambiguity during execution. In practice this could be avoided by being careful with the naming of the actions inside I/O Automata.

For Petri Nets, $\pi$ Calculus and Constraint Automata the expressiveness
has been shown through workflow patterns. For business workflow management systems there are overviews of which system supports what patterns. This is a very significant help when defining the model of computation for a certain workflow. A similar analysis for scientific workflow management systems should be done. This will probably yield a less diverse set of patterns as most are data flow based. For the other formalisms accurately describing the model of computation will be more complicated work. There is some past work which can be of help. In [94] it is shown how the Kahn principle can be modeled using I/O Automata. The main constraints are that the automata have to be deterministic and all connections are one to one. Kahn process networks are similar to many of the data flow models of computation used in scientific workflow management systems.

It is preferable that any tools associated with a formalism exist in a form suitable to workflows. This is most certainly the case for Petri Nets which have long been used for reasoning about workflows. Reo, the tool associated with Constraint Automata is developed for distributed systems composed of heterogeneous mobile black-box components. While these systems encompass more than just workflows, they do fit well within that description. Tools for Constraint Automata are thus very well suited to workflows. The tools for I/O Automata are more general still and lack specific workflow features. Workflow specific tools for π Calculus are still largely under development.

### 4.5 Conclusions & future work

We presented an approach to workflow design at multiple levels of data and process abstraction. It is desirable to formally reason about workflows in this way and prove the correctness of complicated workflows: for instance those workflows involving multiple workflow engines and execution models. We can determine which workflow systems can safely be used to provide sub workflows for other workflow systems. In case this is not safe in general we can determine whether a specific sub workflow can still be used safely. An overview was given of formalisms which can be used for workflow design. From this overview it becomes clear that Constraint Automata appear to be most suited to our approach at this time, they have the best support for compositionality: they are the most expressive and have tools available suitable for use with workflows. π Calculus has its place for different design approaches where there is less emphasis on compositionality of data interaction. I/O Automata offer almost the same support composition, but are less suited to expressing data interaction. Furthermore they need a lot of work both in proving expressive capability as well as in the development of workflow specific tools. Petri Nets are well established as a workflow formalism, but they were not intended to be used in a compositional manner from the outset, this shows in diminished compositional ability both in the
formalism itself and associated workflow tools.
Chapter 5

Workflow Systems Analysis

5.1 Introduction

E-science environments, which enable scientists to achieve the e-science goals of massive computing, massive data handling, virtual organizations and shared resources consist of a basic framework of different layers of middleware. These different layers of middleware manage Grid resources, computing tasks, data and information. On top of these there is a scientific workflow management system (SWMS), which gives the scientist support in executing experiments using the resources made available by the different middleware. A SWMS is crucial for the introduction of e-Science to scientists in specific application domains [80, 52]. There are many reasons for employing a SWMS:

- Facilitating scheduling and management of computing tasks
- Automating flow control between computing tasks, thereby reducing the administrative work for scientists
- Through the use of meta data, allow interfacing with distributed resources and services for integration and customization within specific application scenarios.

The most important one however is to allow a scientist to interact with an experiment in an e-Science environment without having to focus on the networking, (parallel) computing and data management details. A SWMS should allow a scientist to focus on the high level domain specific aspects of the experiments being performed. In this chapter an overview is presented of the elements that make up a SWMS, and how they relate to sharing software resources. This is followed by a look at the state of the art in workflow systems for E-science, which includes discussion on which state of the art system provides the most suitable context for shared software resources. Finally important future directions in research are given for areas
CHAPTER 5. WORKFLOW SYSTEMS ANALYSIS

of SWMSs which need further development in order to achieve the e-Science goals.

5.2 Scientific Workflow Management Systems

There are many workflow management systems in existence today, both in business and scientific fields. Business workflow systems differ from scientific systems in that they focus primarily on the business processes. Scientific systems on the other hand focus mainly on massive data involved in e-Science[93]. As the research in this thesis concerns scientific workflow management systems the following description is specific to these systems, although many elements might also be found in a business oriented system.

5.2.1 Workflow lifecycle

A workflow being performed within a SWMS has a lifecycle consisting of three distinct phases[82]:

- Design
- Execution
- Analysis

Each of these phases imposes a different set of requirements on the SWMS. Within the design phase the workflow is defined at a high level by an application scientist. The order of all of the steps involved, the way these steps are connected and their parameters for a particular experiment are defined in this phase. Most SWMSs offer a graphical programming environment to keep this task accessible to scientist who are not experts at programming languages. Workflows can either be defined as abstract, without explicit reference to the underlying resources used, or concrete where these resources are defined by the user. Some SWMSs allow for hierarchical composition where one workflow-step is actually an interface to an underlying sub-workflow. This is done to keep a clear overview in complex workflows. Also included in the design phase is resource development. This involves programmers who make software components or other resources such as scientific instruments available for use in a workflow. This usually entails developing the resource as a web or grid service according to an API which is supported by the SWMS. The result is that a resource is able to communicate, according to defined standards, with other resources and that meta data is defined which defines how a resource can be used.

The execution phase handles the enactment of a workflow. This involves mapping high level workflow steps to appropriate low level grid resources and orchestrating the runtime behavior as defined in the composition phase.
5.2. SCIENTIFIC WORKFLOW MANAGEMENT SYSTEMS

Analysis phase involves actions after a workflow has finished its execution such as abstracting a template from a successful experiment for future use or changing resource meta data based on experiment results.

From the above description of the workflow lifecycle it should be clear that a SWMS involves different layers of middleware. In order to support these three phases three functional components can be distinguished within a SWMS. These are the workflow model, workflow engine and user support. Figure 5.1 gives a schematic overview of the middleware and the place of the three functional components. We will now proceed to describe each of the three functional components in detail.

![Figure 5.1: Functional components of a Scientific Workflow Management System.](image)

5.2.2 Workflow model

The workflow model is an essential component for separating the application logic from resource functionality, allowing a scientist to model a workflow from a high level of abstraction. This is achieved by defining a standard to describe application scenarios. We will start by viewing this from high level concepts and work down towards actual languages used. Two main types of workflow models can be distinguished: data flow and control flow oriented. In data flow based models the order of execution is determined by data arriving at the next step in the workflow. This means that in the composition phase workflow steps are most logically arranged in a temporal view: each workflow step can only start executing if the previous step has started to produce data. In control flow based models the execution is determined by control statements such conditional statements (if then else) or loops (while, for). This means that in the composition phase a workflow is most intuitively edited in a process logic description view, allowing for easy
explicit definition of parallelism and more complex workflows in general. For the actual description of the processes different methods can be employed. Directed Acyclical Graphs (DAG) are the common process definition for Data flow based models [13, 3], while control flow based models are usually based on Directed Graphs [31, 32, 68] or in some cases on Petri Nets [70]. This process definition is captured in some kind of formal language, almost always XML based. Some SWMSs actually allow a user to directly define a workflow in this language [68]. Most SWMSs automatically derive the XML based workflow definition from a graphical programming environment. In some cases the workflow model not only describes the application logic but also some parts of resource functionality for instance the preferred scheduling strategy can be defined or a specific resource can be coupled to a workflow step.

5.2.3 **Workflow engine**

The workflow engine is responsible for the execution of workflows. It maps workflows onto computing resources, generates concrete computing tasks, schedules flow execution, controls runtime behavior and ensures quality of service. These tasks will be more elaborately explained in the following four topics: enactment and planning, scheduling, orchestration and service quality.

Workflow enactment also sometimes known as planning takes a workflow description and maps unto underlying resources. For this task it needs intelligence which takes the workflow semantics and the availability of resources into account when performing this mapping. The complexity of dealing with workflow semantics can vary between different SWMSs: some have workflow descriptions at multiple levels, an abstract one for the end user to compose and a concrete one to execute, others let the end user compose a concrete workflow. In cases where an abstract workflow, without any references to specific resources, is presented to the engine a lot more intelligence is needed to generate a concrete workflow. To reduce this complexity human assistance can be used to move from abstract to concrete workflows. The other important function that has to be performed during enactment is the discovery of resources. The complexity of this task can vary according to the homogeneity of the e-Science environment. This task can be a complex proposition when dealing with virtual organizations consisting of multiple partners, each having different restrictions associated with gaining access to their resources.

Scheduling a workflow presents another challenge to the workflow engine. Whether the SWMS is a centralized or a decentralized system plays an important role in this system. In a centralized system with one scheduler for the entire e-Science environment scheduling is simpler to realize than for a system in which the workflow engine has to deal with multiple sched-
5.2. SCIENTIFIC WORKFLOW MANAGEMENT SYSTEMS

ulers. This last scenario can however be unavoidable in a heterogeneous environment. Thus apart from offering a scheduler with scheduling strategy a workflow engine can also offer additional features such as reserving cycles on resources not directly under its control, or offering dynamic scheduling that changes the workflows schedule while it is executing.

The orchestration of a workflow is the process of controlling the runtime behavior of resources as defined in the workflow description. This can be done through a centralized coordinator or conductor, a practice common in control flow oriented SWMSs, or it can be done implicitly through the dependencies and information flow between computing tasks, which is an approach taken by many data flow oriented SWMSs.

Finally, service quality deals with an issue already hinted at in scheduling: fault tolerance. There are many cases imaginable where the workflow has to deal with dynamic execution. When a resource suddenly becomes unavailable or the execution of a software component unexpectedly fails, the workflow engine has to handle the situation. For instance: through dynamic (re)scheduling of a workflow, but also by offering fault tolerance features such as rollback and checkpointing. Moving up from fault tolerance, a workflow engine should allow the dynamic composition of workflows, changing workflow components or data while the workflow is executing. Although this last item has been discussed in the context of some SWMSs [104, 130] no usable implementation exists at this time.

5.2.4 User support

At all phases in the workflow lifecycle a SWMS user can expect different forms of user support. The support can have different aims. Therefore we have divided this support into three categories:

- Passive support, giving greater understanding of what is involved in an experiment.
- Interactive support, enabling easy manipulation of the processes involved in an experiment.
- Automated support, completely shielding the user from certain processes allowing him to focus on other things.

What follows are details on the user support available at each phase of the workflow lifecycle and the demands this support places on the workflow model and workflow engine. The design phase has been split into workflow and resource design.

Workflow design

Workflow design is the phase in which an end user scientist will spend most of his time. Most of the user support will therefore be offered here. First of
all an end user should have a clear idea of what the SWMS and the resources he plans to use are capable of. This is achieved through passive user support consisting of clear documentation for the composition environment as well as the workflow steps from which the workflow is composed. During the composition the SWMS can offer interactive support, pointing out errors in the validity of a workflow through type and protocol checking. For instance: when one step outputs integers and it is connected to a step which expects doubles this is clearly wrong. This can be pointed out to the user either by refusing to make the connection in the first place or by giving some form of warning. This can be extended by the use of more sophisticated meta data, for instance when a step that outputs temperature in Fahrenheit is connected to one which expects Celsius as input. This can also be pointed out to the composer of the workflow. A second form of interactive support which is closely related, is semantic search, where a user can search for resources based on the meta data associated with it. For instance: when looking for a next step in a workflow, semantic search can support the user by offering only those workflow steps which can connect with the current workflow step. Finally parts of the composition process can be automated. In SWMSs with a multi-layer model where there is a high level abstract workflow description and a lower level concrete description, the transformation from abstract to concrete can be performed automatically. This process as it is implemented in systems such as ICENI[72] is meant for situations where there are multiple actual instances of one computational component running in different locations. The abstract description is without reference to a particular instance or location, the automatic concrete composition finds the most suitable running instance and uses that for its concrete workflow.

Resource design

The part of the design phase, in which resources are made available for use in a SWMS is usually not performed by the scientist end-user who composes a workflow. A resource developer requires a different type of support since it is assumed he has a lot more programming expertise. Passive support at this phase comes in the form of a well documented API which allows the developer to let a resource communicate with the different layers of middleware associated with an SWMS. Interactive support comes in the form of a rapid prototyping environment in which individual resources can be tested and debugged without executing a complete workflow or using the composition environment. Finally automatic support can be offered in defining meta data associated with a particular resource. Statistics on run time behavior can be automatically gathered and added to the meta data for instance.
5.2. SCIENTIFIC WORKFLOW MANAGEMENT SYSTEMS

Execution

During the execution phase the emphasis of user support is mainly on interaction between user and workflow execution. In this phase the user is assumed to be a scientist end-user just as in workflow design. Providing information on the progress of execution or monitoring is a form of passive support that can be offered. Interactive support allows the user to interact with the execution. This support comes in different forms. First the user can be allowed to manipulate the content of the data flow at runtime. Also the user can steer the execution by choosing the direction a workflow should take at certain points during execution. VCR like controls can allow a user to pause, resume or stop execution. A user can interact with certain steps in the workflow for instance by manipulating parameters, associated with a particular step, at runtime. Finally in some cases a user can be allowed to directly manipulate underlying middleware, for instance changing scheduling strategies. Automated support can also take place with scheduling strategies, in the adaptive workflow execution. For reasons such as a resource suddenly becoming unavailable or the unexpected failure of a software component, the workflow needs to be dynamically rescheduled. Alternatively when rescheduling is impossible execution should be halted in a user-friendly manner. For instance allowing a restart of execution from intermediate results.

Analysis

The analysis phase, from the workflow lifecycle point of view, deals mainly with the reuse of workflow results. Passive support in this case should offer a clear presentation of the results of workflow execution, allowing the scientist to judge the success of an experiment. Interactive support comes in the form of allowing the partial or complete replay of workflow execution. Classifying the success of an experiment according to predetermined criteria can be done automatically supporting the user when he has to perform a certain workflow many times. Finally for SWMSs with multi-layer models abstract templates of successful concrete workflows can automatically be generated.

User support can place particular demands on the other two functional SWMS components. In figure 5.2 these demands are made explicit. While the demands on the workflow model are not particularly great, some forms of support that only apply to multi layer models, are quite demanding on the engine, especially for the execution phase. In the next section we give a brief overview on how SWMSs are implemented in practice.
**Figure 5.2:** User support at different stages in a workflow lifecycle and its influence on the workflow model and engine

<table>
<thead>
<tr>
<th>FLOW COMPOSITION</th>
<th>USER SUPPORT</th>
<th>APPLICABILITY IN WORKFLOW MODELS</th>
<th>APPLICABILITY IN WORKFLOW ENGINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>Provide documentation and description definition, e.g., X:Shell (Taverna), MoML (Kepler)</td>
<td>Applicable to all models</td>
<td>Applicable to all engines</td>
</tr>
<tr>
<td>Interactive</td>
<td>• Validate workflow through type &amp; protocol checking, e.g., SPA, Kepler</td>
<td></td>
<td>Requires that engine has enactment service and be able to communicate with knowledge-based middleware</td>
</tr>
<tr>
<td>Automation</td>
<td>(partial) automatic composition of abstract workflow, e.g., Triana</td>
<td>Multilayer model, e.g., Pegasus</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEVELOPMENT</th>
<th>USER SUPPORT</th>
<th>APPLICABILITY IN WORKFLOW MODELS</th>
<th>APPLICABILITY IN WORKFLOW ENGINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>API for resource development → present in most systems</td>
<td>Applicable to all models</td>
<td>Applicable to all engines</td>
</tr>
<tr>
<td>Interactive</td>
<td>Debugging or rapid prototyping, e.g., Kepler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automation</td>
<td>Automatic generation of components, e.g., ICENI for metadas</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXECUTION</th>
<th>USER SUPPORT</th>
<th>APPLICABILITY IN WORKFLOW MODELS</th>
<th>APPLICABILITY IN WORKFLOW ENGINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>Monitoring execution → most systems</td>
<td>Applicable to all models</td>
<td></td>
</tr>
<tr>
<td>Interactive</td>
<td>• Manipulate flow content</td>
<td></td>
<td>• Capability to execute partial workflows, interrupts, handling intermediate results, interface for interaction</td>
</tr>
<tr>
<td></td>
<td>• Steer up/down</td>
<td></td>
<td>• Capability to stop and restart execution</td>
</tr>
<tr>
<td></td>
<td>• VCR control</td>
<td></td>
<td>• Provide user interface to component parameters</td>
</tr>
<tr>
<td></td>
<td>• Interact with components</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Interact with the e-science middleware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automation</td>
<td>Adaptive workflow execution</td>
<td></td>
<td>Dynamic scheduler needed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POST PROCESSING</th>
<th>USER SUPPORT</th>
<th>APPLICABILITY IN WORKFLOW MODELS</th>
<th>APPLICABILITY IN WORKFLOW ENGINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>Workflow template abstraction</td>
<td>• Distinction between abstract and concrete experiments needed</td>
<td></td>
</tr>
<tr>
<td>Interactive</td>
<td>• Replay experiment → all methods</td>
<td>• No influence → all models</td>
<td>Applicable to all engines</td>
</tr>
<tr>
<td></td>
<td>• Reproduce experiment</td>
<td>• Deterministic models</td>
<td></td>
</tr>
<tr>
<td>Automation</td>
<td>• Automatic template abstraction</td>
<td>• Applicable to multilayer model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Classification</td>
<td>• Applicable to all models</td>
<td></td>
</tr>
</tbody>
</table>
5.3 State of the art

The overview of the state of the art given in this section is based on a survey carried out within the VL-e project [41] and included the following SWMSs: Askalon[62], GEODISE[130], GridAnt[33], GridBus[13], ICENI[68], Kepler[31], Pegasus[58], SPA[32], Taverna[104], Triana[96]. The aim of this section is not to give an exhaustive overview of all SWMSs but rather to show which direction research in the field is moving in. For more detailed information see the previously mentioned survey [41] or one of the other surveys that have been done in this area [22, 134, 43, 45]. For each functional component we will give examples of how existing SWMSs deal with the issues mentioned in previous sections.

SWMSs seem evenly split in their choice of workflow model with about half choosing a control flow based approach while the other half is based on data flow. This may seem surprising because in the beginning of this chapter it was mentioned that most SWMSs are focused on massive data involved in e-science and that the data flow oriented model should be the logical choice. While the choice of model is evenly split, most SWMSs represent the workflow to the user in a temporal view thereby emphasizing data flow, regardless of the actual model being used underneath. Most SWMS offer abstract workflow composition and therefore a distinction between abstract and concrete workflows, but only a few offer the possibility to automatically transform this abstract description into a concrete one, the best example being Pegasus.

For workflow engines there is a trend to extend existing engines instead of building from scratch: for instance both SPA and Kepler are build upon Ptolemy[79], while Pegasus is build on top of DAGMan[3]. Semantic based search is currently provided for GEODISE and Taverna. The use of meta data in general is an area in which much research is taking place but most of this research has not yet led to mature implementations. Scheduling is an area where big differences exists: ICENI for instance offers a sophisticated launching service which takes care of executing and scheduling jobs on the grid, while in Kepler it is up to the user to provide extensions to the SWMS for scheduling and grid access. Orchestration of runtime behavior can happen in two ways: first there is the centralized approach where all orchestrating messages originate from a central component in the engine, e.g. in Taverna and Kepler, secondly there is the decentralized approach where orchestration is handled implicitly by the workflow components themselves. Service quality is an area where much improvement is possible. Although some form of dynamic scheduling is possible in most SWMSs, for the handling of fault tolerance most rely completely on underlying middleware layers. Thus offering fault tolerance support tailored to workflow execution is lacking. Dynamic execution features, allowing for the dynamic composition of workflows, are being researched in the context of the GEODISE SWMS,
however a mature implementations are not available at this time. Taverna offers user controlled fault tolerance schemes at the SWMS level.

There is quite a lot of diversity in the types of user support offered by different SWMSs. This can in part be explained by the different origins of the systems. Systems build upon the Ptolemy engine inherit a very sophisticated user interface originally developed for signal processing. While systems that have their origins in high performance computing research, such as ICENI have only relatively recently started to develop their front end. Passive user support is well developed for all systems. They all offer documentation for the systems, their APIs and in most also monitoring of execution and its results is implemented. More could be done in offering passive support for individual workflow components, for instance offering links to online documentation concerning a component used during workflow composition could improve an end users understanding of what is possible. Interactive assistance is where many SWMSs differ, though all offer some form of workflow validation. For instance SPA does type and protocol checking on workflows, as well as pinging resources used in the workflow to see whether they are alive. Semantic search is offered for far fewer systems: for instance GEODISE and Taverna. Interactive support during execution is an area where much improvement is still possible as can be deduced from the lack of support for dynamic workflow execution in workflow engines. Interaction with components is often limited to manipulating parameters. A VCR like control to stop start or pause execution is common though.

5.4 Towards a shared software resource

Scientific workflow management systems offer solutions to problems facing scientist when doing experiments in an e-Science environment. Because of the different origins of different SWMSs there is diversity in the support offered by each of them. What they have in common is that their support is mostly focused on the electronic part of e-science, hiding the complexities of grid computing from the normal scientist. They do also present an opportunity to offer more support for the science part of e-Science. Especially the increasing trend for using meta data, an approach known under several names such as the semantic grid\textsuperscript{1} and cognitive grids\textsuperscript{2}, should offer opportunities to support end user scientists in performing experiments which contain components from scientific fields not their own. The use of meta data can enable the creation of a scientifically sound experiment without the active participation of experts from all fields involved in that particu-

\textsuperscript{1}http://www.semanticgrid.org originated from UK e-Science programme, but now has wider support
\textsuperscript{2}http://www.isi.edu/ikcao/cognitive-grids University of Southern California Information Sciences Institute
lar experiment. Instead experts make their knowledge explicit in a generic form. In [81] William E. Johnston identifies three categories of capability which semantic solutions will have to meet.

- Check the validity of structures created by a user and assist in correcting errors.
- Automatically build simple composite operations from primitive operations on the basis of their semantics.
- Provide high-level constraints to facilitate correct interaction between complex models from different disciplines.

These three capabilities seem to be ordered by their likeliness of being implemented first. The first capability is already met by several systems, while research such as [84] points the way to how the second capability can be realized. Johnston acknowledges that though the first two capabilities are within the scope of current technology, the third one at this time is not. Sharing software resources fits within this trend of increased use of semantics in workflow composition. Nevertheless the steps presented by Johnston focus mainly on automation.

As it was shown in chapter 3 that automatic workflow composition is not possible in the general case, a shared software component should aim to aid the scientist in integrating it in a workflow, rather than just providing information aimed at automated composition. The first steps in this direction have already been taken: a prototype of an interactive assistant which provides semantics based advice to the user during composition has been described in [51]. This approach differs from the shared software resource in that it aims at composition within one domain. Also it provides assistance at each individual step, but does not provide a methodological overview. It does reinforce the view that facilitating the exchange of components between different domains, is feasible with current technology, through the use of an interactive semantic approach instead of the far more complicated automated approach.
Chapter 6

Data Assimilation

6.1 Introduction

In the upcoming chapters we will study workflows in practice with the help of two case studies. One case study deals with the prediction of bird migration, while the other is concerned with the prediction of road traffic. Both of these case studies use the technique of data assimilation to enable continuous predictions. In this chapter we will describe the origins of this technique. An overview of the essentials of data assimilation is given. Both the history and essentials of data assimilation are derived mainly from the books of Kalnay[83] and Daley[56]. The chapter is concluded with a discussion on the suitability of current toolkits for use in e-Science.

6.2 Weather Prediction

The term data assimilation originated in the field of weather prediction. Numerical weather prediction was first made a practical proposition in the mid 1950’s by the realization in 1904 by Bjerkness[44] that there were no simple causal relationships that relate the state of the atmosphere at one instant of time to that at another. Thus he defined weather prediction as nothing less than the integration of the equations of motion of the atmosphere. This was followed in 1922 by a practical method of performing this integration numerically. The predictions using this method at the time failed because of the inadequacies of the available observations. In 1950 aided by adequate observations and - of more interest to us - one of the first computers, the ENIAC, Charney, Fjörtoft and von Neuman computed the first successful one day numerical weather forecast[50]. One of the largest challenge for achieving a good prediction was the estimation of the initial state of the predictive model. While the observations were “adequate” they did not match the grid points in the model. Furthermore there were many more grid points in the model than there were observations. Finally observations
were not equally distributed, with many more observations in Eurasia and North America than in other areas. For the first experiments the values for these grid points were interpolated manually from the available observations. These grid points were then manually digitized, a very labor intensive process, that also lacked objectivity because human interpretation was involved in the manual interpolation. Soon weather prediction switched to a process where interpolation was done by computer. To compensate for the lack of observations a first guess of the state of the atmosphere was needed. At first this was based on climatological data, later upon short range forecasts. At each prediction cycle this state estimate was adjusted based on the observations available. The prediction of the model could be used as a first guess of the state for the next cycle. The resulting process as illustrated in figure 6.1 is now known as data assimilation. In 1997 data assimilation was defined by Talagrand[116] as: “Assimilation of meteorological or oceanographical observations can be described as the process through which all available information is used in order to estimate as accurately as possible the state of the atmospheric or oceanic flow. The available information essentially consists of the observations proper, and of the physical laws that govern the evolution of the flow. The latter are available in practice under the form of a numerical model. The existing assimilation algorithms can be described as either sequential or variational.” Since the 1950’s weather prediction has grown more sophisticated employing better models with increasing resolution using more computing power. With this sophistication have come better predictions: 72 hour predictions are now of the same quality as 36 hour predictions of 15 years ago, and current 36 hour predictions
are considered perfect by the standards of the mid 1950’s. Interestingly the
performance of subjective weather predictions made by experts continue to
improve as well. In fact the improvements follow the same pattern as that
of numerical forecasts with subjective forecasts always having slightly better
performance.

To understand what is meant by sequential and variational algorithms,
and more generally how these algorithms adjust the state estimate we need
to go into some more detail as explained in the next section

6.3 Data Assimilation Algorithms

From the history of weather prediction it has become clear that data assim-
ilation concerns itself with the minimization of error in the initial conditions
of the model. Historically it was first used to minimize errors in observation
and thereby the estimate of the current state. The model which is used
for predictions can also lack accuracy and therefore need adjustment. Data
assimilation can also be employed to minimize model error by way of per-
forming parameter optimization. The observant reader will notice that the
estimator algorithms used for minimizing model and observation error also
recur in fields other than weather prediction that do not use the term data
assimilation to describe their process. For instance some of the estimators
used also occur in (electronic) control theory as well as many other fields
where error minimization is important. Although The term data assimila-
tion is not used universally, it is commonly used within field related to earth
sciences such as atmospheric research (including weather prediction)[56],
oceanography, hydrology. What sets data assimilation apart from most re-
search which does not use this term for error minimization is the use of a
relatively large computational model and the large amount of input data
required to run this model.

To more clearly explain the data assimilation process we will first enu-
merate its main components and then discuss each in detail with the empha-
sis on estimation. The main components in the data assimilation process
are:

- Observation Data
- Computational Model
- State estimate
- Prediction
- Estimator
6.3.1 Observation Data

Data from observations is usually not in the form that the computational model needs. Like observations used in any scientific process the data can suffer several problems. Noise through inaccuracy inherent in the sensors employed. Noise through human errors caused by some transformation. For instance wrong date or wrong unit of measurement which can potentially result in extreme values far out of normal range. More often than not observational data is not a direct measurement of what the computational model needs as input. For instance if a model needs a flow rate of objects as an input whereas only observations of individual objects are available, these observations need to be aggregated in order to derive a flow rate. Another possibility is that observations contain information on several phenomena at different timescales and only one of them is of interest to the model. In this case others have to be filtered out.
6.3.2 Computational Model

The computational model incorporates the available theories on which prediction is based. It takes the estimate of the current state as input and in addition can have parameters which affect the working of the model. These can be useful when there is some uncertainty on how two modeled phenomena influence each other. The estimator can adjust these parameters to minimize model error.

6.3.3 State Estimate

The state estimate consists of all data the computational model needs to perform a prediction. The initial state estimate can be based on transformed observation data or be derived from a different, possibly less accurate, model if not enough observations are available to provide a complete estimate. State estimates of subsequent cycles are based on previous predictions. They are adjusted through the estimator based on current observational data.

6.3.4 Prediction

The prediction is the output of the computational model. It is both output of the data assimilation process as well as data used by the estimator to adjust both state estimate and model parameters.

6.3.5 Estimator

The estimator can be used to estimate two things:

- State: minimizing the error in the estimate of the current state, by weighting the influence of observations in the state estimate according to their error co-variance.
- Parameter: minimizing the error introduced by the computational model through weighting the various parameters in the model.

Most data assimilation systems are used for the prediction of unstable systems and employ state estimation. It has been shown by Lorenz[92] that even the slightest error in the initial state estimate for weather prediction will eventually result in totally chaotic predictions. There is a theoretical maximum of about two weeks to weather prediction. Thus for the prediction of unstable systems an accurate state estimate is of great importance. Optimal methods for parameter estimation can be very expensive computationally as it brings a cost increase in the same order as the degrees of freedom in a computational model. This means that in practice for complex models which have to run under real time constraints, non optimal estimators are employed. Especially for weather prediction Kalman filters are not
a practical proposition and therefore less complex but also less optimal esti-
mators are used. In other areas such as hydrology where models are either
less complex or real time constraints are less demanding Kalman filters are
employed to achieve the best results. What follows is an overview of com-
mon estimators, the first three are more computationally efficient and the
last three have increasing computational complexity but higher accuracy.

- 3dVar: three dimensional variational analysis, uses a cost function to
  minimize state error.

- 4dVar: four dimensional variational analysis, uses a cost function to
  minimize state error allows for observations within a time interval
  instead of just one discreet point in time.

- Optimal Interpolation: does state estimation using an error covariance
  matrix. It is proven to be equivalent to 3dVar with certain assump-
tions.

- Kalman Filter: does state estimation using an error covariance matrix.

- Extended Kalman Filter: does optimal state and parameter estimation
  using an error covariance matrix.

- Ensemble Kalman Filter: does a non optimal more efficient way of
  parameter estimation using an error covariance matrix.

6.3.6 Use of ensembles

In the prediction of unstable system where even slight error in the initial
state estimate can have a large impact on longer term predictions ensembles
are often employed. An ensemble consists of several runs of the computa-
tional model each with slightly different initial state estimates. This gives
more insight in the effect of error propagation in the final prediction as it
shows a whole possible range of outcomes. It also allows the most likely
outcome can be chosen by taking an average of all the runs in the ensemble.

6.4 Data assimilation toolkits

Within this section an overview is given of toolkits for data assimilation. We
will go into detail on the two toolkits used in the case studies, CAPTAIN
and SOS, but we will also look at other available toolkits. By means of a
comparison of available features and the intended uses of these toolkits, it
is our aim to show how data assimilation can benefit from implementation
inside an e-Science environment.
6.4. DATA ASSIMILATION TOOLKITS

6.4.1 Overview of Toolkits

Data assimilation tools can be divided into two types, those dedicated to modeling one domain specific phenomenon, and those that provide a more generic framework. Within weather prediction and oceanography many national weather and climate institutes provide access to their model for free.

6.4.2 Toolkits in detail

In this section a detailed view of the toolkits used in the case studies is provided.

Captain Toolbox

Captain[133] is a matlab toolbox for non-stationary time series analysis and forecasting. It was developed at the university of Lancaster and can be obtained directly from there for a license fee. It contains several different modeling methods, all based on Unobserved Components. These modeling methods assume that a time series is composed of a combination of different additive or multiplicative components which cannot be observed directly. It also includes algorithms for data pre processing, system identification and model validation as well as a manual explaining the proper use of the included tools and their theoretical background. The Captain toolbox is very well documented a comprehensive manual is provided with the toolkit. The development of this manual as well as providing support in the use of this toolkit is the main reason that this software is licensed.

SOS toolkit

The SOS Toolkit[128] was developed by Heemink and Verlaan at the Technical University of Delft. Its main use has been in the analysis of various new estimators and it has several estimators build in: a reduced rank square root estimator, an ensemble Kalman filter as well as several estimators implementing a hybrid of both. The details of these algorithms as described and implemented by Verlaan and Heemink can be found in [28]. Like the Captain toolbox it was written in matlab, unlike Captain the source is available making it easier to develop it into services. The toolkit provides a matlab based GUI for running experiments and viewing results. One standard interface exists which allows a model to be used with all available estimators when implemented. Similarly more estimators can be added by implementing their side of the interface. No manual or support is available for this toolkit, only related papers and documentation inside of the matlab code.
### Table 6.1: Data assimilation toolkits compared

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>supported models</td>
<td>many geophysical models</td>
<td>advanced research WRF and non-hydrostatic mesoscale model</td>
<td>various hydrological models</td>
<td>various ocean and ocean-atmosphere models</td>
<td>many</td>
<td>many</td>
</tr>
<tr>
<td>supported DA algorithms</td>
<td>various ensemble kalman filters</td>
<td>3d-Var/4d-var</td>
<td>ensemble kalman filter</td>
<td>numerous smoothing estimators</td>
<td>4d-var</td>
<td>reduced rank square root, ensemble kalman filter and combinations of both</td>
</tr>
<tr>
<td>Model interface</td>
<td>sparsely documented model interface</td>
<td>documented framework</td>
<td>documented model interface</td>
<td>api</td>
<td>only models developed with toolkit</td>
<td>sparsely documented model interface</td>
</tr>
<tr>
<td>GUI</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Programming language</td>
<td>fortran90</td>
<td>fortran90</td>
<td>matlab</td>
<td>fortran</td>
<td>matlab</td>
<td>matlab</td>
</tr>
<tr>
<td>Parallel computing</td>
<td>MPI</td>
<td>MPI</td>
<td>no</td>
<td>MPI</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Current development</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>License</td>
<td>Open Source</td>
<td>Open Source</td>
<td>Open Source</td>
<td>Open Source</td>
<td>closed source</td>
<td>Open Source</td>
</tr>
<tr>
<td>Documentation</td>
<td>only tutorial</td>
<td>only tutorial</td>
<td>manual</td>
<td>manual</td>
<td>manual</td>
<td>no</td>
</tr>
<tr>
<td>Grid support</td>
<td>no</td>
<td>yes, teragrid</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Problem solving</td>
<td>no</td>
<td>custom workflow system</td>
<td>matlab</td>
<td>no</td>
<td>matlab</td>
<td>matlab</td>
</tr>
<tr>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.4.3 Grid use

In this section we look into more detail at what it takes to run any of the toolkits from the overview, given in table 6.1, on the grid. There is a clear dichotomy to be made in the toolkits reviewed here. SOS, Captain and Daihm[60] are toolkits which are used in the development of either models estimators or both. IOM[53], WRF-VAR[40] and Dart[4] are tools which are used in a setup where a continuous prediction is needed and are aimed at specific applications such as weather prediction. These are more of a production based system where performance and meeting realtime constraints is more important than exploring new model or estimator possibilities. When performance becomes an issue any overhead or inefficiencies, brought about by using the grid and especially existing e-Science workflow systems, become an important reason to be conservative and look for more efficient alternatives to workflow systems. Let us look for instance at the example of using ensembles in data assimilation. For each prediction step the data used for the calculation of each member in an ensemble changes relatively little. Also each ensemble calculation finishes in more or less the same time. After each prediction step there is a synchronization of the data in all steps and then the next prediction step is started. To do this efficiently each of the nodes calculating an ensemble should keep its state and only the information that has changed for the next step should be altered in this state. For greatest efficiency the ensemble calculation should be kept in memory ready to start immediately with the state update also being performed directly in memory. This is something that can be done easier and more efficiently in a dedicated cluster than in a grid environment. The underlying reason is that until now, there is relatively little support for optimization in the scheduling and especially synchronization of more complicated workflows running on the grid.

Matlab toolkits on the grid

Running the Captain toolbox as a set of services is not easy. The toolkit consists of compiled closed source functions which are not easily converted to stand alone programs through the use the Matlab C compiler. Nor can they be run in Octave, an open source alternative to Matlab. Running in Octave is attractive because it saves on license costs when using multiple instances at the same time. The SOS toolkit is able to run in Octave in a limited form after some modification to the code. The SOS toolkit uses some Matlab functions and libraries for which no equivalent is available in Octave. In the clusters available to the VL-e project there is a limited number of Matlab licenses available while Octave is installed on all nodes: its use is therefore unlimited. The only available option for Captain is a wrapper around Matlab which is able to call Matlab functions, for instance...
using software such as JMatlink\textsuperscript{1}. This means that an instance of Matlab
has to run for each service that is in use.

\section{Conclusion}

Data assimilation is suitable for prediction problems where the phenomenon
one is trying to predict is unstable. Unlike phenomena like tides and sun-
rises which have clear periodicity. For the former problems error propa-
gation will in the end always lead to completely chaotic predictions if the
prediction is a certain amount of (time) steps ahead. Another prerequisite
is that the computational model available is accurate. If the model error is
such that it eclipses errors in state estimation then the sensible approach
is to develop a better model. Data assimilation is very much suited to an
e-Science environment because it deals with massive amounts of data and
massive computation, certainly in the case of ensembles or parameter op-
timization. As illustrated in this introduction, it is also a technique that
can be shared between very different scientific domains like: atmospheric
research\textsuperscript{40}, hydrology\textsuperscript{60}, oceanography\textsuperscript{53}, geophysics\textsuperscript{4}, and as the case
studies in the next chapter will show biology and traffic research.

\textsuperscript{1}http://jmatlink.sourceforge.net
Chapter 7

Data Assimilation Case Studies

7.1 Introduction

This chapter presents two case studies into the use of data assimilation for two very different applications: the prediction of bird migration and the prediction of traffic. The chapter consists of two main sections each dealing with one case study and presenting the conclusions particular to that case study. The chapter ends with more general conclusions on the lessons learned from these case studies that are relevant for implementing data assimilation as a shared software resource in a scientific workflow management system.

7.2 Bird migration model

In this case study we investigate the possibilities to predict north-south autumn bird migration over the Netherlands. The work is performed in the context of the Bird Avoidance Model, Bird Avoidance System or BAMBAS project[112]. The aim of this project is to provide accurate information to the Royal Dutch Air force on the presence of birds over the Netherlands. They can use this information to plan their training missions in such a way that the minimize the risk of bird strikes. Bird strikes to aeroplanes are a serious hazard and can cause a large amount of damage and have caused fatalities in the past. The BAMBAS project is a cooperation between multiple institutions, SOVON, University of Amsterdam and The air force. It combines scientists from different disciplines, biology, physical geography and computer science. The case study here is a first investigation into the use of data assimilation for predictive bird migration models. Results of this case study were presented at the DARE 2004 workshop [119].
CHAPTER 7. DATA ASSIMILATION CASE STUDIES

7.2.1 Data

Several kinds of data are are collected in the BAMBAS project which can be used for prediction of bird migration. We will briefly describe them here:

- Wind data is obtained from the KNMI the royal Dutch meteorological institute. It contains wind data over the relevant period with a resolution of 10km covering the whole of Europe and northern Africa.

- Geographical barriers include mountains such as the Alps, deserts such as the Sahara as well as seas and oceans. Birds cannot land in barriers and consequently are not able to rest or collect food in these areas.

- Vegetation/food data contains information on the amount of food available as well as the caloric value of this food.

- Radar data originates from the Dutch royal air force base in the province of Friesland. It has a maximum range of 150 km. Detailed information is available for two smaller sub samples in the radar image. Measurements are done each hour and the detailed (processed) information is based upon 10 radar rotations.

  This detailed information consists of raw radar data which has been analyzed with a program that can detect birds through motion analysis. This means an estimate of speed and direction of the birds is available. The radar is not detailed enough to detect individual birds but rather groups of birds so exact bird counts are impossible[46]. Through the SOVON organisation bird counts of migratory birds are available[91]. For the collection of this data observations were made at set times of the year at 120 different locations in the Netherlands. Under the guidance of professional ornithologists, volunteers go out into the fields and for each separate area count the number of each species of bird they see within a set amount of time. This produces a record of the seasonal occurrence of migratory birds in the Netherlands.

7.2.2 Model

The model used to predict bird migration is an extended version of the model developed by Erni et. al.[38]. It is a model based on expert knowledge, where each bird is simulated individually. At each timestep the behavior of each bird is simulated based on the input data. Time-variable model inputs are wind speed and direction, and constant inputs are the terrain and habitat quality over Europe. The main process underlying the complex patterns of migratory flight is the bird’s energy balance. According to the terrain it is flying over and the energy it has left it can have different behaviors. The model is initialized with a population of birds leaving Scandinavia. Each bird is initialized with a different endogenous direction towards which it will try
7.2. **BIRD MIGRATION MODEL**

to migrate. Currently the model is implemented in Matlab, an adaptation of an original implementation in BASIC.

### 7.2.3 Estimator

For the data assimilation algorithm and specifically the estimators the SOS toolbox for Matlab by Verlaan & Heemink (TU Delft)[128] was employed. Due to practical circumstances the real radar observations were unavailable for the experiments. Thus the bird observations used were generated by creating a ”truth” dataset with the Erni model. Four observations areas were defined that more or less matched the range of existing radar observations for these areas. From the ”truth” data set bird densities were derived for each timestep. White noise was added to simulate observation error. For the initial state estimate of the birds in Scandinavia no observations were available at all. Thus a guess was made consisting of a random distribution of birds over the start area in Scandinavia with bird parameters initialized according to expert knowledge.

### 7.2.4 Experiment

In this study real weather and terrain data were used, the radar observations were artificially generated. We first investigated the behavior of the predictions, by varying density and location of the radar observation stations (with constant, white noise observation errors). Observed Birds were simulated until the next time step, which was twelve hours in length. Every twelve hours an observation was made, after each observation the model simulated what the observed number of birds would do in the next twelve hours. At the end of the timestep an observation is made above the Netherlands and the number of birds which is predicted at this point according to the model is also noted. These two types of data are input for the estimator which adjusts the state estimate. For the following timesteps the process is repeated until all birds have migrated. The data assimilation experiment produces predicted bird densities above Netherlands.

Figure 7.1 shows our first experiment: here one very large observation point above Scandinavia is used. The number of birds observed at this point (shown in the top right graph of figure 7.1) are entered into the Erni model. This results in a predicted number of birds over the Netherlands.

Figure 7.2 shows the results of an experiment where two types of observation are combined. In this case (radar) observations from a smaller area in Scandinavia are extrapolated using bird count data to an area the size of the first experiment.

Figure 7.3 shows the effect of varying the timestep. For this experiment the observation point was placed over northern Germany, an area where a real radar observation point exists. This experiment shows that a short
Figure 7.1: One large observation point in Scandinavia. The top graph shows the number of observed birds over Scandinavia, the bottom graph shows bird numbers over the Netherlands in three ways. The solid line represents the "truth", plus signs are (radar) observations and the dashed line is the prediction produced by data assimilation.
Figure 7.2: Combining two types of observation. The top graph shows number of birds over Scandinavia in two ways: the solid line represents real number of birds, the dashed line represents extrapolated number of birds. The bottom graph shows bird numbers over the Netherlands in three ways, the solid line represents real numbers, the plus signs represent (radar) observations, the dashed line is the prediction produced by data assimilation.
Figure 7.3: The effects of varying timestep. The graphs show the number of birds over the Netherlands. The top graph has a timestep of four hours, the middle eight hours and the bottom twelve hours. The solid line represents the real number of birds, plus signs the (radar) observations and dashed line is the prediction produced by data assimilation.
time interval means that not all birds at the first observation point have the opportunity to fly to the second observation point. Furthermore the predicted values are low compared to the real and observed values. This is partly due to the fact that birds coming in over the North Sea are not observed. Another reason is that birds are initialized with an endogenous direction. This direction lies within a certain range that was kept the same, for this experiment with observation above Germany, as the first experiments with observations above Scandinavia. However birds flying due south from Scandinavia would not reach the German observation point, thus with unchanged initialization a disproportionate number of birds will predicted by the Erni model to fly south and thus not over the observation point in the Netherlands. Most importantly though the estimator does not seem to have any significant effect in adjusting the low predictions made by the model.

Figure 7.4 shows the same type of experiment as in figure 7.2. This time the observation area is over Germany which has the disadvantages described before.

Figure 7.5 shows what happens when the predictions based on observations from the North Sea are added to the predictions based on observations over northern Germany. The prediction improves slightly as it takes away one of the problems mentioned in the first experiment where only the observations above northern Germany were used, the other conditions were still applicable.

7.2.5 Conclusions bird migration

These experiments were early explorations of how data assimilation can be used with a biological model that describes behavior. While performing these experiments, reproducibility was ensured by keeping track of the methods and data used in each experiment. This was a process that was performed manually. None of the software development done in this experiment was directly reusable in the traffic prediction case study that will be described in the next section.

More interesting experiments could be to extend our analysis to the more realistic situation where the radar measurement error depends on both bird densities (especially during migration peaks) and weather conditions (especially rainfall). The trade-offs between model error - measurement error - measurement density and predictive uncertainty could be described for different weather conditions. Also, using data sources complementary to the radar observations, such as visual counts of birds at resting places, can potentially improve the initial state estimates. These observations do not suffer from state-dependent observation errors, but on the other hand they are less easily linked to the model scale. The real test for this model still has to take place, it has to be validated against real observations in order to determine its accuracy. Only after validation can truly significant
Figure 7.4: combining two types of observation. The top graph shows true number of birds (solid line) and extrapolated number of birds (dashed line). The middle graph shows the true number of birds in the radar observation area. The bottom graph shows number of birds over the Netherlands, where the solid line represents the "truth"; plus signs are observations and the dashed line is the prediction produced by data assimilation.
Figure 7.5: Combining two observations. The graphs all show number of birds above the Netherlands where the solid line represents the "truth"; plus signs are radar observations and dashed line is the prediction produced by data assimilation. The top graph is based on observations above the North Sea, the middle graph is based on observations in Germany, the bottom graph shows the two combined.
experiments be done. However as a way to gain experience in the use of data assimilation techniques the experiments proved very useful. For future experiments which will use more data sources and involve more experiment runs, keeping track of (intermediate) results and ensuring reproducibility will be increasingly difficult.

7.3 Traffic Forecasting

In this section we will look at another application of data assimilation in behavior prediction. This time the behavior being predicted is that of car drivers, more precisely the moment their behavior leads to traffic congestion. This application is presented to give insight into the current practices within data assimilation: what types of problems are encountered during setup and execution of a typical experiment. To provide the background for this research first a brief introduction will be given into the area of Intelligent Transport Systems, the area in which traffic predictions will be used. Then the specifics of prediction within ITS will be explained, including some other state of the art predictive systems. This will be followed by the description of the data assimilation solution performed in the context of this thesis. This section ends with a discussion on how this solution can benefit from an e-Science solution and what the additional requirements for such a solution are. Results of this case study were presented at the ITS worldcongress 2006[118].

7.3.1 Intelligent Transport Systems

Since the beginning of the industrial revolution, ever more complex transportation infrastructure has emerged, starting with mass transportation using trains and getting even more complex with the the adoption of the car as a method for personal transportation. To keep this infrastructure working and accommodate an ever increasing number of travelers requires smart solutions. Intelligent Transport Systems (ITS) try to provide technologically enabled solutions to these transportation challenges. ITS concern themselves with three main areas:

- Information
- Management
- Automation

The area of information contains such subjects as providing accurate information to travelers. This can be information on congestion or delays in public transport. It can take many forms: websites, roadside displays, mobile services etc.
7.3. TRAFFIC FORECASTING

Management in ITS deals with keeping traffic moving. This can take the form of traffic management systems that monitor the situation on the road through various sensors and take appropriate action when a problem is detected. For instance, closing a lane when an accident is detected. It also includes systems to control traffic flow such as variable speed limits or traffic lights at highway access ramps. Probably the most well known subjects in this area are electronic toll collection systems such as “Rekeningrijden”, “Kilometerheffing” in the Netherlands and the congestion charge in London.

Automation concerns itself with assisting or fully automating the driver of a vehicle. An example of assistance is adaptive cruise control where the cruise control automatically reduces a vehicle’s speed when the distance to the vehicle in front gets too small. Prototypes for automated highway systems, where cars drive themselves totally autonomously, also exist. But they are still far from widespread adoption.

In the area of providing information as well as traffic management being able to accurately predict where congestion is going to occur is a great asset. The traffic situation on the roads is a constantly changing process: thus a stretch of road which is free from congestion at the time the information is provided can be congested by the time a driver arrives there. Similarly in traffic management prediction is instrumental in taking preventative measures and also in knowing whether a certain traffic management decision will have the desired effect.

7.3.2 Prediction for ITS

The ITS domain has its specific properties and demands where prediction is concerned. What follows is an elaboration on these properties and demands with regard to the data, the modeling techniques employed and finally the prediction itself.

Data

There are many different ways of obtaining traffic data. Almost all of them fall into two main categories:

- Fixed sensors
- Floating car

Important measurements are:

- Traffic density: cars per time unit at one location.
- Speed
- Travel time: time needed to travel a route.
Fixed sensors such as induction loops in the road and cameras can observe cars at one location. This results in accurate measurements of both the speed as well as the density of traffic. The downside is that measuring every road in this way is very expensive, which is why these types of measurements are mainly done on busy highways.

Floating car data, where the speed and position of a few cars in the traffic-flow is measured through some form of transponder, offers the advantage that traffic on all roads can be measured. The downside of floating car data is that deriving an accurate density is more difficult. However when enough measurements are available traffic speed can be estimated relatively accurately. Furthermore previous research into modeling traffic has shown that there is a relation between the speed of traffic, the capacity of the road and the density of traffic [73]. Thus, knowing the capacity of a road and the speed of traffic traveling on it, an estimate of the density can also be made. For modeling larger areas or long time spans, origin destination matrices are commonly used data. These matrices contain data on where cars start their journey, on what time they start and what their destination is. This data can be estimated through the combination of traffic measurements from multiple locations, but also by conducting interviews with motorists on their driving habits.

Cars themselves are not the only measurable thing that can be of importance when predicting traffic. Other factors exist that influence the behavior of drivers[123]. These factors are divided into three distinct categories.

- First there is nature, which in this case is almost entirely accounted for by the weather. Different weather conditions such as rain, fog and snow cause different driver behavior and therefore different traffic patterns.

- Second there are the human causes, consisting of two sub-categories of global and local influence. The global events are calendar related phenomenon such as the difference between weekends and work-days but also less regular events such as holidays. These events have an effect on the society or country in which this calendar is used, unlike the first category which is bound to one geographical location. Large public events such as pop festivals, parades and most commonly roadworks, can cause localized increases in traffic.

- The third category are incidents and accidents, which have a bidirectional relationship with traffic patterns. They cause extra congestion and capacity reduction, while creating traffic patterns that influence the chance of accidents occurring. Incidents and accidents are a stochastic process where one small event sometimes has large consequences for overall traffic patterns.
7.3. TRAFFIC FORECASTING

Modeling

Within ITS and traffic research in general three different scales of traffic model exist.

- Microscopic
- Mesoscopic
- Macroscopic

Microscopic models model the behavior of each driver individually and are therefore very detailed. Behavior in this case can be: when a driver changes lane, how much distance is kept to the car ahead and at what time a driver brakes to preserve this distance.

Macroscopic models on the other hand regard the traffic stream as a whole and model it as if it was a gas or liquid. This offers less detail but is also less computationally intensive. In [73] macroscopic models are explained in great detail.

Mesoscopic models are hybrids between micro- and macroscopic models. They use a macroscopic approach to straight roads but revert to microscopic methods at more complicated areas such as junctions. An example of this type of modeling can be found in the DynaMit project [42]. For a slightly dated but very comprehensive overview of traffic models one can look at the results of the "smartest" project¹.

Modeling factors which influence traffic can be done in multiple ways. Within microscopic models different behaviors for varying weather conditions can be modeled, while parameters in macroscopic models can be adjusted to account for the different dynamic of traffic under varying weather conditions.

Prediction

Choosing the right type of model for a particular application can depend on multiple factors. In case of traffic prediction the size and complexity of the roadnetwork. These factors determine what type of models are computationally feasible.

Another important factor is the amount of time ahead a prediction is required to be. If a prediction is further into the future it is less likely that a predicted traffic situation at a certain point can be based on direct measurements. If measurements of the current situation of the traffic situation are taken as the basis for a model that propagates the movements of measured cars into the future, then at some point in the future all cars will have disappeared from the area of interest. The simple reason behind this is that

¹http://www.its.leeds.ac.uk/projects/smartest/
traffic is always modeled as an open system. The problem we are dealing with here is known as the courant condition\cite{54}.

This is a problem for forecasts with a larger scope in time, or for a small modeled road network. To overcome this problem the factors influencing traffic mentioned earlier in this section need to be taken into account when constructing a model. Especially the social factors determine how many cars are going to enter and leave the network and therefore have to be included in the model to compensate for the lack of relevant direct observations. It is also important to use the available direct observations to correct the estimate of traffic based on social factors; an application that lends itself to data assimilation.

7.3.3 Current solution

To get a better idea of how data assimilation can currently be used in the ITS field we look at a suitable case study concerned with longer term traffic prediction. In this case direct observations of traffic do not provide enough information for prediction and social factors have to be modeled as well. First the available data will be described in detail. The modeling effort for this case study involves two methods. First Auto Regressive Integrated Moving Average (ARIMA) will be introduced, a common technique in time series analysis which will be used as a baseline to compare to the Dynamic Harmonic Regression (DHR) method. DHR was chosen because it has been used successfully in comparable applications, both involving longer term prediction\cite{125} and involving shorter term traffic prediction\cite{117}. Finally the results of the experiment are analyzed.

Data

For this case study we use data concerning one 500 meter stretch of highway. This allows for (future) comparison with detection loop data. The data covers the whole of October 2004. It includes three normal working weeks and a one week school holiday. The raw data consists of speed measurements at a one minute granularity. However since we are interested in longer term predictions the data is first aggregated to one hour data.

As a first step towards building a predictive system we will look at properties of the data that will have an impact on any modeling attempt. First we looked at the autocorrelation, to see which cycles, if any, were present in the data. In figure 7.6 one can see correlations for 24 hours and all subsequent multiples of 24 which decay slowly with each multiple except for the 168 hours point (one week), where it is a lot stronger. The correlation at one week is the most interesting, as it implies that not all days of the week have similar traffic patterns. Furthermore there are strong negative correlations to be seen at 12 hours after the 24 hour peaks. This could be explained as an
7.3. TRAFFIC FORECASTING

Figure 7.6: Auto correlation for speed measurements in observation area that is aggregated to hourly data

imbalance between morning and evening rush hour. This is very plausible since the data concern only one direction of traffic flow where one would expect just one rush hour per day. As a next step we looked at the correlation between individual days to determine for which day this weekly correlation was strongest. In figure 7.7 we show the correlation between all Mondays in October 2004. This is achieved by taking the traffic pattern of each Monday in October and calculating the correlation coefficient for all possible pairs of Mondays. It can be seen that Mondays are strongly correlated. Fridays, although not shown in these figures, produce results similar to Mondays, also showing a strong correlation. Days in the middle of the working week, such as Thursdays are much less correlated as can be seen in figure 7.8, which was calculated in the same manner as figure 7.7. On the other hand these midweek days (Tuesdays, Wednesdays and Thursdays) show much stronger one and two day correlations. This is demonstrated by figure 7.9 where the correlation between three consecutive midweek days is shown.

Finally Saturdays and Sundays do not show much correlation at all. They don’t have a weekly correlation nor a correlation with the other consecutive day in the weekend nor with any other day of the week. The explanation for this is twofold. Firstly in weekends there is much less traffic than in other days of the week. This does not result in regular traffic jams, which form the basis of the weekday correlations. Secondly, because there is less traffic, there are far less observations making the the resulting traffic patterns less accurate and more erratic, even when aggregated over 1 hour intervals.

There are several lessons to be learned from the data analysis presented above, with regard to modeling. Weekends are, with the current data set, best ignored. Our main interest is in predicting congestion with a certain degree of reliability. Including data from weekends would do more harm
Figure 7.7: Correlation coefficients for the pattern of traffic speeds on the four Mondays of October 2004
7.3. TRAFFIC FORECASTING

Figure 7.8: Correlation coefficients for the pattern of traffic speeds on the four Thursdays of October 2004
Figure 7.9: Correlation coefficients for the pattern of traffic speeds on the first Tuesday, Wednesday and Thursday of October 2004
than good to an overall prediction because of a lack of accuracy and the absence of congestion in the data. The correlation between different weekdays suggest the possibility of modeling midweek days, Mondays and Fridays as three separate entities. Lastly, when evaluating the results of forecasts we should not forget the school holiday in the last week of our dataset. Earlier in this section calendar based events were given as an instance of human factors of influence on traffic. This influence is noticeable in the dataset through a reduction of congestion during the holiday period and also by the distinct traffic patterns of different weekdays, especially Mondays and Fridays. It should be noted that the correlation even for these weekdays is quite weak (below 0.85). However, the difference between Mondays, Fridays and weekdays for traffic data is well known within traffic research. Other studies have made the same categorization based on their traffic data\[69\] as well.

**Dynamic Harmonic Regression**

The modeling method chosen for this traffic prediction application was Dynamic Harmonic Regression (DHR)\[132\], one of the methods available in the Captain toolbox\[133\] that was introduced in the previous chapter. The Captain toolbox has been used for all kinds of time series analysis, including traffic prediction\[117\]. DHR will now be explained in more detail, followed by an explanation of ARIMA(not part of Captain) which was used as a baseline method to compare it with. DHR can be used for forecasting, backcasting, interpolation over gaps in the data, signal extraction and adaptive seasonal adjustment. In the past it has been successfully used in another time-series application to forecast telephone call patterns up to several weeks ahead \[125\]. The unobserved components within the DHR model are the trend, seasonal and cyclical components. These hidden components will match with some of the calendar based influences on traffic mentioned in section 7.3.2. Each of the hidden components has Time Variable Parameters (TVP) associated with them to deal with non-stationarity: for instance changes of phase and amplitude over time present within a time series. A Kalman filter is used to optimize these TVP’s thereby minimizing model error. Thus if a DHR model is used for prediction at each time step we in effect have a data assimilation system minimizing model error. To model traffic using DHR and other tools in the captain toolbox, the following steps are performed. First the data is prepared, the granularity of the model has to be chosen. Observations have to be aggregated into discrete timesteps. These timesteps have to be short enough to capture the shortest (in time) traffic jam but they should not fall outside the limits allowed by the method of observation. In this case five minute timesteps were chosen. If gaps exist in the data they can be filled in. In this case the method of observation is such that a lack of data suggests there is no congestion. Thus traffic can
be assumed to be at or near the maximum speed allowed for the stretch of road concerned. Next the data is analyzed. The Akaike Information Criterion (AIC) as implemented within the Captain toolbox, can be used to find a good compromise between model fit and model complexity. From the AIC output an appropriate number of Auto regressive parameters can be chosen for use in DHR. Further analysis can be provided by viewing the auto regressive spectrum of the data through the "arspec" (auto regressive spectrum) function. What "arspec" does in the frequency domain, the “period” function can do in the time domain. This provides evidence of cycles in the data, and verifies whether expected cycles are present. The period of these cycles is used as an input for DHR. Other inputs for DHR such as the optimization method of the TVP’s have to be selected on the basis of the knowledge gained from either the manual, expert advice or from previous experience. The modeling for DHR consists of two steps. First the initial Noise Variance ratio (NVR) of the hyper parameters is determined using the "dhropt" (dynamic harmonic regression optimization) function. These hyper parameters determine the statistical properties of the Time Variable Parameters. Then the DHR function itself is run to perform the actual modeling and any fore- or backcasting which may be called for.

ARIMA

The name of the model used as a baseline, ARIMA, is an abbreviation for Auto Regressive Integrated Moving Average. It models a time series by
first removing trend and periodic components from the data, then fitting an ARMA model [48], and for the final result adding, or 'integrating', the previously removed trend and periodic components. Predictions are made by applying a transfer function to the model that is fitted to the known data. For more details on ARIMA see [48]. ARIMA is a very well known but relatively basic technique. If DHR is used correctly and is suitable for the domain of traffic prediction it should produce significantly better results than ARIMA.

Results

Having described the DHR and its associated methodology we will now look at the actual results of experiments using DHR. We start with preparation of the data. Weekends contained very little useful information, because very few measurements are made during that period. They were ignored for the experiments because of the aforementioned reason and because weekends contain few to no traffic jams in any case and thus are not of interest in predicting traffic jams. The weekly correlation that is mainly present in Mondays and Fridays is not modeled well when using the complete data set. Thus for the results presented below we separated the data set into three different sets, for Mondays, Fridays and midweek days respectively. The data used for predicting Mondays only consists of Mondays, the one for Fridays only consists of Fridays while the data for predicting midweek days consists of Tuesdays, Wednesdays and Thursdays. As a first experiment traffic speeds were predicted for Tuesday the 12th of October. The results in figures 7.11 and 7.14 show that big slowdowns indicating traffic jams are absent. Both methods do well, but DHR is slightly closer to the observed data. A more interesting example is shown in figures 7.12 and 7.15. This shows a twenty-four hour prediction for Friday the 22nd of October. For this day a traffic jam is clearly visible in the data with a peak at 17:00. The prediction from the ARIMA model has little relation with the true data. DHR on the other hand shows more promise, its prediction contains a traffic jam of the same amplitude, however the phase is shifted resulting in a prediction that is three hours early. Finally in figures 7.13 and 7.16 a sixteen hour prediction is shown where the first 8 hours of Friday the 22nd were also included in data on which the prediction is based. This shows that the difference between the predicted traffic jam and the actual traffic jam has become less for DHR but with ARIMA no clear prediction of a traffic jam can be observed. It is clear from the previous section that DHR shows promise as a modeling method, outperforming ARIMA especially when it comes to predicting traffic jams. These experiments have been a first step towards a traffic forecasting system, however further improvements are possible and needed.

In this experiment Data assimilation was used to minimize prediction
Figure 7.11: Twenty-four hour prediction for Tuesday 12-10-2004 using ARIMA model with midweek data from the previous week.
Figure 7.12: Twenty-four hour prediction for Friday 22-10-2004 using ARIMA model with data from three previous Fridays
Figure 7.13: Sixteen hour prediction for the last sixteen hours Friday 22-10-2004 using ARIMA model with data from the first eight hours and three previous Fridays
Figure 7.14: Twenty-four hour prediction for Tuesday 12-10-2004 using DHR model with midweek data from the previous week.
Figure 7.15: Twenty-four hour prediction for Friday 22-10-2004 using DHR model with data from three previous Fridays.
Figure 7.16: Sixteen hour prediction for the last sixteen hours Friday 22-10-2004 using DHR model with data from the first eight hours and three previous Fridays.
error, that is to try to minimize the error inherent in the modeling method. Data assimilation can also be employed to minimize the observational error. To do this, knowledge about the way data is collected has to be modeled as well. For instance the number of observations behind an input value for the predictive model should be taken into account. Especially during nighttime when few observations are made, a few unusual observations can lead to a big distortion. The predictive model itself could be improved by switching to a multi variate approach where one or more of the external influences mentioned in section 7.3.2 is included.

7.3.4 Discussion and Requirements

The current solution is implemented entirely inside Matlab, it could benefit from a move to an e-Science environment. The methodology involved in using DHR is clearly suitable for implementation as a workflow. Benefits to this approach will be greater in particular areas. Increased computational power through the use of the grid is an area which offers relatively little potential improvements. In its current form additional computational power is not needed. The model is not very computationally intensive. The two scenarios in which grid computing could become useful are first when modeling a far bigger road network, or secondly when taking a brute force approach to analyzing the best model parameters. Both of these cases require many different instantiations of the model to be run, a task that is “embarrassingly parallel” and thus very well suited to grid computing. There are however far better methods for choosing the right parameters with the Captain toolbox. The toolbox offers analysis tools and an extensive manual which aid in finding the right parameters. Further knowledge is contained in papers concerning the individual tools of the toolbox[132]. The greater benefits of using an e-Science environment in this case can be found in the availability and sharing of knowledge. Easy access to the “provenance” of the workflows, data and tools involved in the experiment will help in disseminating implicit knowledge on how to properly use DHR in a traffic application. Knowledge can also be made available explicitly, in the simplest case by making existing documentation available in an on-line form, through a help function for each tool for instance. Expert knowledge on methodology can also be explicitly shared in a more interactive form. An expert in DHR can make template workflows that describe common ways of combining tools from Captain. An expert in traffic trying to employ the Captain toolbox in an e-Science environment can use both implicit and explicit knowledge to more efficiently create his experiment. Finally the specific dataset used in this experiment places demands on use in an e-Science environment. The data is stored on a remote database and as it originates from a commercial party the data has to be kept secure. Access to this data needs to be seamless to someone using it in an e-Science environment, but at the same time its confidentiality
has to be ensured. Only users with permission should be able to access it. Protection has to be robust enough for the commercial party to trust that access to its data will only happen under the terms that were agreed upon.

7.3.5 Conclusions for traffic prediction

We have looked at how we can enable accurate travel advice based on the prediction of the future state of traffic. We described the effect of the calendar on traffic and showed how modeling these effects explicitly can result in sensible predictions. We presented a data assimilation approach to creating a forecasting system. Furthermore we have experimentally shown the use of Dynamic Harmonic Regression for traffic forecasting and described how Dynamic Harmonic Regression fits within the data assimilation approach. Based on the results of the experiments we have discussed how the traffic forecasting system can be further developed within the data assimilation approach. The last ten years of research in Traffic Information Systems has shown that it is difficult to create useful services, unless the services that are offered have a high degree of accuracy and consistency. We are aware that experimental validation is key to this type of research.

7.4 Conclusion

The two case studies presented in this chapter generate very similar requirements if they are to be implemented in an e-Science environment. Both would benefit from features that ensure reproducibility, such as provenance recording, both for the data used and produced in experiments as well as for computational elements which comprise the experiment. The second benefit are features which help in the composition of experiments. In this regard there is a difference between the two case studies in that the SOS toolkit used for bird migration is relatively minimal in making explicit knowledge about the way it should be used where as the Captain toolbox is as well documented as one can expect for a stand alone toolbox. Yet even in the Case of the Captain toolbox much more knowledge about its use can and should be made explicit for use as a shared software resource in an e-Science environment. The documentation for Captain is focussed on the use of individual tools rather than the methodology needed to combine them all. The next chapter will show how such a methodology can be supported through a workflow in an e-Science environment.
Chapter 8

Ideal Workflow for Data Assimilation

8.1 Introduction

In this chapter a theoretical study is presented of what would constitute an ideal workflow for setting up a data assimilation system. This will be done along four main themes, first of all workflow representation: what needs to be expressed in the workflow and which basic building blocks are needed to achieve this. The second issue is the basic ingredients for workflow composition, the workflow constructs and definitions that are needed for building and composing a workflow. The third issue is the methodology for building a workflow. Within the framework of this third issue, data assimilation as a shared software resource will be explained. In particular what role is played by all the basic workflow constructs and how this achieves the goal of letting a non-expert end-user scientist construct a valid workflow for a technique outside his area of expertise. The construction of experiments within an e-science domain has already been addressed in the chapter “Analysis of Requirements for Virtual Laboratories” from the thesis of Ersin Cem Keletas[82]. It explains the specific tasks of the various players (scientist, domain expert, tool developer and administrator) have in constructing a domain specific experiment. The aim of the ideal workflow for data assimilation is to expand on this work to allow scientists from multiple domains to share their tools through methodologies made explicit by the domain expert. The emphasis in this chapter will therefore be on the scientist and domain expert since it is their role which is expanded.

8.2 Workflow representation

To properly express a workflow we need to look at what is needed for a workflow. First of all you need people to create the workflow and perform
experiments. Next, a goal or research question is needed for the experiments that will be performed. Then there is the data that you need in experiments, as well as a means to analyze and manipulate this data. A task that can be performed by man or machine. Finally computational resources are needed, on which to perform experiments described in the workflow.

- **Users**
  As was mentioned in chapter 2 several types of user exist: domain experts, end-user scientist, tool developers and administrators. In the representation it should therefore be clear what type of user is supposed to perform which parts of the workflow.

- **Goal**
  The goal of the workflow has to be clearly expressed, in the case of data assimilation it should be made clear what measure is to be predicted and how far ahead should the prediction be. Furthermore the granularity and accuracy of all the parameters should be clear, for instance how big is each time step and what is the maximum allowable prediction error.

- **Data**
  Data used in the workflow has to be able to come from any source which can keep up with the demands a workflow places on it, whether this is a relational database, a comma separated file or a human typing in numbers by hand. The location and nature of the data source has to be expressed in the workflow in order for the end user scientist to know what data he is using.

- **Data analysis & manipulation**
  Steps within the workflow which manipulate or analyze data should be clearly recognizable. Furthermore it should be clear from the workflow representation whether such a step is performed entirely by software or requires some form of human interaction.

- **Resources**
  The exact location of the computational resources and the storage resources used in the execution of the workflow do not need to be explicitly defined. The workflow management system should be able to take care of reserving and scheduling appropriate resources for execution. However if debugging has to be performed one should be able to find the specifics of actual execution.

The most intuitive way to present and compose a workflow for data assimilation uses a temporal view. This means that steps in the workflow are ordered chronologically and that a connection between two steps implies that one is performed after the other. As data assimilation is an iterative process
which contains feedback loops, the occurrence of loops should be allowed in the workflow representation. To maintain a good overview a hierarchical representation of a workflow should be possible. At the highest level a workflow consists of just a few main steps which hide underlying subworkflows. When a data assimilation experiment has many steps in data preparation a good overview can be maintained when this stage of the workflow can be collapsed into one composite step in the workflow representation.

8.3 Workflow composition

The composition of a workflow can be a complicated process, especially for an end user scientist who composes a workflow that employs resources outside of his area of expertise. Within e-Science it is an important goal to enable an end user scientist to do just that. In this section we will explore how composition would ideally happen with data assimilation as the technique that has to be implemented by a non domain expert. First all the prerequisites before the actual composition starts will be investigated, followed by the composition itself.

8.3.1 Defining data

For data to be useful in e-science it needs to be properly defined. This is a task that has to be performed by whomever introduces data into the e-Science environment: most likely an end user scientist who wants to perform an experiment. Most important is the definition that will directly affect the functionality of a workflow. This functional definition consists of:

- data type(int, float, string)
- data form(file, stream, database)
- data location (url, database query)

The type and form definition will allow the SWMS to determine whether data is compatible with resources in the workflow. While data location is essential not just for execution but also for reusability and reproducibility. Apart from the need to define this basic information, potential semantic user assistance at the composition stage can only be exploited if semantics for the data are defined. Another benefit of adding semantics is to make workflows more meaningful in a collaborative environment where more than one end user scientist is working with a workflow. Important meta data to be added for data assimilation is the following:

- Data origin: Data assimilation deals with making predictions in time and or space, therefore it is important to define when and where data
was gathered. Apart from knowing where and when data was gathered it is also important to know how this was done. For instance what type of instrument or sensor was used, and what was the setup. When setting up a data assimilation experiment for real time use, it is important to know if there is a delay involved in the data collection method. Other end user scientists should be able to easily determine the nature of the data when working in collaborative environments or when offering a workflow for reuse.

- **Data classification**: Classifying data according to domain terminology is information that can be used later on to provide user assistance. Within data assimilation “time series data” and “GIS data” are broad terms that can be helpful. But more application specific terms such as “floating car data” should also be defined here.

- **Data uncertainty**: If anything is known beforehand about the quality and especially the uncertainty of data, for instance the error distribution produced by a sensor is often known, it should be stored in meta data. In data assimilation the minimization of uncertainty is the main goal, therefore knowing the uncertainty in the input data is a very important consideration when composing a workflow. Formalizing this knowledge in meta data could help the SWMS provide user assistance in this area.

### 8.3.2 Defining resources

Similar to data, workflow resources also need to be defined. In this case it is the person who has created the resource, most likely the domain expert, who has to perform the definition. The functional definition in this case consists of the following:

- resource location (url)

- for each input:
  - expected data type
  - expected data form

- for each output:
  - produced data type
  - produced data form

For providing better user assistance and facilitating collaboration the following meta data can improve resource definitions:
8.3. WORKFLOW COMPOSITION

- **Runtime behavior:** Whether the resource is fully automated or it requires some form of human interaction during execution. How much time execution is estimated to take.

- **input/output characteristics:** What class of data the resource expects as input and produces as output, for instance: “time series data”. What minimum quantity of data is needed for a meaningful result. What degree of uncertainty this resource can deal with. What the relation is between input and output data, in terms of quantity and uncertainty.

- **Context:** Which other resources are usually combined with this resource. Which other (partial) workflows employ this resource.

- **Documentation:** Links to textual documentation of the resource itself and links to background information on the techniques used in this resource e.g. scientific papers.

- **Access:** Who is allowed to use this resource? Users belonging to a project, all people belonging to one organization or even people outside of the organization for which the resource was developed.

- **Cost:** What is the price attached to using this resource for different users, e.g. it is free to users from the within a project while outside users might be charged a fee.

### 8.3.3 Defining goals

When constructing a workflow, a clear goal for that workflow is needed. The research question for an experiment needs to be clearly formulated at the beginning of an experiment. This task falls clearly under the responsibility of the end user scientist who is constructing a workflow. In the case of Data Assimilation the required goal is always a prediction of some sort. At the start of workflow construction the required prediction should be specified at the very least at the functional level. Type and form of required output data should be specified as well as where the output should be directed: a database, a visualization on screen etc. In addition higher level characteristics as described in the data (8.3.1) and resource definition (8.3.2) subsections can be added. For instance the maximum allowable uncertainty can be pre-defined or restrictions on computation time can be imposed for real time systems.

### 8.3.4 Provenance

Until now this section has made clear what meta data can and should be defined and by which type of user. But not all meta data has to be added
by a user, some of it can be added automatically. Provenance data based on the context in which a resource was previously used is well suited for automatic addition. Also information on average execution time of a resource can be gathered automatically. The Provenance modules within a SWMS continually gather important statistics, currently mostly about data, but the module could also automatically derive many more useful meta data of the types just mentioned.

8.3.5 Partial workflows

Certain parts of a new workflow will have much in common with previous workflows. The combinations of resources can often be the same for many experiments. Therefore it is useful to have support for partial abstract workflows. Partial because they do not cover a whole experiment but just a commonly reoccurring topology of resources. Abstract because they have to be independent of the specific details of the experiments being performed in the workflow thereby making them suitable for reuse. They need to have the same definition as a resource would have to allow proper user assistance for employing them. The creation of these partial workflows can happen in several ways, first and most importantly the domain expert should define the most common partial workflows for his domain when he wants to make a set of his tools available as a generic software component. In case of non-trivial patterns he can use a formal approach using one of the formalisms described in chapter 4. Based on this formal analysis the domain scientist could implement these patterns in a suitable SWMS. Secondly an end user scientist can construct an abstract workflow from a concrete workflow when there is a need for sharing it. Finally many workflow patterns are not domain specific but do occur frequently. For instance running an ensemble estimator within a data assimilation experiment constitutes a pattern that is in essence a parameter sweep: a very common e-Science pattern. An expert in formal analysis of workflows can therefore provide standard patterns for workflow formalisms that are formally checked. As second step scripted versions of these patterns can be made. These can generate patterns for arbitrary numbers of inputs and/or outputs where applicable. An ensemble estimator is in essence the same pattern whether it involves 10 or 30 ensembles.

8.3.6 Dissemination

Workflows and software resources need to be published somewhere in order to be disseminated. A software resource can exist as a set of web or grid services, but potential users need to be made aware they exist. Currently this is done by writing scientific publications about them, publishing them on dedicated webpages, or including them in the list of standard services in a SWMS. The meta data associated with these services aids their discovery.
by other users.

### 8.3.7 Meta workflows

The goal of the workflow design methodology for data assimilation is to allow an end user scientist to properly build a workflow for using this technique even if it is outside the area of his expertise. In order to do this he not only needs help in connecting individual resources, also a specific methodology for the shared software resource is needed. For this a high level workflow, or meta workflow can be employed. In figure 8.2 the meta workflow for data assimilation is presented. It shows the main high-level steps for a data assimilation system as well as the data flow between these steps and the iterative activity order. For each of the steps involved in this workflow another more detailed meta workflow exists (figures 8.5, 8.6, 8.7). These meta workflows should be created by the domain expert who knows the methodology involved. As a visualization method for these meta workflows a common modeling standard such as UML could be used. This though is not common practice within actual SWMS implementations. The method chosen for visualizing these workflows is one similar to the methods employed in many SWMS's. As shown in figure 8.1 data is represented by rounded rectangles, activities are represented by normal rectangles while data flow is represented by dashed arrows and the execution order of activities is represented by solid arrows. In the next section a description is given on how meta workflows and all the other workflow ingredients mentioned in this section come together in a workflow design methodology.

### 8.4 Workflow Design methodology for Data Assimilation

Now that all the prerequisites for building the workflow have been listed, a methodology is needed to put it all together. First we will explain how we view a shared software resource, then we will show the meta workflows expressing the workflow design methodology for data assimilation.
CHAPTER 8. IDEAL WORKFLOW FOR DATA ASSIMILATION

Figure 8.2: Meta workflow

Figure 8.3: Executable workflow for data assimilation
8.4.1 Shared software resource

The essence of turning a software resource into a shared one from the perspective of the domain experts entails making their software generic, enabling end user scientists to make use of their expert knowledge without them being present. In the previous section it was mentioned what a domain expert should do to make his knowledge explicit and accessible to end user scientists constructing a workflow. The collection of meta workflows, partial workflows and resource definitions each with their own meta data and documentation together with the resources themselves is what constitutes a shared resource. In this chapter we look at what the workflow design methodology is for a workflow using data assimilation as the shared software resource.

8.4.2 Methodology

We will now look at the shared software resource from the perspective of the end user scientist. In particular we will detail the methodology he has to follow to use this shared resource to create a concrete executable workflow. As a first step he needs to define the data he wants to use and the goal he wants to achieve, as detailed in 8.3.1 and 8.3.3. The method followed can be schematically expressed as represented in figure 8.4. First the meta workflow comes into play: each high level step in the meta workflow needs to have details added to make it more concrete. For the Data assimilation workflow in figure 8.2 this means filling in the data preparation, estimator and model steps.

- **1.1** The Data preparation step which will be detailed in section 8.4.3 requires data (1.1a) and a data preparation goal (1.1b) as input. This data preparation goal can be the same as the overall goal, but when the development of the concrete workflow is done in several iterations the estimator workflow can put additional constraints on the data preparation goal. The end result is an executable workflow for data preparation.

- **1.2** The Estimator step which will be detailed in section 8.4.4 requires prepared data (1.2a), a model (1.2b) and a goal (1.3a). The prepared data is the output produced by the executable data preparation workflow (1.1c) and therefore also equivalent to the prepared data (2.1c) in figure 8.3. The end result is an executable estimator workflow (1.2c).

- **1.3** The model step which will be detailed in section 8.4.5 requires the goal (1.3a) and a current state estimate (1.3b). This current state estimate is the result of the executable estimator workflow and therefore also equivalent to the current state estimate (2.2a) in figure 8.3. The end result is an executable model workflow.
The methodology for data preparation is not unique to data assimilation. Processing complex data structures and combining data from heterogeneous sources is an active area where work has to be performed on all steps in parallel. At each level of the composition process the end user scientist can be assisted by a semantic search engine, which uses the constraints imposed by the (sub)goals together with a statistical analysis of previous workflows to suggest concrete resources or even partial workflows. Furthermore the meta-workflow, partial abstract workflows and concrete resources all come with their own documentation which can be used as an extra source of information. In the following subsections we will elaborate on this methodology by detailing the meta-workflows for each of the main data assimilation steps. The steps within these meta-workflows are in most cases not simple computational steps, but rather human activities supported by computational tools. The result of this meta-workflow though should be a workflow that is almost completely computational.

8.4.3 Data preparation

The first step in the meta-workflow for data assimilation is data preparation. The methodology for data preparation is not unique to data assimilation. It is a process that needs to be performed in many e-science applications. Different fields of research are moving towards on-line data processing, processing complex data structures and combining data from heterogeneous sources. This is for instance identified in [25] for the field of data mining. Existing methodologies for data preparation in data mining should to a
large extend also be applicable to data assimilation. Thus instead of developing a methodology from scratch the CRISP-DM (CRoss Industry Standard Process for Data Mining)[49] methodology which extensively covers data preparation is used as a basis. It is assumed that the end user scientist who composes a workflow already knows the important properties of his data and also the goals he wants to achieve with the execution of the workflow. The business understanding and data understanding which precede data preparation in the CRISP-DM methodology are therefore assumed to be completed. In figure 8.5 the workflow for data preparation is presented, Data (3.1a), Goal (3.1b) and Concrete Data Preparation Workflow correspond with their namesakes in figure 8.2 (1.1a, 1.1b and 1.1c respectively). What follows is a step by step explanation.

- **3.1** The first task that has to be addressed is data selection. Based on

![Workflow Diagram](image-url)
CHAPTER 8. IDEAL WORKFLOW FOR DATA ASSIMILATION

the data preparation goals (3.1b) a selection of the data(3.1a) can be made. For instance the scientist might only be interested in predicting events in a certain area or time period, and can therefore exclude a lot of data which he deems irrelevant. Data can also be selected on the basis of quality, or quantity when there is more data than the subsequent data assimilation system will be able to handle. The reasoning for this selection is expressed in a selection report(3.1c). The other output of this step is the selected data(3.1d) itself.

- **3.2** From the Selection report relevant meta data can be abstracted and added to the selected data.

- **3.3** In the data cleaning step, the quality of the data is improved to meet the requirements set by the (sub)goal. For instance gaps in the data may need to be filled, either by sensible default values or by filling in the gaps using modeling techniques. In fact the model that will be used for data assimilation may very well be suitable for filling in gaps as well. The result of the data cleaning process is a set of cleaned data(3.3b) and a report(3.3a) explaining how and why the data was cleaned and what impact this may have on later stages of the experiment.

- **3.4** From this report important meta data can be extracted for instance marking which data was filled in and which data was unaffected by the cleaning process.

- **3.5** Within data construction existing data is used to generate or derive new data. The goals may demand that data which is implicit in the dataset be made explicit when generating new data. It is also possible that the modeling technique that will be employed demands that new data attributes are derived from existing ones, for instance transforming speed measurements from m/s to km/h. Associated meta data will also have to be generated/transformed for the resulting set of constructed data(3.5a).

- **3.6** Data integration takes care of both merging data and data aggregation. Merging data from different sources is employed when the data preparation goals require one data source while the required information in reality comes from multiple data sources. For instance combining bird observations for a certain location with the wind-data for that same location. Data aggregation is a very important process within data assimilation. Usually the observations are not of the same granularity as the model expects. Choosing the way in which data is aggregated to suite the needs of the employed model is of great influence on the eventual accuracy of the system. Apart from merging and
aggregating the data itself the associated meta data also needs to be transformed. The result is an integrated data set (3.6a).

- **3.7** The final step in data preparation is data formatting. This is a syntactical operation in which all the data is set in the order expected by the data preparation goals resulting in a formatted data set (3.7a).

The result of this meta workflow should be a concrete workflow that prepares the data for use by the estimator. For most data assimilation cases this will be a purely computational workflow selecting cleaning and transforming data within real time constraints.

### 8.4.4 State Estimation

Within data assimilation the choice for a particular type of estimator is an important one. The task of the estimator is threefold:

- Filtering, minimizing the error of the current state
- Prediction, forecasting into the future
- Smoothing, backcasting using the observations available up to the present

The emphasis in this thesis is on prediction. The other two goals however can play a part in improving the accuracy of prediction as well. The estimator tries to achieve its goals by minimizing the model error and the observational error. The former is done by adjusting parameters within the model or parameter estimation, the latter is done by estimating the current state on the basis of previous observations known as state estimation. To determine which estimator is appropriate the properties of the data, model and data assimilation goal have to be well known. The workflow presented in figure 8.6 therefore mainly concerns itself with this. It is possible that all the information needed is already available in the meta-data, however in practice some things will need to be checked. The prepared data (4.1a), model (4.2a), goal (4.4a) and integrated estimator workflow (4.7a) match their namesakes from figure 8.2 (1.2a, 1.2b, 1.3a and 1.2c respectively). What follows is a step by step description of the estimator meta workflow.

- **4.1** The nature of the noise in data is important since many estimators assume this noise to be Gaussian and some estimators are able to deal with non Gaussian noise better than others.

- **4.2** In principle estimators are not suited to non-linear problems. However there are estimators which are able to deal with some forms of non-linearity. It is therefore important to know if a problem is non-linear and if so in what way.
• **4.3** When using parameter estimation a number of model parameters can be adjusted by the estimator. A decision has to be made on which parameters should be adjusted: all of them or just a subset. Furthermore there is a need to determine how these parameters can be modified by the estimator.

• **4.4** In many cases data assimilation is used for real-time predictions. This places limits on the computing time available. Some estimators, for instance ensemble Kalman filters, require the model to be run many times. The models computational requirements can severely limit how many times the model can be run for each prediction.

• **4.5** Constraints on which estimators are suitable can follow from the nature of the desired predictions and/or model: are they continuous or discrete.

• **4.6** After having determined the constraints placed on estimator choice by data (4.1b) and model (4.2b), an actual choice has to be made. Through the use of these constraints the search-space can be limited allowing a semantic search engine to suggest suitable candidates to the end user scientist composing the workflow.

• **4.7** The suitable estimator has to be integrated with the model. If they are a perfect match this should be plug and play, but in practice it may require alterations to either the model or the prepared data.

Once integration is complete a concrete workflow for a state estimator exists. It needs no user interaction to run, only prepared data and feedback received from the part of the estimator that executes after the model has made its predictions.

### 8.4.5 Model

Within the meta workflow for data assimilation the model is an optional precondition. In case a model is not available there is support for building a model. In this workflow design methodology there is support for every modeling technique that is not constrained by the basic demands of data assimilation: the ability to make predictions and having adjustable parameters with known functionality. As the number of modeling techniques involved is huge, the workflow for modeling is limited to a general methodology based on the modeling part of CRISP-DM. The Current State Estimate (5.1a), Goal (5.1b) and Integrated Model Workflow (5.6c) match their namesakes in figure 8.2 (1.3b, 1.3a and 1.3c respectively). What follows is a step by step explanation of this methodology which is shown in figure 8.7.

• **5.1** As a first step a modeling technique has to be selected. This choice is already somewhat limited by the Data Assimilation criteria listed
8.4. WORKFLOW DESIGN METHODOLOGY FOR DATA ASSIMILATION

Figure 8.6: Meta Workflow for estimator
above, however it can be limited further by looking at the specific goals for the experiment that is being constructed and the available data. The result is a modeling technique and a report listing the modeling assumptions. This report can be used to find and correct any mismatches with the prepared data later on in the model integration step.

- **5.2** The next step is generating a test design, the purpose of this is to determine how a model should be tested. What is the quality measure, how large should a training set be etcetera. The resulting test design is a plan for training testing and evaluation.

- **5.3** With this in place a model can be build. It needs to be tested according to the test design by determining a good set of initial parameters. The model should be described in all its important aspects such as expected accuracy, robustness and computational complexity.

- **5.4** The model should be assessed to ensure that it meets all of the demands that will be placed on it. This can be done by further testing, checking by a domain expert, checking the plausibility and reliability of the model results. This step can also lead to further insight into the effect of parameter settings, leading to revised initial parameters.

- **5.5** The model description and parameter settings can be used to generate meta data for use in the rest of the workflow.

- **5.6** In the model integration step the model needs to be matched to the prepared data and parameter estimation, especially in the case where a model was developed outside of the meta workflow.

This results in a concrete workflow for a model that can be adjusted and produce predictions without the need for human interaction.

### 8.4.6 Workflow Patterns

The combination of data preparation, estimator and model should lead to a concrete workflow. In the beginning of this section the simplest form of an executable workflow for data assimilation was shown in figure 8.3. Different more complicated patterns are also possible. Sometimes the choices in sub workflows that create the steps in the executable workflow can influence another. The most notable example in data assimilation is the interaction between model and estimator. This interaction comes from two sides. First, within the model the need can arise to do parallel computation: that is divide the data that is used as input and run many parallel instances of the model each with different data sets. Second, the estimator chosen can be of the ensemble type requiring many instances of the model to be run
Figure 8.7: Workflow for the creation of a model that can be used in data assimilation
each with slightly different initial conditions. In the most extreme case both can occur in the same workflow. In Figure 8.8 such a workflow with multiple model instances is illustrated. As can be seen from this figure, the execution diverges into three separate threads and converges again. Before the estimator can run again all model computations need to be finished as the global state estimate has to be determined before another iteration of the loop can occur. The data set and initial conditions for the next model iteration have to be determined which puts an extra burden either upon the estimator in case of ensembles or on data preparation in case of parallel computation in the model.

8.5 Optimization

A completed concrete computational workflow is by no means the end of the design process. The performance of the workflow needs to be evaluated so that adjustments to the workflow design can be made. This evaluation will focus on the following points:

- The data has to be clean enough and contain enough useful information for the model to work.
- The model has to be validated against the data to gauge its accuracy.
- The error minimization has to perform well enough to improve the models predictions.
- The speed at which the estimator converges. In other words: how fast the size of the adjustments the estimator has to make at each time step reduces. This convergence continuous until a (near) optimal error minimization is reached.
- The results of the workflow need to meet realtime constraints.

The remedy to these problems can be found in each of the three basic steps: model, estimator and data preparation. In order to explore alternatives in either three steps, parameter sweeps on the entire workflow can be performed by running many different instances of the computational workflow each with different parameter settings.

8.6 Requirements for Scientific Workflow Management Systems

After showing what an ideal workflow for data assimilation looks like, we will conclude this chapter by analyzing what requirements this workflow places on real SWMS if it were to be implemented. These requirements have been divided into meta-data, expressivity composition and grid requirements.
Figure 8.8: Workflow with multiple model instances
8.6.1 Meta-data

Maintaining meta-data during the execution of a workflow has two goals: reproducibility of the whole workflow and creating a history of experiments which can be used as a blueprint to assist in the composition and the execution of future workflows. Reproducibility of a particular run of a workflow is important within an e-science context as scientific experiments need to be reproducible. Furthermore properly maintained provenance data can help other scientist to reproduce a particular workflow using their own resources which is an important capability for the peer review of workflows. The other use of provenance data is to help in the composition of a new workflow. For instance: a scientist implementing a new data assimilation workflow using an ensemble Kalman filter can learn how it was implemented in the past by using the provenance data associated with an ensemble Kalman filter. Three forms of meta data are required: first provenance of data, what particular data set was used at what time. Secondly, provenance of topology, meaning a workflow component was connected to other components at a certain time and which of these connections were active. Finally meta data descriptions of the interfaces of components are desirable: e.g. an input port takes integers which describe temperature in degrees Celsius with the capability of processing a certain number of these inputs per second.

8.6.2 Expressivity

The expressivity of a SWMS and more specifically the workflow language it employs determine what workflows can be created. For data assimilation the obvious construct that is needed is a loop because the model and the estimator are iterated many times during a typical data assimilation workflow. Less obvious but still needed is support for parallel execution, in terms of the workflow patterns[24] mentioned in chapter 3. Both the "AND split", "OR split" and some form of synchronization are needed. The "AND split" is needed for distributing the same data to different instances of the model, for instance when transporting observational data to different model instances in a workflow utilizing an ensemble Kalman filter. The "XOR split" is needed when transporting different data to each instance, for instance when using a parallelized model where each instance needs a different part of the observation data. After all model instances have finished, synchronization of execution is needed. In order to determine the global current state estimate, the estimator needs input from all models. If data communication is not explicitly modeled and a globally accessible data store is used just "AND split" and synchronization suffice. This is however an undesirable situation as data assimilation is driven by streaming data, thus not showing the main driving force explicitly obscures the way the workflow operates to the user.
8.6.3 Composition

The composition of workflows involves more than connectivity and synchronization as described above. The data assimilation workflow as presented in this chapter involves hierarchy, abstract workflows, abstract to concrete composition, dynamic workflow generation and human in the loop computing. Hierarchy is needed to implement the part of the ideal workflow where one workflow component can represent an entire subworkflow, for instance the data preparation step in figures 8.3 and 8.8 represents the subworkflow depicted in figure 8.5. Furthermore many workflow steps in the ideal workflow do not at first represent a real executable workflow step but rather an abstract representation that has to be instantiated to a concrete one at some point during the composition process. So apart from hierarchy abstract workflow components and abstract to concrete composition should be supported. The optimal number of parallel model instances used in a workflow utilizing ensembles or a parallelized model can differ depending on the input data used. Dynamic workflows can be desirable if this optimum can be computed in a dynamic workflow where the user does not have to specify or compose by hand all the parallel instances needed. During the execution of a workflow an end user scientist will often want to analyze intermediate results in data preparation and adjust workflow component parameters based on his expert opinion. The SWMS can support this through computational steering; parameter adjustments at runtime as well as the user pausing and restarting the workflow to analyze results, or even steps in the workflow which are human activities.

8.6.4 Grid support

Data assimilation often has to deal with massive data and associated massive computation, especially if more compute intensive estimators such as the ensemble Kalman filter are employed. Thus when executing such a data assimilation workflow, grid support is desirable. Each workflow component can be deployed to a different grid node according to its computational needs. The data in data assimilation should be routed directly from component to component and not via a central engine. The scheduler associated with the SWMS should be able to deal with dynamic workflows as described above. Finally it is desirable that when a model instance is run multiple times it can maintain its state on the node on which it is running while communication only consists of the changes in its state instead of sending all data for each iteration.
8.7 Overview of features in existing SWMS

Presented in table 8.1 is an overview of the extent to which current SWMS’s support the features required for the ideal data assimilation workflow. The table is explained along the three main requirements of provenance, expressiveness, computation and grid support.

Meta-Data

As can be seen in Figure 8.1 currently only two SWMS more or less support all provenance requirements. These two systems, Pegasus and Taverna, also happen to be the most constraint in expressiveness. Their workflow topologies are based upon directed a-cyclical graphs (DAG). This makes collecting provenance data on topology relatively easy. One could argue that if the data and component provenance is rich enough, topology provenance could be derived, certainly in combination with the workflow description itself. This has not yet been done however and for more expressive control-flow oriented systems or for a system such as Kepler which has multiple execution models this is a far from trivial task.

Expressiveness

The ”AND split” is supported by all systems however XOR and Synchronization, two other important patterns for data assimilation, are supported only implicitly by some systems. In these data-flow oriented systems one can express these patterns by embedding functionality inside workflow components. The systems are thus able to perform these patterns but have no explicit representation for them. Creating the patterns in this way also limits the potential for reuse of these workflow components because they implicitly embed this functionality. Kepler supports multiple models of execution which are called directors. It supports synchronization implicitly in its SDF (Synchronous Data Flow) director, explicitly in the DE (Discrete Event) director but not in the PN(process network) director. XOR split is supported explicitly in DE, however this is not a director generally used for workflows. Kepler supports directors which are not suitable for workflow because it is build on top of Ptolemy II, a hardware simulator.

The case of loops is far more clear cut. In cases where they are not supported it will be difficult to create a data assimilation workflow as a loop is inherent within data assimilation. Something which is clearly illustrated by the two case studies presented in chapter 7 as well as most other data assimilation scenarios.
<table>
<thead>
<tr>
<th></th>
<th>GVLAM</th>
<th>Kepler</th>
<th>Taverna</th>
<th>Triana</th>
<th>Pegasus</th>
<th>ICENI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta-Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data provenance</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>topology provenance</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>component meta-data</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Expressiveness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>loops</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>XOR split</td>
<td>implicit</td>
<td>depends on director</td>
<td>implicit</td>
<td>implicit</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>AND split</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>synchronization</td>
<td>implicit</td>
<td>depends on director</td>
<td>yes</td>
<td>implicit</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Composition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hierarchy</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>abstract workflows</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>refinement</td>
<td>no</td>
<td>no</td>
<td>limited</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dynamic workflows</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>limited</td>
<td>limited</td>
</tr>
<tr>
<td>grid support</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>grid scheduler</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 8.1: Scientific Workflow Management Systems compared
CHAPTER 8. IDEAL WORKFLOW FOR DATA ASSIMILATION

Composition

Hierarchy is supported by all the systems in this comparison, however other features for composition receive less support. Abstract workflows are only supported in DAG based workflow systems. GVLAM does support the concept of "study" which can be viewed as a form of abstract workflow. In a "study" all steps performed by the scientist are made explicit while actual computational steps are handled in a separate workflow representation. The study itself is not refined into an executable workflow: rather one step of a study can represent an entire computational workflow. Both Iceni and Pegasus offer the ability to dynamically change the number of parallel instances of a workflow component. Computational steering, altering of aspects of the workflow at runtime, is not directly supported in any of the systems. Some workarounds are possible by implementing workflow components which offer an interface that is independent of the SWMS to offer steering of parameters associated with that component.

Execution

The massive computing power needed by data assimilation workflows dealing with large amounts of data and complex models, can be accessed through the grid. All systems except Taverna offer some form of grid support. Kepler currently has the possibility of grid actors which can run on the grid, it does not offer its own scheduler to optimize the running of an entire workflow of grid actors on the grid. Dynamic workflows where the topology of the workflow can change during execution are to some extent possible in ICENI and Pegasus. In these systems the topology could potentially change based on input data to create more parallel jobs when required. Dynamic workflows which allow for computational steering are not currently supported. For computational steering a user should be able to stop certain workflow components while the workflow is running and possibly exchange them for others.

8.8 Discussion & conclusion

The methodology for sharing resources as presented in this chapter puts a large burden on the domain expert who has to make his resources available. What would be the motivation for him to do this? There is only a small direct benefit to his own research having a larger group of users. Benefits include raising the profile of his research and receiving feedback from the user community. This alone probably does not warrant the effort.

There are examples of how, through a change in the way science is funded, the needed incentives can be offered. The National Centre for Text mining[34] in the United Kingdom is a good example of this. This Na-
tionally funded center offers support for textmining tools to scientists in the whole of the country. Among other things its developers provide support for interoperability and sharing of these tools. By directly funding the support for shared tools instead of only individual research the conditions have been created to make text mining a viable shared software resource. A similar setup could be used for data assimilation.

While in earlier chapters the focus was on formal methods to aid workflow design, in this chapter we can see that the largest amount of work in sharing software resources actually goes into making domain knowledge explicit and accessible to the end user scientists. Data assimilation workflow design can benefit from formal methods in validating the design patterns needed for ensemble estimators and for job farming of entire workflows. Currently there is no single SWMS that can support all the features that are needed to implement the ideal workflow presented in this chapter. There is a clear dichotomy between systems offering all the meta data support needed, such as Taverna and Pegasus, and those offering the expressivity that is needed such as Kepler and Triana. Taking into account grid support and composition removes Taverna form consideration. The fact that a data assimilation workflow does not execute without a loop leaves Triana and Kepler as the most likely systems to use at this moment. In the near future one can expect development of additional features in all of the systems. Especially provenance is a very active research area. Even with more features it is unlikely that there will be one system that supports all the features needed for data assimilation. Conventional wisdom might lead to the development of a specific workflow system for data assimilation. A better approach would be to exploit and encourage interoperability between systems. One answer to greater interoperability can be found in the development of the workflow bus[135] which can connect different SWMS together. This allows a scientist to exploit the features of multiple SWMS in one experiment. Another way to achieve greater interoperability is to take a more modular approach in the development of SWMS, for instance the searching in provenance data could be made workflow system agnostic. A possibility explored in the Second Provenance Challenge[99].


Chapter 9

Conclusions & Future work

9.1 Introduction

This thesis has looked into what is a proper design methodology for both workflow components and workflow topologies that supports the sharing of software resources. In workflow design we have found the desirable properties of maximal simplicity, generality and consistency to be mutually exclusive in the representation of a workflow. Furthermore we have found that automatic workflow composition is not possible in the general case. We have shown how abstraction and refinement play an important role in formal approaches to workflow design. The use of these notions requires the use of a new formal approach. Based on the lessons learned from the two case studies we presented an ideal workflow design method for data assimilation. This method uses refinement of abstract workflow components to derive a concrete computational workflow for data assimilation. This design method can be used to offer data assimilation as a shared software resource. However current Scientific Workflow Management Systems do not have all the features needed to actually support this design method and use it for the implementation of a data assimilation workflow. This concluding chapter starts with an overview of the original research performed for this thesis. The results from the previous chapters will be shown as well as an overview of the role of e-science in workflow and the current state of workflow in e-science. The thesis ends with suggestions to improve workflow design in e-science in the future.

9.2 Work Performed

In chapter 3 an extensive study of the formal representation of workflow systems was presented. It identified and analyzed the formal aspects of the workflow design problem. A lattice was derived, setup by the four most extreme cases of workflow representation. Using this lattice it was proven
that maximal simplicity, generality and consistency are mutually exclusive in workflow representation. A formal version of the general workflow design problem was defined and we showed that this problem is undecidable for the general case.

An extensive analysis of workflow formalisms and workflow systems was given. In chapter 4 five formalisms were compared for reasoning about workflow design within the design space setup by the lattice from chapter 3. While reasoning about workflow design in the past focused on expressiveness, we emphasized the importance of abstraction. An example of a problem in which abstraction plays an important part is a workflow with multiple models of execution. This problem occurs when using multiple SWMS in one experiment. In the comparison we found constraint automata to be a particularly suitable formalism for this kind of problem. A general overview of available SWMS was given in chapter 5, while at the end of chapter 8 features specific to implementing data assimilation in a SWMS were provided, based on the previously given design method.

For chapter 7 experiments were setup and performed for real life data assimilation applications. The bird migration case study was an early exploration in how data assimilation techniques could be used with an expert based biological model describing behavior. The traffic prediction case study presented a data assimilation approach to creating a traffic forecasting system. These two case studies give insight in how two data assimilation tools, the SOS toolkit and Captain toolbox, work in practice. Based on this experience and the theoretical design lessons from the first chapters in the thesis an ideal workflow for data assimilation was designed. In chapter 8 this ideal workflow was presented and features needed to implement this workflow were compared for existing SWMS.

9.3 Role of Workflow in e-Science

This thesis will not provide a design method for sharing resources in general. Through the analysis of workflow design and its application to data assimilation we have given an answer for one particular domain. From this one domain lessons can be learned for workflow design methodology in general.

9.3.1 Resource Sharing

The benefit of e-Science and in fact a possible paradigm shift in science in general lies in the increased area of the search space that can be covered and the associated increase in the number of hypothesis that a scientist can expect to test when conducting e-science experiments. This benefit can only be obtained through the sharing of resources. In this thesis the focus has been on the sharing of software components that are used in e-science workflows. From the data assimilation case studies presented in chapter
9.3. ROLE OF WORKFLOW IN E-SCIENCE

the need for resource sharing might not be immediately obvious. It is clear however that the two case studies are from entirely different scientific disciplines, yet they both center on the same type of computational tools. If data assimilation tools are available as a shared resource they reduce the development time needed to do experiments in any of the different scientific fields that use them. Scientists can thus create more complex experiments with the same effort and cover a wider search space when answering their research questions. The downside of this approach is an increased reliance on the work of others: when using a shared resource a scientist needs to know he can trust its accuracy.

9.3.2 Dissemination and publishing

In the future publishing workflows and associated shared resources will become at least as important as publishing papers. Workflow is an important mechanism in the process of dissemination. Currently publishing scientific papers is the main method for dissemination. If the adoption of e-Science takes hold this may change. Within e-Science more than just results are shared: in areas such as bio-informatics publishing workflows along with results is becoming a common practice. Currently shared (software) resources and workflows are mainly published by the scientists themselves or the institutions they work for. The mechanism of peer review should not be limited to scientific papers. To go someway towards resolving the issue of trust shared resources and workflows could be a more substantial part of the peer review process or be peer reviewed separately and published by independent publishers.

9.3.3 Reproducibility

Workflows enable scientists to reproduce results, but this is not a trivial matter. The provenance of a workflow needs to be recorded, both the data provenance, which steps were taken to create a piece of data as well as process provenance describing the manner in which processes interacted during the execution of a workflow. A formal grounding for a workflows design can more easily help to show that the recorded provenance is adequate for reproduction.

9.3.4 Workflow design

Designing a workflow is analogous to formulating a hypothesis. In formulating a hypothesis a scientist starts with a general requirement, a research question, and moves to a precisely defined testable hypothesis. Workflow design is similar in that it starts the general requirements of a workflow which are then refined into an executable concrete design which can answer (part of) a research question. The empirical cycle for e-science, as defined
in chapter 2, shows how workflow design is one of the fundamental steps in the scientific process as it is performed within e-Science.

9.4 Current state of Workflow in e-Science

This research in this thesis has touched upon many different aspects of workflow in e-Science. In this section an overview is given of the current state of workflow in e-Science and in particular the state of workflow design.

9.4.1 Workflow design

Workflow design as understood and supported in current workflow systems [93, 72, 113, 96, 104, 87] is lagging behind established design methodologies for related fields such as parallel computing. In workflow design many of the same issues relating to both the control of processes and data flow exist that have long been present in parallel computing. Yet a formal approach to software design, as it exists in parallel computation to deal with these issues, has not been adopted for workflow. Formalisms have been used for automated checking of workflows[71, 84], to see whether the workflow is grammatically correct and can finish. There are also workflow languages for which a formal definition has been made[102, 47]. There is however no support for designing workflows using a formal approach in current SWMS.

9.4.2 Formalisms for Workflow

The review of workflow formalisms in chapter 4 found that constraint automata[39] was the most suited formalism for workflow design within the workflow design lattice. However tools to support constraint automata, or other formalisms for that matter, are not sufficient. The tools for constraint automata lack a stable user friendly GUI for composition. The code generation that is possible from constraint automata does not have a stable implementation, and currently there are only a few standard workflow patterns[98] available. Tools for other formalisms suffer from similar shortcomings. The theoretical aspect of the formalisms are well developed but they are still lacking usability.

9.4.3 Scientific Workflow Management Systems

Current scientific workflow management systems all have some basic common features. They have a mechanism to define a workflow and to execute it. There is a whole array of different additional features possible of which each system has its own unique combination. This makes each system suitable only to a certain subset of e-Science applications. For instance Taverna has features which focus on bio-informatics applications that need to search
through large worldwide distributed data bases. The feature set of a system such as Pegasus is very different, it is focussed on efficient grid scheduling of computationally intensive jobs. This means there is no single system that can be used for all e-Science applications. Furthermore a means of letting multiple systems interact is needed for experiments that need the unique features of more than one system.

9.4.4 Sharing of resources

From the research in this thesis it is difficult to make any general conclusions on the current state of sharing resources. However we can look at the software resources used in the case studies. In the analysis of the suitability of current SWMS for implementing an ideal workflow for data assimilation, where data assimilation is a shared resource, it became clear that none of the current systems support all required features. In fact there is a clear trade off between systems that support all provenance features on the one hand and systems that support all expressiveness requirement on the other. As expressiveness and especially the ability to support loops is the most critical factor for actually being able to implement a data assimilation workflow, support for provenance and thus reproducibility will be limited for current data assimilation workflows.

Sharing data assimilation tools as a shared resource is not easy and it is not likely it will be easier for other resources of similar complexity. Even though none of the existing data assimilation toolkits[4, 67, 40, 53, 60, 128, 133], as presented in chapter 6 offer the support needed for being used as an e-Science resource, we can still note some interesting facts about the way they are published. The two data assimilation toolkits in the case studies are very different in their approach to sharing.

The SOS toolkit[128] is supplied as is, with all source code provided but no documentation other than associated scientific papers and comments in the code. It is published on the personal web page of the author. The Captain toolbox[133] on the other hand is closed source for the essential algorithms and requires a license. In return the user gets an extensive manual, tutorials as well as support from the developers. Between these two approaches lies the entire spectrum of data assimilation toolkits which were compared earlier.

One of the primary concerns when sharing resources is trust. In the case of these toolkits the question is to what extend these tools offer the functionality their publisher advertises. In the case of a toolkit that is presented as open source but without further support, functionality can theoretically be checked and corrected by the end user. For a closed source licensed toolkit this is not the case. If a shortcoming is suspected or detected the support offered can correct this. However it is hard to judge the effectiveness of either mechanism based on these two isolated cases.
Ideally there should be a way to objectively find out the size of the user community and how they rate the tools on key aspects to determine trust. In the ideal workflow for data assimilation from chapter 8 it can be seen that there is a large burden on the domain expert for creating a shared resources that can be re-used effectively. This burden consists on the one hand of (formally) checking whether the components in a shared resource and associated workflow patterns are correct, and on the other of making his expert knowledge explicit. This large burden is not directly justified by the benefits the domain expert receives. Some form of funding for this work needs to be found for instance if something should be published in a form similar to the Captain toolbox. The National Centre for Text mining\[34\] in the United Kingdom is a good example how the sharing of text mining tools can be encouraged through a different way of funding research.

9.4.5 When to use an e-Science approach

The important consideration is to decide at what point an experiment is performed in an e-science environment. There are four reasons to make this decision. First, does a need for reuse exist at the time the experiment is designed and performed? This can be the case when the experiment is performed by multiple parties that need to cooperate and thus reuse each others resources. Second, are there shared resources available in the e-science environment which can be used in the experiment. Third, is there a case for reuse of parts of the experiment in the future? This can be the case when there are incentives or obligations to support a much larger user community. A fourth reason is the need for sharing workflows and/or software resources in the scientific review process and dissemination.

9.5 Future Work

9.5.1 Standardization

Workflow environments for e-science need standards. Currently every scientific workflow management system has its own standard for describing a workflow, usually XML based. Each of these systems have their own unique features, but there is so much common ground that interoperability and sharing between systems would be greatly enhanced by at least a standard for the basics of workflow descriptions. This could come through the adaptation of an industrial standard such as BPEL\[35\] an attempt to reach standards for business process workflows. There are some efforts to support BPEL in SWMS\[61\], but it is by no means a standard by which workflows can be exchanged between different systems. Visual programming conventions in workflow environments need to be established and implemented. Especially in the area of control flow there are now many ad-hoc solutions while more
commonly used visual programming languages such as visual C++ visual basic etc. have shown that logical constructs exist that are well thought through.

9.5.2 Connectivity

While common standards for workflow description and interaction would be ideal it is not likely, that all needed standards will appear at once or for that matter soon. A different solution is to accept the fact that there will not be one single standard and that different workflow systems will keep evolving to support specific types of experiment. Efforts such as the workflow bus [135] first introduced in chapter 4 can provide a means to let different workflow systems communicate without the need for multiple standards. Furthermore connections made by the workflow bus are based upon Reo [19] which in turn uses constraint automata as a formal basis. The workflow bus is thus very expressive and can for instance be used to connect several instances of less expressive workflow systems. In this way these combined workflow systems can be made to perform experiments which they cannot define in their own workflow languages. For instance, DAG based workflow systems can be made to perform loops, and systems with little grid support can be made to do job farming and parameter sweeps.

9.5.3 Data assimilation in SWMS

The research in this thesis does not include a data assimilation workflow implemented in an SWMS. Before implementing this in future work there are a few points to consider when doing this. First of all in cases where data assimilation is used for prediction, as in weather prediction, the eventual outcome is a production system that is constantly producing predictions within real time constraints. These are not scientific experiments anymore and not very well suited to implementation in a workflow management system. However in choosing an estimator or during model development SWMS’s can have a role to play. Currently parameter sweeps are used to find optimal starting parameters for models, and job farming can be employed to try out many different model and estimator combinations. At the moment these are done as batch jobs without any possible interaction once the scientist has started them. The expansion of an SWMS with computational steering capabilities can be used to let the scientist interact in this process. Based on his expert knowledge a scientist can often quite quickly judge which jobs are not going to produce results and need to be stopped, just by looking at intermediate results. Similarly a scientist can change parameters or even a model or estimator in a running experiment based on his experience. Thus adding computational steering can be of real benefit for data assimilation experiments, and provide an additional reason to use SWMS. Computational steering can
also be of benefit to other techniques that use job farming and/or parameter sweep.

9.5.4 Formalisms

The tools associated with workflow formalisms need to become mature reliable and user friendly. In addition an analysis is needed of what the common workflow patterns are in all of e-Science and what formal representations are needed to express them all. Similar work has been done for business process workflows[23]. Still it would be interesting to create a generalized formal description of common e-Science workflows such as parameter sweeps and job farming, and at the same time use the same formalism to describe the capabilities of different SWMS. This can help scientists in choosing which SWMS is suitable for their experiment based on the general e-Science workflow pattern they intend to use. Having these generalized workflow patterns can also significantly increase the ease with which formal workflow analyses can be done.
Appendix A

List of abbreviations

**AIC**  Akaike Information Criterion

**API**  Application Programmers Interface

**ARIMA**  Auto Regressive Integrated Moving Average

**arspec**  auto regressive spectrum

**BAMBAS**  Bird Avoidance Model Bird Avoidance System

**BASIC**  Beginners All-purpose Symbolic Instruction Code

**BP**  Business Process

**BPEL**  Business Process Execution Language

**BPEL4WS**  Business Process Execution Language for Web Services

**CERN**  Conseil Europen pour la Recherche Nuclaire

**DAG**  Directed A-cyclical Graph

**DAGMan**  Directed Acyclic Graph Manager

**DAIHM**  Data Assimilation In Hydrological Modeling

**DARE**  Data Assimilation Recursive Estimation

**DART**  The Data Assimilation Research Testbed

**DE**  Discreet Event

**DHR**  Dynamic Harmonic Regression

**dhropt**  dynamic harmonic regression optimization

**EGEE**  Enabling Grids for E-sciencE
ENIAC  Electronic Numerical Integrator And Computer

e-Science  enhanced Science

GEODISE  Grid Enabled Optimization and Design Search for Engineering project

GIS  Geographical Information System

Griphyn  Grid Physics Network

GUI  Graphical User Interface

HP  Hewlett-Packard

IBM  International Business Machines Corporation

ICENI  Imperial College e-Science Networked Infrastructure (named after keltic tribe\(^1\))

I/O Automaton  Input Output Automaton

IOM  Inverse Ocean Modeling system

ITS  Intelligent Transport Systems

Jmatlink  Java matlab link

KNMI  Koninklijk Nederlands Meterologisch Instituut

MoML  Modeling Markup Language

MPMD  Multiple Process Multiple Data

MPSD  Multiple Process Single Data

MTM  Multi-Tape Turing Machine

NVR  Noise Variance Ration

OGSA  Open Grid Services Architecture

Pegasus  Planning for Execution in Grids

PN  Process Network

SCUFL  Simple Conceptual Unified Flow Language

SDF  synchronous Data Flow

SOA  Service Oriented Architecture

\(^1\)http://www.lesc.ic.ac.uk/iceni/iceni.html
SOAP  Simple Object Access Protocol
SOVON  SOVON Vogelonderzoek Nederland
SPA  Scientific Process Automation
SPMD  Single Process Multiple Data
SPSD  Single Process Single Data
SUN  Sun Microsystems (derived from Stanford University Network)
SWMS  Scientific Workflow Management System
TM  Turing Machine
TU Delft  Technische Universiteit Delft
TVP  Time Variable Parameters
UML  Universal Markup Language
VCR  Video Cassette Recorder
VL-e  Virtual Laboratory for e-Science
WRF-Var  Weather Research and Forecasting (model) Variational data assimilation system
WSDL  Webservice Description Language
XOR  eXclusive OR
Appendix B

Turing Completeness I/O Automata

To show the Turing completeness of I/O automata we will show that any computation which can be performed by a Turing-Machine can be performed by an I/O automata. It is important to note that I/O automata are not Turing equivalent in the general case as they have no restrictions on computability for the individual steps of the I/O automaton.

**Definition B.0.1** A non-deterministic Turing Machine (TM) is defined in the usual way by a 7-tuple:

\[ T_i = (Q_{\text{turing}}, \Sigma, \Gamma, \delta, q_{0_{\text{turing}}}, B, F) \]

Here \( i \) is the index of the machine, \( Q_{\text{turing}} \) is a set of finite states, \( \Sigma \) a set of finite input symbols, \( \Gamma \) the complete set of tape symbols \((\Sigma \subseteq \Gamma)\), \( \delta \) the transition function, \( q_0 \) the start state, \( B \) the blank symbol and \( F \subseteq Q \) the set of accepting states. Furthermore we refer to the set of all possible tapes with all possible head positions as \( \text{Tapes} \).

**Definition B.0.2** An input/output automaton is denoted by a tuple \( A = (S, Q, Q_0, T, \sim) \) where

- \( S \) is an action signature consisting of input- internal- and output-actions
- \( Q \) is a set of states
- \( Q_0 \subseteq Q \) is a nonempty set of start states
- \( T \subseteq Q \times S \times Q \) is a set of transitions, with the property that for every state \( q' \in Q \) and all input action \( \pi \) there is a transition \((q', \pi, q)\) in \( T \)
- \( \sim \) is an equivalence relation, partitioning the set \((A^{\text{out}} \cup A^{\text{int}})\) into at most a countable number of equivalence classes
We will use $Q(A)$, $Q_0(A)$, $T(A)$ and $\sim (A)$ as shorthand for the action signature set of states, start states, transitions and equivalence relation of automaton $A$. An equivalence class of internal actions $(A^{\text{out}} \cup A^{\text{int}})$ consists of actions under the control of one primitive component within the system being modeled. This means that in composition of automata the functionality of each specific automaton can still be identified in the automaton representing the composition. The equivalence relation is used in the upcoming definition of farexecution.

An execution of an I/O Automaton is a finite or infinite sequence of alternating states and actions which starts in a start state.

A state $q$ is reachable if it is the final state of a finite execution. The notion of execution can be reduced to the more interesting one of farexecution by demanding that in a final state no more actions are enabled. A farexecution of automaton $A$ is defined as an execution $\alpha$ of $A$ such that the following conditions hold for each equivalence class $C$ of $\sim (A)$:

- if $\alpha$ is finite, no action of $C$ is enabled in the final state of $\alpha$
- if $\alpha$ is infinite, either $\alpha$ contains infinitely many events from $C$, or $\alpha$ contains infinitely many occurrences of states in which no action of $C$ is enabled.

A farexecution ensures all internal and output actions of an automaton can potentially be performed at some point during an (infinite)execution. The use of the equivalence relation makes sure all primitive system components get treated fairly regardless of the abstraction level of the automaton. A finite execution of automaton $A$ is considered a fair execution when its final state $A$ cannot perform anymore of its locally controlled actions.

The behavior $\text{beh}(\alpha)$ of execution $\alpha$ is the sequence of external actions in $\alpha$. This means only actions which are contained in input and output actions of automata involved in the execution or schedule. A fairbehavior is the sequence of external actions contained in a farexecution.

Problems to be solved by I/O Automata are formally represented by a behavior, an arbitrary set of (finite or infinite) sequences of external actions. An I/O Automaton $A$ is said to solve a problem $P$ when $\text{fairbehaviors}(A) \subseteq P$. When this is achieved solely through finite fair behaviors $A$ implements $P$. An I/O Automaton has to have a non-empty set of fair behaviors, thus preventing an I/O Automaton with no fair behaviors from solving all problems. Not only can a Problem $P$ be implemented or solved by one automaton, this can also be done by a set of automata. Even more general, the set of automata $A$ can implement or solve set of automata $B$ if the fair behaviors of $A$ are a subset of $B$.

Theorem B.0.3 I/O Automata are Turing complete
Proof: To show an I/O Automaton is Turing complete we will define an I/O automaton \( A_t = (S, Q, Q_0, T, \sim) \) which emulates a Turing machine.

- the tape of the Turing machine is indexed as follows, the cell being read in \( q_0 \) is given index zero moving the head to the left decreases the index by one, moving it right increases by one.
- a state of \( A_t \) is an instantaneous description of Turing machine \( T_i \), the name of the current state the, index of the tape and the symbol at that index, all index symbol pairs read so far during execution.
- The set of all states \( Q \) corresponds to the set of all possible instantaneous descriptions of \( T_i \)
- the set of starting states \( Q_0 \) correspond to the set of all possible instantaneous descriptions of \( T_i \) which contain a state that is in \( q_0 \).
- an action consists of writing a symbol and moving the tape head.
- The action Signature \( S = (S^{in}, S^{int}, S^{out}) \) where \( S^{in} \) one action \( a_0 \) for each state \( q_0 \in Q_0 \), where the action corresponds to what is defined for \( q_0 \). For each instantaneous description for which the starting state \( q_0 \) of the Turing machine defines a symbol to be written and a tape movement, a separate action exists in \( S^{in} \).
- \( S^{out} \) contains all actions \( a_{final} \) which correspond to the writing of a symbol and tape movement of accepting states \( F \) of the Turing machine.
- \( S^{int} \) contains all actions \( a_{internal} \) which result in a state \( q_{result} \in (Q \setminus Q_{final}) \) and are enabled by a state \( q_{enabled} \in (Q \setminus Q_0) \).
- The transition relation \( T \subseteq Q \times S \times Q \) is such that for each possible transition that is defined by \( \delta \) of \( T_i \) there is a transition in \( T \).
- the properties of the \( \sim \) equivalence relation remain unchanged.

Due to the fact that each state is an instantaneous description of a Turing machine and all transitions correspond to the \( \delta \) of the Turing machine all fair executions of \( A_t \) correspond to an of the Turing machine. All executions of \( A_t \) end in a reachable state which contains an accepting state from \( F \). Thus proving an I/O automata can always be constructed which is equal to any Turing Machine.
Bibliography


Acknowledgements

When I started my PhD I had no idea what I was letting myself in for. I may have received some warnings, but chose to totally ignore them. However thanks to the help of my colleagues, supervisors, professors, friends and family I did finally manage to bring it all to a happy end. Let me try and start by thanking my colleagues in chronological order. It all started with Andy Pimentel whom I would like to thank for suggesting to me and Bob Hertzberger that I could do a PhD after I mentioned I somewhat missed the artificial intelligence component as a researcher in the field of embedded systems, my job at the time. Bob is the second person I would like to thank for actually giving me the opportunity to do my PhD and giving me a lot of freedom in determining the subject. He provided me with Arnoud Visser (who also supervised my Masters thesis) and Geleyn Meijer as my supervisors.

Arnoud I would like to thank for support early on in my research and especially for helping me in attracting and supervising Peter Schouten (Thanks for your work as well Peter) as a masters student for my research.

Geleyn I would like to thank for sticking with me all the way, he has been the most constant factor in my PhD research, but also for taking me outside the world of academic research by involving me in multiple commercial academic cooperations.

Many thanks also go out to my former colleagues from Andy’s embedded system research. Joe Coffland and Cagkan Erbas and especially Simon Polstra whom I kept sharing a room with long after I started my PhD.

Next I would like to thank Willem Bouten for giving me the opportunity to get started with models and data from the BAMBAS project when the arrival of traffic data took a little longer than expected. Special thanks goes out to Emiel van loon who provided me with lots of help and advice, but I also should not forget Scott, Jelmer, Judy, Luit and Hans.

Then there’s the VLAM team which I joined very early on in my research. My thanks go out to Dmitry, Vladimir, Piter, Victor, Hakan, Wibi, Zhiming and Adam. I would like to thank Adam Belloum and Zhiming Zhao especially for all the input they gave for my research. For his insights into control theory and for supporting my traffic research I would like to thank Hans Hadderingh at Logica.
The downside of thanking people in chronological order is that some of the most important people get mentioned relatively late. Fact is though that I had already met a lot of people before Pieter Adriaans first helped me with my research. First just as an adviser but later as my promotor when Bob Hertzberger went into retirement. Without his help though I would probably not have finished this work, the theoretical insight he thought me really helped in getting some much needed science into my work. One hour brainstorming on a whiteboard with Pieter can keep you busy for weeks putting the ideas of one such session to paper. Many thanks to you Pieter for your input and for being my promotor.

With Pieter becoming my Promotor I also moved to the HCS lab and into a different research environment with many new colleagues. Suddenly I was surrounded by a very large number of fellow PhD students, the social activities with Sophia, Wico, Vera, Niels, Victor, Jochem, Henriette, Simon and Katja provided a welcome diversion. Frits de Vries was a great roommate for the last few years. Maarten van Someren I would like to thank for his help and feedback on my work, he is a great source of knowledge.

Scott Marshall deserves a special thank you for pointing me towards Constraint Automata and Farhad Arbab, who also deserves a thank you for explaining his work concerning them. Sometimes the best solution is so close it is easily overlooked.

Finally of all my colleagues I would like to thank Marco Roos. It was great cooperating with you in the very last part of my work at the UvA.

Now that I have thanked all my colleagues, and I sincerely apologize to anyone I have forgotten. I would like to thank all of my friends and family for their support. A special thanks goes out to Roger Latton, who is a good friend of the family but might just as well be family, for proof reading my thesis.

Justus, thank you for all the effort you have put into the cover design.

Last but certainly not least I thank both my parents, who provided critical and important input in the finishing stages of my thesis.
Summary

This thesis is about practical and theoretical issues in scientific workflow design. In order to explain the importance of scientific workflow design one needs to know the concept of e-Science. This stands for enhanced Science (not electronic as one might expect). It aims to enhance the scientific experiments of scientists from many domains by offering them access to Grid resources. These can enhance experiments through massive computing and massive data handling capabilities. The other enhancement e-Science offers is the sharing of (software) resources that run on the Grid between different scientists. Both for scientists that are collaborating in an experiment but also for scientists from different domains that can re-use each others work. The interface through which scientists from these different domains access Grid resources and share them with their peers is a Scientific Workflow Management System. E-Science is performed by designing and running workflows in such a system. The main research question asked in this thesis is "What is a proper design methodology for both workflow components and workflow topologies that supports the sharing of software resources?". To answer this question we look both at the theoretical and practical sides of scientific workflow design. On the practical side we look into sharing one particular type of software resource, Data assimilation. The idea is that lessons from this specific case can tell us more about the problem of sharing resources in general. First however we turn our attention to the theoretical aspects involved.

The scientific method in general is characterized by the empirical cycle. In this cycle a scientist starts from an existing theory or a research question, from this he derives a hypothesis for testing in an experiment. After performing the experiment a scientist analyzes the result, this can lead to verification or falsification of parts of the existing theory. The theory is adjusted accordingly and the whole cycle can start again performed either by the same scientist or a peer performing a peer review of the results of the experiment. In e-Science designing a workflow is analogous to formulating a hypothesis, just as performing a workflow equates to performing an experiment in general science. There also are differences between science in general and e-Science, in particular concerning the sharing of (software) resources. The main method of sharing in science in general is through peer
reviewed scientific papers. Within e-Science the work of scientists can also be shared directly as resources ready to be used in a workflow. Widespread adoption of this type of sharing can lead to a paradigm shift for science as a whole, by allowing a far greater search space to be covered in scientific experiments. The start of this can be seen in bio-informatics where the sharing of research through webservices is leading to many new discoveries. Currently the sharing of resources has not yet risen to significant levels for science as a whole. While in bio-informatics mainly data is shared, sharing of methods and models as needed for sharing resources related to data assimilation, is more complicated. Currently e-Science can not convincingly be called a paradigm shift however if sharing of more complex resources becomes widespread this could change in the future.

In order to do workflow design one needs to know what the workflow design space looks like. In this thesis an overview is given of the different approaches to workflow design, concrete, abstract to concrete and automatic workflow composition. Using a formal approach we show that automatic workflow composition in the general case is not possible. Furthermore limits of abstraction in workflow representation are proven. Maximal generality and simplicity are mutually exclusive with consistency. In other words the most general and abstract representation of a workflow can not be consistent. From this formal look at the workflow design space a number of criteria are derived that need to be met by a formalism for reasoning about specific workflow instances. As was mentioned earlier workflow design is analogous to formulating a hypothesis. A scientist needs to formulate a hypothesis in such a way that experiments based on this hypothesis will provide answers to the research question. The execution of workflow needs to confirm or falsify existing theories. Formal reasoning about a workflow can help a scientist determine whether his workflow is consistent and actually provides relevant answers especially in more complicated workflows.

To this end five formalisms are compared for use in abstract to concrete workflow design, the method that is most suitable to scientific workflows, especially when sharing resources. First Petri nets, a formalism used extensively in concrete workflow design, especially in the business process workflow field. Second π Calculus a more recent alternative to Petri nets used for the same applications. Third there are Turing machines which provided the underpinnings of the proofs in the formal analysis of the workflow design space. The final two I/O Automata and Constraint Automata have not been applied to workflows yet but have many of the properties that are needed. From this comparison Constraint Automata emerge as the most suitable formalism for the abstract to concrete approach to workflow design.

From the theoretical side of workflows and workflow design we move to the practical side. An analysis is given of the types of support workflow systems can give to a scientist that is designing a workflow. This support can be during composition of a workflow, the development of resources, the
execution of a workflow as well as the dissemination of a workflow. Current workflow systems are compared on the basis of the features they provide in supporting a scientist when creating a workflow.

For the practical aspects of sharing one particular software resource, we look at the technique of data assimilation. This is done through the use of two case studies. Before going into the details of the case studies the background to data assimilation is explained. How it originated in weather prediction as a way to minimize errors in both the predictive model as well as in the observations used for prediction. The elements which make up data assimilation are explained, the predictive model as well as the estimator which can adjust parameters of this model as well as correct errors in observation. An overview of current data assimilation toolkits is given. Within this overview features, relevant for employing them as a shared resource in an e-Science environment, are compared. Two of these toolkits are used in the case studies.

The first case study is about bird migration. In this study data assimilation is used to minimize observation error. The aim is to predict bird densities of migratory birds over the Netherlands using a model for bird migration from Scandinavia to Africa. The second case study looks at the prediction of car traffic for one particular stretch of Dutch highway. The aim is to predict traffic several hours ahead. In this study data assimilation is used to minimize error in the employed predictive model. From these case studies we learn what types of knowledge about implementing data assimilation should be made explicit in order to use data assimilation as a shared resource.

This knowledge is combined with the earlier lessons learned about the workflow designspace and the use of formalisms in an abstract to concrete design methodology to produce a methodology for implementing data assimilation as a shared resource. It is a workflow detailing how to implement each step in the data assimilation process in a scientific workflow. Data preparation model development choice of estimator, use of parallelism are all detailed.

The thesis is concluded by looking at what lessons can be drawn from the specific case of data assimilation as well as from the exploration of the workflow designspace to answer the original research question: what is a proper methodology to share software resources in the general case. Not only are proper methodologies needed but also changes in the way science is funded. Such a change can ensure there is more incentive for scientists to share resources. Funding is not the only measure that can provide incentive, a change in the scientific review process and the role of publishers can also help. Not only scientific papers need to be reviewed, but also shared resources as well as workflows used that were employed in scientific discoveries. Publishers of scientific journals can offer more than just scientific papers, they can also publish peer reviewed resources and workflows.
Samenvatting

Dit proefschrift gaat over praktische en theoretische onderwerpen in wetenschappelijk workflow ontwerp. Voor het uitleggen van het belang van wetenschappelijk workflow ontwerp is het belangrijk bekend te zijn met het concept e-Science. Dit is een afkorting voor "enhanced-Science" (en niet electronic science zoals men wellicht verwacht). Het heeft als doel wetenschappelijke experimenten van wetenschappers uit verschillende vakgebieden te verbeteren door hen toegang te verschaffen tot Grid resources. Deze kunnen experimenten verbeteren door de mogelijkheden die zij bieden op het gebied van "massive computing" en "massive data". De andere verbetering die e-Science biedt is het delen met andere wetenschappers van (software) resources die op het Grid draaien. Die andere wetenschappers kunnen werken aan hetzelfde project, maar het kunnen ook wetenschappers uit verschillende vakgebieden zijn die elkaars werk hergebruiken. De interface waarmee deze verschillende wetenschappers toegang krijgen tot Grid resources en ze delen met anderen is een Scientific Workflow Management System. E-Science wordt bedreven door het ontwerpen en uitvoeren van workflows in een dergelijk systeem. De belangrijkste onderzoeksvraag die in dit proefschrift wordt gesteld is: "Wat is een juiste ontwerp methodologie voor zowel workflow componenten als topologieën die ondersteuning biedt voor het delen van software resources?". Voor het beantwoorden van deze vraag kijken we zowel naar theoretische als praktische aspecten van wetenschappelijk workflow design. Aan de praktische zeide kijken we naar het delen van een specifiek type software resource, te weten Data Assimilatie. De achterliggende gedachte is dat lessen van deze specifieke resource ons meer vertellen over het delen van resources in het algemeen. Voordat we in detail op de praktische kant ingaan richten we eerst onze aandacht op de theoretische aspecten van wetenschappelijk workflow design.

De wetenschappelijke methode wordt gekarakteriseerd door de empirische cyclus. In deze cyclus start een wetenschapper met bestaande theorie of een onderzoeks vraag, hieruit leidt hij een hypothese af die getest kan worden in een experiment. Nadat het experiment voltooid is analyseert de wetenschapper het resultaat, hetgeen kan leiden tot bevestiging of falsificatie van de hypothese en achterliggende theorie of vraagstelling. De theorie wordt naar aanleiding van de analyse bijgesteld en de hele cyclus kan opnieuw wor-
den uitgevoerd. Dit kan worden gedaan door dezelfde wetenschapper of een collega wetenschapper ten behoeve van een ”peer review” van de resultaten. Voor e-Science is het ontwerpen van een workflow analoog aan het formuleren van een hypothese, net als het uitvoeren van een workflow gelijk staat aan het uitvoeren van een experiment binnen de wetenschap in het algemeen. Tussen wetenschap en e-Science zijn er ook verschillen, in het bijzonder als het aankomt op het delen van resources. De gebruikelijke methode voor delen bij wetenschap is het publiceren van ”peer reviewed” wetenschappelijke artikelen. Binnen e-Science bestaat ook de mogelijkheid om resultaten van werk te delen in de vorm van resources die direct in een workflow gebruikt kunnen worden. Het uitgebreid toepassen van deze manier van delen kan leiden tot een paradigma verschuiving voor de wetenschap in algemene zin, doordat een veel grotere zoekruimte kan worden bestreken in wetenschappelijke experimenten. Een begin van deze verschuiving is waarneembaar in de bio-informatica waar het delen van onderzoeksresultaten door middel van webservises al heeft geleid tot vele nieuwe ontdekkingen. Op dit moment echter worden resources, binnen de wetenschap als geheel, niet op grote schaal gedeeld. Terwijl binnen de bio-informatica vooral data wordt gedeeld is het delen van methodes en modellen zoals bijvoorbeeld vereist voor data assimilatie veel complexer. Op dit moment kan e-Science niet met overtuiging een paradigma verschuiving genoemd worden, echter als het delen van meer resources mogelijk wordt gemaakt kan dit veranderen.

Voor het ontwerpen van workflows is het belangrijk om te weten hoe de ontwerpruimte voor workflows er uitziet. In dit proefschrift wordt een overzicht gegeven van de verschillende manieren waarop workflows worden ontworpen: concreet, abstract naar concreet en automatische workflow compositie. Door gebruik te maken van een formele benadering wordt aangetoond dat automatische workflow compositie in het algemene geval niet mogelijk is. Bovendien laat de formele benadering de limieten van abstractie in workflow representatie zien. Maximale algemene toepasbaarheid, simplificatie en het consistent zijn van een workflow representatie sluiten elkaar wederzijds uit. Oftewel de meest simpele en algemeen toepasbare representatie van een workflow kan niet consistent zijn. Met behulp van deze formele kijk op de workflow ontwerpruimte worden een aantal criteria afgeleid. Deze zijn bedoeld voor het vinden van een formalisme dat gebruikt kan worden om over workflows, die in de praktijk gebruikt worden, te redeneren. Zoals al eerder genoemd het ontwerpen van een workflow is analoog aan het formuleren van een hypothese. Een wetenschapper moet zijn hypothese kunnen formuleren zodanig dat hij antwoord kan geven op de onderzoeksvraag. Formeel redeneren over een workflow kan een wetenschapper helpen te bepalen of deze daadwerkelijk de antwoorden kan geven die hij zoekt, in het bijzonder als er complexe workflow constructies meespelen.

Voor dit doel worden vijf formalismen vergeleken voor gebruik in een abstract naar concreet workflow ontwerp. De ontwerp methode die het meest
geschikt is voor het delen van resources in e-Science. Als eerste zijn er Petri nets, een formalisme dat veelvuldig gebruikt is concreet workflow ontwerp, in het bijzonder voor business process workflows. Het tweede formalisme is $\pi$ Calculus, een recenter alternatief voor Petri nets en gebruikt voor dezelfde toepassingen. Als derde formalisme hebben we Turing machines welke het fundament vormden voor de bewijzen in de formele analyse van de workflow ontwerpruimte. De laatste twee formalismen zijn I/O Automata en Constraint Automata, deze zijn tot op heden nog niet uitgebreid toegepast op workflows maar hebben veel van de vereiste eigenschappen. Uit deze vergelijking komen Constraint Automata naar voren als het meest geschikte formalisme voor de abstract naar concreet aanpak van workflow ontwerp.

Van de theoretische kant van workflows en workflow ontwerp bewegen we ons vervolgens naar de praktische zijde. De verschillende manieren waarop een workflow systeem een wetenschapper kan ondersteunen bij het ontwikkelen en uitvoeren van een workflow worden geanalyseerd. Deze ondersteuning kan plaats vinden tijdens het componeren van een workflow, het ontwikkelen van resources het uitvoeren van de workflow alsmede de disseminatie van de workflow. Huidige workflow systemen worden vergeleken op basis van de functionaliteit die ze bieden voor het ondersteunen van wetenschappers.

De praktische kanten van het delen van een resource komen aan bod door te kijken hoe dit mogelijk is voor een specifieke resource, data assimilatie. Dit wordt gedaan door middel van twee "case studies". Alvorens deze in detail te beschrijven wordt eerst ingegaan op de achtergrond van data assimilatie. Hoe het zijn oorsprong vond in weersvoorspelling als een manier om de fout in zowel het voorspellende model als in de gebruikte observaties te minimaliseren. De elementen waaruit data assimilatie bestaat worden uitgelegd, het voorspellende model en de schatter welke de parameters van het model kan bijstellen alsmede de fout in observaties kan corrigeren. Beschikbare data assimilatie toolkits komen aan bod en worden vergeleken op eigenschappen die ze geschikt maken om als resource op het Grid geïmplementeerd te worden. Twee van deze toolkits worden gebruikt in de case studies.

De eerste case study gaat over de migratie van trekvogels. Hierbij wordt data assimilatie gebruikt om de fout in observaties te minimaliseren. Het doel is het voorspellen van vogeldichtheden boven Nederland, gebruikmakend van een model voor vogeltrek van Scandinavië naar Afrika. De tweede case study heeft als doel het voorspellen van files op een specifieke stuk snelweg in Nederland. Hierbij wordt de situatie meerdere uren vooruit voorspeld. De rol van data assimilatie is hier het minimaliseren van de fout in het gebruikte voorspellend model. Uit deze case studies worden lessen getrokken over welke soorten kennis, met betrekking tot het implementeren van data assimilatie, expliciet moeten worden gemaakt. Waarbij het doel is data assimilatie te gebruiken als gedeelde resource.

Deze kennis wordt gecombineerd met de eerdere lessen die geleerd zijn
over de kenmerken van de workflow ontwerpruimte alsmede het gebruik van formalismen in een abstract naar concreet workflow ontwerp methodologie. Op basis van dit alles wordt een methodologie gepresenteerd voor het implementeren van data assimilatie als een gedeelde resource. Het is een workflow voor het implementeren van iedere stap in het data assimilatie proces binnen een wetenschappelijke workflow. Data preparatie, model ontwikkeling, de keuze van een schatter het gebruik van parallellisme komen allen aan bod.

Dit proefschrift eindigt door te kijken in hoeverre de originele research vraag ("Wat is een juiste ontwerp methodologie voor zowel workflow componenten als topologieën die ondersteuning bied voor het delen van software resources?") beantwoord kan worden uit voorafgaande hoofdstukken. In het bijzonder hetgeen geleerd is met de formele verkenning van de workflow ontwerpruimte en de specifieke case van data assimilatie. Een van de belangrijke conclusies is dat er niet alleen correcte ontwerp methodologieën nodig zijn maar ook motivatie voor wetenschappers om deze te ontwikkelen en toe te passen. Dit kan bereikt worden door de financiering van de wetenschap aan te passen zodat het delen van resources aantrekkelijker wordt. Het veranderen van de manier waarop wetenschappelijk werk beoordeeld wordt kan ook bijdragen aan het delen van resources. Zo zouden niet alleen papers onder peer review moeten vallen, maar ook gedeelde resources en de workflows die gebruikt zijn bij wetenschappelijke ontdekkingen. Uitgevers zouden meer dan alleen papers kunnen aanbieden, ze kunnen ook gedeelde resources publiceren die het peer review proces goed doorstaan hebben.
SIKS Dissertatierens

1998

1998-1 Johan van den Akker (CWI)
    DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
    Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
    A Contribution to the Linguistic Analysis of Business Conversations
    within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
    Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
    Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
    Physiology of Quality Change Modelling;
    Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
    Classification using decision trees and neural nets

1999-3 Don Beal (UM)
    The Nature of Minimax Search

1999-4 Jacques Penders (UM)
    The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
    Empowering Communities: A Method for the Legitimate User-Driven
    Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
    Re-design of compositional systems

1999-7 David Spelt (UT)
    Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
    Informed Gambling: Conception and Analysis of a Multi-Agent
    Mechanism for Discrete Reallocation.

2000
2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennis-technologie; een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup (EUR)
Sensitivity Analysis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

====

2001
====

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization
2001-8 Pascal van Eck (VU)  
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan 't Hoen (RUL)  
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)  
Modeling and Simulating Work Practice  
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)  
Knowledge Management: The Role of Mental Models in Business Systems Design

====

2002
====

2002-01 Nico Lassing (VU)  
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)  
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)  
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)  
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)  
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)  
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)  
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijns (VU)  
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel(KUB)  
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)  
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)  
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (Uva)  
Processing XML in Database Systems
2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models
2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

====

2004
====

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play
2004-14 Paul Harrenstein (UU)  
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)  
Multi-Relational Data Mining

2004-16 Federico Divina (VU)  
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)  
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)  
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)  
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)  
Learning from Design: facilitating multidisciplinary design teams

====

2005

====

2005-01 Floor Verdenius (UVA)  
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM))  
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)  
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)  
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)  
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)  
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)  
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)  
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)  
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)  
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)  
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
2005-12 Csaba Boer (EUR)  
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)  
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)  
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)  
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)  
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)  
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)  
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)  
Situated Representation

2005-20 Cristina Coteanu (UL)  
Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)  
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

=====

2006
=====

2006-01 Samuil Angelov (TUE)  
Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)  
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)  
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)  
Building Web Service Ontologies

2006-05 Cees Pierik (UU)  
Validation Techniques for Object-Oriented Proof Outlines

2006-06 Ziv Baida (VU)  
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling

2006-07 Marko Smiljanic (UT)  
XML schema matching – balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)  
Forward, Back and Home Again - Analyzing User Behavior on the Web
2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor’s Opinion

2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems

2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types

2006-12 Bert Bongers (VU)
Interactivation - Towards an e-ology of people, our technological environment, and the arts

2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents

2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change

2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain

2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks

2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device

2006-18 Valentin Zhizhkun (UVA)
Graph transformation for Natural Language Processing

2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach

2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining

2006-21 Bas van Gils (RUN)
Aptness on the Web

2006-22 Paul de Vrieze (RUN)
Fundaments of Adaptive Personalisation

2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web

2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources

2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC

2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval

2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval
2007

2007-01 Kees Leune (UvT)  
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)  
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mika (VU)  
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)  
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)  
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)  
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic’ (UT)  
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)  
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)  
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)  
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)  
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)  
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)  
Meetings in Smart Environments: Implications of Progressing Technology

2007-14 Niek Bergboer (UM)  
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)  
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)  
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)  
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)  
On the development an management of adaptive business collaborations

2007-19 David Levy (UM)  
Intimate relationships with artificial partners
2007-20 Slinger Jansen (UU)  
Customer Configuration Updating in a Software Supply Network

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

2007-22 Zlatko Zlatev (UT)  
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)  
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramrez Camps (CWI)  
Structural Features in XML Retrieval

2007-25 Joost Schalken (VU)  
Empirical Investigations in Software Process Improvement

====

2008
====

2008-01 Katalin Boer-Sorbn (EUR)  
Agent-Based Simulation of Financial Markets: A modular,continuous-time approach

2008-02 Alexei Sharpanskykh (VU)  
On Computer-Aided Methods for Modeling and Analysis of Organizations

2008-03 Vera Hollink (UVA)  
Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT)  
Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT)  
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN)  
On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU)  
Supporting the tutor in the design and support of adaptive e-learning

2008-08 Janneke Bolt (UU)  
Bayesian Networks: Aspects of Approximate Inference

2008-09 Christof van Nimwegen (UU)  
The paradox of the guided user: assistance can be counter-effective

2008-10 Wauter Bosma (UT)  
Discourse oriented summarization

2008-11 Vera Kartseva (VU)  
Designing Controls for Network Organizations: A Value-Based Approach

2008-12 Jozsef Farkas (RUN)  
A Semiotically Oriented Cognitive Model of Knowledge Representation
2008-13 Caterina Carraciolo (UVA)  
Topic Driven Access to Scientific Handbooks

2008-14 Arthur van Bunningen (UT)  
Context-Aware Querying; Better Answers with Less Effort

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.

2008-16 Henriette van Vugt (VU)  
Embodied agents from a user’s perspective

2008-17 Martin Op ’t Land (TUD)  
Applying Architecture and Ontology to the Splitting and Allying of Enterprises

2008-18 Guido de Croon (UM)  
Adaptive Active Vision

2008-19 Henning Rode (UT)  
From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search

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

2008-21 Krisztian Balog (UVA)  
People Search in the Enterprise

2008-22 Henk Koning (UU)  
Communication of IT-Architecture

2008-23 Stefan Visscher (UU)  
Bayesian network models for the management of ventilator-associated pneumonia

2008-24 Zharko Aleksovski (VU)  
Using background knowledge in ontology matching

2008-25 Geert Jonker (UU)  
Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency

2008-26 Marijn Huijbrugs (UT)  
Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled

2008-27 Hubert Vogten (OU)  
Design and Implementation Strategies for IMS Learning Design

2008-28 Ildiko Flesch (RUN)  
On the Use of Independence Relations in Bayesian Networks

2008-29 Denni Reidsma (UT)  
Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans

2008-30 Wouter van Atteveldt (VU)  
Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content

2008-31 Loes Braun (UM)  
Pro-Active Medical Information Retrieval

2008-32 Trung H. Bui (UT)  
Toward Affective Dialogue Management using Partially Observable Markov Decision Processes