Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diplomarbeit Nr. 3618

# Development of a Pattern Library and a Decision Support System for Building Applications in the Domain of Scientific Workflows for e-Science

Stephan Passow

**Course of Study:** Informatik

**Examiner:** Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
**Supervisor:** Dipl.-Inf. Michael Hahn
**Commenced:** Februar 4, 2014
**Completed:** August 1, 2014

**CR-Classification:** D.2.11, H.4.1, H.4.2, I.5.0, I.6.3, I.6.7

## Abstract

Karastoyanova et al. created eScienceSWaT (eScience SoftWare Engineering Technique), that targets at providing a user-friendly and systematic approach for creating applications for scientific experiments in the domain of e-Science. Even though eScienceSWaT is used, still many choices about the scientific experiment model, IT experiment model and infrastructure have to be made. Therefore, a collection of best practices for building scientific experiments is required. Additionally, these best practice need to be connected and organized. Finally, a Decision Support System (DSS) that is based on the best practices and enables decisions about the various choices for e-Science solutions, needs to be developed. Hence, various e-Science applications are examined in this thesis. Best practices are recognised by abstracting from the identified problem-solution pairs in the e-Science applications. Knowledge and best practices from natural science, computer science and software engineering are stored in patterns. Furthermore, relationship types among patterns are worked out. Afterwards, relationships among the patterns are defined and the patterns are organized in a pattern library. In addition, the concept for a DSS that provisions the patterns and its prototypical implementation are presented.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

The field of e-Science has the goal to provide comprehensive IT support for scientists throughout the life cycle of scientific experiments [56]. Several supporting systems, initiatives and literature are proof for the rapid increase of e-Science [47, 108]. Furthermore, e-Science aims at shortening the time to new discoveries by providing software tools and infrastructures to natural scientists from different fields [56]. Typical for e-Science is large scale science that is carried out through distributed global collaborations enabled by the Internet [67]. Further characteristics are very large data sets, large scale computing resources and high performance visualization [67]. In addition, projects in e-Science often involve the interdisciplinary cooperation of many different research groups [67]. All these characteristics lead to a high complexity. Thus, this thesis focuses on the identification and collection of best practices in building applications in the domain of e-Science. These best practices are stored in patterns, connected to each other and organized in a pattern catalogue. Furthermore, this thesis provides the concept of a DSS and its prototypical implementation for provisioning the patterns. The DSS enables decisions about the various choices that can be made for an e-Science solution. The following sections provide the problem statement and a motivating scenario, which the thesis is based on. Furthermore, the scope of work is presented.

## 1.1. Problem Statement

Natural scientists that are not experts in the fields of computer science and software engineering create systems for e-Science, which are not reusable and lack interoperability [57]. Additionally, anti patterns are implemented and only a few available technologies and concepts are considered when developing code for their experiments [57]. The e-Science community realizes that IT support is required for all phases of a scientific experiment, e.g. the phases *capturing and validating data*, *curation*, *processing and analysis* and *permanent archiving*. Karastoyanova et al. detect, that a lot of research is done for the e-Science *processing and analysis* phase [56]. However, the analysed systems have limitations like outdated IT techniques and technologies, domain-specific concepts covering only some scientific domains or a minimal degree of experiment automation. For this reason they identified the necessity for focused fundamental research in the data *processing and analysis* phase. As stated by Karastoyanova et al., there is currently "no unified software engineering methodology and corresponding body of knowledge to support the development and execution of eExperiments in a generic manner that meets the requirements of natural scientists towards the experiments themselves and the IT infrastructure" [56]. Hence, they developed the eScienceSWaT, a methodology that helps to develop and execute scientific experiments. However, they identified challenges related to the

realization of future systems supporting eScienceSWaT [56]. First of all, many choices can be made, when creating an application for the *processing and analysis* phase. Questions about how to build the scientific experiment, which IT techniques, technologies and infrastructures to choose arise. Therefore, best practices need to be collected that help to answer these questions and that can be used as recommendations. Decisions for the scientific experiment influence the decisions for the IT technique. Then again decisions about the IT technique influence the decisions about technologies. Furthermore, the decisions about technologies influence the decisions for infrastructures. Therefore, the best practices need to be connected to each other and decision chains need to be captured as well. Certainly, it is not clear which abstraction levels are useful for storing the best practices [56]. In addition, people from different domains build an e-Science application together. If they use their domain-specific language, when talking to people from other domains, this can bring additional challenges.

Thus, this thesis focuses on the development of an e-Science pattern catalogue. We use the terms pattern catalogue and pattern library interchangeably. The library comprises knowledge and best practices from natural science, software engineering and computer science. One advantage of using patterns is, that they have a common structure and people from all domains can understand patterns. Furthermore, this thesis introduces a DSS, which provides these patterns to the people that want to develop an application in the domain of e-Science. In doing so, the DSS enables decisions about how to build a scientific experiment model, IT experiment model and which infrastructures to choose [56].

## 1.2. Motivating Scenario

Several natural scientists want to conduct a scientific experiment in the domain of e-Science. They require an IT solution that is able to analyse the massive data stream from an instrument that collects data all the time without breaks. Past projects show, that the self-made IT solutions are not reliable and also have difficulties to process large amounts of data. Therefore, they decide to use a software engineering technique, which enables the separation of natural science tasks from IT tasks like eScienceSWaT [56]. If there is a clear separation of tasks, each stakeholder only works in his domain of expertise. For example, an IT professional can perform the IT specific tasks and does not have to work on the scientific experiment model. This methodology comprises the phases *experiment modelling*, *creation of software artefacts*, *deployment on IT execution environment* and *execution* [56]. Decisions in one phase influence the decisions in a later phase. For instance, decisions made in the experiment modelling phase influence the decisions for software artefacts. In the first phase of this software engineering technique, the natural scientists have to model the scientific experiment. However, they have many choices for creating the scientific experiment model. They would benefit from a collection of best practices for modelling the scientific experiment. In the second phase, software artefacts are created for the scientific experiment. Hence, an IT solution has to be developed. The natural scientists have hard times to explain to the IT professionals what kind of IT solution they require. The natural scientists and IT professionals speak in their domain languages and this is why they talk at cross purposes. Since the IT professionals have many choices for the IT solution, it takes them weeks to narrow down the amount of

possible solutions. Finally, they come up with four different concepts for the IT solution. In a workshop, the natural scientists should choose the solution which fits their needs best However, none of the IT solutions provides a web based graphical user interface, although this is the preferred interaction style of the natural scientists. In addition, two of the four concepts are not applicable at all, because the IT professionals misunderstood the natural scientists. With a few months delay, natural scientists and IT professionals agree on the concept for the IT solution. Meanwhile, most of the project budget is spent already and phases three and four of the software engineering technique are still missing. Hence, the project leader decides to terminate the project, because the costs are too high and the project is delayed too much.

We can identify the need for a collection of best practices and communication issues in the above scenario . Therefore, we see the clear necessity to develop an e-Science pattern catalogue. The patterns comprise knowledge and best practices from natural science, computer science and software engineering. These patterns build the language that both the natural scientists and IT professionals can use. This helps them to communicate more efficiently and get the requirements right. Furthermore, both the natural scientist and the IT professionals have many possibilities to create the scientific experiment model and the IT solution. As soon as an e-Science pattern catalogue is available, that comprises many patterns and even more relationships among them, all stakeholders require decision support for choosing the most suitable patterns. A DSS can handle a big amount of patterns and the relationships among them. Therefore, we see the need for developing a DSS that provisions these patterns. By doing so, the DSS supports both the natural scientists and IT professionals in decision making. Furthermore, the decisions of natural scientists can directly be used for making proposals for IT solutions to the IT professionals.

## 1.3. Scope of Work

A simplified approach for developing the e-Science pattern library and the DSS is presented in Figure 1. We can identify the following steps:

1. First of all we analyse various e-Science solutions. We focus on systems for data processing and analysis in the domain of scientific workflows for e-Science.
2. We collect information about various systems and transform this information into problem-solution pairs.
3. By abstracting from these problem-solution pairs, we generate the e-Science patterns. These patterns contain the abstract problem and the abstract solution among others.
4. Then we organize the patterns. Additionally, we define the relationships among them.
5. Finally, we create a DSS that uses the patterns of the e-Science pattern library.

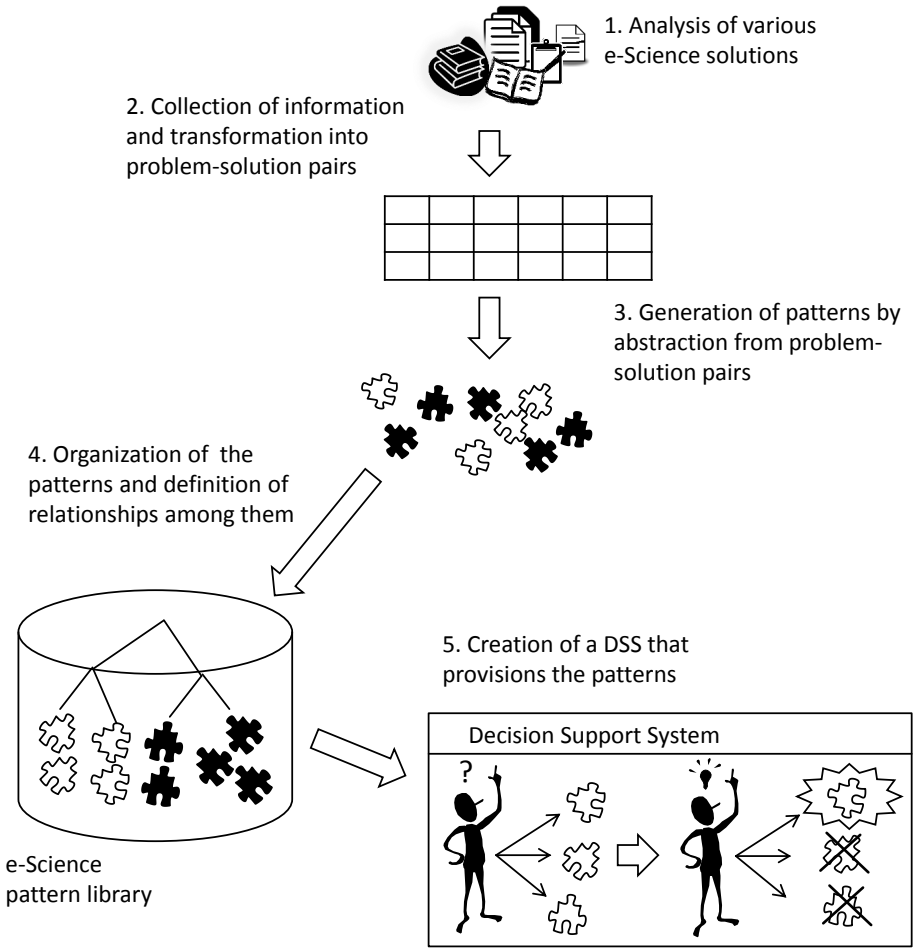


**Figure 1.:** This is an overview of the steps performed for the creation of the e-Science pattern library and the DSS.

## 1.4. Outline

In order to accomplish the given goals and to draft future tasks on the topic, the work is structured as followed:

- **Chapter 2 – Background:** This chapter gives an overview about patterns and e-Science in general. Furthermore, eScienceSWaT and scientific workflows are introduced.
- **Chapter 3 – Related Work:** Existing approaches and state of the art in the fields of patterns and DSSs are presented. Additionally, we analyse various relationships among patterns and position our research to each related work.
- **Chapter 4 – e-Science Pattern Catalogue:** This chapter contains the core deliverable of this thesis – the e-Science pattern catalogue. First of all, we present our approach for developing the e-Science pattern catalogue. In addition, the resulting pattern format and the pattern relationships are presented. Finally, the patterns for the *scientific experiment model*, *IT experiment model* and *infrastructure* are introduced.

- **Chapter 5 – Decision Support System**: At the beginning, our concept for the DSS is inducted. Furthermore, we present a general architecture for the DSS, which is prototypically implemented in the context of this thesis.
- **Chapter 6 – Conclusion and Future Work:** The last chapter summarizes the outcomes of this work. Additionally, suggestions for extensions of the pattern catalogue and the DSS are provided.

## 1.5. Definitions and Conventions

This section contains a list of abbreviations used in this diploma thesis.

| | |
|---|---|
| **CBR** | Case-Based Reasoning |
| **DoE** | Design of Experiments |
| **DPR** | Design Pattern Recommender |
| **DPRS** | Design Pattern Recommendation System |
| **DSS** | Decision Support System |
| **eScienceSWaT** | eScience SoftWare Engineering Technique |
| **FEM** | Finite Element Method |
| **GMF** | Graphical Modeling Framework |
| **GUI** | Graphical User Interface |
| **HPC** | High Performance Computing |
| **HTC** | High Throughput Computing |
| **IaaS** | Infrastructure-as-a-Service |
| **IAAS** | Institute of Architecture of Application Systems |
| **IC** | Implicit Culture |
| **MVC** | Model-View-Controller |
| **MPI** | Message Passing Interface |
| **NIST** | National Institute of Standards and Technology |
| **OFAT** | one-factor-at-a-time |
| **GQM** | Goal-Question-Metric |
| **PaaS** | Platform-as-a-Service |
| **SaaS** | Software-as-a-Service |
| **SICS** | Systems for Implicit Culture Support |

| | |
|---|---|
| **SQL** | Structured Query Language |
| **UML** | Unified Modeling Language |
| **VSM** | Vector Space Model |
| **WfMS** | Workflow Management System |
| **XML** | eXtensible Markup Language |

# 2. Background

This chapter gives relevant background information that is required to understand the concepts presented in this thesis. Section 2.1 comprises different approaches to patterns. In Section 2.2, we introduce the term e-Science with different definitions and present thesis relevant e-Science dimensions. After that we present e-Science research results from the University of Stuttgart. This also contains the eScienceSWaT methodology. Finally, in Section 2.4, we expound scientific workflows.

## 2.1. Patterns

We introduce the concepts of patterns here, because the identification of patterns in e-Science solutions is one major deliverable of our thesis. To be more exact, we identify patterns in computer science, software engineering and in natural science that play a role in e-Science solutions. The requirement to base the DSS on patterns originates from the task description of this diploma thesis. However, there are reasons for using patterns and we want to work out the benefits of storing best practices in patterns within this section. In addition, we want to give different examples of pattern related research to underline the wide applicability of patterns in general. We also cover the core elements of a pattern in this section. Different pattern formats are introduced and we show possible ways to connect the patterns to each other. Usually, experienced experts of a domain store the best practices in patterns. We have experiences with patterns and software architecture, but do not have many years of experience in the domain of e-Science. This is why we examine different pattern approaches. We expect that this improves our ability to detect patterns for e-Science. Later in this thesis, we define a pattern format for the e-Science patterns (see Section 4.2.1).

### 2.1.1. Christoph Alexander

Christoph Alexander can be seen as the *father of design patterns* [61, 74]. During his work as an architect, he recognized that he had to solve certain problems in a different context again and again. Out of this observation he defined a pattern as follows [3]:

> *„Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."*

A pattern here is the entity. Together these entities build the pattern language. The structure of Alexander's pattern is shown in Figure 2. In addition, the connection to other patterns is visualized. Alexander's pattern language is structured hierarchical. Patterns on a lower level are needed to fill out/complete the patterns on the higher level. Alexander was an architect and his patterns comprise best practices for architecture. He starts with patterns for *regions*. The patterns for *regions* are filled/completed by patterns for *towns*. Patterns for *towns* are filled with patterns for *neighbourhoods*. *Neighbourhood* patterns are filled with patterns for *clusters of buildings* and *buildings*. Going further down, there are patterns for *rooms* and *alcoves*. The patterns on the lowest level comprise *details of construction*.

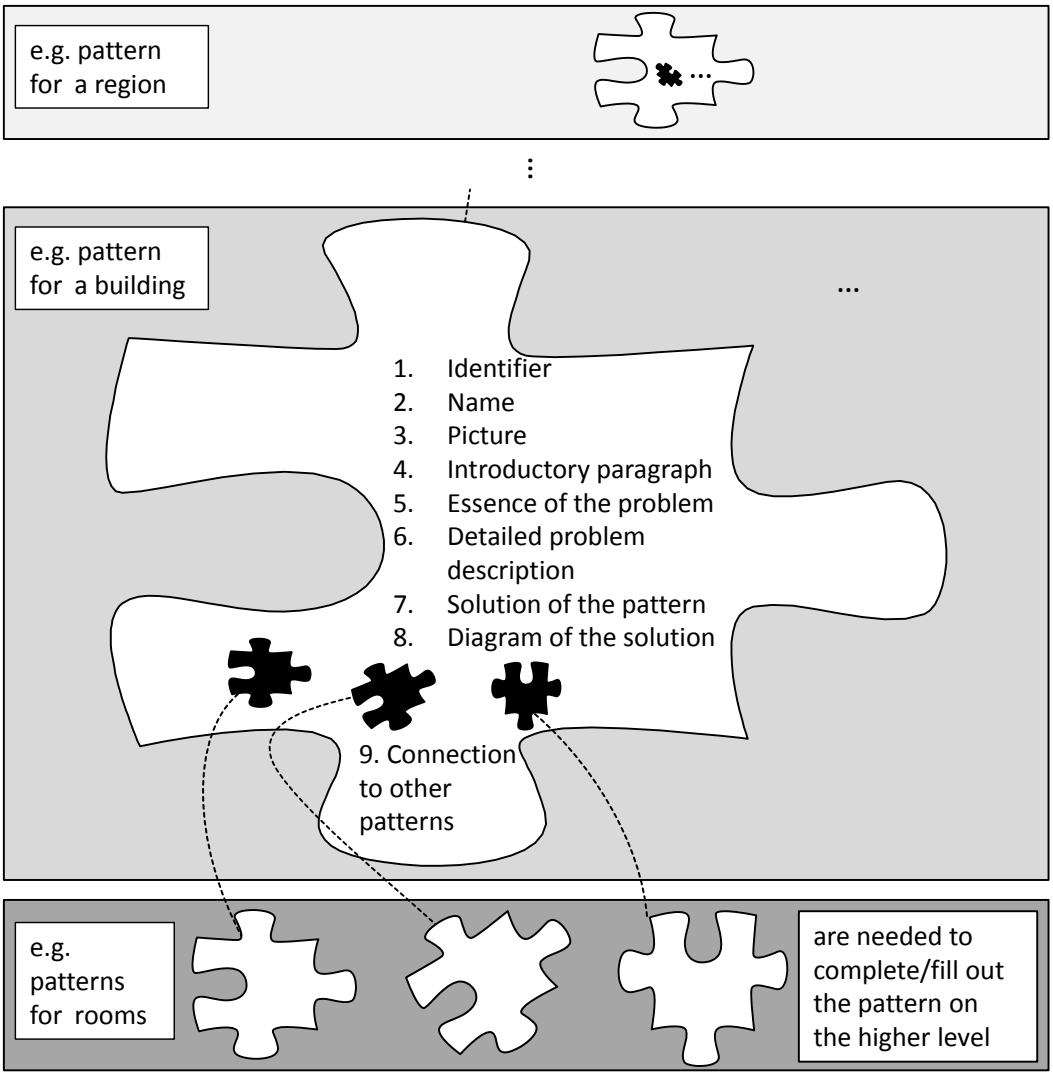


**Figure 2.:** Graphical representation of the pattern language introduced by Alexander in [3].

Alexander used the same structure for each and every pattern he defined. This was done for convenience and clarity reasons. A pattern consists of text and images. Contentwise a pattern can be divided into 9 segments. The segments are also shown in Figure 2. We explain the different segments in the following:

1. **Identifier:** A unique number for each pattern is used.
2. **Name:** The name of the pattern.
3. **Picture:** The picture shows an archetypal example of the pattern.
4. **Introductory paragraph:** The context for the pattern is set. It is explained how this pattern helps to complete larger patterns.
5. **Essence of the problem:** In a short summary of two or three sentences, the problem is described.
6. **Detailed problem description:** Here, the empirical background and the evidence for its validity are described. Additionally, different ways of using the pattern in certain contexts are described.
7. **Solution of the pattern:** This is the heart of the pattern. It comprises the solution to the problem in the given context. The solution is given in the form of an instruction. This helps the user of the pattern to execute the required steps.
8. **Diagram of the solution:** Here the solution is shown in a diagram. Labels indicate the main components of the solution.
9. **Connection to other patterns:** Due to the hierarchical structure there can be patterns on a lower level that help to complete the current pattern. The connection to these patterns build the last segment. The connection is achieved by a text that refers to the pattern and its unique number.

In another book, *The Timeless Way Of Building* [2], Alexander describes the theory behind the pattern language and gives instructions for using the pattern language. Within this book he describes the core elements of a pattern:

> *„Each pattern is a three part rule, which express a relation between a certain context, a problem, and a solution."*

Furthermore, a pattern can be seen from two angles. Firstly, it can be seen as a rule or process. If the rule is followed, the thing which is described within the pattern is created. Secondly, a pattern is a thing that exists in the world. A thing that is alive. This means that a pattern can be created from real things. Since patterns are abstract, the problem and solution in the real world can be transformed to a pattern with the help of abstraction. In addition, Alexander points out that "patterns can exist at all scales" [2].

### 2.1.2. Gang of Four

The book *Design Patterns: Elements of Reusable Object-Oriented Software* [37], written by Gamma et al., is one of the foundations for design patterns in object-oriented development. The authors describe it as "book of design patterns that describes simple and elegant solutions to specific problems" [37]. In their opinion, the solutions captured in design patterns have evolved and developed over time. The solution is special, because the architect would not

have chosen this solution at first. The goal of the book is to give design insights and inspiration to the reader. Gamma et al. work out the characteristics of a good pattern. The authors see a graphical notation for the pattern as an important component but not sufficient enough since it only captures the end product of the design process. Besides, it is also essential that the reader understands the decisions that led to the pattern. Possible alternatives and trade-offs should also be captured within the pattern. In order to make the solution more tangible to the reader, a concrete example is required [37].

In order to make the patterns reusable, comparable and easy to learn, a consistent pattern format needs to be defined. Gamma et al. use the following format [37]:

- **Pattern name:** The pattern name will become part of the design vocabulary. Therefore, a name that describes the pattern should be used.
- **Classification:** There needs to be a schema that organises the different patterns. Hence, a classification that shows the position within the pattern scheme is required.
- **Intent:** Here the design problem is shortly described.
- **Also known as:** If there exist synonyms for the pattern, their names can be placed here.
- **Motivation:** Description of a scenario that makes the design problem more tangible to the reader.
- **Applicability:** Gives the reader insights, in which cases he can use the pattern.
- **Structure:** The authors use a notation based on the Object Modeling Technique (OMT) and interaction diagrams [15, 53].
- **Participants:** The patterns are about object-oriented design. Here, the participating objects and classes with their responsibilities are described.
- **Collaborations:** The participants have certain responsibilities. In order to achieve their goals, they need to collaborate with other objects. The collaboration is described here.
- **Consequences:** Here, possible trade-offs that are coming with the usage of the patterns can be discussed.
- **Implementation:** This section keeps best practices for implementing the pattern (e.g. language specific issues).
- **Sample code:** They used *C++* or *Smalltalk* code for illustration purposes.
- **Known uses:** This section comprises real world examples, in which the pattern can be identified. This makes the pattern more tangible.
- **Related patterns:** Patterns that are closely related should be named here. Also patterns that should be used in combination with the current pattern.

Gamma et al. identify, that patterns can exist on different abstraction levels. In addition, patterns vary in granularity. Therefore, a classification that is organizing the patterns is required. This helps the user to find the patterns easier that can solve his problem. The authors use two criteria for the classification of the patterns. The criterion *purpose* reflects what the pattern is doing. Purpose can take three values: *creational, structural* or *behavioural*. *Scope* is the second criterion. It specifies the application area of the pattern. Possible values are *object* or *class*.

### 2.1.3. Patterns at IAAS

Patterns are used in many research projects of the IAAS[1] at the University of Stuttgart. In order to underline the applicability of patterns, we present a selection of research projects that involve patterns in the following. In addition, we present a book, which is a collaboration result between the IAAS and the Daimler AG[2]. This shows, that both the academic researchers and the real world architects do benefit from patterns. Furthermore, we extract thesis relevant background information out of the research from IAAS.

The following projects from IAAS are related to patterns. We choose the categorization of the research projects. Another grouping is also possible.

**Cloud related patterns:**

- Cloud Data Patterns for Confidentiality [113]
- Capturing Cloud Computing Knowledge and Experience in Patterns [30]
- Non-Functional Data Layer Patterns for Cloud Applications [111]
- Using Patterns to Move the Application Data Layer to the Cloud [112]
- Pattern-based Runtime Management of Composite Cloud Applications [17]

**Processes related patterns:**

- Process Viewing Patterns [102]
- Applicability of Process Viewing Patterns in Business Process Management [100]
- Pattern-driven Green Adaptation of Process-based Applications and their Runtime Infrastructure [78]
- Green Business Process Patterns - Part II [79]

[1] http://www.iaas.uni-stuttgart.de/
[2] http://www.daimler.com/

**Costumes in films related patterns:**

- A Pattern Language for Costumes in Films [101]

Leymann et al. give an example of the relationship between a concrete costume and costume patterns in [101]. We used this example as a basis for Figure 3 that shows the abstract approach for the interrelation between patterns and concrete problem-solution pairs. Concrete solutions for problems are the basis for patterns. By means of abstraction, commonalities in reoccurring problem-solution pairs can be filtered out. This leads to an abstract problem-solution pair: the *pattern*. The pattern again refers via the attribute *Known use* to the problem-solutions pairs that build the basis. It is also shown that a problem can be solved by applying and concretizing a pattern which contains the abstract solution.

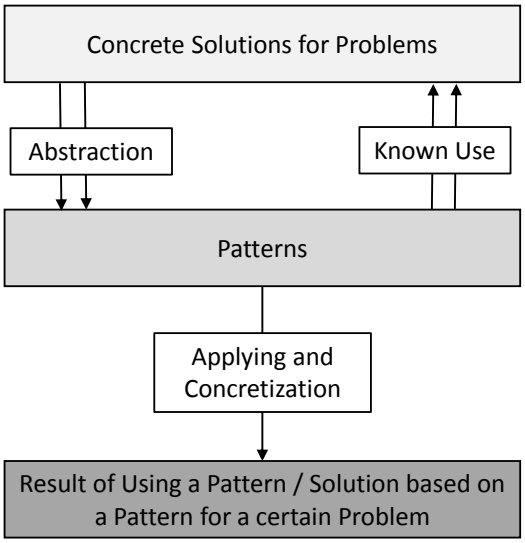


**Figure 3.:** Pattern abstraction and concretization based on [101].

The recent cooperation of the University of Stuttgart with the Daimler AG in the area of Cloud computing led to the book *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications* [32]. This proves, that storing best-practices in a pattern format is not only valuable for research projects, but also valuable for real world application architecture. The quotation of Dr. Michael Gorriz, the Chief Information Officer (CIO) of the Daimler AG, about the book underlines the benefits from using patterns:

> *„During our collaborative research with the University of Stuttgart, we identified a vendor-neutral and structured approach to describe properties of Cloud offerings and requirements on Cloud environments. The resulting Cloud Computing Patterns have profoundly impacted our corporate IT strategy regarding the adoption of Cloud computing. They help our architects, project managers and developers in the refinement of architectural guidelines and communicate requirements to our integration partners and software suppliers."* [3]

[3] `http://www.amazon.de/Cloud-Computing-Patterns-Fundamentals-Applications/dp/3709115671`

Previously, we have shown that the usage of patterns adds value and that this researchers are experts in their field. The overview of pattern related research underlines the experience with patterns. Now we want to extract some valuable information out of the research projects, that we can reuse for the thesis.

Leymann and Fehling use the following general definition of a pattern [31]:

> *„Patterns are structured text describing abstract problem-solution pairs."*

In a concrete domain like Cloud computing, they refine their general definition to the below [31]:

> *„The Cloud Computing Patterns describe **abstract solutions to recurring problems** in the domain of cloud computing to capture **timeless knowledge** that is **independent of concrete providers**, products, programming languages etc."*

Furthermore, Fehling et al. also present a format for patterns. Their approach is shown in Figure 4. A pattern map that visualizes pattern groups and connections among patterns is shown in Figure 5. This pattern map is used to provide an overview about patterns. The number in brackets represents the starting page of the pattern within the book and therefore helps the reader to access a specific pattern faster.

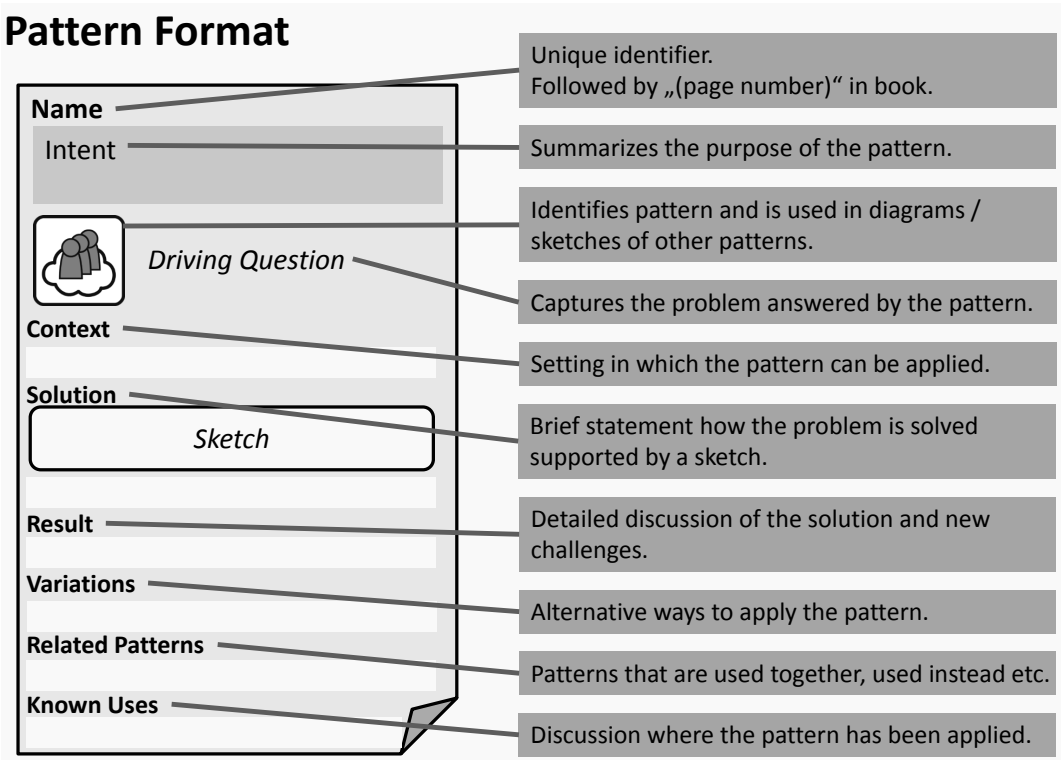


**Figure 4.:** This is a possible pattern format [31].

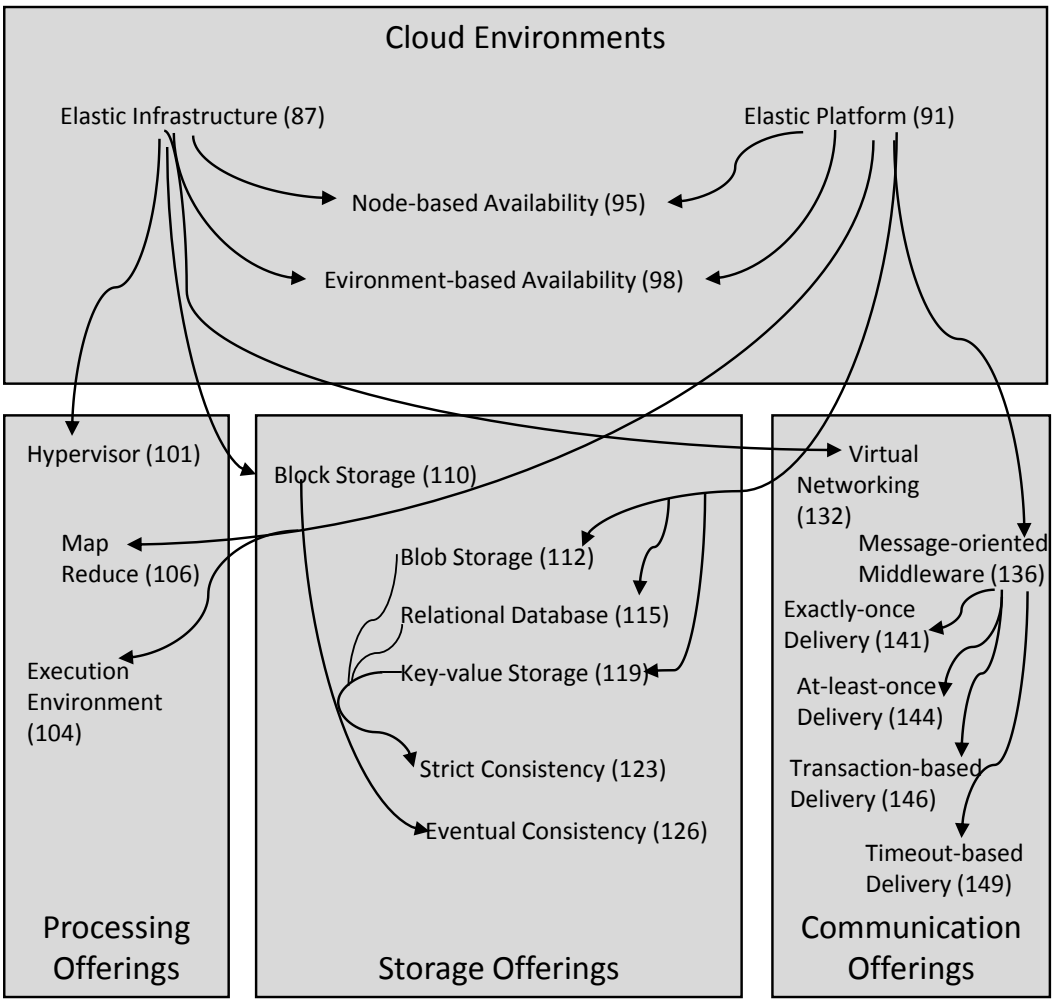


**Figure 5.:** This shows a Cloud computing pattern map based on [32].

## 2.2. e-Science

In common literature different notations like *e-Science* or *eScience* are used. We will use e-Science in the context of this thesis. The *UK e-Science Programme* began in 2001. One part of it was the *e-Science Core Programme* that should support the development of generic technologies to enable different resources to work together seamlessly across networks and to create computing Grids [90]. A good overview of the UK e-Science Core Programme can be found in [48]. We will not focus on the history of e-Science and predecessors within this thesis.
An overview of the different components that enable e-Science is shown in Figure 6. Laws and theories are the research results of theoretical scientists. The experimental scientists' research in laboratories leads to scientific data. Within simulations the computational scientists can make use of both the theories and laws and the scientific data. At the same time the computational scientist can give feedback to theoretical scientists and experimental scientists regarding their research. Scientists from different fields work in new kinds of collaboration in key areas of science. These scientists are called *e-Scientists*. The e-Scientists share hardware,

data, resources as well as approaches and knowledge. Their research is enabled by next generation infrastructures and advanced tools and devices [92].

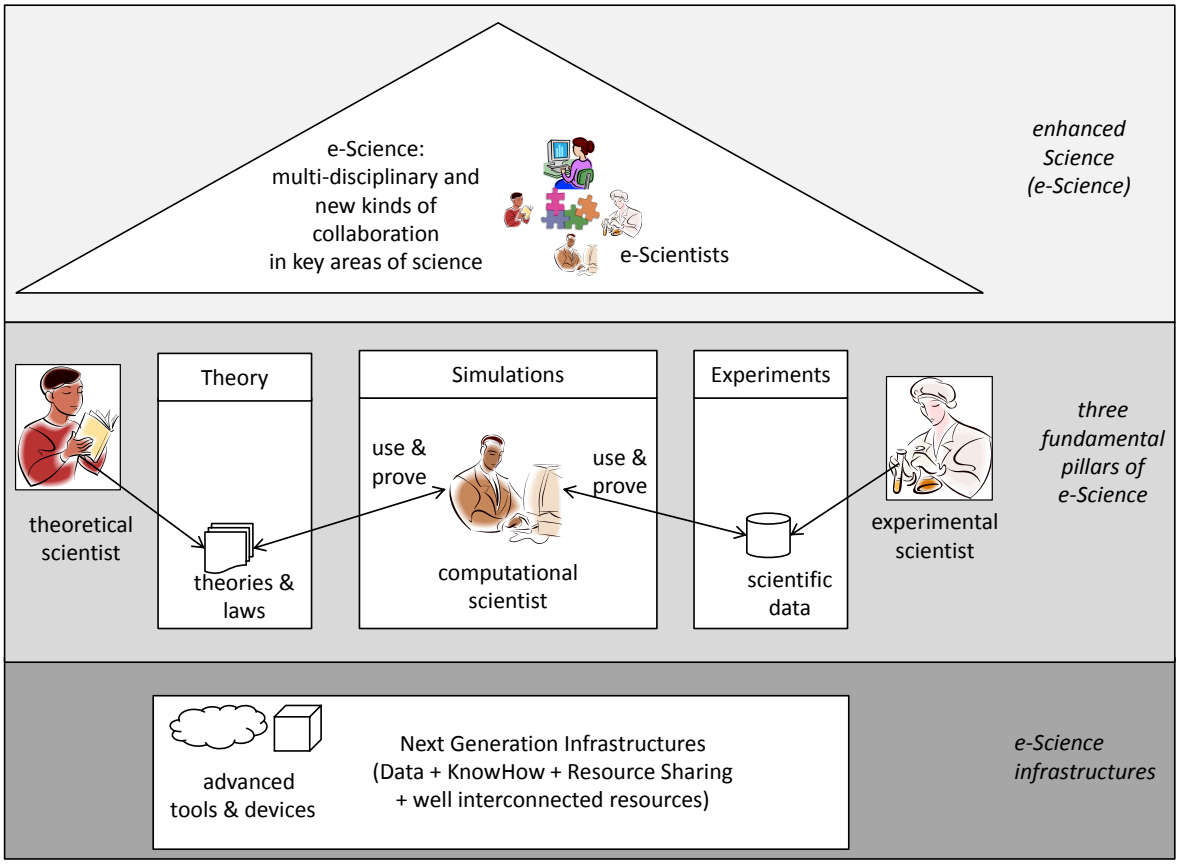


**Figure 6.:** An e-Science Overview based on [92].

Dr. John Taylor, Director of Research Councils in the Office of Science and Technology (UK), introduced the term *e-Science* with the two following statements [119]:

> *„e-Science is about global collaboration in key areas of science and the next generation of infrastructure that will enable it."*

> *„e-Science will change the dynamic of the way science is undertaken."*

A widely accepted definition of e-Science is "e-Science is the shared usage of information and resources for computing and data intensive experiments of geographically distributed researches" [45]. Karastoyanova et al. put the focus more on the scientists with their e-Science definition. They define that "e-Science is the field trying to provide IT support to scientists throughout the life cycle of scientific experiments" [57]. In addition, they describe the major focus of e-Science on shortening the time to new discoveries and revealing knowledge about natural phenomena by providing software systems for different scientific tasks and for many domains. Hay et al. consider e-Science as a new paradigm for science, the so called *fourth paradigm* or *data-intensive science*, which unifies theory, experiments and simulation for data exploration with the goal of scientific discovery [47, 57]. The goal of this thesis is to build a

DSS for scientists. Therefore, we find the definition of Karastoyanova et al. most suitable, because this definition contains the support of the scientists explicitly. Since we will analyse various e-Science solutions in the context of this thesis, a holistic understanding of e-Science solutions is required. Therefore, we want to look at e-Science solutions in more detail from various angles. Pursuant to [57], e-Science solutions differ in *application domains*, *modelling approaches*, *technologies used to implement experiments* and *underlying infrastructures*. In the following we will have a closer look on the different aspects of an e-Science solution.

### 2.2.1. Application Domains

E-Science solutions can be found in many different domains. In each domain there exists domain specific knowledge. Scientists within a domain use special terms and vocabulary to describe their research and the objects of interest. In addition, their research is characterized by specific tasks. Many e-Science solutions are targeted at a specific domain [56]. We just name a few domains to highlight the widespread use of e-Science solutions: astronomy, biology, biodiversity informatics, bioinformatics, medicine, chemistry, environmental studies, geology, healthcare, physics, materials science, manufacturing engineering [47, 54, 56]. To illustrate differences between domains we take the example of *biodiversity informatics* and *bioinformatics*, which was introduced by Jones in [54].
*Biodiversity informatics* differs from *bioinformatics* in the data being used and the typical tasks to be performed. Within biodiversity, research is more collaboratively and simultaneously. Scientists use data from wet lab experiments and other sources in complex analysis. One definition of biodiversity is: "variability among living organisms from all sources [...] and the ecological complexes of which they are part; this includes diversity within species, between species and of ecosystems" [23]. Therefore, the scientists need to have access to many different kinds of data like species catalogues, species information sources, geographical data or climate data. Since the data is coming from many sources and the data format was specified based on the requirements of the original user, the data formats are not standardized [54].
The National Institutes of Health (NIH) Biomedical Information Science and Technology Initiative (BISTI) Consortium defined *bioinformatics* in [50] as follows:

> *„Research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral or health data, including those to acquire, store, organize, archive, analyze, or visualize such data."*

Typical activities within bioinformatics include sequence similarity searching, functional motif searching or the search for non-coding Deoxyribonucleic Acid (DNA) [110]. Significant standardization efforts have led to standards for representing data within *bioinformatics* [54]. An example for data standards in bioinformatics is data from European Nucleotide Archive (ENA)[4], where a comprehensive record of the world's nucleotide sequencing information can be accessed. The data standards also incorporate the efforts to standardize meta-data. The Gene Ontology (GO)[5] project provides for instance an audited vocabulary of terms for

[4] http://www.ebi.ac.uk/ena/
[5] http://www.geneontology.org/

describing gene product characteristics. Moreover, tools to access and process this data are provided.
With this comparison of *bioinformatics* and *biodiversity* we showed that there exist differences between application domains. In this comparison it was the typical tasks that are performed by scientists, different data and the different levels of data standardization.

### 2.2.2. Modelling Approaches

Scientists start with the modelling of the experiment as a first step. In different domains, different models for creating experiments are used by scientists. Experimental design, another expression for experimental modelling, is described as follows [99]:

> *„Experimental design is the process of planning a study to meet specified objectives. Planning an experiment properly is very important in order to ensure that the right type of data and a sufficient sample size and power are available to answer the research questions of interest as clearly and efficiently as possible."*

According to [99], the design of an experiment starts with the definition of the problem and the questions to be addressed. In addition, scientists have to define the non-functional properties that are required within the modelling phase [99, 56]. Examples for non-functional properties are the processing time or the costs of the experiment. Every experiment requires the definition of input parameters. Additionally, the scientists define which parameters take which values and in which order the parameters are changed. In keeping with Giger, there exist several ways of modelling an experiment with respect to the input parameters [40]. *Trial and error* circumscribes changing of several parameters at the same time. In contrast to trial and error stands one-factor-at-a-time (OFAT). In OFAT one input parameter is modified and the output is analysed. The configuration of the input parameter, that leads to the best result, is stored. After that the next input parameter is modified. OFAT is easy to implement, but it is not a systematic approach and an optimum is only discovered by accident. Design of Experiments (DoE) is a method for planning and statistical analysis of experiments and tries to overcome the obstacles of *trial and error* and *OFAT*. Giger describes the goal of DoE to understand dependencies between input parameters and outputs with minimal effort. One way of modelling an experiment within DoE is *full factorial designs*. For each input parameter different levels are defined. In full factorial designs, all possible configurations of the different input parameters with their levels are tested. In *fractional factorial designs* only a subset of all levels of input parameters is used [40].

Karastoyanova et al. point out, that even within the same domain the modelling approach for an experiment can differ [56]. As an example they bring in a solid body simulation. A solid body simulation can be modelled using a simulation that is based on the Finite Element Method (FEM). Herein, simulation scales can be represented by a single mathematical model [89], or by separate simulations for each scale that communicate by exchanging relevant data [105].

### 2.2.3. Technologies

In [57], the authors picture that scientists typically choose the technology by themselves . Their decision is influenced by organizational policies, financial means, available technology, software and skills at the research organization or other factors [57]. In general, scientists can choose from a wide variety of technologies for the e-Science solutions. Possible approaches for e-Science are *simple scripts*, *application plug-ins* or *complex workflows* [94]. But even if the scientist decides to use a specific technology (e.g. workflow technology), he can choose from many different systems, which support the selected technology. A few well known systems that support the workflow technology are Taverna[6], Triana,[7] Pegasus[8] and Kepler[9].

### 2.2.4. Infrastructures

Finally, concrete infrastructure like storage, network resources or computing power has to be chosen. Based on our literature research, the most important infrastructures are Cloud computing platforms, clusters, Grids, High Performance Computing (HPC) infrastructures or High Throughput Computing (HTC) infrastructures [56, 92]. There exist overlaps between theses concepts. Our goal is not to give a detailed distinction between the different concepts, but more providing an overview of the main characteristics. We also provide literature references, so that the reader can dive deeper in the respective areas. For example a more detailed comparison of Cloud computing, Grids and clusters can be found in [21]. The following sections comprise the concepts of the formerly introduced infrastructures for e-Science.

#### Cloud Computing

The most common definition of Cloud computing was developed by National Institute of Standards and Technology (NIST) [73]. Pursuant to NIST, the main characteristics of Cloud computing are *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity* and *measured service*. Furthermore, three different service models within Cloud computing can be distinguished:

- Infrastructure-as-a-Service (IaaS) provides the user with computing power, network resources and data storage. Since we focus on infrastructures for e-Science, this is the most important service for us.
- Platform-as-a-Service (PaaS) provides the user with programming languages and libraries to build own applications.
- Software-as-a-Service (SaaS) provides applications that are running in the infrastructure of the provider to the user.

[6] http://www.taverna.org.uk/
[7] http://www.trianacode.org/
[8] http://pegasus.isi.edu/
[9] https://kepler-project.org/

Another way to segment Cloud computing is the distinction in four different deployment models: *private Cloud*, *community Cloud*, *public Cloud* and *hybrid Cloud*. A good overview of Cloud computing can be found in [4]. An example for an e-Science solution based on Cloud technologies is presented by Lezzi et al. in [66]. Well known Cloud infrastructure hosting providers are Amazon Web Services[10], Microsoft Azure[11] or Computer Sciences Corporation (CSC)[12]. Leong et al. compare different Cloud infrastructure providers in [65].

#### Grids

Grid computing is a special form of distributed computing. The *Grid problem* was defined by Foster et al. as "flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources" [36]. They see the dynamic collections of individuals, institutions and resources as *virtual organizations*. According to Leymann and Karastoyanova, a virtual organization "can comprise different groups of scientists working on a common goal but can as well be different companies that join forces or are in a supplier-requester relationship in order to reach a certain (mostly computing intensive) business goal" [58]. Another definition for a Grid was given by Foster et al. back in 1998. In [35], they define a Grid as the following: "A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." This is not the only definition for a Grid and therefore, Foster reviewed different definitions for Grids in [34]. As a result he presents three essential characteristics for Grids:

- Coordination of resources that are not subject to centralized control.
- Use of standard, open, general-purpose protocols and interfaces.
- Delivery of non-trivial qualities of service.

#### Clusters

Based on the work in [116] and [86], Buyya et al. define that "a cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource" [21]. This means, that the whole system with all nodes behaves like a single system and all resources are managed by a centralized resource manager. A cluster is more a *tightly coupled system* with a *single system image* and a *centralized job management* and *scheduling system* [55].

[10] `http://aws.amazon.com/de/`
[11] `http://azure.microsoft.com/de-de/`
[12] `http://www.csc.com/cloud`

#### High Performance Computing and High Throughput Computing

Riedel defines that "a HPC-driven e-Science Infrastructure is based on computing resources that enable the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance Central Processing Unit (CPU)/core interconnections" whereas "a HTC-driven e-Science infrastructure is based on commonly available computing resources such as commodity Personal Computers (PCs) and small clusters that enable the execution of farming jobs without providing a high performance interconnection between the CPU/cores" [92]. He sees the major difference between HPC and HTC in the quality of the connection between the resources. HPC resources like supercomputers or large clusters provide a better interconnection of CPUs/cores whereas HTC resources like PC pools do not [92]. Boisseau defines the goal of HTC to integrate multiple computing systems. This makes it possible to schedule lots of jobs. If a resource is available, it can perform the scheduled job. In HPC computers or processors are aggregated and the work is divided so parallel computing at large scale is possible [14]. Ahronowitz points out that HPC resources are tightly coupled. This influences the software that is written for HPC machines. In many cases this leads to code that is bad with regards to portability. HPC code incorporates often Message Passing Interface (MPI) and Graphics Processing Unit (GPU) libraries. In many cases these libraries have library specific components or even machine specific components. This specifics need to be considered when migrating code from one HPC infrastructure to another [1]. Examples for HTC are the European Grid Infrastructure (EGI)[13] or the Open Science Grid(OSG)[14]. The Distributed European Infrastructure for Supercomputing Applications (DEISA)[15], the Partnership for Advanced Computing in Europe (PRACE)[16] and the Extreme Science and Engineering Discovery Environment (XSEDE)[17] are examples for HPC projects.

### 2.2.5. Further Complexity and Challenges

In the previous sections we described different dimensions in e-Science solutions. Certainly, we think we need to point out further challenges and complexity that come across IT solutions for e-Science. First of all, there is heterogeneous data involved in e-Science solutions. This data might be distributed across several data sources (e.g. databases or file systems) in heterogeneous formats and different sizes. In addition, some systems work with live data from sensors which add additional complexity, e.g. how to make changes to a system which should process data all the time. Ultimately, the e-Science solution should support the natural scientist in discovering new knowledge. When designing IT solutions, the scientist with his competencies needs to be considered. In general, natural scientists are no experts in computer science or software engineering. Normally, they are also no experts when it comes to programming and therefore, the creation of scientific experiments within an e-Science solution should not require programming of the scientist. Rather than programming, the

[13] https://www.egi.eu/about/egi-inspire/
[14] http://www.opensciencegrid.org/
[15] http://www.deisa.eu/
[16] http://www.prace-ri.eu/?lang=en
[17] https://www.xsede.org/

scientist should be able to model the experiment in his domain language [57]. At the same time, the massive (distributed) data and the heterogeneity of distributed systems adds a lot of complexity. This is a major challenge in e-Science solutions: providing an interface to the scientist, that he can handle and which supports him in his way of doing research, but at the same time handling the complexity of e-Science in the background.

## 2.3. e-Science at IAAS

In the previous section we presented the dimensions *application domains*, *modelling approaches*, *technologies* and *infrastructures* of e-Science. This showed the variance in e-Science solutions and that many choices are possible in each dimension. After this overview we want to outline now the thesis relevant e-Science research of the IAAS. This helps to understand where the tasks for this thesis originate. Firstly, the *e-Science life cycle* is introduced. The *e-Science life cycle* gives us insights in scientists' operating principles. The second block is oriented towards an application model for eExperiments and the requirements for eScienceSWaT. Finally, in Section 2.3.3 we present eScienceSWaT. This is the methodology that was developed by Karastoyanova et al. to achieve better e-Science solutions.

### 2.3.1. e-Science Life Cycle

In Section 2.2.5 we introduced additional requirements for an e-Science solution that arise out of operating principles of scientists. At the end, the DSS should support the natural scientist. Therefore, we need to have a better understanding how scientists are doing research. The *e-Science life cycle* depicts on a high level the approach of scientists doing experiments.

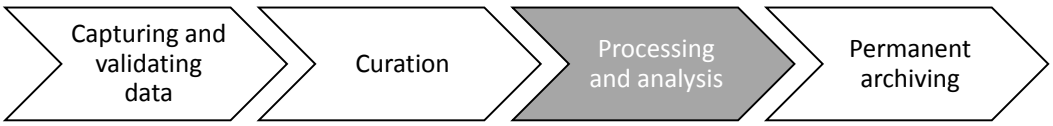


**Figure 7.:** The e-Science Life Cycle Phases based on [57].

The four phases of the e-Science life cycle are shown in Figure 7. We give a short explanation on the most important tasks:

1. **Capturing and validating data:** First of all the data is collected. The data can come from various sources e.g. data from sensors and instruments or data from simulations. In order to ensure the data quality, the data needs to be validated. In some cases additional meta-data is collected, e.g. about status of sensors or calibration offsets for sensors [124].

2. **Curation:** A suitable format for the data is defined. The data is stored in a way that it should not get lost. But scientists do not only use databases, but also use excel-sheets or any other format like simple text files for storing their results.

3. **Processing and analysis:** In this phase the raw data is analysed to reveal knowledge and gain new insights. In many cases this involves different IT systems and the manual intervention of the scientist, e.g. copying over intermediate results from one server to another [57].

4. **Permanent archiving:** After the analysis, the data and the processed results get stored permanently.

Many research projects build systems to target the *processing and analysis phase* in particular [56]. However, many of the IT solutions are built with deficiencies from the view of software engineering and enterprise computing. Outdated IT technologies are used, a low degree of modularization or the non-interoperability are only a few of the identified issues. Therefore, the authors recognized the necessity for focused fundamental research in the *processing and analysis phase*. They want to revolutionize the way software for e-Science is built and used by scientists [56].

### 2.3.2. Rationale behind eScienceSWaT

This section is based on the work in [56]. We introduce here the deficiencies within e-Science solutions development. This helps to understand why eScienceSWaT was developed. According to Karastoyanova et al., the cooperation between natural scientists and IT needs to be improved. The tasks from natural scientists need to be separated from the IT tasks. This helps to build better e-Science solutions. Natural scientists can focus on their strengths and their domain, whereas the IT tasks can be executed by IT professionals. Particularly, systematically organized best practices for building e-Science solutions are missing. In addition, the authors identified that there is no unified application model for eExperiments. In some publications the term eExperiment is used instead of scientific experiments. In this thesis we also use the terms scientific experiments and eExperiments interchangeably as synonyms. A unified application model makes it possible to develop a unified software methodology. This again makes it possible to use the methodology across different domains.

Therefore, in [56] an *eExperiment Model* is defined (see Figure 8). It consists of the following 3 layers: *scientific experiment model*, *IT experiment model* and *infrastructure*. One can observe that a scientist is able to make decisions on each layer based on his requirements and knowledge. There is also a high variety on possible solutions on each layer (see Section 2.2). Here, both scientists and software developers require tool support for making decisions to reduce the complexity of e-Science. Furthermore, there exist dependencies between the layers. The methodology should also capture these dependencies. In addition, decision chains need to be captured as well. Out of the eExperiment model the different steps for the methodology can be retrieved. If the methodology is more in line with the eExperiment model, it is closer to the scientists' way of doing research. Classic software engineering methods like the *Boehm-Waterfall software engineering methodology* are out of harmony with the scientists' way of experimenting [13]. The methodology should support the scientist to build an experiment model in his preferred domain language. Then the model of the experiment needs to be transformed to the next lower level. Ideally, this is done in a (semi-) automatic way. This

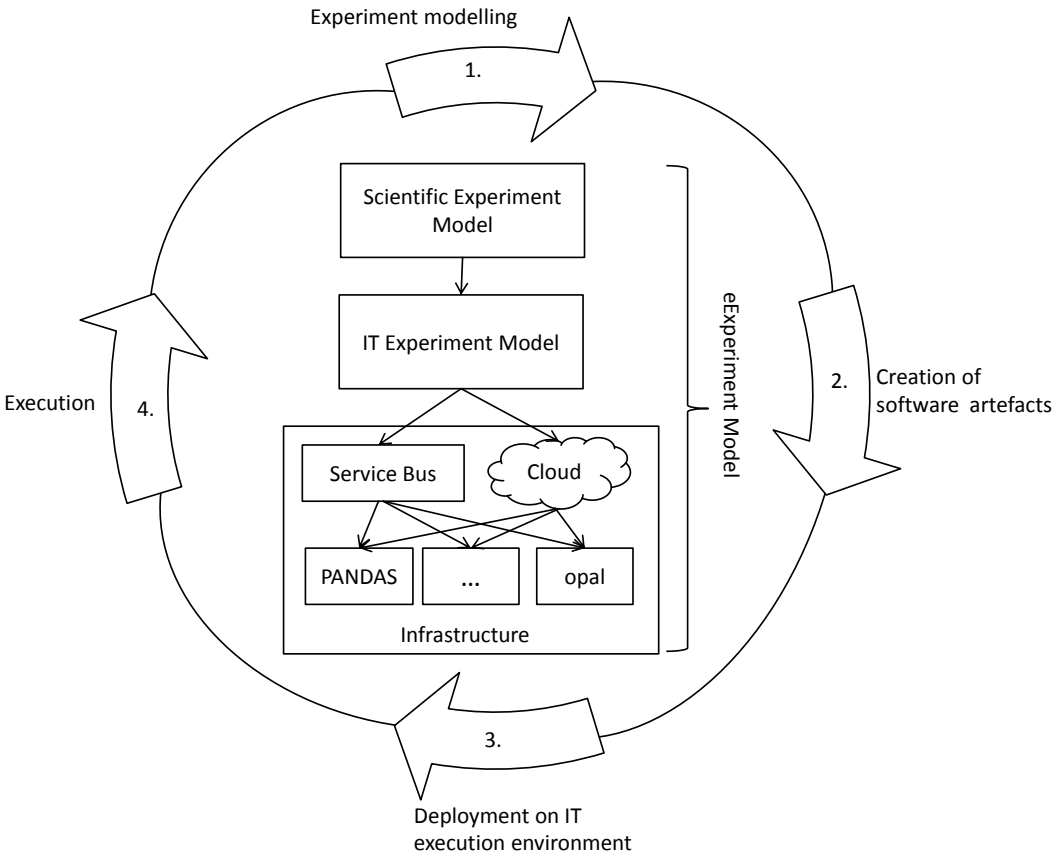


**Figure 8.:** This is the eExperiment life cycle and the eExperiment model with its three layers based on [56].

transformation step leads to executable artefacts. These artefacts have to be mapped onto an execution environment. In the end, the experiment has to be executed [56]. The following section shows the resulting methodology.

### 2.3.3. eScienceSWaT

First of all we present the methodology for eScienceSWaT and then we explain how a DSS is supporting decision making within eScienceSWaT.

#### Methodology

In Section 2.3.2 requirements for an e-Science methodology based on a unified application model for eExperiments in the *processing and analysis phase* of the *e-Science life cycle* are derived. In addition, observations of deficiencies in current e-Science solution development also lead to requirements. Out of that requirements, Karastoyanova et al. defined the *eExperiment life cycle*, which is built around the *eExperiment model* (see Figure 8). The eExperiment life cycle builds the basement of eScienceSWaT and consists of four phases:

1. **Experiment modelling:** The scientist models the eExperiment in his preferred modelling environment. For supporting this phase, patterns and best practices for modelling/design of experiments are required. An example for an eExperiments is a simulation that is based on FEM for different scientific fields like solid body or porous body simulations. Since non-functional properties for the IT infrastructure play an important role for making decisions in the next steps, they also have to be considered here in this phase.

2. **Creation of software artefacts:** In this step the pattern-based model of the scientific experiment is transformed into software code and additional meta-data. Choices made in the previous step influence the decisions for this step. Different IT technologies require different transformations. These transformations again require specific information. This approach might require several transformations until executable software artefacts are derived. The methodology does not require the fully automation of the transformation. Semi-automated derivation as well as manual creation of the IT experiment model are also possible. For instance the experiment model of the previous step is transformed into a workflow model.

3. **Deployment on IT execution environment:** Here, the previously created software artefacts are deployed on the selected IT environment. An example is the deployment of a workflow model on a Workflow Management System (WfMS). A broad range of complexity in the scenarios can be found. The installation of a single program on a single computer is rather simple. Provisioning of an execution environment as a first step and then deploying software artefacts on it is a more compound task. The decisions made in the previous steps lead to requirements for the IT execution environment, that need to be integrated.

4. **Execution:** In the preceding step software artefacts were deployed on an IT execution environment. Now these deployed artefacts are executed. By way of example the workflow model is instantiated and executed with a WfMS on a Grid infrastructure. The selection of infrastructure for execution depends also on decisions about technology and non-functional requirements from previous steps.

Scientists from different domains, with different backgrounds and skills work together in e-Science (see Section 2.2). Within eScienceSWaT, four succeeding phases lead to the execution of an eExperiment. The eScienceSWaT methodology enables that different scientists can work in their domain of expertise. We showed, that there exist interdependencies between the different phases of the methodology. This also makes the decisions in a specific phase dependent on the decisions made in previous phases.

In Figure 9 we present a simplified decision making process in eScienceSWaT based on an example. The natural scientists make decisions in the *experiment modelling* phase, when they are modelling an experiment (see *Decision 1 to Decision 3*). Scientists with computer-science background have to make the choice about IT technology within the *creation of software artefacts* phase. Their choice is limited by their own knowledge about technologies in general. In addition, they need to understand the decisions of the natural scientists' which lead to requirements for the creation of software artefacts. We assume, that they can extract all

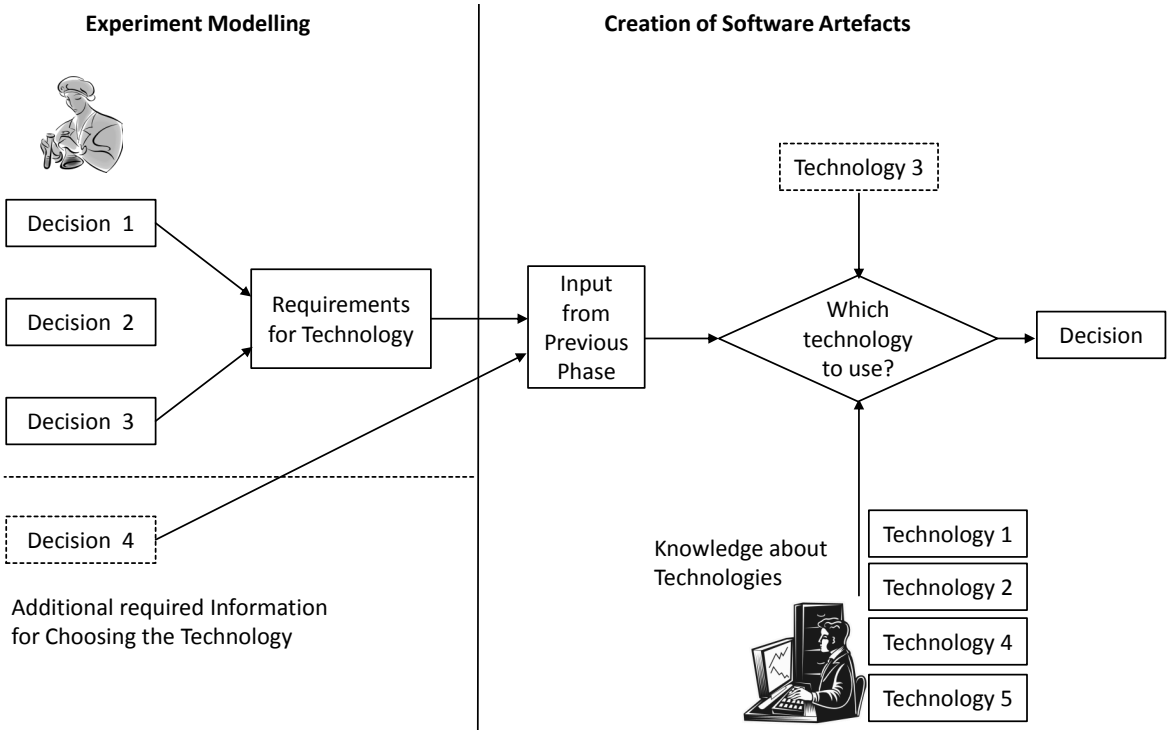


**Figure 9.:** This is an example for the decision making process within eScienceSWaT.

requirements out of the natural scientists' modelling decisions. However, they realize that they cannot make a decision without additional information from the natural scientists. This additional information is represented by *Decision 4*. Furthermore, there exists another technology that matches the requirements of the natural scientists (represented by *Technology 3*). Initially, the computer-science researchers were not thinking of *Technology 3*. The driving question is now how to support the scientists in the decision making process.

#### Decision Support System

Karastoyanova et al. propose to use a DSS within eScienceSWaT [56]. We explain how the scientists in the above example would benefit from a DSS in the following. Firstly, possible choices for modelling an experiment are presented to the natural scientists. They can select the solutions that fit best for their way of modelling experiments. Furthermore, they have to make additional decisions that help the IT professionals/scientists in the following phases to make decisions (see *Decision 4*). Their choices are captured in the DSS. Based on these choices, the DSS now proposes technologies, that do fit the requirements of the natural scientists. *Technology 4* and *Technology 5* do not match the requirements and therefore, the DSS is not showing them. *Technology 1*, *Technology 2* and *Technology 3* are suitable and therefore proposed by the DSS. However, the computer-science researchers are not experienced with *Technology 3*. Since the DSS also holds information about best practices for the use of the different technologies, they can read which scenarios do benefit most by using *Technology 3*. Finally, the scientists with computer-science background can make a decision about the most suitable

technology. This decision is based on the proposals of the DSS and their experience. In this case, the DSS is supporting by capturing relevant requirements and providing additional solutions packed with best practices.

In Section 2.1 the advantages of patterns were discussed. The eScienceSWaT methodology also makes strong use of patterns. This is why also the DSS makes strong use of patterns. Karastoyanova et al. especially see the definition of a meta-model for documenting the patterns and the relationships among the patterns as a challenge from knowledge management and software engineering point of view. Additionally, the architecture and implementation of the pattern catalogue for decision support is another challenge [56]. The core element of the DSS is to provide relevant information for decision making within each phase of the methodology. This information is represented in form of patterns. Karastoyanova et al. point out, that the DSS should help to select the best patterns across all layers in order to retrieve the most suitable realization of an eExperiment. The proposals of the DSS should take previous decisions of the user into account. Furthermore, the DSS should also be able to propose decision chains and related patterns [56].

## 2.4. Scientific Workflows

During our literature research we were confronted with different maturity levels of *scientific workflows*. We came across scientific workflows ranging from workflows that contain many manual steps to workflows with a high degree of automation. Thus, we identify the demand to present this term in a separate section. Vouk et al. define scientific workflows as a general concept for describing a sequence of structured activities and computations that emerge in scientific problem-solving. Often these structured activities are named studies or experiments [123]. This definition is formulated in a more general manner and matches our experiences of the broad range of scientific workflows. In [68], scientific workflows are defined as follows:

> *„These are networks of analytical steps that may involve, e.g., database access and querying steps, data analysis and mining steps, and many other steps including computationally intensive jobs on high performance cluster computers."*

This definition is more specific compared to the definition of Vouk et al. and already includes characteristics of e-Science like the *computationally intensive jobs*. Additionally, a vision of scientific workflows is presented in [68]. This vision incorporates that scientists can plug-in all kinds of scientific data resources. In addition, they are able to inspect and visualize the data on the fly. If a scientist changes the parameters for the eExperiment, only the affected components are re-run. Furthermore, meta-data is captured. This helps to make the eExperiment more understandable and reproducible by other scientists. Finally, a scientific workflow system becomes a scientific problem-solving environment. The components of this environment are distributed and a service-oriented Grid infrastructure is used [68].

For our purpose we do not need to stick to a specific definition of *scientific workflows*. Our goal is to identify patterns for e-Science. On top of these patterns we build a DSS. As we can see in Section 2.2, e-Science is built on top of traditional science and traditional concepts. Hence,

both the more manual processes and the complex and highly automated scientific workflows can be valuable for our research if they contain best practices, which also can be applied in e-Science.

# 3. Related Work

Power defines a DSS in [88] as an interactive computer-based system, which supports people in using computer communications, documents, data, models and knowledge with the goal of solving problems and decision making. The DSS for eScienceSWaT should enable decisions about the modelling of the experiment, the choice of IT models and underlying infrastructures [56]. As described in Section 2.3.3, eScienceSWaT is strongly based on patterns, which store knowledge and best practices. Therefore, the DSS is also based on patterns. In [56], Karastoyanova et al. point out, that the chain of patterns across all layers of an eExperiment are responsible for the efficiency and suitability of the eExperiment realization. Hence, we see the major function of the decision support system to help with the choosing of the most suitable patterns for the eExperiment. For the related work in the next sections, we state which things we can reutilize and which things we can transform so that they fit for our purpose. Section 3.1 comprises different approaches and techniques for decision support in the domain of design patterns. Furthermore, we examine different relationships in the domain of patterns in Section 3.2.

## 3.1. Decision Support Systems for Choosing Design Patterns

During our literature research for related work in the domain of DSSs, we came across several frameworks, expert systems and recommendation systems in the area of selecting design patterns. The approaches are ranging from very simple to sophisticated approaches, that also involve data mining and machine learning. The following sections introduce the different concepts first. At the end of each section, we state which concepts we can reuse in the context of this thesis.

### 3.1.1. Case Based Reasoning

In order to automate design pattern application, Gomes et al. created the REBUILDER framework. REBUILDER is based on Case-Based Reasoning (CBR). In CBR past experiences are used for understanding and finally solving new problems [62]. REBUILDER has the purpose to support software developers in the design phase with intelligent help. A case in REBUILDER consists of a software design, a design pattern that helps to improve the design and information about how to apply the design pattern. The software design is described in form of an Unified Modeling Language (UML) class diagram [42]. A user imports the UML class diagram for which he seeks support into REBUILDER. Then the relevant cases

are identified by REBUILDER. After that, the most suitable case is selected with the help of a similarity metric. Then, the corresponding design pattern is applied automatically to the UML diagram of the provided software design. Together the initial design problem, the pattern and the information how to apply the pattern build a new case. This case is then stored in the case library.

In the domain of e-Science many different representations for experiments exist (see Section 2.2). Therefore, we do not use UML. Without such a fixed format it is hard to automate the pattern selection process. However, full automation for pattern selection is not a requirement for the DSS. As reported by Thabasum et al., REBUILDER's system performance strongly depends on the quality and the diversity of the case library [98]. This means, that on top of creating all the e-Science patterns, we would also have to model such cases in which the patterns are applied. Therefore, we do not use CBR for e-Science pattern selection support. When a pattern is applied for a concrete software design, Gomes et al. store this information in form of a case. We will also provide an example, where the e-Science pattern is applied. However, an eExperiment involves several patterns and therefore we also need to store decision chains. Decision chains contain different patterns that build a solution together. For future work, we propose to store all patterns that are used in an eExperiment in a case. All these cases can be browsed by the users for decision support.

### 3.1.2. Simple Recommender System for Design Patterns

Guéhéneuc et al. see the major task of a recommender system in information filtering that leads to the items of interest, which can be presented to the user [46]. They developed a simple recommender system for design patterns. The user selects words, which are related to his design problem. The system searches for the design patterns, that match the selected words. Then the best matching patterns are presented to the user. In order to realize such a system, the textual description of design patterns have to be analysed as a first step. The result of this analysis is a set of the most important words for each design pattern. Then this words are associated with the design pattern [46]. In another research project, Guéhéneuc et al. conclude, that this system cannot handle complex design problems. Furthermore, the user has no chance to provide feedback to the recommendation system [84]. We also think that this simple approach is not sufficient for the rather complex designs that are required for eExperiments. However, adding meta-data to design patterns in term of tags might be a good idea, if we want to enable browsing over the pattern catalogue.

### 3.1.3. Design Pattern Recommender

The Design Pattern Recommender (DPR) developed by Palma et al. is an expert system [84]. An expert system simulates the behaviour and judgement of a human or an organization. In addition, it has expert knowledge and experience in a specific field [95]. The main goal of DPR is to support designers in finding the most suitable pattern for a particular design [84]. Palma et al. use a ranking based selection mechanism for DPR. The ranking is calculated by

a simple Goal-Question-Metric (GQM) approach [33]. Four major developing steps lead to DPR, that are explained in the following [84]:

1. **Identify the conditions in which patterns can be applied:** Here individual patterns are studied in depth. Also experts in the domain of design patterns can be interviewed in order to gain knowledge. Finally, for each pattern the conditions are defined, in which they can be applied. For each pattern a tree is created that connects the pattern with the circumstances (see Figure 10).
2. **Refine the conditions with sub-conditions:** Sub-conditions for each pattern are defined. They hold information where the pattern can be applied. The sub-conditions are inserted into the tree for the pattern (see Figure 10).
3. **Formulate questions to ask designers:** The tree representation stores knowledge about the pattern. The nodes are transformed into questions that can be asked to designers. At the end of this step for each tree a set of questions is generated.
4. **Formulate GQM model with the defined questions:** The pattern names are the goals. The formerly derived questions in Step 3 build the questions.

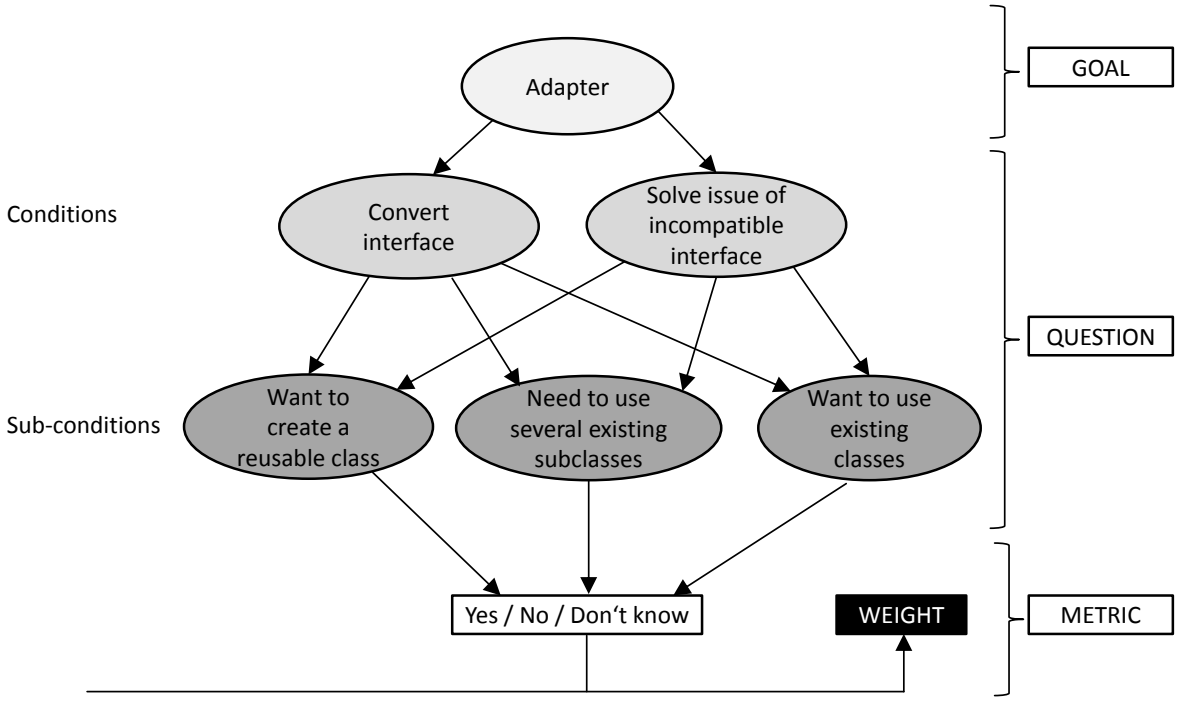


**Figure 10.:** The adapter pattern with conditions (pattern-intents) and sub-conditions (pattern-applicabilities) and the mapping to GQM, based on [84].

For finding the most suitable design pattern, the designers have to answer the bottom questions with *yes*, *no* or *don't know*. In addition, they have to weight the questions with an integer between 1 and 9 (O only for *don't know*). Their answers and the weighting are used by DPR that finally proposes the most suitable pattern with the help of a metric. We do not want to go in more detail here. The full metric and the details of DPR can be found

in [84]. The user interacts with the DSS by answering questions. This interaction process has a big advantage, because it is based on questions and people from all domains should be able to answer questions. Therefore, GQM approach seems suitable for a DSS in the domain of e-Science. The GQM approach is suitable, when one pattern has to be selected from a group of possible patterns that can solve the problem. However, the users of our DSS have to select several patterns on different layers for an eExperiment. Hence, GQM is not applicable on a global scale. We propose to use GQM at specific decision points, when the user requires additional decision support. Finally, we do not use GQM for the thesis, because it requires a lot of effort to define the questions for each pattern. In addition, GQM is not applicable for the whole experiment, but only at specific *decision points* that have to be defined as well. The concept of GQM is part of future work. Guéhéneuc et al. use an eXtensible Markup Language (XML) representation of the DPR for the patterns in their prototype [84]. To use an XML representation seems suitable, since this enables portability and platform independence.

### 3.1.4. Design Pattern Recommendation System

Suresh et al. want to support novice and experienced users by the selection of design patterns [115]. They use the concept of *social recommendation* in their Design Pattern Recommendation System (DPRS). The user has to select his level according to his knowledge: *Novice*, *Beginner*, *Advanced beginner*, *Expert* and *Wizard*. This is helpful for the analysis of the acceptance level of the different user groups [115]. The user starts by sending a query text to the system. The query is a collection of words, that describe the problem scenario. The text is analysed in order to find out the *intent* of the query. From here, two different scenarios are possible.
In the first scenario, knowledgeable questions are retrieved from the pattern repository based on the intent [115]. The user has to answer these questions. Then the system is calculating a score for the possible patterns. Based on the score, the system recommends either *strongly recommended*, *satisfactory recommended* or *not recommended*. The interaction of the user with the system is stored. For details see [115]. This is helpful for performance evaluation of the system and is required for the second scenario.
In the second scenario, the intent of the query is compared to queries from the past. The goal is to find a similar query. If a query satisfies the minimum thresholds of support and confidence, then this query is chosen. The old query led finally to a pattern. This pattern is recommended. The system always executes scenario 2 first. If no pattern can be recommended automatically, the first scenario is executed.

We think the idea to use a query to narrow down the amount of possible patterns is also applicable for most domains. This is because the query is free text and not a *query language* like Structured Query Language (SQL). If SQL is used, only certain users can use the DSS, because not everyone knows how to use SQL. However, there comes overhead due to the parsing of the text of the query. There needs to be a mechanism for retrieving the intent of the query. Finally, the patterns need to be stored in a way, that they can be parsed as well in order to find matching patterns. Therefore, we will not use a query based DSS for this thesis. An automatic mechanism, that suggests patterns to the user is helpful. Certainly, this

comes with lots of overhead. First of all we need to store the interaction from all users. This builds the basis on which patterns are selected. We also see the thresholds for *support* and *confidence* crucial, because they lead to a recommendation. Therefore, we would first need to simulate different users and then define the thresholds. This is why we decide not to use a fully automated DSS for patterns, which is based on social recommendation. However, since the users of our DSS do not build a homogeneous group of people, we also need a mechanism for supporting the different users. Whereas DPR by Guéhéneuc et al. (see Section 3.1.3) only suggests the highest ranked pattern as a solution, here more than one pattern can be recommended. We think it is useful to provide an alternative and therefore, we will provide not only one solution in our DSS.

### 3.1.5. Systems for Implicit Culture Support

In [10], Birukou et al. present the architecture of a multi-agent system for design pattern selection. Their concept is based on Systems for Implicit Culture Support (SICS). In general, knowledge can be sub-divided into *explicit* and *implicit* knowledge. Knowledge is *explicit*, when it can be described and shared through any kinds of documents. In contrary, *implicit* knowledge is contained in the abilities and capabilities of community members [10]. So the basic idea is to give inexperienced users decision support for selecting design patterns through the implicit knowledge of more experienced users. This relation between the single agent (inexperienced user) and the group is called *Implicit Culture (IC)* [12]. The knowledge transfer is enabled by a SICS and finally leads to inexperienced users behaving similar to the community culture [10]. Universally, the architecture of a SICS consists of an *observer*, an *inductive module* and a *composer* [10]. The *observer* observes the actions that are performed by users and stores them in a database. The stored observations are analysed by the *inductive module*. Additionally, learning techniques like data mining or machine learning are applied. This leads to theories, which keep information about the actions performed in different situations. The *composer* utilizes the information kept by the *observer* and the *inductive module* in order to make suggestions to the user.
In the domain of choosing design patterns, the *observer* stores information about the design problem, the potential patterns to solve the problem and the selected pattern, which solved the problem. The *inductive module* analyses the history of users' interaction with the system. It groups related problem-solution pairs to a theory. The *composer* compares the design problem of the user with the problem part of different theories. If a matching theory is found, the corresponding solution part is suggested [10].
In the following, we describe the main differences between SICS and systems that are based on CBR. CBR uses a specific situation from the past and suggests the solution to this specific situation. IC supports the user by solving a problem, but not producing a solution directly [10]. In IC only the implicit information about the problem and the solution is available, whereas in CBR the problem and the solution is represented explicitly as the case.

We consider this approach as the most complex approach for decision support in the domain of design patterns so far. It involves not only storing information about user interactions, but the even more compound task to apply machine learning and data mining algorithms on

this information to retrieve knowledge. The presented SICS can even be further extended, so that it also provides support for more complex scenarios [98]. Therefore, it would fit for our purpose, because we identified complexity in the domain of e-Science (see Section 2.2). However, the effort to implement a system with the above described functionality is beyond the scope of a thesis. Hence, we decide not to base our DSS on IC.

### 3.1.6. Design Pattern Selection, A Solution Strategy Method

Sahly et al. developed a strategy method for obtaining an appropriate recommendation for the selection of design patterns [97]. Their approach uses different algorithms.
*Query-Matching-Pattern (QMP)* parses the query of the user and tries to find a matching between pattern intents and the query. This is done by using Vector Space Model (VSM). This is an algebraic model in which texts can be represented as vectors [52]. Both the query and the pattern can be represented as vectors. Now similarity between query and pattern is the cosine value of the angle between their vectors. First of all, we decided not to use a query for our DSS. Secondly, the additional work that is coming with the use of VSM is not worth the effort for calculating similarity in our case. Hence, we will not use QMP and VSM.
*Question-Answer-Session (QAS)* has the goal to narrow down the amount of suitable patterns by asking questions. The questions that are asked, are dependent on the answers to previous questions. To make this efficient, Sahly et al. divide the questions among four levels: *Pattern-domain (Level 1)*, *Pattern-category (Level 2)*, *Pattern-intent (Level 3)* and *Pattern-specific (Level 4)*. Questions of the domain-level have to be answered first. This narrows down the questions of the category-level and so on. Since the *eExperiment model* is structured hierarchically (see Section 2.3.3), this approach would fit well for the DSS. However, QAS is similar to GQM, which we decided not to use due to the additional effort that comes with the definition of suitable questions. Therefore, we do also not use QAS for our DSS in this thesis.
*Query-Similarity-PreviousQuery (QSPQ)* is an algorithm, that is based on users' queries from the past. Since this approach is similar to DPRS in Section 3.1.4 and we decided not to use a query based DSS, we will not explain it in more detail.
The last algorithm is called *Collaborative-Implicit-knowledge (CIK)*, which is quite equal to SICS, which is described in Section 3.1.5. Since we decided not to base our DSS on IC, we also do not use CIK and we do not go into more detail here.

The proposed method of Sahly et al. supports three different types of recommendations [97]. First of all, one single pattern is recommended for solving one particular problem. In addition, recommendations regarding the implementation of this pattern are provided. The third type of recommendation are pattern sequences that contain patterns, which are often applied together. We have stated several times already, that the recommendation of one single pattern is not sufficient for an eExperiment. However, the DSS also will be able to recommend pattern sequences, which we call decision chains. Information regarding the implementation of the pattern is stored directly within the pattern. We use a specific attribute in the pattern format, that can provide this information in form of an example.

### 3.1.7. Patterns 2.0: a Service for Searching Patterns

Pattern selection can be seen as an instance of the problem of retrieving relevant information from large documents collection [11, 29]. However, patterns have specific characteristics, e.g. containing different parts that express different kinds of information or patterns are linked in a pattern language [11]. Therefore, various approaches have been developed to support pattern selection. Weiss et al. identify the following shortcomings to existing approaches in the domain of pattern selection support [11]:

- Additional effort in the authoring and selection process (e.g. meta-data required for patterns) is required
- Pattern repositories require effort in maintaining and updating information
- Patterns are targeted at one specific user group (for instance developers)
- Collaboration and personalized recommendations are rarely supported

This is why they build the *Pattern 2.0 service*[1], which is a composite service for simplifying pattern selection. *Pattern 2.0* addresses the previously outlined shortcomings with the following concepts [11]. The search is improved by using *tagging* and *usage history*. In addition, *community-generated content* is used. This minimizes the effort in updating and maintaining the pattern repositories. The service is orthogonal to the domain of patterns and their format. Furthermore, it can be used in collaboration (usage data is shared between communities) or in an isolated way (each community only uses recommendations that are based on their own usage) [11]. Further details to the services can be found in [11].

The collaborative characteristics of e-Science are supported by the *Patterns 2.0 service*. Additionally, the concepts of Weiss et al. take the different domains within e-Science into account. Therefore, we strongly recommend to consider these approaches of the *Patterns 2.0 service*, when implementing a DSS in the domain of e-Science. However, these approaches are too sophisticated for implementing them in this thesis and we do not use them.

[1] 2.0 is used in the name because there exist similarities to Web 2.0: the service includes tagging and other community-generated content [11]

## 3.2. Relationships Among Patterns

In Section 2.1, we can identify different kinds of relationships between patterns. Alexander uses a hierarchical structure to organize the patterns. Therefore, there can be patterns on a lower level that help to complete the patterns on the level above (see Figure 2). Gamma et al. use a pattern format, that includes the attribute *related patterns*. Related patterns are closely related or should be used in combination with the current pattern (see Section 2.1.2).
We see that different relationships are possible among patterns. However, we only focus on relationships that are useful with regard to decision support. Decisions for one layer of the eExperiment model influence the decisions on the layers below (see Section 2.3.3). We transfer this concept to the domain of patterns, so that the status of a pattern influences the decisions for the related patterns. After the selection of one pattern, all relationships of this pattern are evaluated. Then the status of the related patterns gets updated. This reduces the amount of possible patterns in the next step or emphasises specific patterns. At the beginning all pattern have a *neutral* status. From a neutral status, a pattern may change to the following status: *highlighted*, *selected* or *deselected* (see Section 5.1.1). By focusing on this pattern status changing relationships, we build a basis for the DSS.

### 3.2.1. Classification of Relationships

We introduce the used terminology for classifying relationships shortly, since different terminologies exist. Abhijit et al. state, that relationships in general can be classified by the *cardinality* of the relationship and the *degree* of the relationship [87].
The *cardinality* of the relationship describes the minimum/maximum number of instances, which must/can be associated with another instance [87]. Examples are *one-to-one*, *one-to-many* or *many-to-many* relationships. We use *one-to-one* relationships for connecting the patterns of the e-Science pattern catalogue.
The *degree* of a relationship describes how many entities participate. If only one entity participates, the relationship is called *unary*. If n entities are participating, we have a *n-ary* relationship. Currently, we do not see the need for self relations and therefore, we do not use *unary* relationships. However, we always connect two patterns with a relationship. Hence, we only use *binary* relationships [87].
Relationships among entities can either be *unidirectional* or *bidirectional* [82]. In *unidirectional* relationships, only one entity has a property that refers to the other entity, whereas in *bidirectional* relationships each entity has such a property [82]. We only use *unidirectional* relationships for the e-Science pattern catalogue.

### 3.2.2. Relationships in the Domain of Patterns

Table 1 contains all relationships, that we want to use for the e-Science patterns. The column *name(s)* contains the different names that are used by different authors for the relationship. If we reuse a name for the e-Science pattern relationships in this thesis, we put it in bold letters. In the column *explanation* we summarize the meaning of the relationship and present

our extensions, if needed. In the introductory phase of this section we pointed out, that only relationships can be reused, which lead to a status change of the related pattern. Therefore, also the resulting status of the related pattern is displayed. Based on the findings here, we will define the relationships for the e-Science patterns in Section 4.2.2.

| Name(s) | Explanation | Status |
|---|---|---|
| **Uses**, Might Use, Comprises | A pattern uses another pattern. | Highlighted |
| Must Use, **Requires** | A pattern must use another pattern. For example a pattern requires another pattern for its own solution. | Selected |
| Similar To, Alternative | Both patterns address a similar problem by using different solutions. We extend this relationship so that also patterns that do not address a similar problem can be connected. We call this modified relationship **Related To**. | Highlighted |
| **Conflicts** | Both patterns address a similar problem by using different solutions. However, we extend the meaning. Also patterns can be in conflict, which are not addressing a similar problem. | Deselected |
| **Refined By** | A more general pattern is refined by a more specific pattern. It is the inverse relationship to the *refines* relationship (see Table 2). | Highlighted |

**Table 1.:** Different pattern relationships that we reuse for our e-Science patterns based on [63, 76, 131].

There are other pattern relationships, which we do not use for this thesis. Since they might be useful in another context, we also present the not used pattern relationships in Table 2.

| Name | Explanation |
|---|---|
| Refines | One specific pattern refines a general pattern. |
| Used By | A smaller pattern is used by a larger pattern. The inverse relationship of the *Uses* relationship. |
| Variant | A *variant* pattern refines a more well-known pattern. When an abstract pattern is instantiated, some ways are more common for this instantiation. The most important variations can be stored in a separate pattern. This pattern gets connected to the original pattern via the *variant* relationship. |
| Variant Uses | A variant of a pattern uses another pattern. This means, that the other pattern is not used all the time. |
| Combines | Two patterns are combined to solve a single problem, which is not addressed yet by any other pattern. |
| Tiling | Here a pattern uses itself. |
| Sequences of Elaboration | This is a sequence of patterns or a fragment of a pattern language. |

**Table 2.:** Different pattern relationships that we do not use for our e-Science patterns based on [63, 76, 131].

# 4. e-Science Pattern Catalogue

This chapter addresses the core deliverable of this thesis – the e-Science pattern catalogue. In Section 4.1, we describe our approach for developing the pattern catalogue. Out of this approach, the concept for the pattern catalogue emerges. It comprises the pattern format and the relationship types, as described in Section 4.2. Another result of the process are the concrete e-Science patterns, the relationships among them and the structure, in which the patterns are organized. Together, they build the e-Science pattern catalogue. The catalogue is oriented towards eScienceSWaT, which is built around the three layers *scientific experiment model*, *IT experiment model* and *infrastructure* of the eExperiment model (see Section 2.3.3). Therefore, it seemed suitable for us to use the layers as a rough structure for the e-Science pattern catalogue. This is why Section 4.3 comprises the patterns on the level *scientific experiment model* and Section 4.4 contains the *IT experiment model* patterns. Finally, the *infrastructure* patterns are introduced in Section 4.5.

## 4.1. Development Approach

The e-Science pattern catalogue is one big deliverable of this thesis. Therefore, we recognize the necessity to present our approach for developing the e-Science pattern catalogue. Our approach starts with an *information collection* phase and building on top of that a *pattern generation* phase, which leads to the e-Science pattern catalogue as shown in Figure 11. The information collection phase is presented in Section 4.1.1 and the pattern generation phase is introduced in Section 4.1.2.

### 4.1.1. Information Collection

The information collection phase evolved to 6 steps as shown in Figure 11. The whole information collection phase is done in an iterative way, because we analyse various applications for e-Science. Basically, for each application we perform an analysis, we extract information and after that we store the information. Then we start over with the next application for e-Science. At one point in time we stop with the collection of new information. We rather focus on transforming the already collected information into patterns and then start with a new information collection phase. The rationale behind this is that we want to avoid collecting similar information that finally leads to an existing pattern. If we identified a pattern already, it is easier to recognize the pattern already in the step *analyse e-Science solutions*. This saves time, because we can skip the other steps of the information collection phase and start

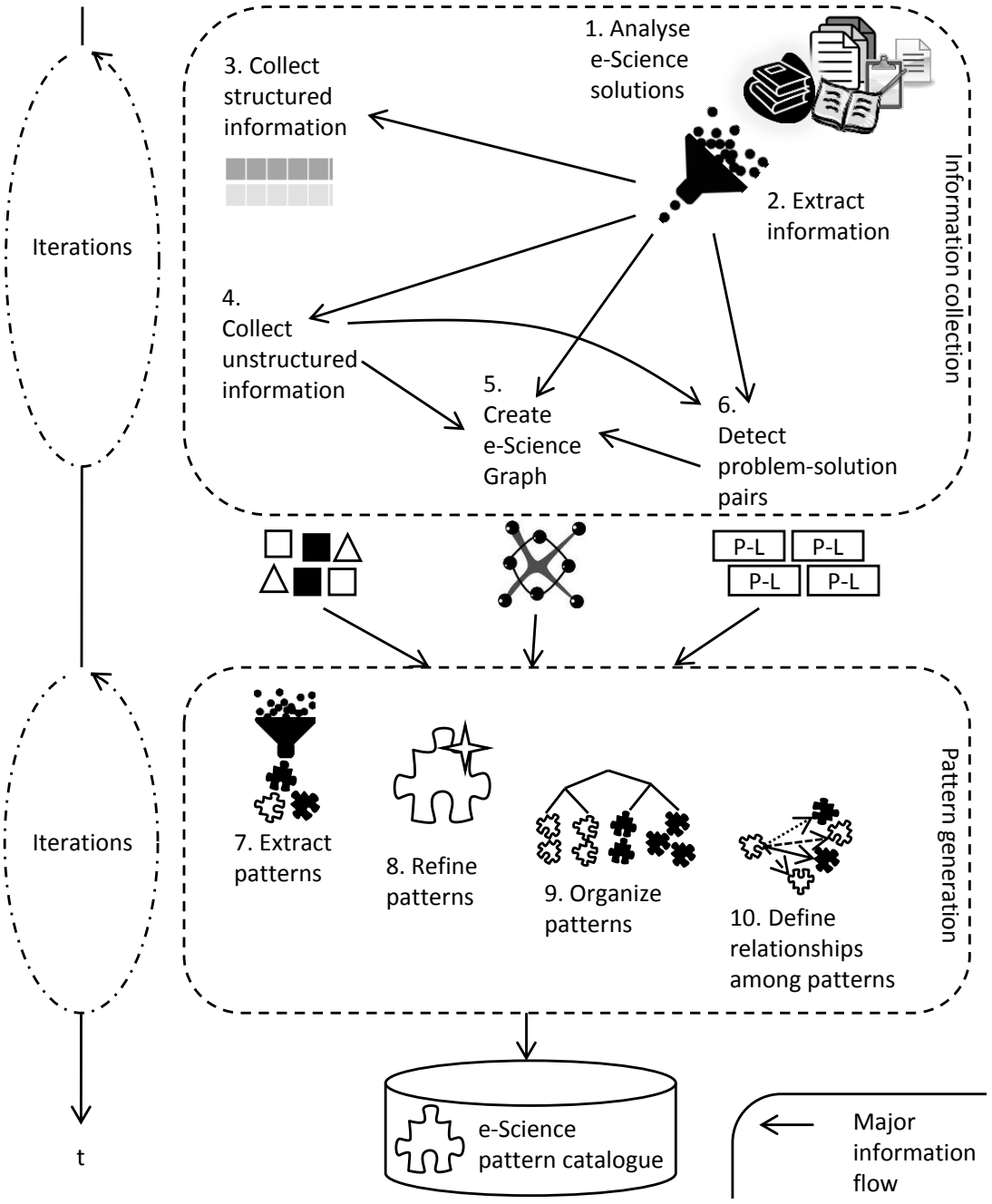


**Figure 11.:** This is an overview of the e-Science pattern catalogue development.

over with new e-Science solutions. The information collection phase leads to an *unstructured document*, an *e-Science graph* and *problem-solution pairs*. All this information builds the input for the *pattern generation* phase, that is introduced in Section 4.1.2.
In the following, we describe each single step of the *information collection* phase and explain how the whole information collection process evolved. However, we only collect *structured information* at the beginning of the first iteration. It is difficult to store all information about e-Science applications in a fixed table scheme. Therefore, we decide to replace task 3 with the task *collect unstructured information* for all following iterations.

**1. Analyse e-Science Solutions**
First of all, we analyse various e-Science solutions in the domain of scientific workflows by doing literature research. We realize, that different authors use varying levels of granularity and diverse perspectives for describing their e-Science solutions.

**2. Extract Information**
After the analysis we need to extract the important information from the e-Science solutions. At the beginning, we extract too much detailed information. For instance we also extract information about the used programming language, middlewares or operating systems. Later in the process, we realize that is hard to detect reoccurring problem-solution pairs on this abstraction level. Therefore, we refine our extraction process and focus on high level information about the e-Science applications.

**3. Collect Structured Information**
At the beginning we collect important information about e-Science solutions in a table. The table contains columns for the *domain*, the used *technology/model* and the *reason* why this technology/model was used. Additionally, *advantages* and *disadvantages* for the technologies/-models are captured in the table. We have difficulties, to store all important information in this table scheme. By using this fixed scheme, we also cannot connect the information chunks to each other.

**4. Collect Unstructured Information**
Because we cannot connect the chunks of information to each other, when using a table with a fixed scheme, we decide to replace the table with a new document. In this document we collect the information in an unstructured way. However, we use colours and visual effects for separating groups of information chunks. In addition, we use indent and colours for grouping information. Unfortunately, the document is getting bigger and bigger, and it gets harder to keep track of all collected information.

**5. Create e-Science Graph**
We collect unstructured information so far and realize that it is hard to stay on top of things. We want to have an overview about the aspects of e-Science solutions that we have already covered. Furthermore, we want to structure the information. This is why we decide to build an e-Science graph, which provides an overview of the collected information. The e-Science graph contains the following clusters of information: *scientists*, *non-functional properties*, *data*, *scientific experiment model*, *IT experiment model* and *infrastructure*. The graph also helps to avoid the collection of redundant information. At the same time we can identify missing aspects by examining the e-Science graph. After that we can start over with step 1, the analysis of

e-Science solutions, with focus on the missing aspects in e-Science solutions. The details and references to the sources are still stored in the document containing unstructured information. We recognize that the unstructured information in combination with the e-Science Graph do not provide us with sufficient support for identifying patterns.

**6. Detect Problem-Solution Pairs**
Currently, we collect information about e-Science solutions in an unstructured manner. In addition, we have an e-Science graph, which provides an overview about the collected information. Nevertheless, we want to generate e-Science patterns from the collected information. Figure 3 in Section 2.1 visualizes the concept of abstraction from concrete solutions and problems for generating patterns. This is why we focus on the identification of concrete problem-solution pairs now. We either highlight problem-solution pairs in the document containing the unstructured information or we transform the collected information into problem-solution pairs.

### 4.1.2. Pattern Generation

Similar to the *information collection* phase, also this phase is done in an iterative way. We extract a few patterns from the problem-solution pairs, we refine the patterns and we structure the patterns. Then we define the relationships among the patterns and insert the patterns in the pattern catalogue. Then we start over with the extraction of new patterns. However, the iteration also affects the pattern in the e-Science pattern catalogue. Patterns can always be refined by using a more suitable pattern name or providing a better example. In addition, when new patterns are inserted into the e-Science pattern catalogue, this can require adding new relationships to existing patterns. Furthermore, the structure of the pattern catalogue changes over the time, when new patterns are added to the catalogue. This can be due to more suitable groupings of patterns or splitting a group of patterns in two groups. Initially, we also thought about iterating over the two phases *information collection* and *pattern generation*. Certainly, we realize that the amount of information that we collected, is sufficient. Only in a few cases we step back to the information collection phase, and therefore we do not visualize this in Figure 11. In the following we describe each of the steps of the *pattern generation* phase.

**7. Extract Patterns**
At one point we decide to stop collecting new information, but rather carve out the patterns. However, before carving out patterns, we need to define a pattern format. Since the pattern format is one final result of the overall process, we put it in a separate section (see Section 4.2). Once the pattern format is available, we abstract from the concrete problem-solution pairs to generate the patterns.

**8. Refine Patterns**
Pattern writing is an iterative approach. We change the values of the pattern attributes again and again to improve the quality of the pattern. In addition, the pattern name can describe the problem, the solution or the desired characteristics. For example, we can name a pattern *low data quality* (describes the problem), *improve data quality* (describes the solution) or *high data*

*quality* (describes the desired state). When the pattern catalogue is growing, we sometimes need to reorder certain patterns. If some patterns of a group use the problem description as name and others use the solution as the pattern name, this can lead to misunderstandings. We need to make sure, that patterns within one group follow the same logic for deriving the pattern name.

**9. Organize Patterns**
In Section 2.3.3 we introduced eScienceSWaT, which is built around the three layers *scientific experiment model*, *IT experiment model* and *infrastructure* of the eExperiment model (see Figure 8). Therefore, it seemed suitable for us to use the layers as a rough structure for the e-Science pattern catalogue. However, we also need to define a more granular structure within each layer. The e-Science graph helps to structure the patterns, because it also contains a structure. Nevertheless, several iterations were required for generating the target structure. If new patters are added to the catalogue, this can require a change of the structure, e.g. a new category has to be defined or an existing category has to be renamed. In addition, patterns need to be moved to a more suitable category. A holistic view on the structure of the e-Science pattern catalogue comprising all patterns is presented in Appendix A.

**10. Define Relationships Among Patterns**
In this step we define the relationships among the e-Science patterns. First of all, we need to determine what kind of relationships are useful for the e-Science patterns. We decided to use *unidirectional binary one-to-one* relationships (see Section 3.2.1). Section 3.2.2 builds the foundation for the relationships among the e-Science patterns. Since these relationships are one important result in the context of this thesis, we put them into a separate section (see Section 4.2.2). Once we defined these relationships, we can apply them to the e-Science patterns.

## 4.2. Pattern Format and Relationships

This section comprises the pattern format for the e-Science patterns, which is presented in Section 4.2.1. In addition, Section 4.2.2 introduces the different relationships that we use to connect the e-Science patterns.

### 4.2.1. Pattern Format

We decide to focus on the most important attributes in the first step. Section 2.1 shows, that more useful attributes exist. Table 3 shows the specified format for the e-Science patterns. We describe a few attributes in more detail, where we think that this is required.
The *name* of the pattern is unique across the catalogue. Furthermore, the pattern name can describe the problem, the solution or the desired characteristics as described in Section 4.1.2. The pattern name has to be unique across the pattern catalogue. This helps to avoid misunderstandings, when natural scientists and IT professionals use the pattern name in their vocabulary, when they talk about e-Science solutions.

| Attribute | Explanation |
|---|---|
| Name | Unique Identifier |
| Classification | Layer of the eExperiment model, to which the pattern is allocated |
| Intent | Summarizes the purpose of the pattern |
| Problem | Captures the problem |
| Context | Setting where the pattern can be applied |
| Solution | Explanation how the problem is solved |
| Relationships | List of relationships to other patterns |
| Example | An example where the pattern is applied |

**Table 3.:** This is the format used to specify the e-Science patterns.

The attribute *classification* can take one of the values: *Scientific Experiment Model*, *IT Experiment Model* or *Infrastructure*.
The attribute *example* provides an example, where the pattern is applied. Often, this is only a reference to another paper or book, where the pattern is used.

### 4.2.2. Pattern Relationships

In Section 3.2.2 we introduced relationships, which were used in related works and we also stated which of them we want to reuse. Now we want to explicitly define, how we reuse the relationships and which semantics each relationship has. Since the relationships are used to support the decision making process, it is helpful to understand the role, which they play in the DSS (see Section 5.1.1). Figure 12 is used to make the explanation of the different relationships more tangible.

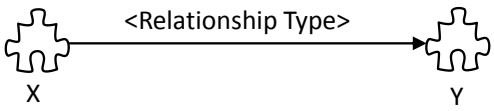


**Figure 12.:** This is an example for a relationship.

**<Uses>** If a pattern X has a <Uses> relationship to Y, it can optionally make use of Y in its solution. For instance the *heterogeneous data* pattern uses the patterns *data transformation* or *semantic mediation*. The relationship <Uses> is not as strong as the <Requires> relationship.

**<Requires>** Pattern X requires Y. This means, that Y also has to be applied, when X is applied. This can be due to side effects, which arise when X is applied. Therefore, Y is required to solve these side effects. Additionally, the <Requires> relationship can have the meaning, that Y has to be selected as a precondition, so that X can be applied. An example in the e-Science pattern catalogue for

the <Requires> relationship is found in the *low data quality* pattern. It is used, when the data quality of the currently available data sets is not sufficient for achieving reliable scientific results. This requires an action to solve this problem and therefore the *low data quality* pattern has a <Requires> relationships to the patterns *data quality gate* and *data quality improvement*.

**<Related To>** <Related To> replaces the *similar to/alternative* relationship, which was described in Section 3.2.2. Whereas *similar to/alternative* connects patterns, which address a similar problem by using different solutions, the <Related To> relationship goes beyond this. The <Related To> relationship connects X with Y, when Y is somehow related to X. For example Y addresses a similar problem by using a different solution or it is quite common to apply Y, when X is applied.

**<Conflicts>** X has a <Conflicts> relationship to Y, when Y solves a similar problem by using a different solution and they both cannot be applied together. In addition, X can have a <Conflicts> relationship to Y, even when they do not solve a similar problem. This just means that these patterns can never be applied together.

**<Refined By>** X is refined by Y. This means that Y is a more specialized version of X. For example Y is specialized for a domain or a specific case. Therefore, the user should check, if the refining pattern is more suitable.

### 4.2.3. Structure Elements


**Figure 13.:** This shows the visualization of a structure element.

In order to organize the patterns, we need to have additional structure elements. These structure elements are part of the e-Science pattern catalogue. They increase the usability by grouping similar patterns. In the textual description, the section build these structure elements, which organize the patterns. The graphical overview contains elements with dotted lines, that represent the structure elements as shown in Figure 13.

## 4.3. Scientific Experiment Model

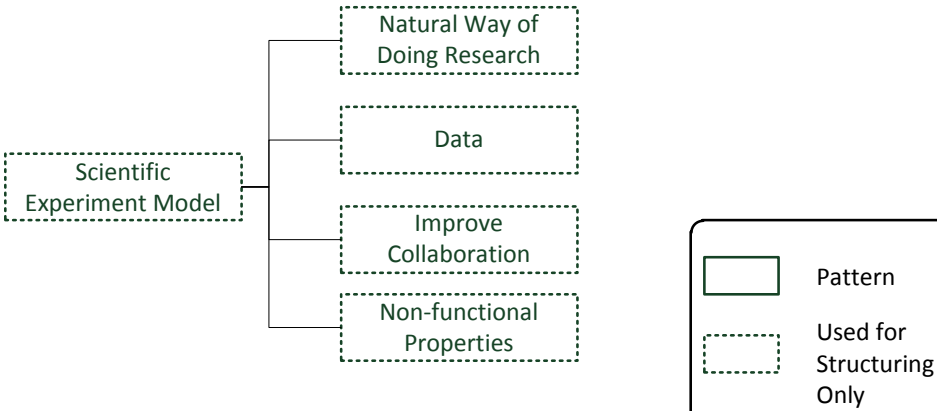


**Figure 14.:** This is an high level overview of the e-Science experiment model patterns.

In Figure 14 we give an overview of the structure of this section. This section comprises our e-Science patterns on the level of the *scientific experiment model*.

### 4.3.1. Natural Way of Doing Research

This section comprises the *natural way of doing research* patterns. Figure 15 gives an overview of the different research approaches.

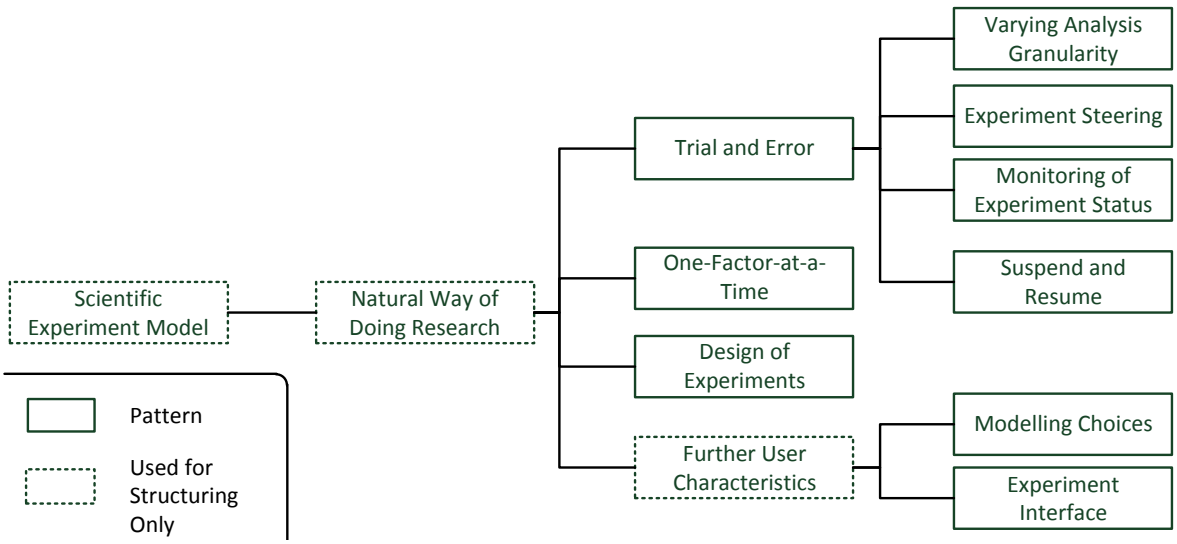


**Figure 15.:** This is an overview of the e-Science natural way of doing research patterns.

#### Trial and Error

| | |
|---|---|
| **Name** | Trial and Error |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to solve problems or discover new things in an unstructured way. |
| **Problem** | Scientists know the goal of their research, but they do not know the exact way for achieving the goal [104]. They try a certain parameter configuration. If this does not lead to a satisfying result, they want to try a variation of the parameters. However, they do not follow a methodology and therefore, the e-Science solution needs to be flexible to support this way of doing research. |
| **Context** | Scientists that want to discover new things. |
| **Solution** | Provide an e-Science solution that supports a trial and error approach. |
| **Example** | Karastoyanova et al. present a workflow environment, which supports the trial and error approach of scientists for developing experiments [104]. |
| **Relationships** | <Uses> Varying Analysis Granularity<br><Uses> Experiment Steering<br><Uses> Monitoring of Experiment Status<br><Uses> Suspend and Resume |

| | |
|---|---|
| **Name** | Varying Analysis Granularity |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to build an eExperiment, which is flexible with regard to the depth of the analysis of data. |
| **Problem** | How to support the scientists in spontaneously analysing data in more detail? |
| **Context** | Scientists that require analysis of data on different levels. There are not enough resources to analyse all the data on the finest level. |
| **Solution** | Different approaches are possible here. The *experiment steering* pattern is used, so the scientists can determine the level of granularity during runtime. The second possibility is to use the *data refinement* pattern. Here, the analysis granularity is determined by the system automatically. The system checks for patterns in the data and it adapts the analysis granularity accordingly. |
| **Example** | In [38], dynamic and adaptive workflows for mesoscale meteorology are described, which use varying analysis granularity. |
| **Relationships** | <Uses> Data Refinement<br><Uses> Experiment Steering |

| | |
|---|---|
| **Name** | Experiment Steering |
| **Classification** | Scientific Experiment Model |
| **Intent** | Enables the steering of an eExperiment during runtime by scientists. |
| **Problem** | During runtime scientists want to steer the eExperiment by changing parameters [94]. |
| **Context** | Scientists that want to make changes to the eExperiment during runtime. |
| **Solution** | First of all the eExperiment status needs to be made visible to the scientists. The information about the status of the eExperiment is required, in order |

| | |
|---|---|
| | to decide which parameters to change. In addition, a mechanism that can process external parameters during runtime needs to be implemented. |
| **Example** | Mattoso et al. describe how users can steer experiments during runtime [70]. |
| **Relationships** | <Requires> Monitoring of Experiment Status<br><Uses> Suspend and Resume |

| | |
|---|---|
| **Name** | Monitoring of Experiment Status |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to make the eExperiment status visible for the scientists. |
| **Problem** | How can the scientists monitor the status of the eExperiment? |
| **Context** | Scientists, that want to get information about the status of their experiment during runtime. This is especially important for long running experiments. |
| **Solution** | Implement a mechanism for making the eExperiment status visible. This can be done by a textual status update or by visualization of the status. |
| **Example** | Activities in the workflow model are inked according to their execution state [104]. |
| **Relationships** | <Uses> Event-based Monitoring<br><Uses> Visual Monitoring |

| | |
|---|---|
| **Name** | Suspend and Resume |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to make the eExperiment more dynamic. |
| **Problem** | During runtime, the scientists decide to make changes on the eExperiment. This is especially important for long running experiments, where the likelihood of changes is higher. Therefore, they have to stop the eExperiment and apply the changes. After that, the eExperiment is run again from the start, although the changes influenced only parts of the eExperiment. |
| **Context** | Scientists that want to make changes to an eExperiment after it was started. |
| **Solution** | Implement a mechanism, that stores intermediate results and makes it possible to start the modified eExperiment model from intermediate results. |
| **Example** | Karastoyanova et al. describe the spontaneously re-execution of activities in workflows within e-Science [107].<br>Concurrent workflow evolution is introduced by Sonntag et al. in [106]. |
| **Relationships** | - |

#### One-Factor-at-a-Time Approach

| | |
|---|---|
| **Name** | One-Factor-at-a-Time |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to understand dependencies between input parameters and outputs by trying all parameter configurations. |
| **Problem** | Several input parameters can be modified. Which configuration of the input parameters leads to the best result? |

| | |
|---|---|
| **Context** | The parameters for an experiment can be controlled by the scientists. |
| **Solution** | Only one input parameter is changed at a time. Then the next input parameter is changed. By trying all the combinations, one configuration of the input parameters will lead to the best result [40]. This can be automated, so the scientists do not have to change the parameters manually. For defining the range of input parameters the *application plug-ins* pattern can be used. |
| **Example** | - |
| **Relationships** | <Uses> Application Plug-ins |

#### Design of Experiments Approach

| | |
|---|---|
| **Name** | Design of Experiments |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to understand dependencies between input parameters and outputs with minimal effort. |
| **Problem** | Several input parameters can be modified. How to identify an approximation for the best configuration of input parameters with less resources? |
| **Context** | The parameters for an experiment can be controlled by the scientists. |
| **Solution** | Define different levels for the input parameters. Now the input parameters cannot take any value, but only the values which are defined by the levels. This reduces the overall amount of possible input configurations compared to OFAT. However, since the input parameters do not take all possible values, the best configuration might not be determined. The design of experiments approach delivers an approximation of the best configuration of input parameters. For defining the range of input parameters the *application plug-ins* pattern can be helpful. |
| **Example** | *Full factorial design:* All possible configurations of the different input parameters with their levels are tested [40].<br>*Fractional factorial design:* Only a subset of all levels of input parameters is used [40]. |
| **Relationships** | <Uses> Application Plug-ins |

#### Further User Characteristics

| | |
|---|---|
| **Name** | Modelling Choices |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to create a modelling environment, that fits the needs of the researchers. |
| **Problem** | Different researchers prefer different ways of modelling. Some researchers are used to model the experiment on a higher level. For instance biologists are used to high-level Web portals [94]. Scientists from the domain of physics are also interested in the low-level computational techniques. In addition, the physicists typically also want to understand the low level details of high-end computers which they use for their research [94]. |

| | |
|---|---|
| **Context** | An eExperiment that involves different scientists which prefer different ways of modelling. |
| **Solution** | Provide a modelling environment, which enables different ways for modelling the same experiment. |
| **Example** | In [28], workflows can be modelled on a high-level in a first step. The high-level representation is independent from the actual execution environment. Later on, once the execution environment is defined, the high-level representation can be refined. The refinement goes down to an executable workflow, which also contains all necessary information about the resources. Dependent on the skills of the scientists, they can choose their level of modelling. The mapping to the lower level is either done in a (semi-) automated way or by experts in this area [28]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Experiment Interface |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to choose a suitable interface to the eExperiment. |
| **Problem** | Some researchers prefer a specific way of interacting with IT systems. E.g. some physicians prefer commando line based tools and some biologists prefer high-level Web portals [94]. |
| **Context** | Scientists that require an interface to the eExperiment. |
| **Solution** | Investigate which kind of interface the different scientists usually work with and prefer. |
| **Example** | Some biologist prefer high-level Web portals [94]. |
| **Relationships** | <Uses> Command Line Tools<br><Uses> Graphical User Interface |

### 4.3.2. Data

This section presents the *data* patterns on the level *scientific experiment model*. Figure 16 provides an overview of these patterns.

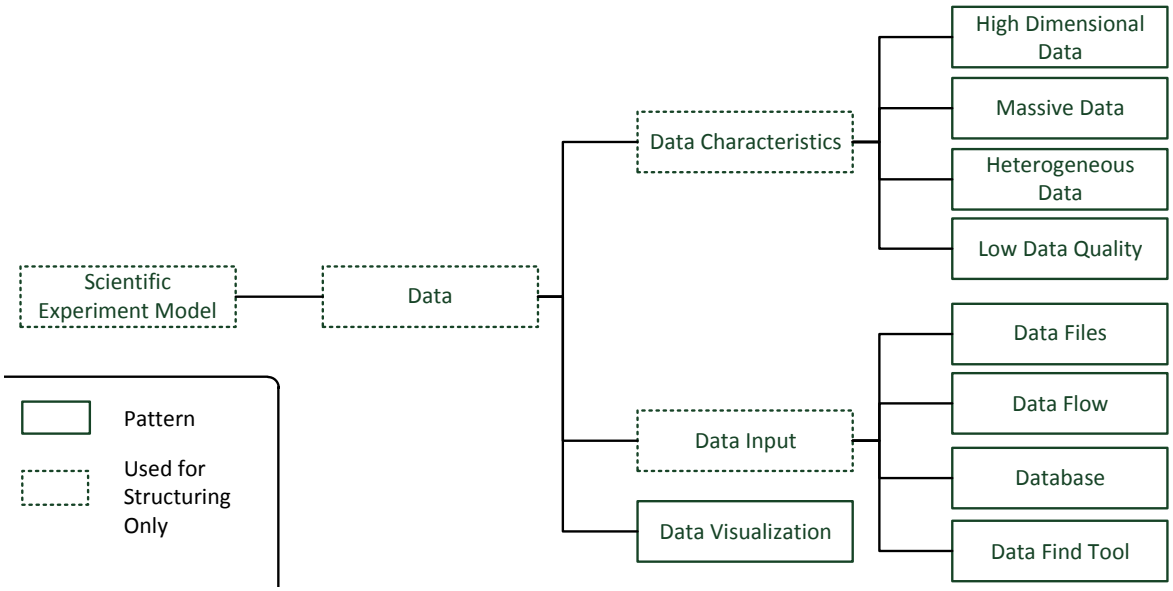


**Figure 16.:** This is an overview of the e-Science data patterns within the level *scientific experiment model*.

#### Data Characteristics Pattern

| | |
|---|---|
| **Name** | High Dimensional Data |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to handle high dimensional data. |
| **Problem** | Data has many dimensions and requires massive resources in order to be analysed [96]. |
| **Context** | Analysis of high dimensional data within e-Science. |
| **Solution** | Use algorithms to reduce the dimensions of the data and transform the data in a format, which eases the data processing. |
| **Example** | In [96], Ruan et al. introduce Multidimensional Scaling (MDS) to reduce the dimensions of the data. |
| **Relationships** | <Uses> Reduce Dimensions |

| | |
|---|---|
| **Name** | Massive Data |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to handle massive data. |
| **Problem** | The analysis of massive data on one machine takes too much time. Additionally, the result set can be very huge and this makes it hard to analyse for the |

| | |
|---|---|
| | scientists. |
| **Context** | Scientists want to analyse massive data within e-Science. |
| **Solution** | The work is distributed to different machines, to reduce the overall data processing time. In order to analyse the huge result set, the result data is visualized. Also check, if the pattern *varying analysis granularity* or *data refinement* can be applied. |
| **Example** | In [38] massive data is analysed. |
| **Relationships** | <Uses> Data Visualization<br><Uses> Master-Worker<br><Uses> MapReduce<br><Uses> Data Refinement<br><Uses> Varying Analysis Granularity |

| | |
|---|---|
| **Name** | Heterogeneous Data |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to handle data that comes from various sources in various formats. |
| **Problem** | In e-Science researches reuse data from other researchers. The data format was defined from the creator of the data. So the data format fits for the original purpose. If scientists reuse data with many different formats, this requires intelligent data transformation for minimizing the (manual) effort. |
| **Context** | Non standardized data is reused for an eExperiment. The data sets have many different formats. |
| **Solution** | Use the pattern *data transformation* and/or the pattern *semantic mediation*. |
| **Example** | Heterogeneous data is analysed in [75]. |
| **Relationships** | <Uses> Data Transformation<br><Uses> Semantic Mediation |

| | |
|---|---|
| **Name** | Low Data Quality |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to improve the data quality. |
| **Problem** | In e-Science researches reuse data from other researchers. Sometimes the data quality is not sufficient for the use. The data needs to be checked, if the quality is sufficient. In addition, many computations contain sequences of format translations and therefore, intermediate results can be incorrect need to be checked as well [123]. |
| **Context** | There is a high chance that data has low quality and this data is reused for an eExperiment. Intermediate results are incorrect. |
| **Solution** | Implement a mechanism that can identify the low data quality. After this, a mechanism to improve the data quality needs to be implemented. |
| **Example** | - |
| **Relationships** | <Requires> Data Quality Gate<br><Requires> Data Quality Improvement |

**Data Input**

| | |
|---|---|
| **Name** | Data Files |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to integrate data into the eExperiment |
| **Problem** | How to integrate data into the eExperiment? |
| **Context** | Scientists that want to integrate data into an eExperiment. |
| **Solution** | The data is stored in data files, which are analysed by the e-Science solution. |
| **Example** | Scientists upload data files via a Web form to an e-Science application in [75]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Data Flow |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to integrate data into the eExperiment |
| **Problem** | How to integrate data into the eExperiment? |
| **Context** | An eExperiment that involves *always on* scientific instruments that produce a continuous data flow [127]. |
| **Solution** | Use the *data refinement* or the *instrument control pattern*. When there is continuous data flow involved, the specifics for updating the experiment model need to be considered as well. In this case check, if the *model update strategies* pattern can be applied. |
| **Example** | Brooke et al. describe a continuous data flow workflow application in [18]. |
| **Relationships** | <Uses> Data Refinement<br><Uses> Instrument Control<br><Uses> Model Update Strategies |

| | |
|---|---|
| **Name** | Database |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to integrate data into the eExperiment |
| **Problem** | How to integrate data into the eExperiment? |
| **Context** | Scientists that want to use data from databases within their eExperiment. |
| **Solution** | Establish connection to databases and retrieve data. |
| **Example** | In [25], biologists use databases containing observation data. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Data Find Tool |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps the scientists to find appropriate data sets. |
| **Problem** | Scientists have access to many different data sets. Searching over the file names is not sufficient for finding the appropriate data set. |
| **Context** | Scientists, that have access to many different data sets. |
| **Solution** | Provide a data find tool, which enables the search over meta-data attributes. |

| | |
|---|---|
| **Example** | In [19], scientists have access to gravitational wave data. This data comes from different gravitational wave data detectors. In this case meta-data attributes like start and end time of the gravitational wave data recording are available. The scientists can also search over this meta-data attributes. |
| **Relationships** | <Requires> Resource Catalogue<br><Requires> Resource Discovery |

#### Data Visualization

| | |
|---|---|
| **Name** | Data Visualization |
| **Classification** | Scientific Experiment Model |
| **Intent** | Supports the scientist in understanding data. |
| **Problem** | Massive data leads to many intermediate and to many end results. If there is too much data, it is hard to understand the data without a visualization. |
| **Context** | Scientists that analyse massive data. |
| **Solution** | Provide a visualization for the data. |
| **Example** | Newman et al. provide a visualization for scientists in the domain of animal tracking [75]. The visualization e.g. shows location data for one or more animals. |
| **Relationships** | - |

### 4.3.3. Improve Collaboration

This section comprises the *improve collaboration* patterns. Figure 17 gives an overview of the different patterns.

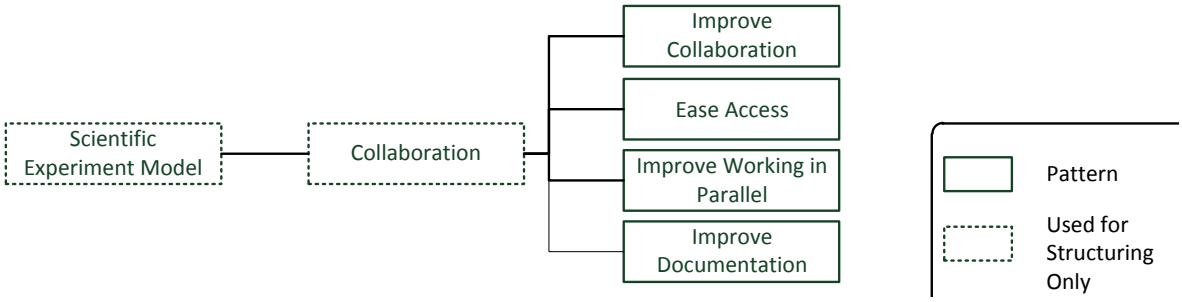


**Figure 17.:** This is an overview of the e-Science improve collaboration patterns.

| | |
|---|---|
| **Name** | Improve Collaboration |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to improve the communication between different scientists that work in an e-Science project. |
| **Problem** | Different scientists work together in an e-Science project. How to make the eExperiment comprehensible for all scientists and how to update them with the latest status of the eExperiment? |

| | |
|---|---|
| **Context** | An eExperiment that involves different scientists. |
| **Solution** | Provide the scientists with a communication platform, where they can discuss the latest status of the eExperiment or where they can put intermediate results. Furthermore the involved scientists should know, which other scientists require which data sets at which point in time. In addition, the *ontologies* pattern should be considered in such research projects. |
| **Example** | Wallis et al. analyse the life cycle of e-Science collaboratory data in [124]. They identify a life cycle of data, that helps to understand the whole data flow within the collaboration.<br>In [8], Belloum et al. introduce the virtual laboratory for e-Science. This framework also improves the sharing across different scientific domains. |
| **Relationships** | <Uses> Ontologies<br><Uses> Communication Platform<br><Related To> Ease Access |

| | |
|---|---|
| **Name** | Ease Access |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to support the different scientists and stakeholders of a research project to access results, intermediate results and IT systems. |
| **Problem** | Scientists work together in an inter disciplinary research project. Scientists and stakeholders have trouble to access results and intermediate results at the right point in time. Firstly, the results are spread across different machines and in addition access rights to the machines are difficult to get [109, 124]. In addition, people fluctuate and other researchers do not have the necessary rights to access the systems and cannot continue the research [109]. |
| **Context** | An e-Science project that involves different IT systems. Additionally, it is expected that scientists join the project and leave the project. |
| **Solution** | Identify the life cycle of data. Then identify which persons need to have access to which data sets. Finally, suitable access rights can be granted. Use the *role based access* pattern in combination with the *single sign on* pattern. |
| **Example** | In an international cancer research project, role based access was implemented [109]. |
| **Relationships** | <Uses> Single Sign-On<br><Uses> Role Based Access<br><Related To> Resource Catalogue |

| | |
|---|---|
| **Name** | Improve Working in Parallel |
| **Classification** | Scientific Experiment Model |
| **Intent** | Supports scientist to work in parallel on one eExperiment. |
| **Problem** | Scientists work on one eExperiment. They want to work in parallel on things like eExperiment model or IT model. This needs to be supported. |
| **Context** | An eExperiment that involves different scientists, that want to work in parallel. |

| | |
|---|---|
| **Solution** | Consider the required parallel working, when designing an e-Science solution. |
| **Example** | If scientists want to work in parallel on a model, this could be realized with different model versions. In [106], the authors introduce a concept for migrating a running workflow instance to a new model. This is not an example for a full solution to the parallel working problem, but it can be used as a starting point. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Improve Documentation |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to improve the documentation and therefore, the reproducibility of results. |
| **Problem** | Results of eExperiments are published, but not the different models and data sets, that were used to achieve the results. Other scientists cannot easily reproduce the results. |
| **Context** | Scientists, that want to improve the documentation of the eExperiment. |
| **Solution** | Implement a mechanism to document the eExperiment on different levels. This involves e.g. the storing of meta-data, intermediate results, end results and the used calculations. |
| **Example** | In [125], a provenance-aware geographic information system is described. |
| **Relationships** | <Uses> Data Provenance |

### 4.3.4. Non-functional Properties

Figure 18 displays the *non-functional properties* patterns, which are presented in this section.

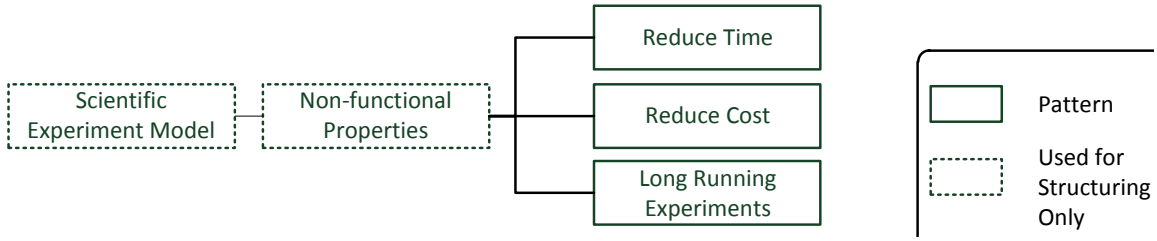


**Figure 18.:** This is an overview of the e-Science non-functional properties patterns.

| | |
|---|---|
| **Name** | Reduce Time |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to reduce the overall time span of an eExperiment. |
| **Problem** | Scientists want to shorten the time to new knowledge. |

| | |
|---|---|
| **Context** | Scientists that want to conduct an eExperiment. |
| **Solution** | Use *design of experiments* to reduce the amount of iterations of the eExperiment. In addition, a parallel data processing pattern can help to reduce the processing time. |
| **Example** | Parallelism is exploited to reduce processing time in [9]. |
| **Relationships** | <Uses> Design of Experiments<br><Uses> Master-Worker<br><Uses> MapReduce<br><Uses> Single Program Multiple Data |

| | |
|---|---|
| **Name** | Reduce Cost |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to reduce the overall cost of an eExperiment. |
| **Problem** | The cost is too high or budget is too low. |
| **Context** | Scientists that want to conduct an eExperiment. |
| **Solution** | Use *design of experiments* to reduce the amount of iterations of the eExperiment. In addition, use the *infrastructure as a service* pattern to reduce cost for the infrastructure. |
| **Example** | Yuan et al. develop a cost-effective datasets storage strategy in [129]. |
| **Relationships** | <Uses> Design of Experiments<br><Uses> Infrastructure as a Service |

| | |
|---|---|
| **Name** | Long Running Experiments |
| **Classification** | Scientific Experiment Model |
| **Intent** | Helps to design a stable long running eExperiment |
| **Problem** | Software as well as hardware might fail. With increasing runtime of an eExperiment, there is a higher chance of a failure and there is a higher chance, that the scientist wants to make changes on the eExperiment. |
| **Context** | Scientists that want to conduct a long running eExperiment. |
| **Solution** | Use the patterns *partition-level failure recovery*. If a failure occurs, only the current partition has to be restarted. For increasing the flexibility of the eExperiment to changes by scientists use the *suspend and resume* pattern. |
| **Example** | Long running workflows are introduced in [28]. |
| **Relationships** | <Requires> Partition-Level Failure Recovery<br><Requires> Suspend and Resume<br><Uses> Static Partitioning |

## 4.4. IT Experiment Model

Figure 19 depicts the structure of the e-Science patterns on the level *IT experiment model*, which are introduced in this section.

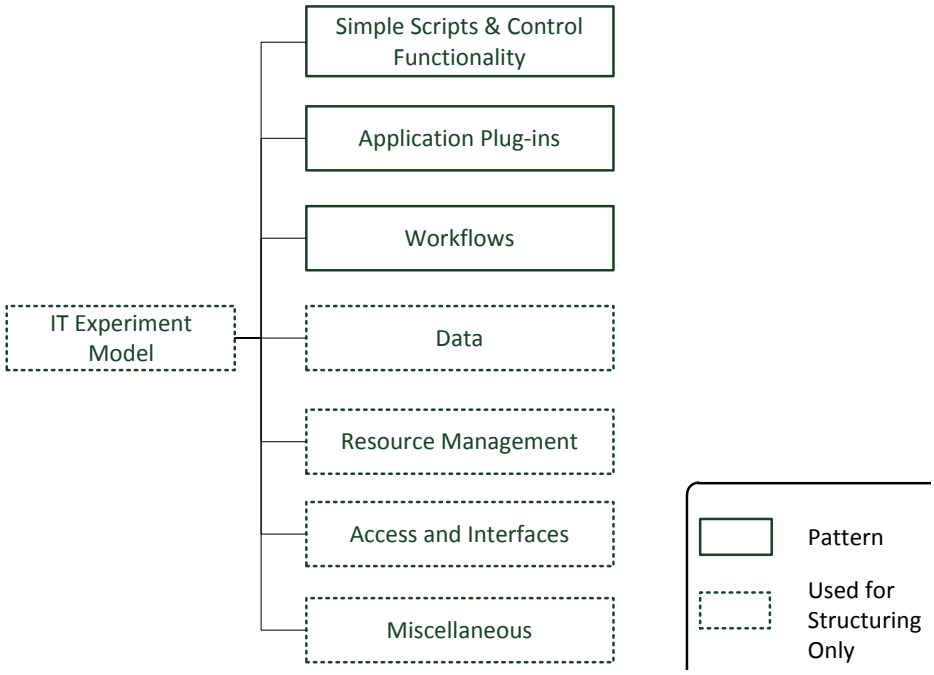


**Figure 19.:** This shows the different groups of patterns within the e-Science IT experiment model patterns.

### 4.4.1. Simple Scripts & Control Functionality

| | |
|---|---|
| **Name** | Simple Scripts & Control Functionality |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to perform e-Science with little programming. |
| **Problem** | How to perform e-Science with little programming for the end-user? |
| **Context** | Scientists, that require automation for their eExperiment. |
| **Solution** | Scientists have typically software code, e.g. C/C++ or Fortran code, that they run on computational resources [94]. In order to control the execution of these codes, they use simple scripts that provide control functionalities. These control functionalities are for example *If-Then-Else*, *Do-N* or *Do Repeat*. |
| **Example** | In [94], an e-Science application is described, that uses the *Do-N* functionality. The output of the previous job is used as input to the subsequent job. |
| **Relationships** | - |

### 4.4.2. Application Plug-ins

| | |
|---|---|
| **Name** | Application Plug-ins |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to perform e-Science |
| **Problem** | How to set parameters for an eExperiment without programming for the end-user? |
| **Context** | Scientists that want to create an eExperiment. These scientists want to control the parameters but they do not want to write any scripts or code. |
| **Solution** | In [94], Riedel et al. define application plug-ins, that make it easy to e.g. submit jobs to Grid infrastructures. In addition, job specific options or the configuration options of the scientific applications can be set via a convenient Graphical User Interface (GUI). For instance controls like check-boxes or lists are used to selected the number of nodes and processors for a job. |
| **Example** | An example is the GridSphere Web portal [77]. |
| **Relationships** | <Uses> Graphical User Interface |

### 4.4.3. Workflows

This section introduces the *workflow* patterns, which are also shown in Figure 20.

| | |
|---|---|
| **Name** | Workflows |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to perform e-Science |
| **Problem** | How to achieve a high automation of an eExperiment, which requires data management and computational tasks in conjunction with several numerous control functionalities [94]? |
| **Context** | Scientists that want to create an eExperiment with a high degree of automation. |
| **Solution** | Use workflows [94]. |
| **Example** | Quantitative structure-activity relationships (QSAR) workflows are used in the domain of healthcare [94]. |
| **Relationships** | - |

#### Modelling

| | |
|---|---|
| **Name** | Abstract Workflows |
| **Classification** | IT Experiment Model |
| **Intent** | Helps the end user to model a workflow on a high level without the need of programming. |
| **Problem** | How to effectively create workflows without programming? |
| **Context** | Scientists want to create workflows for an eExperiment. |
| **Solution** | Oinn et al. point out, that in such a case, a useful high-level representation of the workflows to the scientists is required. The scientists should be able |

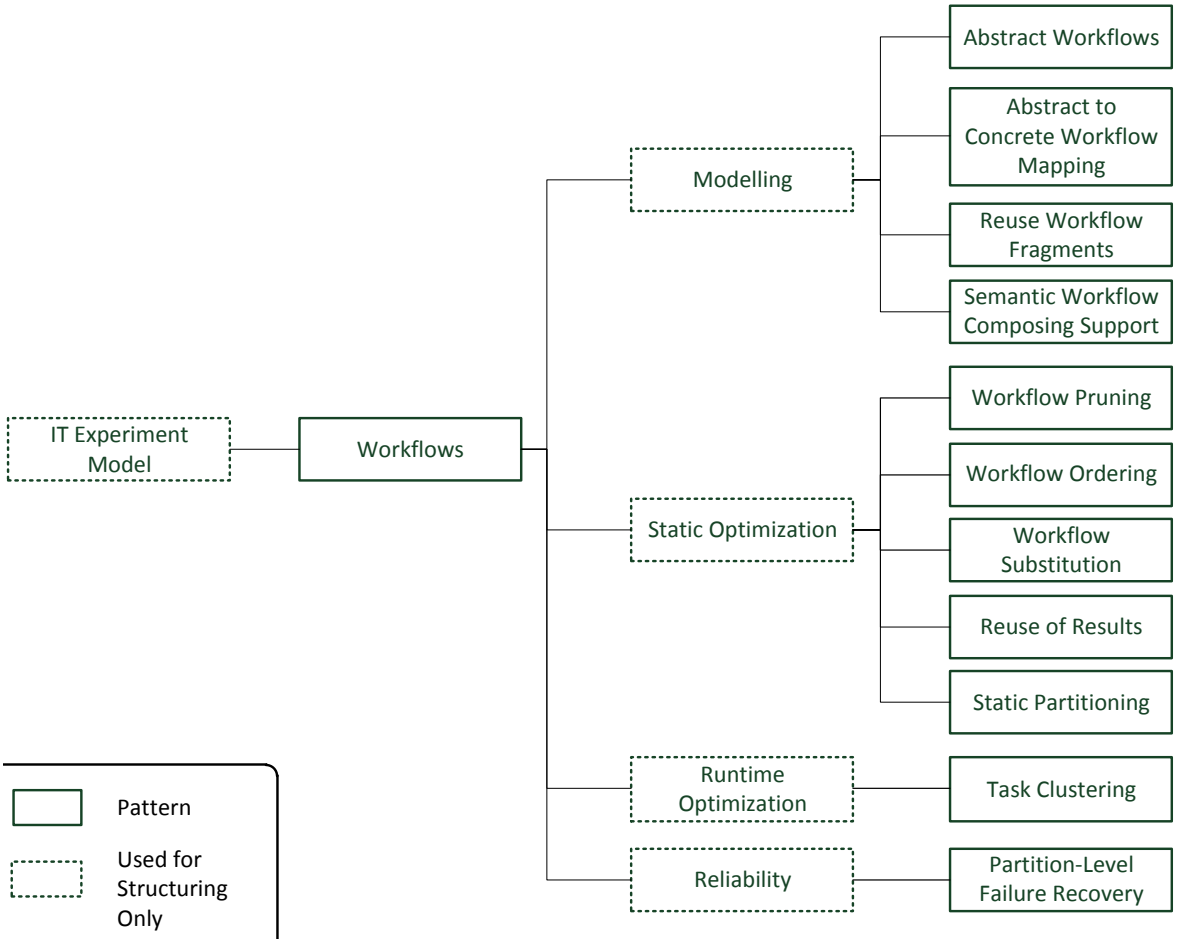


**Figure 20.:** This is an overview of the e-Science workflow patterns.

to coordinate a variety of resources. Especially, it is important to present the workflows from a problem-oriented view. The scientists are used to think in terms of the data, which is consumed and produced. According to this data, they want to connect the workflow components [81].

**Example** In [81], the architecture of a workflow system in the domain of life science is presented. For the end-user abstraction level the Simple Conceptual Unified Flow Language (SCUFL) is used. SCUFL is a workflow language, that links applications from a data flow point of view. It defines a graph, that reflects the data interactions between different components [80]. The scientists also have to choose, which analysis they want to do on the selected data. Rather than using services directly for the analysis, Oinn et al. added another layer, which provides shallow descriptions of the services [81]. This helps to hide the low level details from the scientists.

**Relationships** <Requires> Abstract To Concrete Workflow Mapping
<Related To> Reuse Workflow Fragments

| | |
|---|---|
| **Name** | Abstract to Concrete Workflow Mapping |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to transform abstract workflows to executable workflows. |
| **Problem** | High level workflows cannot be executed. They need to be mapped to executable workflows. |
| **Context** | Scientists that model abstract workflows within e-Science. |
| **Solution** | Implement a mapping from abstract to concrete workflows. |
| **Example** | Oinn et al. use SCUFL at the end-user abstraction level [81]. SCUFL has an extensible processor plug-in architecture. In these processors, the concrete mappings to the lower level are implemented. For instance, a processor can use a Java Database Connectivity (JDBC) connection to access predefined queries over a relational database [81]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Reuse Workflow Fragments |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to fasten the creation of workflows. |
| **Problem** | How to fasten the creation of workflows? |
| **Context** | Scientists that want to create a workflow. |
| **Solution** | Provide workflow fragments to the scientists, so they can reuse them. |
| **Example** | Hategan et al. describe a workflow component repository, which enables the reuse of components. These components can either be maintained by a community of researchers or by a single researcher [64]. |
| **Relationships** | <Uses> Resource Catalogue<br><Uses> Resource Discovery |

| | |
|---|---|
| **Name** | Semantic Workflow Composing Support |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to reduce the manual effort for the scientists when composing different services into a workflow. |
| **Problem** | How to reduce the manual effort for the scientists when composing different services into a workflow? |
| **Context** | Scientists that want to create a workflow by composing services, that require semantically and structurally different data sets [16]. |
| **Solution** | Bowers et al. propose to annotate the input and output of services with a structural type, a semantic type and a registration mapping. Allowable data values for output and input are described in the structural type. The semantic type contains conceptual information of input and output. Properties and concepts of an ontology express the semantic input and output. The registration mapping links the structural types with the semantic types [16]. |
| **Example** | In [16], a framework that provides semantic workflow composing support is presented. |
| **Relationships** | <Uses> Semantic Mediation<br><Requires> Ontologies |

**Static Optimization**

| | |
|---|---|
| **Name** | Workflow Pruning |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to preoptimize the runtime execution. |
| **Problem** | How to shorten the execution time of a workflow? |
| **Context** | A workflow that contains elements which do not contribute to the final result. |
| **Solution** | Implement a mechanism, that detects components of a workflow, which do not contribute to the final result [72]. This can be e.g. redundant components within a workflow. |
| **Example** | - |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Workflow Ordering |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to preoptimize the runtime execution. |
| **Problem** | How to shorten the execution time of a workflow? |
| **Context** | A workflow, in which the order of tasks is not optimized yet. |
| **Solution** | Implement a mechanism, that automatically reorders the components of a workflow in order to improve efficiency [72]. |
| **Example** | - |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Workflow Substitution |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to preoptimize the runtime execution. |
| **Problem** | How to shorten the execution time of a workflow? |
| **Context** | A workflow that contains inefficient tasks. |
| **Solution** | Implement a mechanism, that can detect inefficient components within a workflow. These inefficient components can be replaced by more efficient components. In addition, some computations can be replaced by results of other eExperiments, that contain the same calculation. |
| **Example** | A finite-element solver gets a sparse matrix as input, which is diagonally dominant. The basic implementation uses a Jacobi Solver. With the help of workflow substitution, the Jacobi Solver is replaced by a Conjugate Gradient Solver that it is more efficient for such a matrix [72]. |
| **Relationships** | <Uses> Reuse of Results |

| | |
|---|---|
| **Name** | Reuse of Results |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to preoptimize the runtime execution by reusing result that were previously calculated. |
| **Problem** | How to shorten the runtime of the workflow execution? |

| | |
|---|---|
| **Context** | A workflow that produces data sets. Data products from other eExperiments are available and can be reused. |
| **Solution** | Determine, which already available data products can be reused in the workflow, so that they do not have to be recalculated. |
| **Example** | Deelman et al. describe the Pegasus system, which replaces calculation steps with data products from previously executed workflows [28]. First of all, a data catalogue is required. It stores results and intermediate results of other eExperiments. Before the workflow is executed, Pegasus consults the data catalogue in order to determine the already available data products. If data products are available, they replace the corresponding components in the workflow. If the final result is already available, the workflow is not executed at all [28]. |
| **Relationships** | <Related To> Data Provenance<br><Related To> Resource Catalogue |

| | |
|---|---|
| **Name** | Static Partitioning |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to reduce the overall runtime of a workflow in a dynamic environment. |
| **Problem** | In dynamic execution environments it is hard to plan which resources are available in the future. In the future, the resources might be overloaded or cannot be accessed anymore [28]. |
| **Context** | Long running workflows in a dynamic execution environment. |
| **Solution** | The workflow is partitioned before execution. This is called static partitioning. For each partition, the WfMS is called and maps the portions inside the partition at a time to resources. This is also called *deferred mapping* [28]. |
| **Example** | - |
| **Relationships** | - |

#### Runtime Optimization

| | |
|---|---|
| **Name** | Task Clustering |
| **Classification** | IT Experiment Model |
| **Intent** | Reduce system overheads for small task computation and optimize the overall workflow execution time. |
| **Problem** | Some workflows contain very fine-grained tasks. The scheduling mechanism needs more resources than the computation of the small single task. |
| **Context** | Workflow that contains many fine computational granularity tasks. |
| **Solution** | Merging multiple small tasks into a single job. This reduces the scheduling overhead [22]. |
| **Example** | Chen et al. describe different methods for balanced task clustering for scientific workflows [22]. |
| **Relationships** | - |

**Reliability**

| | |
|---|---|
| **Name** | Partition-Level Failure Recovery |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to minimize the overall runtime, because when a failure occurs not the whole workflow needs to be restarted. |
| **Problem** | How to handle long running workflows that can fail? |
| **Context** | A long running workflow that might fail. |
| **Solution** | Partition the workflow. If the tasks within a partition are executed, the parameters are saved. Also the intermediate results that belong to this partition need to be stored. If a failure happens inside the partition, only the partition has to be restarted and not the whole workflow [28]. |
| **Example** | In [28], Deelman et al. implement a partition-level failure recovery in Pegasus. |
| **Relationships** | - |

### 4.4.4. Data

This section comprises the *data* patterns on the level *IT experiment model*. They are shown in Figure 21.

**Data Quality**

| | |
|---|---|
| **Name** | Data Quality Gate |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to detect data quality issues in the used data. |
| **Problem** | How to determine, if the data quality is sufficient for the eExperiment? |
| **Context** | Data quality issues within an eExperiment. |
| **Solution** | Implement a mechanism, that checks the data quality and makes low data quality visible. |
| **Example** | In [25], several algorithms are used to detect data quality issues in biological observation databases. |
| **Relationships** | <Related To> Data Quality Improvement |

| | |
|---|---|
| **Name** | Data Quality Improvement |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to improve the quality of the used data. |
| **Problem** | Data quality gates detect data quality issues. The data quality needs to be improved in order to achieve reliable scientific conclusions [25]. |
| **Context** | Data quality issues within an eExperiment. |
| **Solution** | Here several solutions are possible. Ideally, the quality of the source data can be improved. In e-Science this can be hard, since data from other researchers is reused. If the source data quality is sufficient and the intermediate results are not correct, the data quality can be improved by changing the computation in between. |

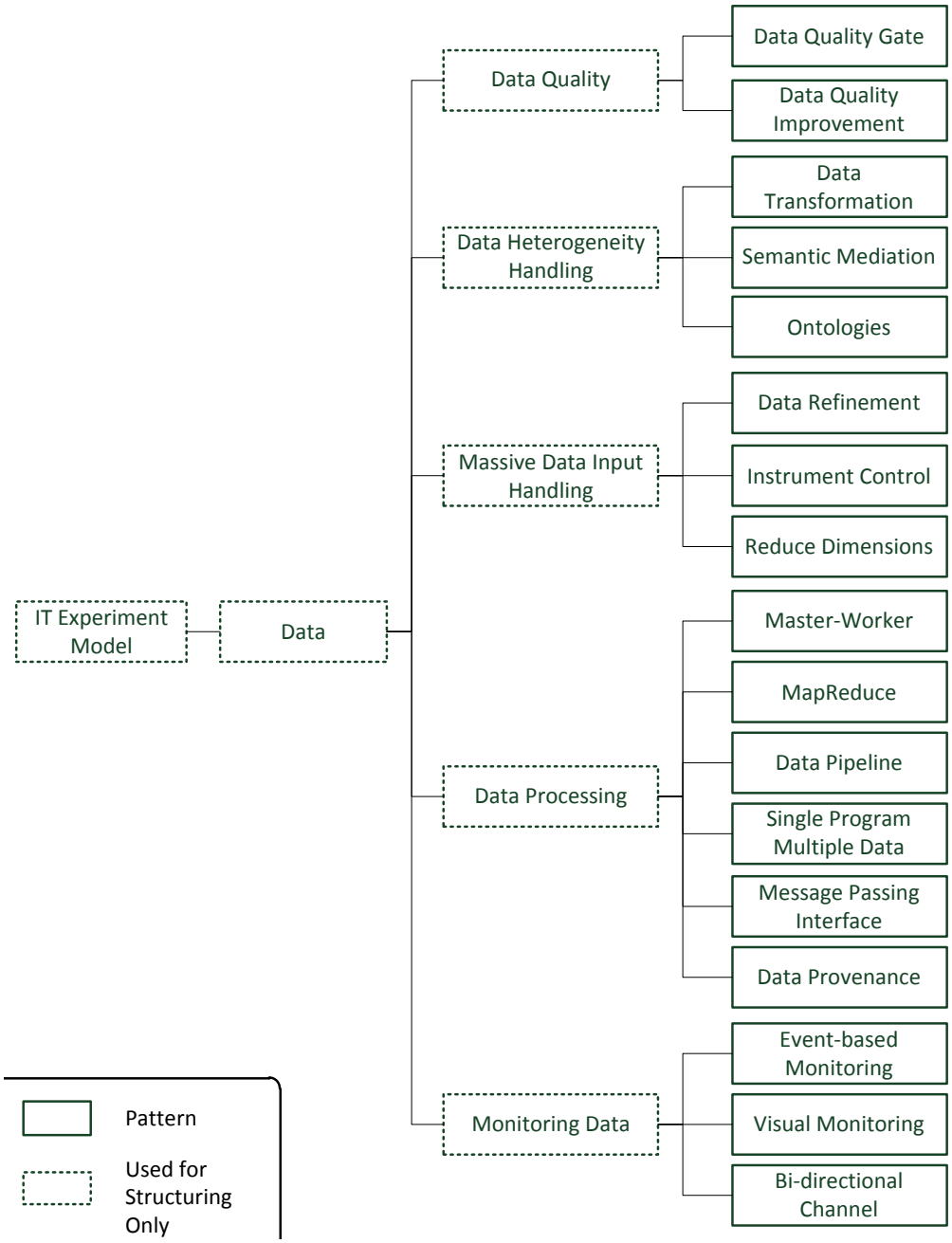


**Figure 21.:** This is an overview of the e-Science data patterns within the level *IT experiment model*.

| | |
|---|---|
| **Example** | In [25], Cugler et al. introduce a geographical approach for improvement of meta-data quality in biological observation databases. |
| **Relationships** | <Uses> Data Quality Gate |

#### Data Heterogeneity Handling

| | |
|---|---|
| **Name** | Data Transformation |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to reduce the manual effort for data transformation. |
| **Problem** | Scientists in e-Science use several data input formats and they apply several calculation steps on this data. A lot of work has to be undertaken, to adapt the calculation logic for every input format and/or transform the data manually into suitable formats. It gets even worse, when the adaptation is done manually. |
| **Context** | Scientists, that apply analysis steps to heterogeneous data. |
| **Solution** | A sophisticated solution is the use of the *semantic mediation* pattern. Sometimes, the definition of an intermediate format might be sufficient. Map all input data to this format first and then do the analysis based on the intermediate format. |
| **Example** | Deelman et al. use an intermediate format for transforming mosaics in the domain of astronomy [9]. |
| **Relationships** | <Uses> Semantic Mediation |

| | |
|---|---|
| **Name** | Semantic Mediation |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to reduce the manual effort for integrating heterogeneous data into the eExperiment. |
| **Problem** | Scientists in e-Science reuse heterogeneous data from other researchers. Before they can use the data, they need to put lots of effort into the integration and synthesizing of the data [16]. |
| **Context** | Scientists, that reuse heterogeneous data within e-Science. |
| **Solution** | Annotate the data with semantic types. Properties and concepts of an ontology express the semantic types. Based on these semantic types, the data can be transformed between heterogeneous local schemas into a global schema [16]. |
| **Example** | In [16], an ontology-driven framework for data transformation in scientific workflows is presented. |
| **Relationships** | <Requires> Ontologies |

| | |
|---|---|
| **Name** | Ontologies |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to formalize knowledge as a hierarchy of concepts within a domain. |

| | |
|---|---|
| **Problem** | How to formalize knowledge? |
| **Context** | Scientists that require a higher degree of automation for data transformations within their eExperiment. |
| **Solution** | Create an ontology for the domain. Studer et al. define an ontology as "a formal, explicit specification of a shared conceptualization" [114]. A *conceptualization* is an abstract model of the relevant concepts of a phenomenon in the world. The types of concepts that are used and the usage constraints have to be *explicitly* defined. In addition, the ontology has to be formal, which means machine readable [114]. |
| **Example** | Bowers et al. adopt the Web Ontology Language (OWL) for expressing ontologies within the SEEK framework [16].<br>Another example is Biological PAthways eXchange (BiPAX)[1], that is based on the Web Ontology Language Description Logic (OWL DL). |
| **Relationships** | - |

#### Massive Data Input Handling

| | |
|---|---|
| **Name** | Data Refinement |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to analyse data in depth during runtime based on external events. |
| **Problem** | There is massive data that should be analysed. For reasons like cost or resource limitations, not all the data can be analysed on the most granular level. |
| **Context** | A massive data stream should be analysed |
| **Solution** | The data is analysed on a higher level at the beginning. If interesting patterns are detected inside the data stream, the interesting portion of the stream gets analysed in more detail. Therefore, additional resources need to be provisioned and/or new analysis logic components need to be started. |
| **Example** | In [38], dynamic and adaptive workflows for mesoscale meteorology are described, which use data refinement. |
| **Relationships** | <Related To> Instrument Control<br><Uses> Infrastructure as a Service |

| | |
|---|---|
| **Name** | Instrument Control |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to detect important information in a data stream that is coming from instruments. The data generation of the instruments is controlled by the results of the data analysis. |
| **Problem** | There is massive data that should be analysed. For reasons like cost or resource limitations not all the data can be analysed in detail. |
| **Context** | Scientists that want to analyse massive data, which is produced by instruments. In addition, the instrument analysis granularity can be adjusted. |

[1] http://www.biopax.org/

| | |
|---|---|
| **Solution** | The data is analysed on a higher level at the beginning. If interesting patterns are detected inside the data stream, feedback is passed to the instruments, which generate the data stream. Because of the feedback the sensors generate more and/or different data. |
| **Example** | In [38], dynamic and adaptive workflows for mesoscale meteorology are described, which are using data refinement, that is based on *instrument control*. |
| **Relationships** | <Uses> Infrastructure as a Service<br><Related To> Data Refinement |

| | |
|---|---|
| **Name** | Reduce Dimensions |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to reduce the dimensions in high dimensional data. |
| **Problem** | The used data has many dimensions and requires massive resources in order to be analysed [96]. |
| **Context** | Scientists that use high dimensional data sets for their eExperiment. |
| **Solution** | Implement techniques to reduce the dimensions so the data can be analysed easier afterwards. |
| **Example** | In [96], Ruan et al. reduce dimensions of the data by using Multidimensional Scaling (MDS) and an interpolation algorithm, that uses a deterministic annealing technique. |
| **Relationships** | - |

#### Data Processing

| | |
|---|---|
| **Name** | Master-Worker |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to speed up the processing of large datasets. |
| **Problem** | How to speed up the processing of large datasets? |
| **Context** | Computation that involves tasks, which can be run concurrently or data that can be processed concurrently. Furthermore, there are no dependencies in the computations and there is no communication between tasks required [44]. |
| **Solution** | Use a master-worker technique. The master splits up the work according to the number of workers. A worker receives its task, processes the task and returns the result to the master [44]. However, if a worker goes down, he stops working [60]. The work that is left undone, is in an unknown stage. Please check, if the *MapReduce* pattern is applicable, that solves the just described issue by a more collaborative effort between master and workers. |
| **Example** | The master-worker technique is used for a signal processing workflow in pulsar astronomy [18]. |
| **Relationships** | <Refined By> MapReduce |

| | |
|---|---|
| **Name** | MapReduce |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to speed up the processing of large datasets. |
| **Problem** | How to speed up the processing of large datasets? |
| **Context** | Within e-Science, scientists want to analyse a very large amount of data sets. |
| **Solution** | According to Fehling et al., the solution can be split into four parts [32]. First, the data set is retrieved. After this, the data set is split into smaller portions, which is called *mapping*. Then, distributed independent components process the small data portions. In the last step, the individual results for the small data chunks are consolidated. This is called *reducing*. In contrast to the *master-worker* pattern, the work in *MapReduce* is done in a more collaborative effort between master and workers [60]. In addition, no portion of work is left unfinished or forgotten. |
| **Example** | Dean et al. give an overview of MapReduce in [27]. |
| **Relationships** | <Uses> Cluster<br><Uses> Data Pipeline<br><Uses> Master-Worker |

| | |
|---|---|
| **Name** | Data Pipeline |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to use MapReduce without the requirement to load the whole data set into the cluster before the analysis starts. |
| **Problem** | Many MapReduce implementations like Hadoop have the restriction, that the entire input dataset needs to be loaded into the cluster, before any analysis can take place [130]. |
| **Context** | Within e-Science, scientists want to analyse a very large amount of data sets. They decide to use MapReduce. |
| **Solution** | Use a data pipeline approach to MapReduce. This approach enables the overlap of data loading and data processing. Therefore, large data sets do not have to be fully loaded into the cluster, before the processing can start. |
| **Example** | Zeng et al. explain the data pipeline approach for MapReduce in [130]. |
| **Relationships** | <Related To> Cluster |

| | |
|---|---|
| **Name** | Single Program Multiple Data |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to organize a parallel program. |
| **Problem** | How to organize a parallel program? |
| **Context** | Within e-Science, scientists want to analyse a very large amount of data sets. |
| **Solution** | According to [71], the Single Program Multiple Data (SPMD) technique is the typical approach for organizing a parallel program. Each node within a parallel computer contains the same single program. However, the copies of the program run independently from each other. |
| **Example** | SPMD is used in [9]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Message Passing Interface |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to write parallel programs. |
| **Problem** | How to write parallel programs? |
| **Context** | Parallel data processing is required within an eExperiment. |
| **Solution** | Use the MPI specification. MPI aims at the message-passing parallel programming model. Data is exchanged in a cooperative manner between processes. To be more exact, one process sends the data explicitly and the other process receives the data explicitly [5]. |
| **Example** | MPI is used for astronomy workflows in [9]. |
| **Relationships** | <Uses> Single Program Multiple Data<br><Uses> Master-Worker<br><Uses> Cluster |

| | |
|---|---|
| **Name** | Data Provenance |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to create a reproducible eExperiment. |
| **Problem** | How to make an eExperiment more reproducible and understandable? |
| **Context** | Scientists that want to increase the reproducibility of their eExperiment. |
| **Solution** | The system should log which data sets are used, which parameters are set by the scientists and which steps are applied in order to derive the result. In addition, persistent identifiers of intermediate results need to be logged as well. Furthermore, the scientists should be able to generate reports with all relevant provenance and runtime information. These reports can be shared with other scientists so that they can understand the eExperiment better. |
| **Example** | Wang et al. implement a provenance aware geographic information system in [125]. Simmhan et al. present a taxonomy of data provenance techniques in [103]. The provenance approaches are categorized with respect to the reason for provenance, what they actually describe and how provenance information is stored and presented. In addition, the provenance dissemination builds another category.<br>Another categorization scheme for provenance can be found in [41]. Here, *provenance model*, *query and manipulation functionality* and *storage model and recording strategy* build the three main categories of the scheme. |
| **Relationships** | - |

#### Monitoring Data

| | |
|---|---|
| **Name** | Event Based Monitoring |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to monitor the execution of workflows. |
| **Problem** | How to monitor the workflow execution? |
| **Context** | Scientists, that want to monitor the execution of a workflow. |

| | |
|---|---|
| **Solution** | Provide the scientists with event-based monitoring of the workflow execution. |
| **Example** | Oinn et al. created an event-based notification for the end user. They implemented a logging mechanism, which captures when for example a workflow activity is started or when an activity fails. All these messages are transformed into XML document format. The results are displayed in a tabular form for the end user [81]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Visual Monitoring |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to monitor the execution of very complex workflows. |
| **Problem** | How to monitor the execution of complex workflows, where an event-based notification is not sufficient? |
| **Context** | Scientists, that want to monitor the execution of a complex workflow. |
| **Solution** | Provide a visualization, that shows the scientists at a look the current status of the execution [81]. |
| **Example** | - |
| **Relationships** | <Uses> Data Visualization<br><Related To> Bi-directional Channel |

| | |
|---|---|
| **Name** | Bi-directional Channel |
| **Classification** | IT Experiment Model |
| **Intent** | Enables the steering of an eExperiment during runtime by scientists. |
| **Problem** | During runtime scientists want to steer the eExperiment by changing parameters [94]. |
| **Context** | Scientists that want to dynamically change parameters during runtime of an eExperiment. |
| **Solution** | Establish a bi-directional channel. Scientific data is transferred to a scientific visualization. In the visualization, the computation status can be observed by the scientists. Then the scientists can set steering parameters. After that the steering parameters are transferred back from the visualization to the simulation. |
| **Example** | Existing frameworks supporting bi-directional channels are *gVID* [59], *e-Viz* [91] and *Collaborative Online Visualization and Steering (COVS)* in the context of Uniform Interface to Computing Resources (UNICORE)[2] [93].<br>In [39], Gibbon et. al implement a bi-directional channel for the Pretty Efficient Parallel Coulomb Solver (PEPC) plasma physics code. |
| **Relationships** | - |

[2] http://www.unicore.eu/

### 4.4.5. Resource Management

In Figure 22 an overview of the *resource management* patterns is given. A resource can for example be data, computing power, workflow fragments, storage or services.

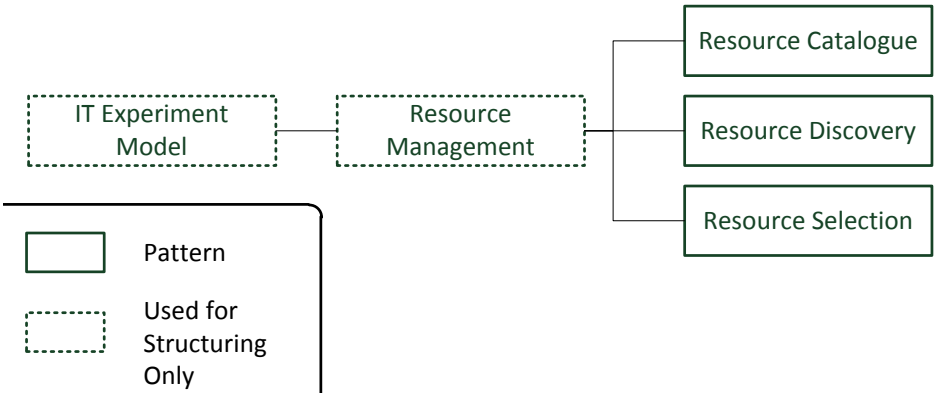


**Figure 22.:** This is an overview of the e-Science resource management patterns.

| | |
|---|---|
| **Name** | Resource Catalogue |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to organize resources. |
| **Problem** | How to organize the resources? |
| **Context** | Scientists that want to use resources. |
| **Solution** | Provide a resource catalogue, where information about resources is stored. Since resources in distributed environments are changing, the resource catalogue cannot be static. It rather has to contain up to date information about the resources. For example, the resource catalogue should contain availability and capability information about computing resources. |
| **Example** | In [69], a site catalogue is used, which describes available computing resources. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Resource Discovery |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to discover resources. |
| **Problem** | How to discover resources? |
| **Context** | Scientists that want to use resources. A resource catalogue that contains information about resources is available. |
| **Solution** | Query the resource catalogue. The resource catalogue is not static, but contains up to date information about services, computing resources, etc. The query can be performed manually by the scientist or in an automated way by |

| | |
|---|---|
| | the IT system. As a result, the resource catalogue delivers a list of suitable resources. |
| **Example** | In [81], a scientist oriented approach for service discovery is presented. The scientists can use their view on services and the domain for searching. The final choice of which service to choose is made by the scientists. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Resource Selection |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to select a suitable resource. |
| **Problem** | Which resources to select? |
| **Context** | A list of resources is available from which one resource has to be selected. |
| **Solution** | The selection can either be done manually by the scientists or with an algorithm by the IT system. The algorithm can select the resource based on different characteristics like availability or the location [28]. |
| **Example** | In [81], a scientist oriented approach for service discovery is presented. The final choice of which service to choose is made by the scientist. |
| **Relationships** | - |

### 4.4.6. Access and Interfaces

Figure 23 shows the *access and interfaces* patterns.

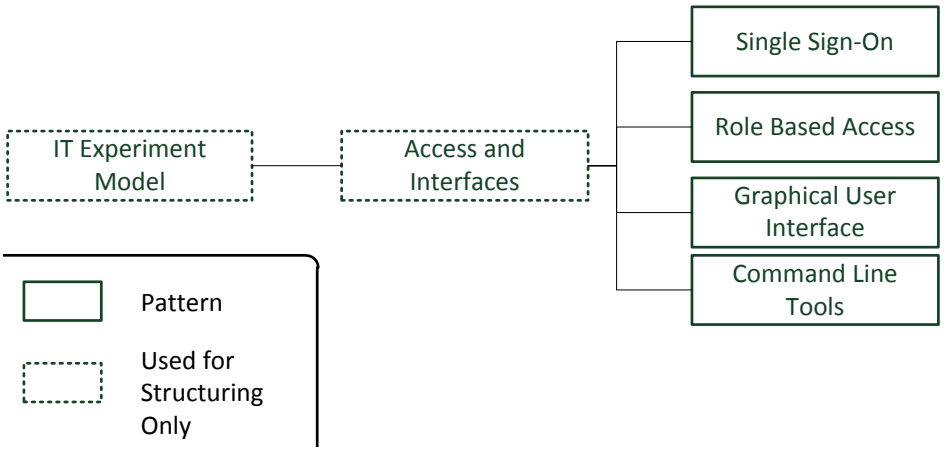


**Figure 23.:** This is an overview of the e-Science access and interfaces patterns.

| | |
|---|---|
| **Name** | Single Sign-On |
| **Classification** | IT Experiment Model |
| **Intent** | Speeds up the research process and increases the acceptance of the eExperiment solution. |

| | |
|---|---|
| **Problem** | Users need to key in a username and a password for every IT system, which they use for the eExperiment. This takes too much time [94]. |
| **Context** | A user wants to access different IT systems within an eExperiment. |
| **Solution** | Implement a mechanism, where the user has to sign on once. After that, this mechanism handles the sign-on to the different IT systems. |
| **Example** | Riedel et al. describe a single sign-on solution, that is based on X.509 certificates of scientists for authorization [94]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Role Based Access |
| **Classification** | IT Experiment Model |
| **Intent** | Ease of user access management. |
| **Problem** | Scientists and stakeholders need to have access to IT Systems. Different people need to have similar access rights within the IT systems. It requires much effort to grant each user every single right. |
| **Context** | Users, who require different access rights for an IT system. There exist several users, which require the same rights. |
| **Solution** | Implement role based access. Rights are granted to roles first. Then a specific user gets the role and therefore the rights, that are granted to the role. |
| **Example** | An example for role based access can be found in [109]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Graphical User Interface |
| **Classification** | IT Experiment Model |
| **Intent** | Helps the scientist to interact with the eExperiment solution. |
| **Problem** | How to interact with the eExperiment solution? |
| **Context** | Scientists that want to perform an eExperiment and are used to GUIs. |
| **Solution** | Provide the scientist with a GUI for conducting the eExperiment.. |
| **Example** | Biologists prefer Web-based portals [94]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Command Line Tools |
| **Classification** | IT Experiment Model |
| **Intent** | Helps the scientist to interact with the eExperiment solution. |
| **Problem** | How to interact with the eExperiment solution? |
| **Context** | Scientists that want to perform an eExperiment and are used to command line tools. |
| **Solution** | Provide the scientists with command line tools for conducting the eExperiment. |
| **Example** | Physicists like command-line interfaces in general and try to avoid GUIs [94]. |
| **Relationships** | - |

### 4.4.7. Miscellaneous

In Figure 24 an overview of the *miscellaneous* patterns on the level *IT experiment model* is given.

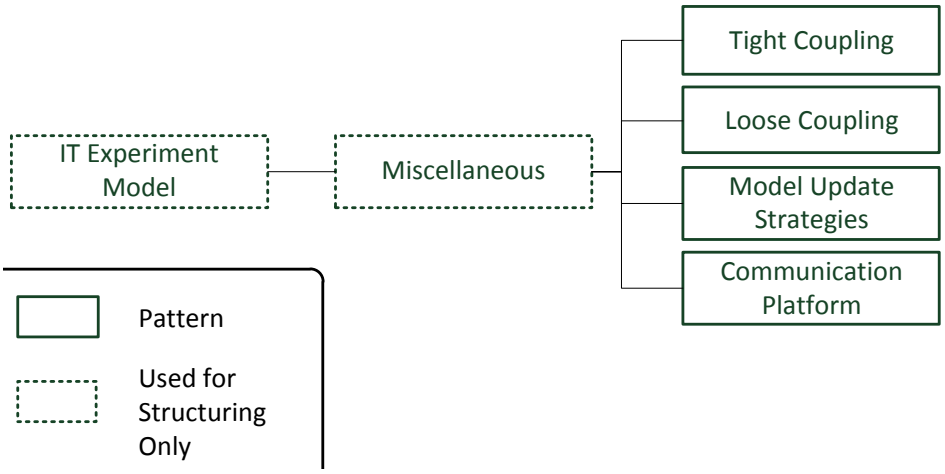


**Figure 24.:** This is an overview of the miscellaneous patterns on the level *IT experiment model*.

| | |
|---|---|
| **Name** | Tight Coupling |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to optimize the performance of applications, that are not frequently changed. |
| **Problem** | How to optimize applications? |
| **Context** | Scientists, that want to optimize the performance of the applications. |
| **Solution** | Use tight coupling of components. This means the components make many assumptions about each other. |
| **Example** | An example for a framework, that was build for tightly coupled, high performance simulations is the Cactus framework [43]. |
| **Relationships** | <Conflicts> Loose Coupling |

| | |
|---|---|
| **Name** | Loose Coupling |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to create a flexible and robust distributed application. |
| **Problem** | How to reduce dependencies between distributed applications and their distinctive components [32]?. |
| **Context** | Components in a distributed environment make assumptions about each other. These assumptions should be limited for flexibility and robustness reasons [32]. |
| **Solution** | Use an intermediary that handles format transformation, routing and addressing [32]. |

| | |
|---|---|
| **Example** | The SimTech WfMS uses a service bus for simulation workflows. This service bus handles data transformation, message routing and service discovery and selection [51]. |
| **Relationships** | <Conflicts> Tight Coupling |

| | |
|---|---|
| **Name** | Model Update Strategies |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to update the models within an eExperiment. |
| **Problem** | The data flow within an eExperiment is continuous. The model of the IT system needs to be changed. In a standard procedure the analysis is stopped. Then the model is updated. After this the analysis is started again. During the update, portions of the data flow cannot be analysed and get lost. |
| **Context** | There is continuous data flow in an eExperiment. The model of the eExperiment needs to be updated. |
| **Solution** | Implement a mechanism to reduce the downtime of the system during the update. |
| **Example** | In [127], Wickramaarachchi et al. formalize different update models. In addition, they provide metrics for evaluating update strategies. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Communication Platform |
| **Classification** | IT Experiment Model |
| **Intent** | Helps to improve the collaboration between the scientists. |
| **Problem** | Scientists want to make comments and exchange thoughts about an eExperiment. |
| **Context** | Scientists that conduct an eExperiment in a collaborative manner. |
| **Solution** | Build a mechanism, that enables communication around an eExperiment. |
| **Example** | Group-to-group communication is supported by the AccessGrid via high-speed networking and state of the art camera and display technology [49]. |
| **Relationships** | - |

## 4.5. Infrastructure

This chapter comprises the patterns on the level *infrastructure*. Figure 25 shows the grouping of the patterns into *concrete infrastructures* and *WfMS*.

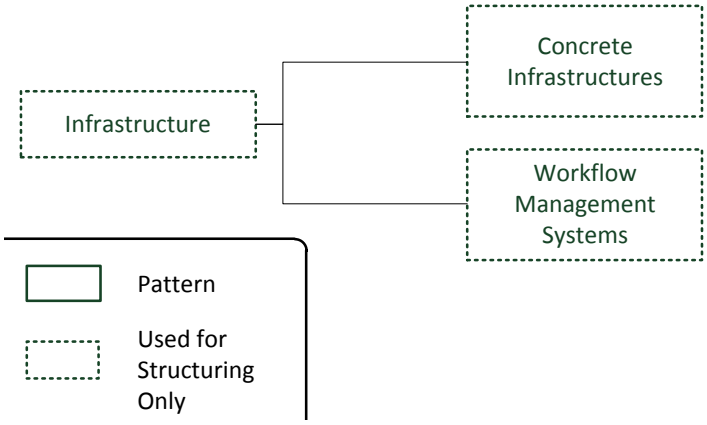


**Figure 25.:** This is an overview of the e-Science patterns on the level *infrastructure*.

### 4.5.1. Concrete Infrastructure

Figure 26 provides an overview of the *concrete infrastructures* patterns.

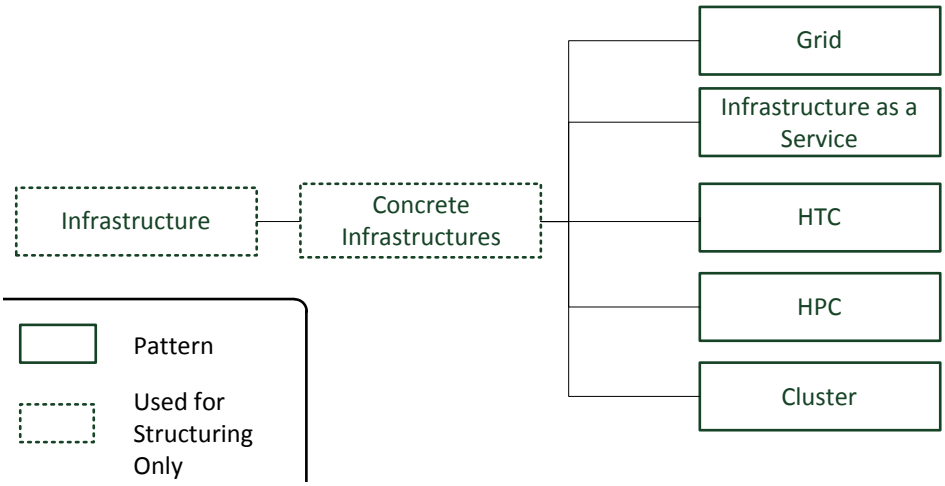


**Figure 26.:** This is an overview of the e-Science concrete infrastructure patterns.

| | |
|---|---|
| **Name** | Grid |
| **Classification** | Infrastructure |
| **Intent** | Helps to provide the scientists with infrastructures. |
| **Problem** | How to share resources in a flexible, secure and coordinated way among dynamic collections of individuals, institutions and resources [36]? |
| **Context** | Scientists that require infrastructures. |

| | |
|---|---|
| **Solution** | Use a Grid. Grids coordinate resources that are not subject to centralized control. They use standard, open, general purpose protocols and interfaces. Furthermore, they deliver non-trivial qualities of service [34] |
| **Example** | A Grid is used in [19]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Infrastructure as a Service |
| **Classification** | Infrastructure |
| **Intent** | Helps to provide the scientists with dynamic infrastructures. |
| **Problem** | How to provide the scientists with dynamic infrastructures? |
| **Context** | Scientists require virtual servers, which should be offered on a pay-per-use basis. In addition, isolation between the different tenants is required [32]. |
| **Solution** | The utilization of resources needs to be monitored. This is required for billing on a pay-per-use basis. In order to achieve isolation between the tenants, access controls need to be established. Then tenants can only access the resources on which they have access rights [32]. Please also consider the other Cloud computing patterns that can be used in combination with the *Infrastructure as a Service* pattern that can be found in [32]. |
| **Example** | Lezzi et al. base an e-Science solution on Cloud technologies [66].<br>An examples of Cloud infrastructure hosting provider is Amazon Web Services[3].<br>A collection of Cloud computing patterns can be found in [32]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | HTC |
| **Classification** | Infrastructure |
| **Intent** | Helps to provide the scientists with computing power. |
| **Problem** | How to provide the scientist with computing power? |
| **Context** | Scientists that require computing power. |
| **Solution** | Provide HTC infrastructure, which "is based on commonly available computing resources such as commodity PCs and small clusters that enable the execution of farming jobs without providing a high performance interconnection between the CPU/cores" [92]. |
| **Example** | Wilson et al. describe a HTC resource in production use [128]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | HPC |
| **Classification** | Infrastructure |
| **Intent** | Helps to provide the scientists with computing power. |
| **Problem** | How to provide the scientist with massive parallel computing power? |
| **Context** | Scientists that requires massive computing power. |

[3] http://aws.amazon.com/de/

| | |
|---|---|
| **Solution** | Provide a HPC infrastructure, that enables "the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance CPU/core interconnections" [92]. |
| **Example** | The *HPCWorld consortium* provides a handbook for effective use of HPC infrastructures [6]. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Cluster |
| **Classification** | Infrastructure |
| **Intent** | Helps to provide the scientists with computing power. |
| **Problem** | How to provide the scientist with computing power? |
| **Context** | Scientists that require massive computing power. |
| **Solution** | Use a cluster, which "is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource" [21]. |
| **Example** | - |
| **Relationships** | - |

### 4.5.2. Workflow Management Systems

Figure 27 provides an overview of the *WfMS* patterns.

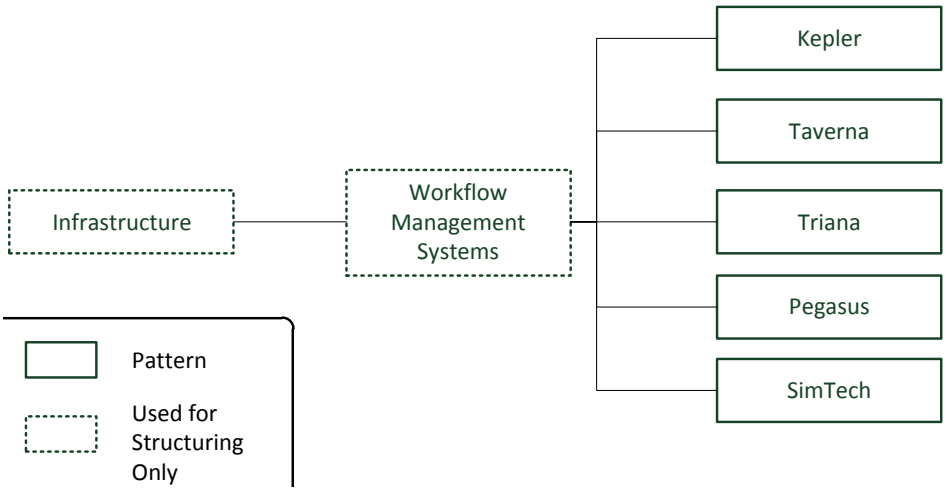


**Figure 27.:** This is an overview of the e-Science WfMS patterns.

| | |
|---|---|
| **Name** | Kepler |
| **Classification** | Infrastructure |
| **Intent** | Helps to conduct data-intensive, computation-intensive and knowledge-intensive experiments [85]. |
| **Problem** | How to conduct experiments, that are data-intensive, computation-intensive and knowledge-intensive [85]? |
| **Context** | Scientists that want to automate their eExperiments. |
| **Solution** | Use the Kepler WfMS [4]. Curcin et al. see the focus of Kepler in data analysis and modelling [26]. In Kepler, the execution engine is separated from the workflow model. Each workflow gets one model of computation assigned (MoC). This MoC is called director. The director triggers the execution of the workflow components. A component in Kepler is called actor. This actor represents data sources or operations in combination with a number of ports, which act as endpoints for connections. Data in Kepler is represented by *data tokens*. To make the Kepler actors reusable, the actors are *data polymorphic*. This means that one actor can process different data types on inputs. For instance one actor can perform string concatenation, the addition of integers or real numbers [26]. The user can start the workflow execution either by a GUI or from a command line [117]. |
| **Example** | In [68] and [85], Kepler is described in more detail. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Taverna |
| **Classification** | Infrastructure |
| **Intent** | Helps to conduct in silico experiments in the domain of life sciences [81]. |
| **Problem** | How to use Grid technology to conduct in silico experiments in the domain of life sciences [81]? |
| **Context** | Scientists that want to automate their eExperiments. |
| **Solution** | Use the Taverna WfMS [5]. Talia sees the goal of Taverna in supporting the life sciences community (medicine, chemistry and biology) in the design and execution of scientific workflows [117]. Taverna is strongly based on Web services and therefore, it can be used by scientists from different domains. The tree main components of Taverna are the Taverna Workbench graphical workflow authoring client, SCUFL as workflow representation language and the Freefluo enactment engine [26]. Taverna is open to many components, which are provided by different organizations. Since there is no common data format assumed, the user needs to resolve the formats manually [26]. |
| **Example** | In [81], Taverna is described in more detail. |
| **Relationships** | <Uses> Grid |

[4] https://kepler-project.org/
[5] http://www.taverna.org.uk/

| | |
|---|---|
| **Name** | Triana |
| **Classification** | Infrastructure |
| **Intent** | Helps to make use of services within multiple environments and to integrate with various types of middleware toolkits [118]. |
| **Problem** | How to make use of services within multiple environments and how to integrate with various types of middleware toolkits [118]? |
| **Context** | Scientists that want to automate their eExperiments. |
| **Solution** | Use the Triana WfMS[6]. Triana provides a visual interface and data analysis tools [117]. Functional components within Triana are called *units*. Units can represent a workflow, legacy code or a file [118]. Directed *cables* are used to connect units. The user is interacting via the GUI with the Triana units. The units specify one section of the system instead of implying a specific implementation methodology [118]. The units in Triana are bound late to the services, which they represent. This ensures a dynamic behaviour [117]. |
| **Example** | In [118], Triana is described in more detail. |
| **Relationships** | - |

| | |
|---|---|
| **Name** | Pegasus |
| **Classification** | Infrastructure |
| **Intent** | Helps to design workflows at the application level without knowledge about the execution environment. |
| **Problem** | How to design workflows at the application level without knowledge about the execution environment [28]? |
| **Context** | Scientists that want to automate their eExperiments. |
| **Solution** | Use the Pegasus WfMS[7]. In [117], Talia introduces Pegasus as a "a set of technologies to execute workflow-based applications in a number of different environments, including desktops, clusters, Grids, and Clouds". The functionality of Pegasus is split across the three main components *mapper*, *execution engine* and *task manager*. The mapper takes an abstract workflow, which can be either created by the user or by the system, as an input. Computational resources and the required input data are located in an automated way by the mapper. Finally, the mapper transforms the abstract workflow into an executable workflow. The execution engine takes the executable workflow as input. Then it executes the tasks in the order of their dependencies on the defined resources. Finally, the task manager supervises the execution of a single workflow task. Pegasus also includes error recovery that is based on workflow-level checkpointing. In addition, provenance information is recorded for increasing the workflow reproducibility [117]. |
| **Example** | In [28], Pegasus is described in more detail. |
| **Relationships** | - |

[6] http://www.trianacode.org/
[7] http://pegasus.isi.edu/

| | |
|---|---|
| **Name** | SimTech |
| **Classification** | Infrastructure |
| **Intent** | Helps scientists conducting eExperiments with focusing on their core competencies. The scientists do not have to care about the technical complexity [51]. |
| **Problem** | Scientists require a WfMS for conducting eExperiments, that is tailored to their requirements. At the same time the technical complexity should be hidden, so that they can focus on their core competencies [51]. |
| **Context** | Scientists, that want to automate their eExperiment. |
| **Solution** | Use the SimTech WfMS [51]. Scientists can model workflows intuitively by using a GUI[8]. In addition, they can execute, monitor and adapt the workflows within the SimTech WfMS. Furthermore, the trial and error approach for conducting experiments is supported by increasing the flexibility of the workflows[9]. For the SimTech WfMS the Business Process Execution Language (BPEL) as workflow language is used. Additionally, Web services technology is utilised for implementing workflow activities [10]. |
| **Example** | In [56], the SimTech WfMS is used for evaluating the eScienceSWaT methodology. |
| **Relationships** | - |

[8] http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_modeling.php
[9] http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_flexibility.php
[10] http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_execution.php

# 5. Decision Support System

Another result of this thesis is the concept of a DSS, which provisions the e-Science patterns. In Section 5.1 we describe the functionality of the DSS and Section 5.2 presents the architecture of the DSS. Finally, the prototypical implementation is introduced in Section 5.3.
In eExperiments different people, e.g. natural scientists, computational scientists or software engineers work together in a collaborative manner. The DSS is targeted at all the participants of an eExperiment. However, for this section we do not see the need to differentiate the various user groups, but we only speak of users.

## 5.1. Functionality

First of all the basic pattern selection support is introduced in Section 5.1.1. Then we present the concept of decision chains within the DSS.

### 5.1.1. Basic Pattern Selection Support

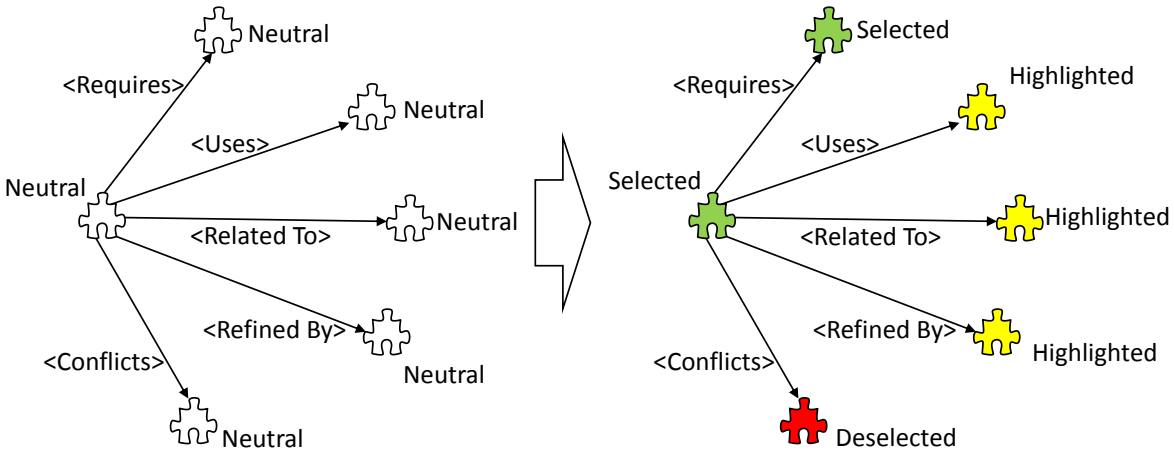


**Figure 28.:** This shows the basic pattern selection support.

The basic pattern selection support makes use of the information in the e-Science pattern catalogue. A pattern can have several relationships to other patterns. If this pattern gets selected, the status of all related pattern gets updated according to the relationship, which is used to connect it. A pattern can have one of the following status: *neutral* (default status), *highlighted*, *selected* or *deselected*. In Section 4.2, we introduced the relationships, which we use

for the patterns. Figure 28 visualizes the basic pattern selection support. All patterns have a neutral status at the beginning. The pattern in the centre, which has outgoing relationships to the other patterns gets selected by the user. After this, the status of all related patterns is updated. For example, the pattern which is connected via the <Requires> relationship then changes its status to *selected* or the pattern which is connected via the <Uses> relationship takes the status *highlighted*. As soon as a pattern takes the status *selected* – the status can either be set manually by the user or via the basic selection support mechanism – all its relationships are evaluated and the status of the related patterns gets updated. This mechanism supports the user by the pattern selection, since certain patterns get selected or deselected. In addition, the users attention is drawn on the highlighted patterns.

### 5.1.2. Decision Chains

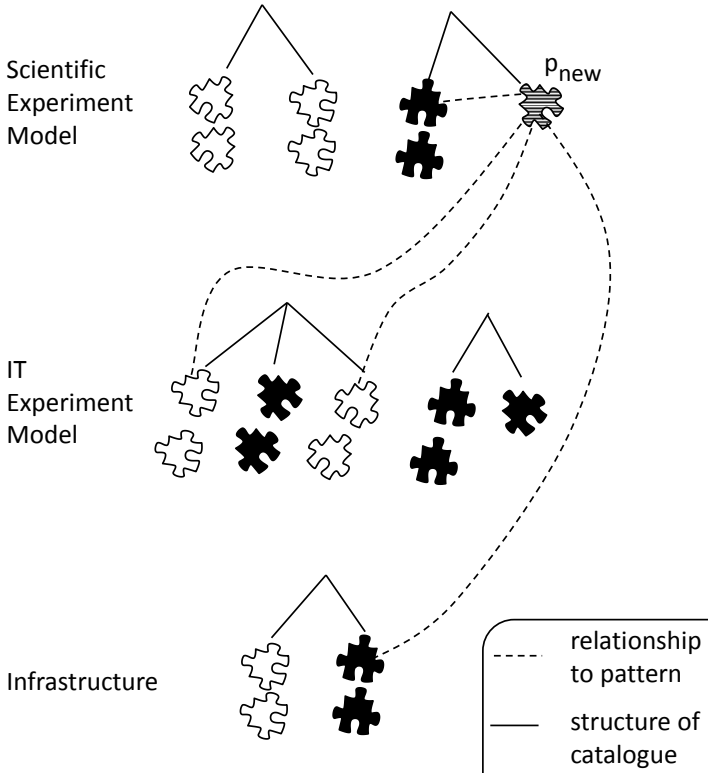


**Figure 29.:** New pattern comprising a decision chain.

Karastoyanova et al. stated, that a knowledge base for eScienceSWaT should also comprise decision chains [56]. Currently, the e-Science pattern catalogue – the knowledge base – only comprises the basic e-Science patterns. With the DSS it is possible to store decision chains. In the context of the DSS, a decision leads to the selection or deselection of a pattern. Consequently, a decision chain leads to a selection or deselection of certain patterns $p_1$, ..., $p_i$. The decision chain can be stored with the help of a new pattern $p_{new}$. $P_{new}$ comprises all patterns $p_1$, ..., $p_i$. The problem of $p_{new}$ describes the problem, which is solved by the

decision chain. The intent of $p_{new}$ specifies the intent of the decision chain. Furthermore, the solution provided by $p_{new}$ comprises the solution of all patterns $p_1$, ..., $p_i$. $P_{new}$ has either a <Requires> or a <Conflicts> relationship to each of the patterns $p_1$, ..., $p_i$. Figure 29 shows the position of $p_{new}$ in the e-Science pattern catalogue. $P_{new}$ should be positioned as high as possible in the eExperiment model, but not higher as the highest comprised pattern. If the user follows eScienceSWaT, he starts with the selection of scientific experiment patterns. This means he can reuse the decision chain early in the process by selecting the pattern $p_{new}$. The $p_{new}$ pattern shown in Figure 29 comprises four patterns. The relationships to these patterns are visualized by dashed lines.

## 5.2. Architecture

We use the Model-View-Controller (MVC) architecture pattern, which was introduced by Buschmann et al. in [20]. By using the MVC pattern, an application is split into the three components *model*, *view* and *controller*. The decoupling of model, view and controller leads to a higher flexibility of the application [83]. One component can be modified or replaced with a new version. This only has minimal impact on the other components. For example the visualization of the pattern catalogue can be changed by using a different *view* component. In addition, there can be multiple views of the same model [20]. Furthermore, the complexity is reduced by separating the different application components. The core functionality and data is kept inside the model. Information is displayed to the user by the view component. The controller receives the input of the user. Figure 30 displays the interaction among model, view and controller. The view can query the model about its status. If the model is changing, it notifies the view. The user can make changes in the view, which are propagated to the controller. The controller selects the view and updates the model according to the changes of the user.

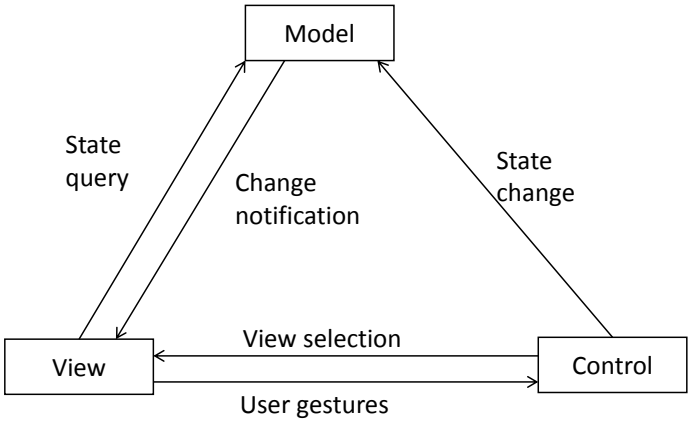


**Figure 30.:** Interaction among model, view and controller based on [83].

In our case, the model of the DSS comprises the e-Science pattern catalogue. The contents of the patterns, the relationships among the patterns and also the logic for determining the pattern status of the related patterns. The view visualizes the e-Science pattern catalogue. It

shows the patterns and their relationships to other patterns. In addition, the pattern status is displayed. The controller receives the user gestures from the view. For example the user requests to see a different section of the e-Science pattern catalogue or selects a pattern. Then the controller selects a different view of the e-Science pattern catalogue and sets the pattern status accordingly. Once the pattern status is set, the model sends a change notification to the view.

## 5.3. Prototype

In the last section we introduced the MVC pattern. We apply this pattern for the DSS prototype architecture. For implementing the prototype, we use the Graphical Modeling Framework (GMF)[1]. GMF is based on the Eclipse Modeling Framework Project (EMF)[2] and the Graphical Editing Framework (GEF)[3]. Section 5.3.1 provides an overview of the model and Section 5.3.2 introduces view and controller. In Section 5.3.3, we describe how the user can work with the DSS prototype.

### 5.3.1. Model

The model of the DSS is created with the help of EMF. In Figure 31 the major steps for generating the Java classes of the domain model and the flow of artefacts are presented. A XML Schema Definition (XSD) description of the domain model of the DSS builds the basis. This XSD description can be seen in Listing B.1. The main elements of the domain model of the DSS are patterns and structure elements. This XSD model is imported into EMF and gets transformed into a model in Ecore format. We make manual adjustments on the Ecore model, because the import mechanism does not set all properties of the Ecore model properly. Based on the Ecore model, a generator model is generated. Finally, the Java classes for the domain model of the DSS are generated with the help of the generator model.

### 5.3.2. Graphical User Interface

By using GMF, a graphical editor is created, which implements the MVC pattern [120]. We use the editor as a *view*. The controller links the model and the view. By using a graphical editor for the view, we get additional benefits. The pattern status can be set in the editor and *decision chains* can directly be created in the editor (see Section 5.1.2). Figure 31 provides an overview of the flow of artefacts and the major steps for generating the graphical editor. Based on the Ecore domain model, we define the graphical notation and a tooling definition. Then we create a mapping model, which maps the different models of graphical notation, tooling definition and domain model generator. Out of this mapping model, we generate the generator model for the graphical editor. Finally, the Java classes for the graphical editor are

[1] http://wiki.eclipse.org/Graphical_Modeling_Framework
[2] http://www.eclipse.org/modeling/emf/
[3] http://www.eclipse.org/gef/

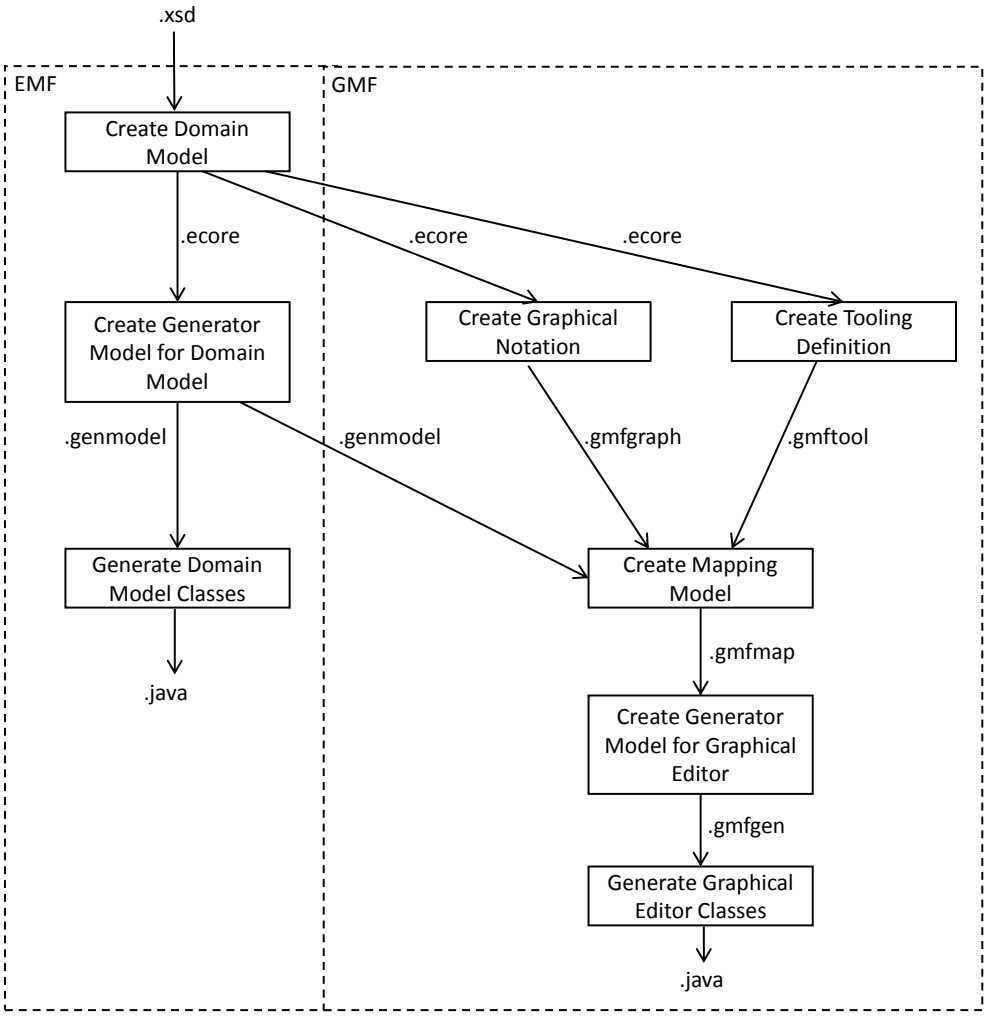


**Figure 31.:** This shows the major steps and the flow of artefacts for generating the DSS prototype based on [122].

generated with the help of the graphical editor generator model. A detailed description of the steps for generating a graphical editor with GMF can be found in [121].

In Figure 32, a screenshot of the GUI of the DSS prototype is presented. The GUI can be partitioned into the four parts *outline*, *palette*, *editor* and *properties*. We explain each part briefly hereinafter.

**Outline:** An overview of the whole e-Science pattern catalogue is provided. In addition, the current position within the e-Science pattern catalogue is visualized, so that the user can use the outline for orientation.

**Editor:** The editor contains the graphical representation of the e-Science pattern catalogue. Since it is an editor, the whole e-Science pattern catalogue is editable. This is a prototype and therefore, we accept the editability. However, for the future DSS the editability needs to be restricted, so that only the pattern status can be edited.

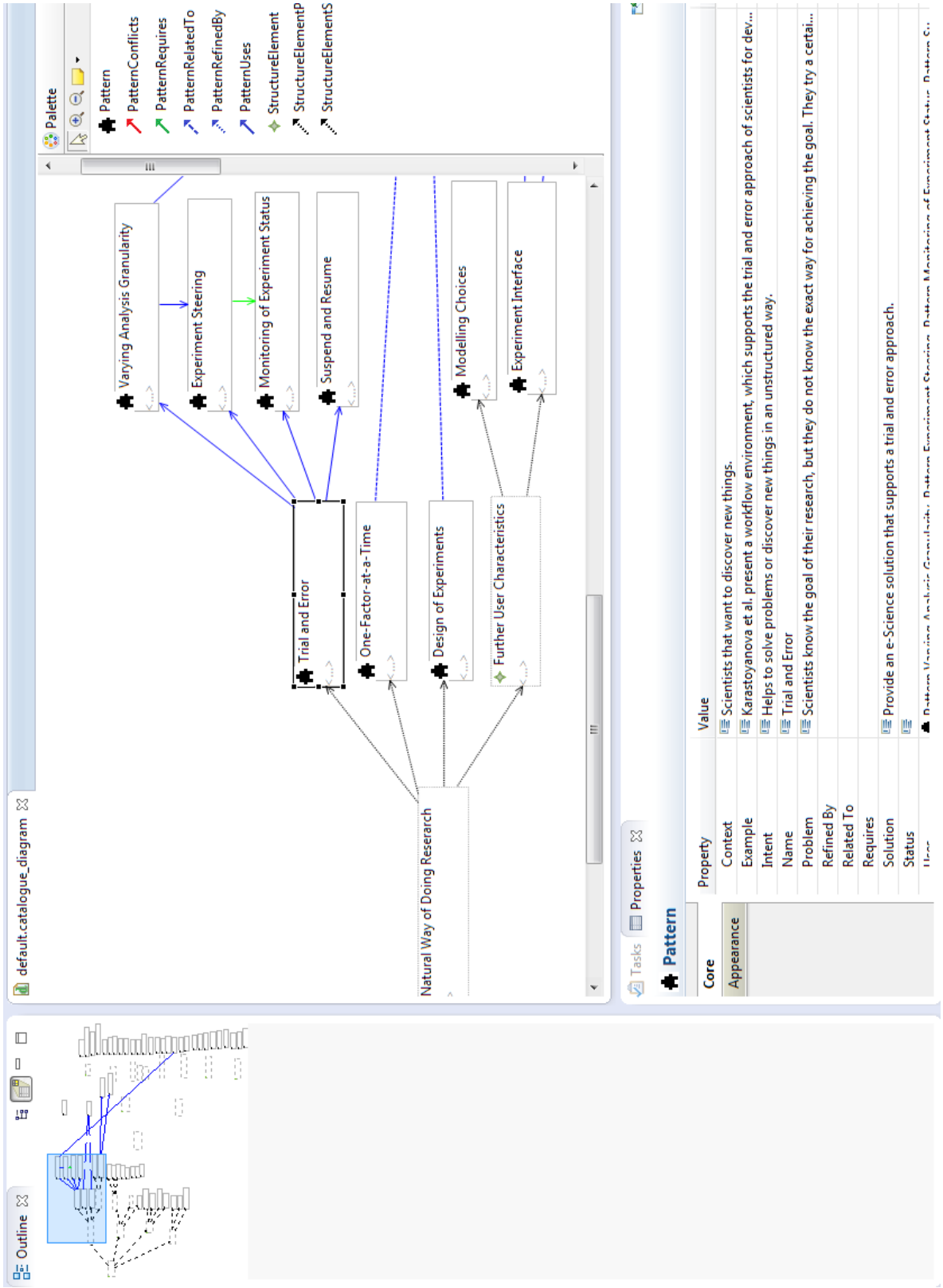


**Figure 32.:** This shows the GUI of the DSS prototype.

The graphical representation of the e-Science pattern catalogue follows the same logic as the e-Science pattern catalogue, which is presented in Chapter 4. It consists of structure elements and patterns. Additionally, the editor visualizes each of the different relationships in the e-Science catalogue. The structure elements and patterns are labeled with their names. The pattern status is visualized via colours. Green means *selected*, *red* stands for *deselected* and blue means *highlighted*.

**Palette:** The palette can be used to create new elements of the e-Science pattern catalogue. New decisions chains can be added by dragging new elements from the palette onto the editor (see Section 5.1.2). We already explained, that the future version of the DSS requires limited editability. Additionally only certain users should be able to extend the e-Science pattern catalogue with new elements. Beside the initial functionality of the palette, it can be used as a legend. It holds the mapping between the graphical representation and the description of the elements.

**Properties:** The properties view provides detailed information in form of the attributes of the e-Science patterns and the structure elements. The structure elements in the e-Science pattern catalogue only had the purpose of organizing the patterns (Section 4.2.3). The structure elements in the DSS prototype contain an additional attribute. This attribute is called *instruction* and contains additional information about the related patterns. For instance the structure element *data characteristics* contains the information "please select patterns, which describe the data, that is used in the eExperiment". This additional information helps the user to understand the purpose of the related patterns better, before he actually reads through the patterns.

### 5.3.3. Use Case

In the following, we explain how the user is working with the DSS prototype. The user wants to create an eExperiment and requires decision support, because he can make many choices for the scientific experiment model, the IT experiment model and infrastructures. This is why he uses the DSS prototype. Basically, the graphical representation of the e-Science pattern catalogue can be seen as a graph. It consists of nodes, which represent patterns and structure elements, and edges that represent the relationships. The user has three starting nodes: *scientific experiment model*, *IT experiment model* and *infrastructure*. From each starting node, the user can reach patterns or structure elements by following the edges. Each structure element can also be seen as an important category for the creation of an eExperiment. By providing the different categories of patterns, the user gets an overview what might be important for his eExperiment. The user can select or deselect the patterns for his eExperiment by setting the pattern status. The status of the related pattern gets updated automatically, because the DSS prototype supports the basic pattern selection support (see Section 5.1.1). Now the user can follow the edges to related patterns and especially focus on the patterns, which are highlighted. In general, the patterns contain more generic information and best practices. This information should be seen as a starting point, since no concrete detailed solutions are provided. If the user is not familiar with a pattern, then he can access its detailed information in the *properties section* (see Figure 32). Furthermore, the *properties section* provides an example,

where the pattern is applied in an eExperiment. This helps the user to understand the pattern better so that he can decide if he selects the pattern or not.

# 6. Conclusion and Future Work

This chapter summarizes the contents of the thesis in Section 6.1. In Section 6.2 we present several possible extensions for the e-Science pattern catalogue and the DSS.

## 6.1. Conclusion

In order to realize systems supporting the eScienceSWaT methodology, Karastoyanova et al. identified the clear need of a DSS, which comprises knowledge and best practices from natural sciences, software engineering and computer science [56]. This thesis provides these best practices in the form of an pattern catalogue. Furthermore, the concept of a DSS, which provisions the patterns and its prototypical implementation are introduced. Chapter 2 provides background information, which is necessary to understand the concepts of this thesis. Patterns, an overview about e-Science and the concept of scientific workflows belong to these. In Chapter 3, related work in the fields of patterns and DSSs is introduced. Additionally, various relationships among patterns are analysed and the work of this thesis is positioned to each related work. Furthermore, the findings in this chapter lay the groundwork for the following chapters. Chapter 4 is the key contribution of this thesis. First of all, we describe our approach for developing the e-Science pattern catalogue. This contains the information collection phase and the pattern generation phase. In the information collection phase we gather relevant information about eExperiments, which we transform into problem-solution pairs. In the pattern generation phase we abstract from the concrete problem-solution pairs and by doing so we carve out patterns. After that the results in form of the relationships among e-Science patterns and the e-Science pattern catalogue are introduced. The catalogue is split into three categories: scientific experiment model, IT experiment model and infrastructure. Each category again contains a substructure in which the patterns are organized. Finally, in Chapter 5 the concept of a DSS is introduced. The core functionality of the DSS is to provide relevant information in form of patterns for decision making within each phase of the eScienceSWaT methodology. Furthermore, an architecture for the DSS is introduced, which is prototypically implemented.

## 6.2. Future Work

This section comprises different suggestions for extending the e-Science pattern catalogue and the DSS. For simplification reasons we do not differentiate among the different user groups of the DSS and the e-Science pattern catalogue in the next section. We only speak of users.

### 6.2.1. Pattern Awareness for Artefacts

At one point in the decision making process, concrete artefacts, e.g.infrastructures, applications, services or middleware have to be selected. All artefacts have certain characteristics. Due to these characteristics, they support certain patterns or do conflict with certain patterns. We propose to store this information about supported patterns and conflicting patterns for each artefact. Now the artefacts are aware of the supporting and conflicting patterns. With the help of this information, the DSS can propose suitable artefacts. In the following we give an example how the DSS can provide decision support for selecting the most suitable WfMS pattern for an eExperiment. The e-Science catalogue comprises five different WfMS patterns from which the user can choose, that represent five different WfMS (see Section 4.5.2). Each WfMS supports certain patterns of the e-Science pattern catalogue. We propose to add a list of all supported e-Science patterns from the e-Science pattern catalogue to each WfMS. With the help of the DSS, the user selects the required patterns for his eExperiment. However, he does not select the WfMS patterns, because he is not sure which of the WfMS fits best for his eExperiment. At the end, the DSS compares all selected patterns with the list of supported patterns of every WfMS. Then the DSS ranks the WfMS according to the number of patterns, which they support. Finally, the DSS can make a proposal for the most suitable WfMS – the WfMS that supports the most patterns – to the user.

Currently, we assume that the user follows the eScienceSWaT methodology. Therefore, he starts with the selection of patterns on the *scientific experiment model*, then he selects patterns for the *IT experiment model* and finally he makes choices for the *infrastructure* patterns. With respect to the eExperiment model, this is a top-down approach (see Figure 8). However, the user knows, that he has to use a cloud computing infrastructure from a certain provider. This infrastructure has specific characteristics and narrows down the amount of possible patterns of the *IT experiment model*. Nevertheless, the user has no choice to provide this information at an earlier stage in the decision making process, because infrastructures are selected at the end. *Pattern awareness* brings an additional benefit. The user can set this specific cloud computing infrastructure as a precondition. This cloud computing infrastructure is aware of its supported and conflicting patterns. Now the DSS can use this information to provide better decision support at an earlier stage in the decision making process. Conflicting patterns of the *IT experiment model* are automatically deselected and required patterns are automatically selected.

### 6.2.2. Extended Pattern Selection Support

Some decisions do only affect one single pattern. If a pattern X has no connection to other patterns and no other patterns have a connection to X, the user only has to consider two choices for X: he can select X or deselect X. However, if X is in relationships with other patterns, so that the status of the patterns are influenced mutually, the decision gets more complicated. The effects on the other patterns due to selection or deselection of X have to be considered as well. Generally speaking, the problem is to select m patterns out of a group of n patterns under certain conditions. For such cases, we propose to provide additional decision

support for the user. An example for this problem is the selection of one pattern among a

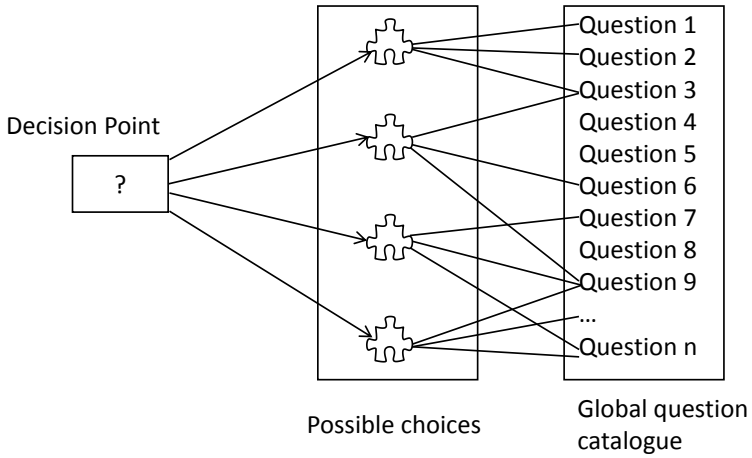


**Figure 33.:** Extended pattern selection support.

group of four patterns, that have a <Conflicts> relationship to each other. We propose to give the user additional guidance for this decision. In Section 3.1.3, the GQM approach was introduced. In GQM every pattern is connected to questions. Figure 33 shows the decision point and the four patterns, from which the user can choose. In addition, the connection of the patterns to the global question catalogue is visualized. At such a decision point, the DSS provides additional support by displaying all the questions, which are related to the patterns to the user. Now the user has to answer each question with *yes*, *no* or *don't know* and has to assign a weight to the question. The weight stands for the relevance of this question to his eExperiment. Now a metric calculates a ranking of the patterns based on his answers and the given weightings. At the end, the DSS can propose the pattern which is ranked on the first place to the user.

### 6.2.3. Case Based Reasoning for Pattern Selection

The DSS is used to select the most suitable e-Science patterns for an eExperiment. We propose to store a list of all selected/deselected e-Science patterns and additional information about the eExperiment. Together this information builds a *case*. A case has certain attributes, in which the additional information is stored. For instance a case can have attributes like domain of the eExperiment, the goal of the eExperiment, number and domain of scientists, available budget. In addition, every pattern in the e-Science pattern catalogue should be connected to all cases, in which it is applied. Then we want to provide this information to the users of the DSS. They can browse this information and check, which patterns were applied in other similar eExperiments. For example biologists have a certain budget for the eExperiment. In addition, they identified that they require the heterogeneous data pattern. Now they can browse all cases in the domain of biology, in which the heterogeneous data pattern was applied within a certain budget. They can see which other patterns were applied together with the heterogeneous data pattern. This helps them to select the most suitable patterns for

their own eExperiment.
For a more sophisticated DSS, we propose to use CBR, which also makes use of the *cases*. In CBR past experiences are used to understand and solve new problems. However, the whole system proposes suitable patterns with the help of a similarity metric (see Section 3.1.1). CBR seems suitable for the DSS and therefore, we propose to do further research in this domain.

### 6.2.4. Pattern Status Conflicts

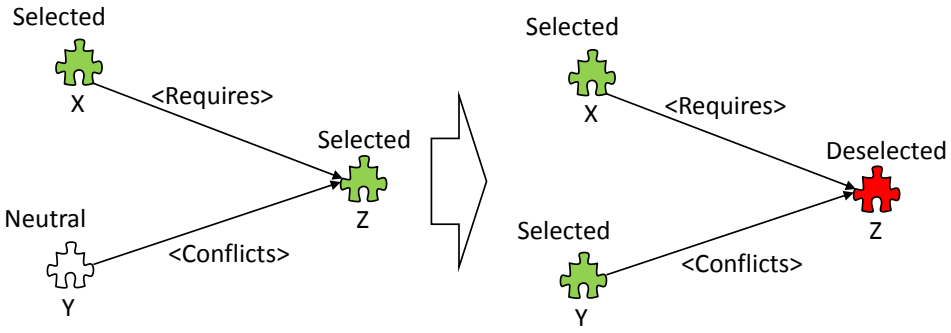


**Figure 34.:** This shows an example for a possible pattern status conflict.

Currently, there is no logic in the DSS, which ensures a correct pattern status. The sequence of actions of the user determine the final pattern status. Although there is no concurrency in the DSS prototype, this kind of faults can be compared with *race conditions*. In [24], *race conditions* are introduced with an example of two parallel processes that attempt to write to the same offset. The process that finishes later, overwrites the assignment of the process that finished first. Figure 34 shows the uncontrolled overwriting of the pattern status in the DSS prototype. Pattern X got selected by the user. Via the <Requires> relationship, Z gets automatically selected by the DSS. If the user selects now the pattern Y, the status of Z changes to deselected due to the <Conflicts> relationship. However, Z cannot be deselected, when X is selected.

We propose to implement a mechanism, that ensures integrity among the patterns. The survey of Beckman about preventing race conditions can be used as a starting point for further research [7]. Another concept, that might be applicable for the DSS prototype, is dead path elimination (DPE). Weidlich states, that DPE supports the explicit skipping of activities [126]. This skipping can lead to the skipping of subsequent activities for *eliminating* the *dead path*. If we apply this concept to the example in Figure 34, we force a correct pattern status. After the deselection of Z by the DSS due to the <Conflicts> relationship, also X gets deselected. Now there are no conflicting pattern status anymore.

### 6.2.5. Global and Local Views on eExperiments

Currently, the DSS supports only one single global view on an eExperiment for all users/stakeholders (e.g. natural scientists, IT specialists, ...). This can lead to conflicts for the pattern selection. For one component, there is a combination of certain patterns that is locally applied, for example on the level of a scientific experiment model. For another component, different patterns are locally applied, for example on the level of an IT experiment model. The patterns of each component are in conflict with each other. However, this is not an issue, since they are applied locally. But if all these patterns get aggregated into the single global view of the eExperiment, this leads to a conflict. For example the whole application of the eExperiment can be loosely coupled. However, one part of the application consists of a tightly coupled HPC. From a global perspective, the *loose coupling* pattern should be selected, but for the HPC the *tight coupling* pattern needs to be selected. We propose to extend the DSS in a way, so that the separate selection of patterns for local components of the eExperiment is possible.

# Appendix A.

# e-Science Catalogue Overview

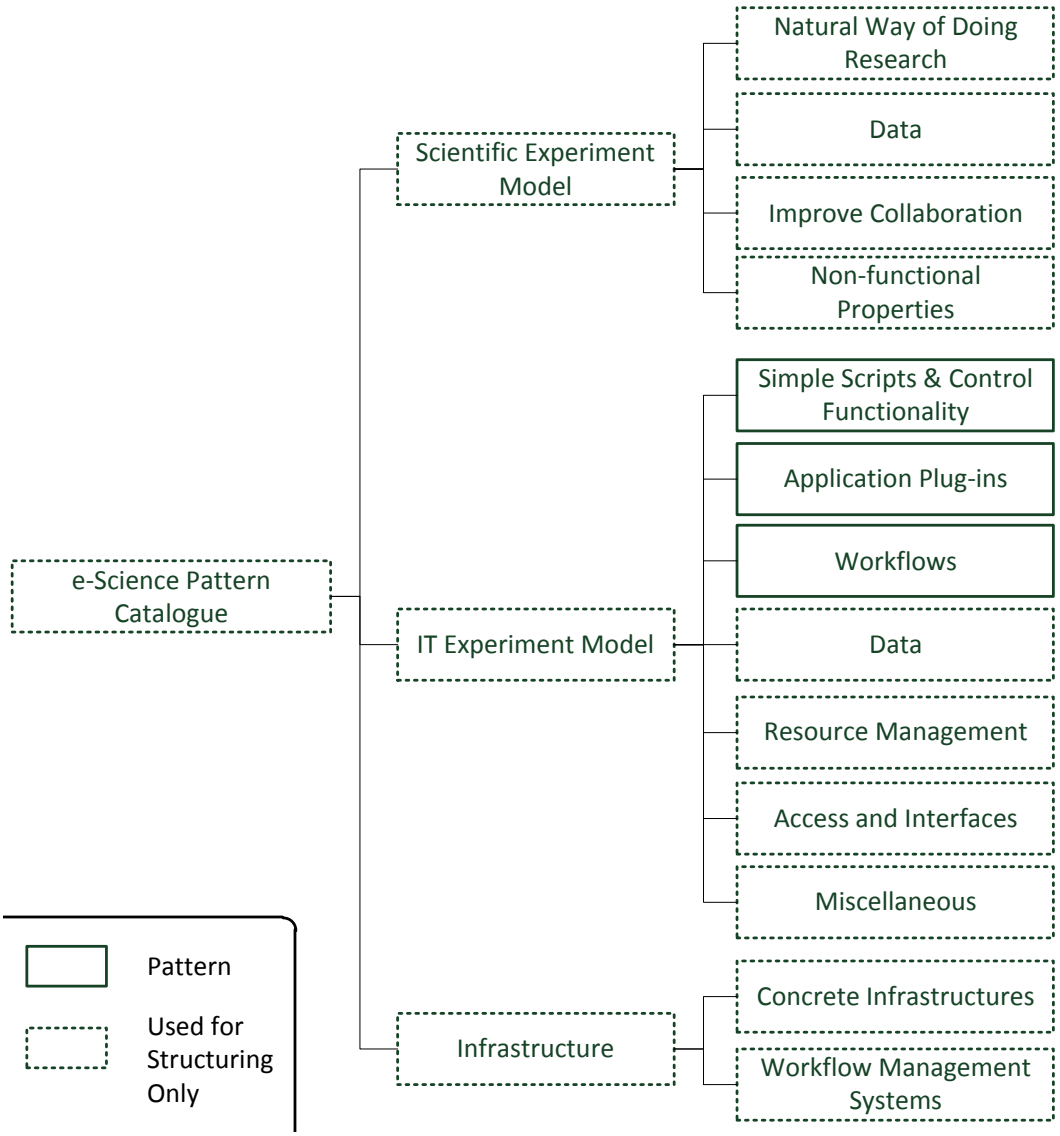


**Figure 35.:** This shows the top layers of the e-Science pattern catalogue.

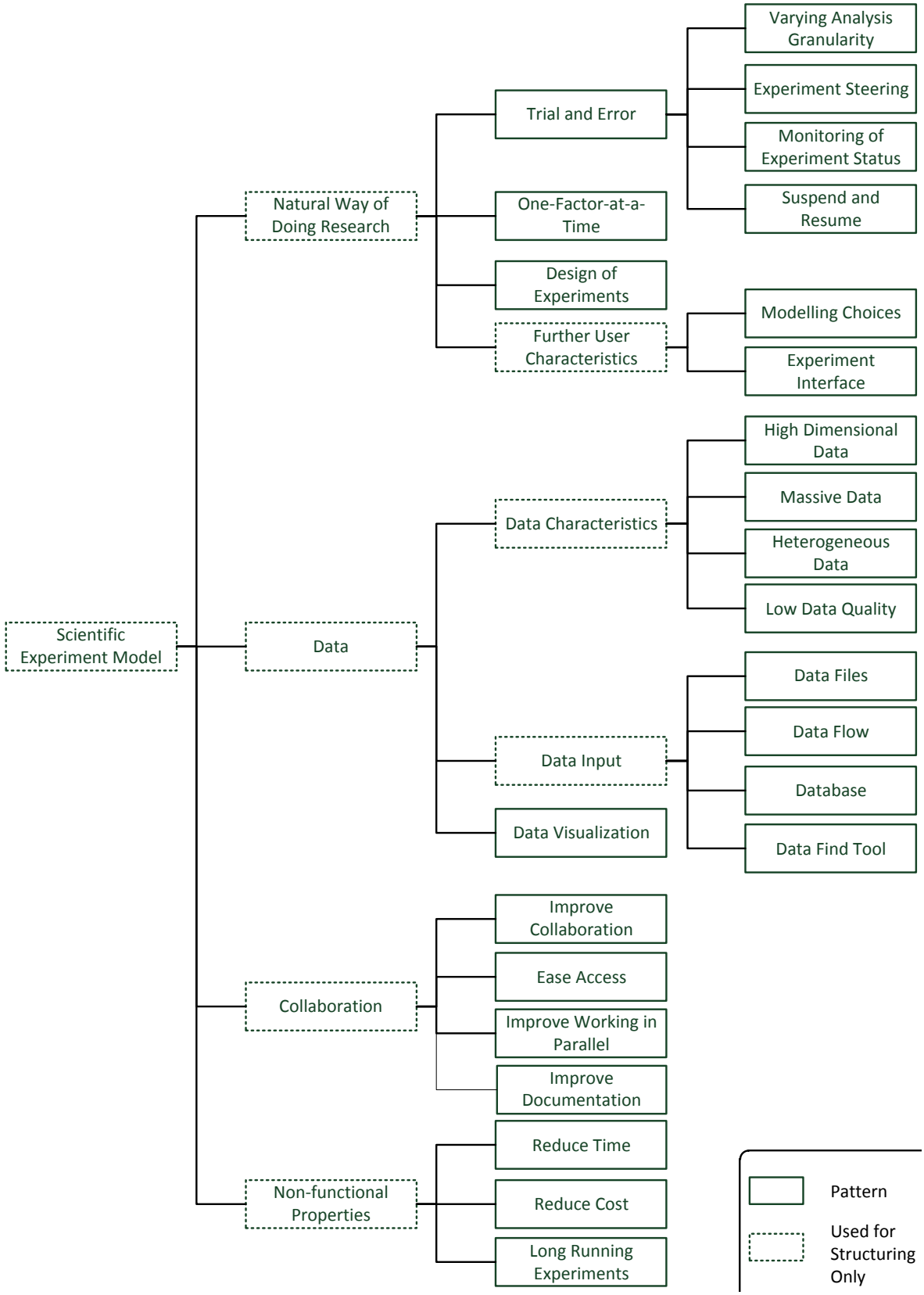


**Figure 36.:** This is an overview of the e-Science patterns on the level *experiment model*.

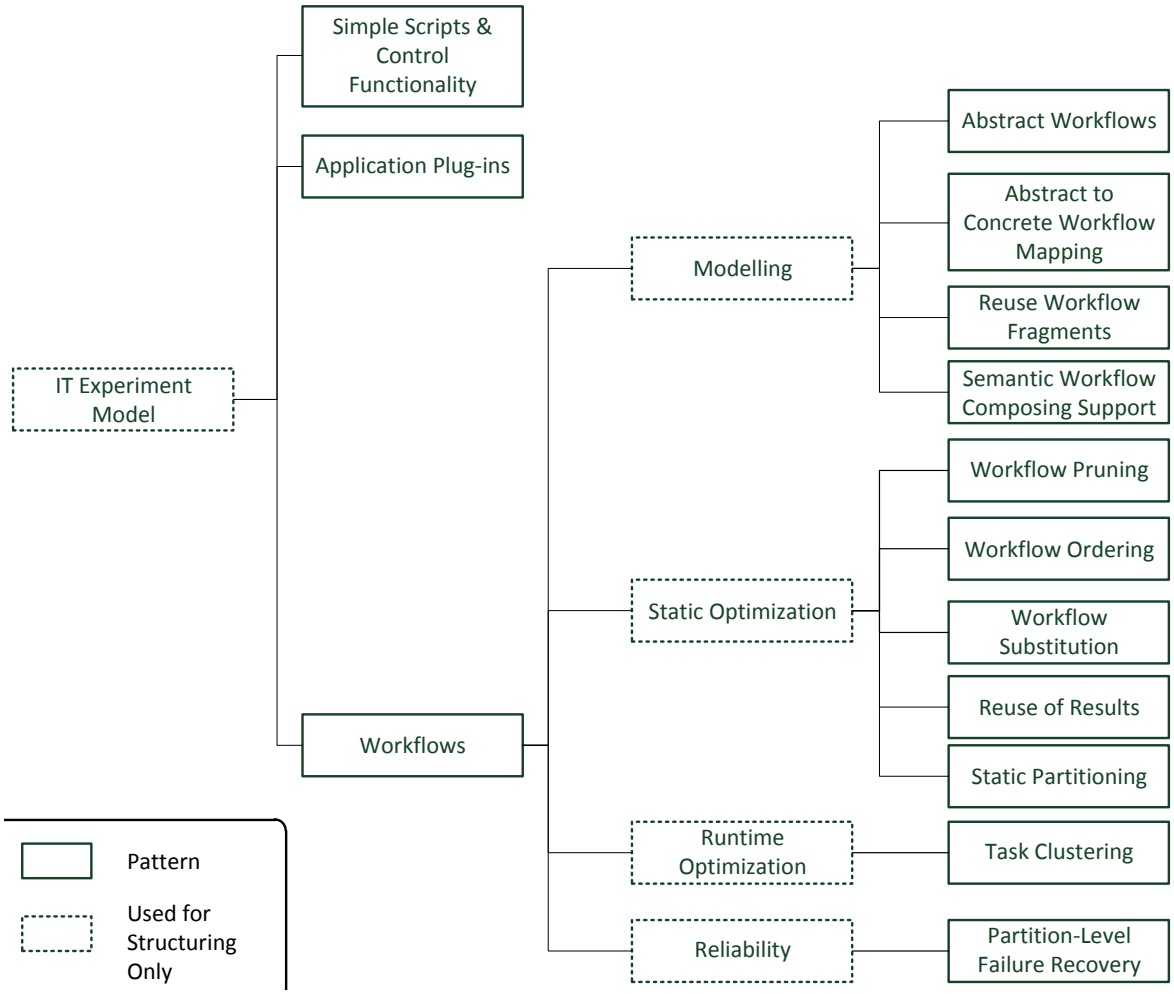


**Figure 37.:** This is the first part of an overview of the e-Science patterns on the level *IT experiment model*.

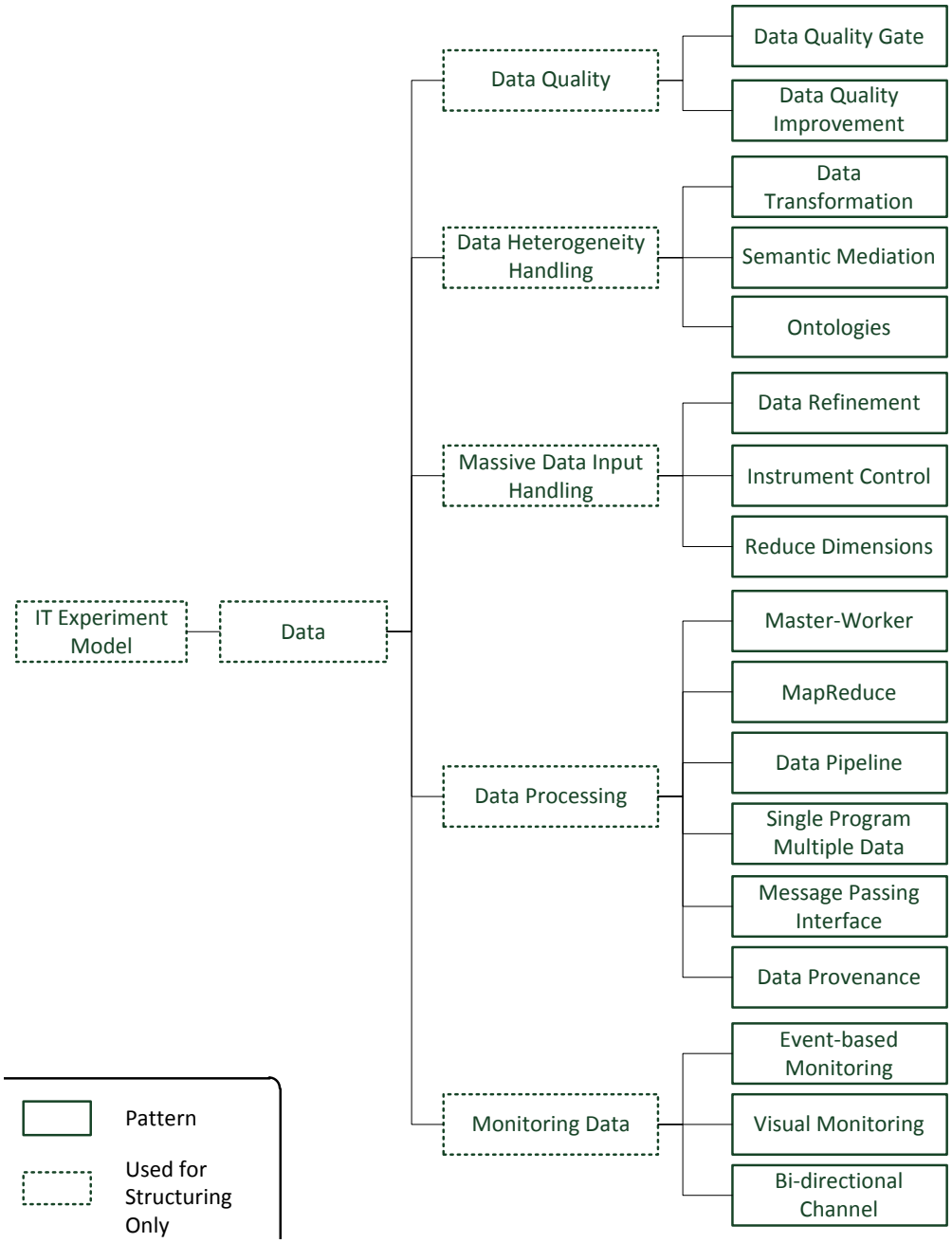


**Figure 38.:** This is the second part of an overview of the e-Science patterns on the level *IT experiment model*.

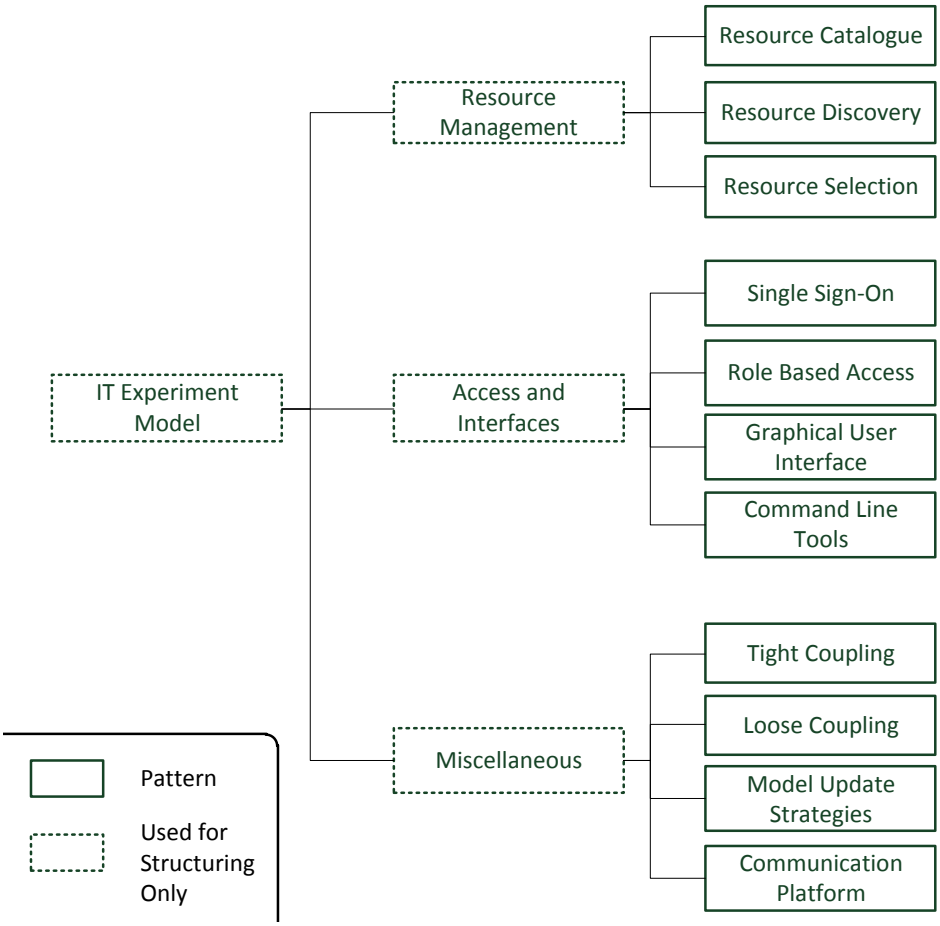


**Figure 39.:** This is the third part of an overview of the e-Science patterns on the level *IT experiment model*.

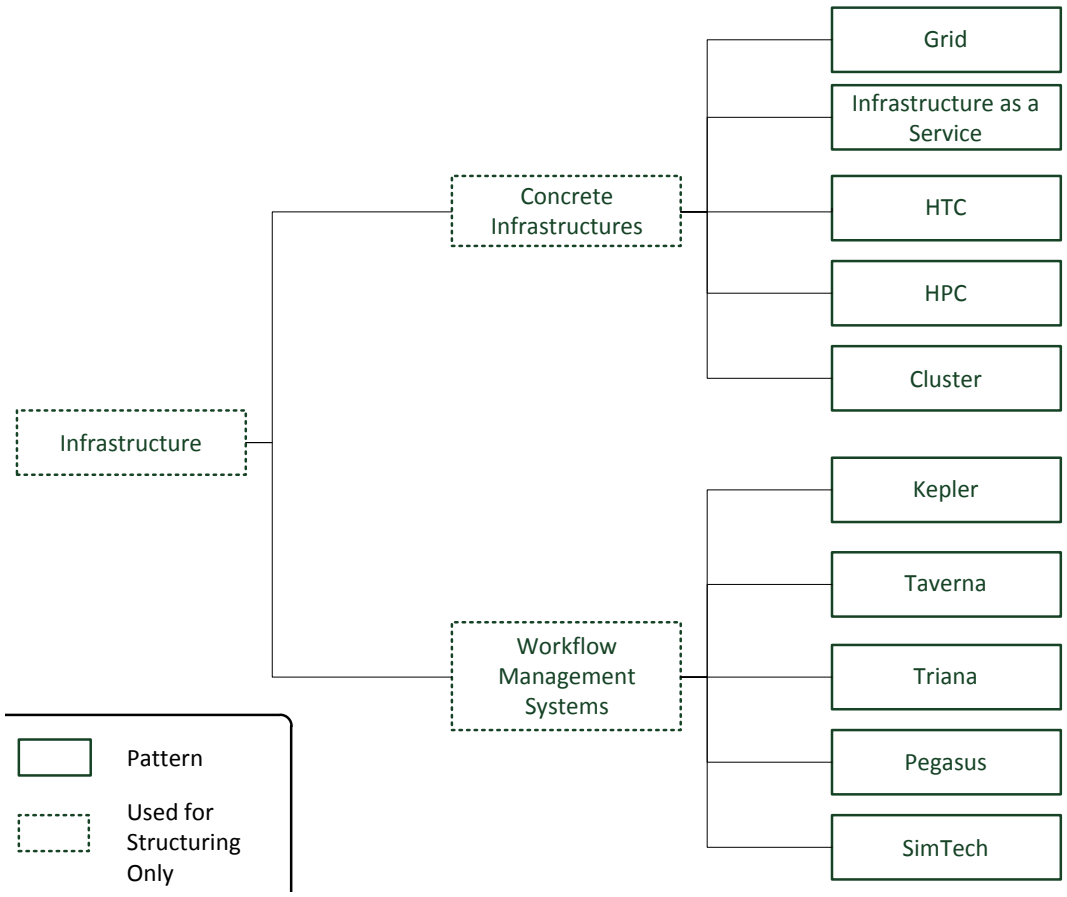


**Figure 40.:** This is an overview of the e-Science patterns on the level *infrastructures*.

# Appendix B.

# Decision Support System

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <xsd:schema xmlns:catalogue="http://www.example.org/catalogue" xmlns:ecore="http://www.
       eclipse.org/emf/2002/Ecore" xmlns:xsd="http://www.w3.org/2001/XMLSchema" ecore:nsPrefix=
       "catalogue" ecore:package="catalogue" targetNamespace="http://www.example.org/catalogue"
       >
3
4   <xsd:element name="patternCatalogue" type="catalogue:PatternCatalogue"/>
5
6   <xsd:simpleType name="Classification">
7      <xsd:restriction base="xsd:string">
8        <xsd:enumeration ecore:name="ScientificExperimentModel" value="Scientific␣Experiment␣
             Model"/>
9        <xsd:enumeration ecore:name="ITExperimentModel" value="IT␣Experiment␣Model"/>
10       <xsd:enumeration value="Infrastructures"/>
11     </xsd:restriction>
12   </xsd:simpleType>
13
14   <xsd:complexType name="PatternCatalogue">
15   <xsd:sequence>
16     <xsd:element maxOccurs="unbounded" minOccurs="0" name="patterns" type="catalogue:Pattern
           "/>
17     <xsd:element maxOccurs="unbounded" minOccurs="0" name="structureElements" type="
           catalogue:StructureElement"/>
18   </xsd:sequence>
19   <xsd:attribute name="title" type="xsd:string"/>
20   </xsd:complexType>
21
22   <xsd:complexType name="Pattern">
23     <xsd:attribute name="classification" type="catalogue:Classification"/>
24     <xsd:attribute name="context" type="xsd:string"/>
25     <xsd:attribute name="example" type="xsd:string"/>
26     <xsd:attribute name="intent" type="xsd:string"/>
27     <xsd:attribute name="name" type="xsd:string"/>
28     <xsd:attribute name="problem" type="xsd:string"/>
29     <xsd:attribute name="solution" type="xsd:string"/>
30     <xsd:attribute name="status" type="xsd:string"/>
31     <xsd:attribute ecore:name="conflicts" ecore:reference="catalogue:Pattern" name="
           CONFLICTS">
32       <xsd:simpleType>
33         <xsd:list itemType="xsd:anyURI"/>
```

```
34        </xsd:simpleType>
35      </xsd:attribute>
36      <xsd:attribute ecore:name="refinedBy" ecore:reference="catalogue:Pattern" name="
            REFINEDBY">
37        <xsd:simpleType>
38          <xsd:list itemType="xsd:anyURI"/>
39        </xsd:simpleType>
40      </xsd:attribute>
41      <xsd:attribute ecore:name="relatedTo" ecore:reference="catalogue:Pattern" name="
            RELATEDTO">
42        <xsd:simpleType>
43          <xsd:list itemType="xsd:anyURI"/>
44        </xsd:simpleType>
45      </xsd:attribute>
46      <xsd:attribute ecore:name="requires" ecore:reference="catalogue:Pattern" name="REQUIRES"
            >
47        <xsd:simpleType>
48          <xsd:list itemType="xsd:anyURI"/>
49        </xsd:simpleType>
50      </xsd:attribute>
51      <xsd:attribute ecore:name="uses" ecore:reference="catalogue:Pattern" name="USES">
52        <xsd:simpleType>
53          <xsd:list itemType="xsd:anyURI"/>
54        </xsd:simpleType>
55      </xsd:attribute>
56    </xsd:complexType>
57
58    <xsd:complexType name="StructureElement">
59      <xsd:attribute name="name" type="xsd:string"/>
60      <xsd:attribute name="instruction" type="xsd:string"/>
61      <xsd:attribute ecore:name="patternref" ecore:reference="catalogue:Pattern" name="
            PATTERNREF">
62        <xsd:simpleType>
63          <xsd:list itemType="xsd:anyURI"/>
64        </xsd:simpleType>
65      </xsd:attribute>
66        <xsd:attribute ecore:name="structureelementref" ecore:reference="
              catalogue:StructureElement" name="STRUCTUREELEMENTREF">
67        <xsd:simpleType>
68          <xsd:list itemType="xsd:anyURI"/>
69        </xsd:simpleType>
70      </xsd:attribute>
71    </xsd:complexType>
72
73  </xsd:schema>
```

**Listing B.1:** Domain model of the DSS in XML Schema Definition (XSD) .

# Bibliography


[1] Ahronovitz, M.: Some Thoughts on the Differences between HTC and HPC. http://my-inner-voice.blogspot.de/2013/03/some-thoughts-on-differences-between.html (2013)

[2] Alexander, C.: The Timeless Way of Building. Oxford University Press, New York (1979)

[3] Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (August 1977)

[4] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009)

[5] Barney, B.: Message Passing Interface (MPI), https://computing.llnl.gov/tutorials/mpi/

[6] Bassini, S., Boyer, E., Browne, M., Dale, T., Desplat, J.C., Bela Dias, A., Eickermann, T., Garofalo, F., Geller, C., Girona, S., González, M., Hart, D.L., Kennedy, A., Norman, M., Patra, A., Towns, J., Webster, P., White, J.C.: Handbook of HPC e-Science Infrastructure Allocation Reviewing, Selection and Management. http://www.hpcworld.eu/files/intranet/handbook.pdf (2011)

[7] Beckman, N.E.: A Survey of Methods for Preventing Race Conditions. http://www.cs.cmu.edu/~nbeckman/papers/race_detection_survey.pdf (2006)

[8] Belloum, A., Inda, M., Vasunin, D., Korkhov, V., Zhao, Z., Rauwerda, H., Breit, T., Bubak, M., Hertzberger, L.: Collaborative e-Science Experiments and Scientific Workflows. Internet Computing, IEEE 15(4), 39–47 (July 2011)

[9] Berriman, G., Deelman, E., Good, J., Jacob, J., Katz, D., Laity, A., Prince, T., Singh, G., Su, M.H.: Generating Complex Astronomy Workflows. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 19–38. Springer London (2007)

[10] Birukou, A., Blanzieri, E., Giorgini, P.: Choosing the Right Design Pattern: The Implicit Culture Approach (2006)

[11] Birukou, A., Weiss, M.: Patterns 2.0: a Service for Searching Patterns. In: Proceedings of the 14th Annual European Conference on Pattern Languages of Programming (EuroPLoP 2009), 8-12 July 2009, Irsee, Germany (2009)

[12] Blanzieri, E., Giorgini, P., Massa, P., Recla, S.: Implicit Culture for Multi-agent Interaction Support. In: Proceedings of the 9th International Conference on Cooperative Information Systems, CoopIS 2001, 5-7 September 2001, Trento, Italy, Lecture Notes in Computer Science, vol. 2172, pp. 27–39. Springer Berlin Heidelberg (2001)

[13] Boehm, B.: Software Engineering. IEEE Trans. Computers C-25, 1226–1241 (1976)

[14] Boisseau, J.: High Throughput Computing, Grid Computing, Cloud Computing, Etc. – Definitions & Thoughts. `http://cfc.fis.uc.pt/events/tacc2008/docs/HTC,%20Grid%20Computing.pdf` (2008)

[15] Booch, G.: Object-oriented Analysis and Design with Applications (2nd Ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA (1994)

[16] Bowers, S., Ludäscher, B.: An Ontology Driven Framework for Data Transformation in Scientific Workflows. In: Proceedings of the 1st International Workshop on Data Integration in the Life Sciences (DILS 2004), 25-26 March 2004, Leipzig, Germany. LNCS 2994, Leipzig, Germany (March 2004)

[17] Breitenbücher, U., Binz, T., Kopp, O., Leymann, F.: Pattern-based Runtime Management of Composite Cloud Applications. In: Proceedings of the 3rd International Conference on Cloud Computing and Service Science, CLOSER 2013. SciTePress (2013)

[18] Brooke, J., Pickles, S., Carr, P., Michael, K.: Workflows in Pulsar Astronomy. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 60–79. Springer London (2007)

[19] Brown, D., Brady, P., Dietz, A., Cao, J., Johnson, B., McNabb, J.: A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 39–59. Springer London (2007)

[20] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture, Volume 1: A System of Patterns. John Wiley & Sons, Hoboken, New Jersey, USA (Aug 1996)

[21] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing As the 5th Utility. Future Generation Computer Systems 25(6), 599–616 (Jun 2009)

[22] Chen, W., Silva, R.F.D., Deelman, E., Sakellariou, R.: Balanced Task Clustering in Scientific Workflows. In: Proceedings of the 2013 IEEE 9th International Conference on e-Science, 22-25 October 2013, Beijing, China. pp. 188–195. IEEE Computer Society, Washington, DC, USA (2013)

[23] Convention on Biological Diversity: Article 2 (Rio Earth Summit). `http://www.cbd.int/convention/articles/?a=cbd-02` (1992)

[24] Corbet, J., Rubini, A., Kroah-Hartman, G.: Linux Device Drivers, 3rd Edition. O'Reilly Media, Inc. (2005)

[25] Cugler, D.C., Medeiros, C.B., Shekhar, S., Toledo, L.F.: A Geographical Approach for Metadata Quality Improvement in Biological Observation Databases. Proceedings of the 2013 IEEE 9th International Conference on e-Science, 22-25 October 2013, Beijing, China pp. 212–220 (2013)

[26] Curcin, V., Ghanem, M.: Scientific Workflow Systems - Can one Size fit all? In: Proceedings of 2008 Cairo International Biomedical Engineering Conference (CIBEC 2008), 18-20 December 2008, Cairo, Egypt. pp. 1–9 (Dec 2008)

[27] Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Commun. ACM 51(1), 107–113 (Jan 2008)

[28] Deelman, E., Mehta, G., Singh, G., Su, M.H., Vahi, K.: Pegasus: Mapping Large-Scale Workflows to Distributed Resources. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 376–394. Springer London (2007)

[29] Fan, W., Gordon, M., Pathak, P.: On Linear Mixture of Expert Approaches to Information Retrieval. Deciision Support Systems 42(2), 975–987 (Nov 2006)

[30] Fehling, C., Ewald, T., Leymann, F., Pauly, M., Rütschlin, J., Schumm, D.: Capturing Cloud Computing Knowledge and Experience in Patterns. In: Proceedings of the 5th IEEE International Conference on Cloud Computing, CLOUD 2012. pp. 726–733. IEEE Computer Society (2012)

[31] Fehling, C., Leymann, F.: Cloud Computing Patterns – Fundamentals to Design, Build, and Manage Cloud Applications, Tutorial at SummerSoC 2013, 1-6 July, Hersonissos, Crete, Greece. `http://www.summersoc.eu/summersoc2013/wp-content/uploads/2013/07/Christoph_Fehling_Part_1.pdf` (2013)

[32] Fehling, C., Leymann, F., Retter, R., Schupeck, W., Arbitter, P.: Cloud Computing Patterns – Fundamentals to Design, Build, and Manage Cloud Applications. Springer (2014)

[33] Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach. PWS Publishing Co., Boston, MA, USA, 2nd edn. (1998)

[34] Foster, I.: What is the Grid? A Three Point Checklist. GRIDtoday (June 2002)

[35] Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)

[36] Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. High Perform. Comput. Appl. 15(3), 200–222 (Aug 2001)

[37] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)

[38] Gannon, D., Plale, B., Marru, S., Kandaswamy, G., Simmhan, Y., Shirasuna, S.: Dynamic, Adaptive Workflows for Mesoscale Meteorology, chap. 9, pp. 129–145. Springer (2007)

[39] Gibbon, P., Frings, W., Dominiczak, S., Mohr, B.: Performance Analysis and Visualization of the N-Body Tree Code PEPC on Massively Parallel Computers, NIC series, vol. 33, pp. 367–374. John von Neumann Institute for Computing, Jülich (2006)

[40] Giger, B.: Design of Experiments - Einführung in die statistische Versuchsplanung). `http://www.tqu-group.com/downloads/doedownload.pdf` (2013)

[41] Glavic, B., Dittrich, K.R.: Data Provenance: A Categorization of Existing Approaches. In: Proceedings of the 12th GI Conference on Datenbanksysteme in Buisness, Technologie und Web (BTW). LNI, vol. 103, pp. 227–241. GI (2007)

[42] Gomes, P., Pereira, F.C., Paiva, P., Seco, N., Carreiro, P., Ferreira, J.L., Bento, C.: Using CBR for Automation of Software Design Patterns. In: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, pp. 534–548. Springer Berlin Heidelberg (2002)

[43] Goodale, T.: Expressing Workflow in the Cactus Framework. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 416–427. Springer London (2007)

[44] Google: Introduction to Parallel Programming and MapReduce. `https://courses.cs.washington.edu/courses/cse490h/07wi/readings/IntroductionToParallelProgrammingAndMapReduce.pdf` (2007)

[45] Grimm, C.: Seminar Aspekte Verteilter Systeme, Teil 1 – Einführung in e-Science und Grid Computing. `http://www.rrzn.uni-hannover.de/fileadmin/ful/vorlesungen/Seminar/ss_08/Grid-Seminar_2_SS08.pdf` (2008)

[46] Guéhéneuc, Y.G., Mustapha, R.: A Simple Recommender System for Design Patterns. In: Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories (2007)

[47] Hey, T., Tansley, S., Tolle, K. (eds.): The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmond, Washington (2009)

[48] Hey, T., Trefethen, A.E.: The UK e-Science Core Programme and the Grid. Journal of Future Generation Computer Systems(FGCS 18, 1017–1031 (2002)

[49] Hinde, S., Wilcock, L.: The Grid as a Platform for Communication, Collaboration and e-Science (2002)

[50] Huerta, M., Haseltine, F., Lio, Y., Downing, G., Seto, B.: Nih Working Definition of Bioinformatics and Computational Biology. `http://www.bisti.nih.gov/docs/CompuBioDef.pdf` (2000)

[51] IAAS, University of Stuttgart: Simulation Workflows. `http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/index.php` (2011)

[52] Intakosum, S., Muangon, W.: Retrieving Model for Design Patterns. In: Proceedings of ECTI Transactions on Computer and Information Technology, Vol. 3, No. 1. pp. 51–55 (May 2007)

[53] Jacobson, I., Christerson, M., Jonsson, P., Overgaard., G.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992)

[54] Jones, A.: Workflow and Biodiversity e-Science. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 80–90. Springer London (2007)

[55] Journal of Theoretical and Applied Information Technology: Comparison of Grid Computing vs. Cluster Computing. http://www.jatit.org/research/introduction_grid_computing.htm

[56] Karastoyanova, D., Andrikopoulos, V.: eScienceSWaT – A Software Engineering Methodology and Architectural Framework for Building eScience Applications

[57] Karastoyanova, D., Andrikopoulos, V.: eScienceSWaT – Towards an eScience Software Engineering Methodology. Proceedings of the 17th International Enterprise Distributed Object Computing Conference Workshops (EDOCW 2013), 0, 229–238 (2013)

[58] Karastoyanova, D., Leymann, F.: Making Scientific Applications on the Grid Reliable through Flexibility Approaches Borrowed from Service Compositions, pp. 635–656. Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications. Volume II., IGI Global (January 2010)

[59] Köckerbauer, T., Polak, M., Stütz, T., Uhl, A.: GVid - Video Coding and Encryption for Advanced Grid Visualization. In: Proceedings of the 1st Austrian Grid Symposium, 1-2 December 2005, Hagenberg, Austria. pp. 204–218 (2006)

[60] Kerzner, M., Maniyam, S.: Hadoop Illuminated. Hadoop illuminated LLC (2013)

[61] Khurana, R., Beniwal, V.: Object Oriented Solution for Industrial ERP Using Design Patterns in .Net. International Journal of Applied Engineering Research (IJAER). http://gimt.edu.in/clientFiles/FILE_REPO/2012/NOV/23/1353645863030/78.pdf (2012)

[62] Kolodner, J.L.: An Introduction to Case-Based Reasoning. Artificial Intelligence Review 6(1), 3–34 (Mar 1992)

[63] Kumar, K., Prabhakar, T.: Design Decision Topology Model for Pattern Relationship Analysis. In: Proceedings of the 1st Asian Conference on Pattern Languages of Programs. AsianPLoP '10, ACM, New York, NY, USA (2010)

[64] von Laszewski, G., Hategan, M., Kodeboyina, D.: Java CoG Kit Workflow. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 340–356. Springer London (2007)

[65] Leong, L., Toombs, D., Gill, B., Petri, G., Haynes, T.: Magic Quadrant for Cloud Infrastructure as a Service. http://www.gartner.com/technology/reprints.do?id=1-1UKQQA6&ct=140528&st=sb (2014)

[66] Lezzi, D., Rafanell, R., Torres, E., Giovanni, R., Blanquer, I., Badia, R.: Programming Ecological Niche Modeling Workflows in the Cloud. In: Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013). pp. 1223–1228 (March 2013)

[67] Library of Congress: Science Reference Services – e-Science. http://www.loc.gov/rr/scitech/tracer-bullets/esciencetb.html

[68] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006)

[69] Maechling, P., Deelman, E., Zhao, L., Graves, R., Mehta, G., Gupta, N., Mehringer, J., Kesselman, C., Callaghan, S., Okaya, D., Francoeur, H., Gupta, V., Cui, Y., Vahi, K., Jordan, T., Field, E.: SCEC CyberShake Workflows—Automating Probabilistic Seismic Hazard Analysis Calculations. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 143–163. Springer London (2007)

[70] Mattoso, M., Dias, J., Costa, F., Oliveira, D., Ogasawara, E.: Experiences in Using Provenance to Optimize the Parallel Execution of Scientific Workflows Steered by Users (2014)

[71] Mattson, T., Sanders, B., Massingill, B.: Patterns for Parallel Programming. Addison-Wesley Professional, first edn. (2004)

[72] McGough, S., Lee, W., Cohen, J., Katsiri, E., Darlington, J.: ICENI. In: Workflows for e-Science: Scientific Workflows for Grids. Springer (June 2007)

[73] Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Tech. rep., National Institute of Standards and Technology (NIST) (2009)

[74] Naveda, J.F., Seidman, S.B.: IEEE Computer Society Real-World Software Engineering Problems: A Self-Study Guide for Today's Software Professional. Wiley (2006)

[75] Newman, P., Ward, N., Campbell, H., Watts, M., Franklin, C., Hunter, J.: OzTrack: Data Management and Analytics Tools for Australian Animal Tracking. http://www.itee.uq.edu.au/eresearch/filething/files/get/projects/oztrack/OzTrack_eResearch_Conference.pdf (2011)

[76] Noble, J.: Classifying Relationships Between Object-Oriented Design Patterns. In: Proceedings of the 1998 Australian Software Engineering Conference, 9-13 November 1998, Adelaide, Australia. pp. 98–107 (Nov 1998)

[77] Novotny, J., Russell, M., Wehrens, O.: GridSphere: A Portal Framework for Building Collaborations: Research Articles. Concurrency and Computation: Practice and Experience 16(5), 503–513 (Apr 2004)

[78] Nowak, A., Binz, T., Fehling, C., Kopp, O., Leymann, F., Wagner, S.: Pattern-driven Green Adaptation of Process-based Applications and their Runtime Infrastructure. Computing 94(6), 463–487 (2012)

[79] Nowak, A., Leymann, F.: Green Business Process Patterns - Part II. In: Proceedings of the 6th IEEE International Conference on Service Oriented Computing and Applications, SOCA 2013, 16-18 December 2013, Kauai, Hawaii, USA. p. TBA. IEEE Computer Society (2013)

[80] Oinn, T., Addis, M., Ferris, J., Marvin, D., Carver, T., Pocock, M.R., Wipat, A.: Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. Bioinformatics 20(17) (2004)

[81] Oinn, T., Li, P., Kell, D., Goble, C., Goderis, A., Greenwood, M., Hull, D., Stevens, R., Turi, D., Zhao, J.: Taverna/myGrid: Aligning a Workflow System with the Life Sciences Community. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 300–319. Springer London (2007)

[82] Oracle Corporation: The Java EE 6 Tutorial - Direction in Entity Relationships). `http://docs.oracle.com/cd/E19798-01/821-1841/bnbqi/index.html` (2010)

[83] Oracle Corporation: The NetBeans E-commerce Tutorial - Designing the Application. `https://netbeans.org/kb/docs/javaee/ecommerce/design.html` (2013)

[84] Palma, F., Farzin, H., Guéhéneuc, Y.G., Moha, N.: Recommendation System for Design Patterns in Software Development: An DPR Overview. In: Proceedings of the 3rd International Workshop on Recommendation Systems for Software (RSSE) (2012)

[85] Pennington, D., Higgins, D., Peterson, A., Jones, M., Ludäscher, B., Bowers, S.: Ecological Niche Modeling Using the Kepler Workflow System. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 91–108. Springer London (2007)

[86] Pfister, G.F.: In Search of Clusters (2nd Ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1998)

[87] Pol, A.A., Ahuja, R.K.: Developing Web-Enabled Decision Support Systems Using VB.NET and ASP.NET. Dynamic Ideas (2007)

[88] Power, D.J.: Decision Support Systems: Concepts and Resources for Managers). Westport, CN: Quorum Books (2002)

[89] Reiter, M., Breitenbücher, U., Kopp, O., Karastoyanova, D.: Quality of Data Driven Simulation Workflows. In: Proceedings of the 8th IEEE International Conference on e-Science. pp. 1–8 (Oct 2012)

[90] Research Councils UK: About the UK e-Science Programme. `http://www.bath.ac.uk/bucs/services/esciencegridwork/`

[91] Riding, M., Wood, J.D., Brodlie, K.W., Brooke, J.M., Chen, M., Chisnall, D., Hughes, C., John, N.W., Jones, M.W., Roard, N.: e-Viz: Towards an Integrated Framework for High Performance Visualization. In: Proceedings of the UK e-Science All Hands Meeting 2005, 19-22 September 2005, Nottingham UK. pp. 1026–1032 (2005)

[92] Riedel, M.: Design and Applications of an Interoperability Reference Model for Production e-Science Infrastructures. Dissertation, Karlsruher Institut für Technologie (KIT), Jülich (2013)

[93] Riedel, M., Eickermann, T., Frings, W., Dominiczak, S., Mallmann, D., Dussel, T., Streit, A., Gibbon, P., Wolf, F., Schiffmann, W., Lippert, T.: Design and Evaluation of a Collaborative Online Visualization and Steering Framework Implementation for Computational Grids. In: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (GRID '07). pp. 169–176. IEEE Computer Society, Washington, DC, USA (2007)

[94] Riedel, M., Streit, A., Wolf, F., Lippert, T., Kranzlmüller, D.: Classification of Different Approaches for e-Science Applications in Next Generation Computing Infrastructures. In: Proceedings of the 4th IEEE International Conference on eScience, eScience 2008, 7-12 December 2008, Indianapolis, Indiana, USA. pp. 198–205 (December 2008)

[95] Rouse, M.: Expert System. `http://searchhealthit.techtarget.com/definition/expert-system` (2005)

[96] Ruan, Y., Fox, G.: A Robust and Scalable Solution for Interpolative Multidimensional Scaling with Weighting. In: Proceedings of the 2013 IEEE 9th International Conference on e-Science, 22-25 October 2013, Beijing, China. pp. 61–69. IEEE Computer Society, Los Alamitos, CA, USA (2013)

[97] Sahly, E., Sallabi, O.: Design Pattern Selection: A Solution Strategy Method. In: Proceedings of the IEEE 2012 International Conference on Computer Systems and Industrial Informatics (ICCSII), 18-20 December 2012, Dubai, Sharjah, United Arab Emirates. pp. 1–6 (Dec 2012)

[98] Saira Thabasum, S., Mani Sundar, U.T.: A Survey on Software Design Pattern Tools for Pattern Selection and Implementation. In: International Journal of Computer Science & Communication Networks, Vol 2(4) (2012)

[99] SAS Institute Inc.: Concepts of Experimental Design: Design Institute for Six Sigma. `https://support.sas.com/resources/papers/sixsigma1.pdf` (2005)

[100] Schumm, D., Anstett, T., Leymann, F., Schleicher, D.: Applicability of Process Viewing Patterns in Business Process Management. In: Proceedings of the International Workshop on Models and Model-driven Methods for Service Engineering (3M4SE 2010), in conjunction with the 14th IEEE International EDOC Conference (EDOC 2010), 25-29 October 2010, Vitória, Brazil. pp. 79–88. IEEE Computer Society (2010)

[101] Schumm, D., Barzen, J., Leymann, F., Ellrich, L.: A Pattern Language for Costumes in Films. In: Proceedings of the 17th European Conference on Pattern Languages of Programs (EuroPLoP 2012) (2012)

[102] Schumm, D., Leymann, F., Streule, A.: Process Viewing Patterns. In: Proceedings of the 14th IEEE International EDOC Conference, EDOC 2010, 25-29 October 2010, Vitória, Brazil. pp. 89–98. IEEE Computer Society (2010)

[103] Simmhan, Y.L., Plale, B., Gannon, D.: A Survey of Data Provenance in e-Science. ACM SIGMOD Record 34(3), 31–36 (Sep 2005)

[104] Sonntag, M., Hahn, M., Karastoyanova, D.: Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In: Proceedings of the Demo Track of the 10th International Conference on Business Process Management (BPM 2012), CEUR Workshop Proceedings, 2012. pp. 1–5. CEUR Workshop Proceedings (September 2012)

[105] Sonntag, M., Hotta, S., Karastoyanova, D., Molnar, D., Schmauder, S.: Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications. In: Proceedings of the 4th European Conference on Towards a Service-based Internet. pp. 242–253. ServiceWave'11, Springer-Verlag, Berlin, Heidelberg (2011)

[106] Sonntag, M., Karastoyanova, D.: Concurrent Workflow Evolution. In: Electronic Communications of the EASST, Volume 37, ISSN 1863-2122. pp. 1–12. Gesellschaft für Informatik e.V. (GI) (März 2011)

[107] Sonntag, M., Karastoyanova, D.: Ad hoc Iteration and Re-execution of Activities in Workflows. International Journal On Advances in Software 5(1 & 2), 91–109 (Juli 2012)

[108] Sonntag, M., Karastoyanova, D., Leymann, F.: The Missing Features of Workflow Systems for Scientific Computations. In: Proceedings of the 3rd Grid Workflow Workshop (GWW), Software Engineering Conference, GI-Edition Lecture Notes in Informatics (LNI), P-160. pp. 209–216. Gesellschaft für Informatik e.V. (GI) (February 2010)

[109] Stell, A., Sinnott, R.: e-Enabling International Cancer Research: Lessons Being Learnt in the ENS@T-CANCER Project. In: Proceedings of the 2013 IEEE 9th International Conference on e-Science, 22-25 October 2013, Beijing, China. pp. 132–139 (Oct 2013)

[110] Stevens, R., Goble, C.A., Baker, P.G., Brass, A.: A Classification of Tasks in Bioinformatics. Bioinformatics 17(1), 180–188 (2001)

[111] Strauch, S., Andrikopoulos, V., Breitenbücher, U., Kopp, O., Leymann, F.: Non-Functional Data Layer Patterns for Cloud Applications. In: Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2012, 3-6 December 2012, Taipei, Taiwan. pp. 601–605. IEEE Computer Society (2012)

[112] Strauch, S., Andrikopoulos, V., Breitenbücher, U., Sáez, S.G., Kopp, O., Leymann, F.: Using Patterns to Move the Application Data Layer to the Cloud. In: Proceedings of the 5th International Conference on Pervasive Patterns and Applications, PATTERNS 2013, 27 May – June 1 2013, Valencia, Spain. pp. 26–33. Xpert Publishing Services (XPS) (2013)

[113] Strauch, S., Breitenbücher, U., Kopp, O., Leymann, F., Unger, T.: Cloud Data Patterns for Confidentiality. In: Proceedings of the 2nd International Conference on Cloud Computing and Service Science, CLOSER 2012, 18-21 April 2012, Porto, Portugal. pp. 387–394. SciTePress (2012)

[114] Studer, R., Benjamins, V.R., Fensel, D.: Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering 25(1-2), 161–197 (March 1998)

[115] Suresh, S.S., Naidu, M.M., Asha Kiran, S.: Design Pattern Recommendation System (Methodology, Data Model and Algorithms) (2011)

[116] SYS-CON Media Inc.: Twenty-One Experts Define Cloud Computing. http://virtualization.sys-con.com/node/612375 (2009)

[117] Talia, D.: Workflow Systems for Science: Concepts and Tools. ISRN Software Engineering 2013, 15 (2013)

[118] Taylor, I., Shields, M., Wang, I., Harrison, A.: The Triana Workflow Environment: Architecture and Applications. In: Workflows for e-Science: Scientific Workflows for Grids, pp. 320–339. Springer London (2007)

[119] Taylor, J.: Defining e-Science. http://www.nesc.ac.uk/nesc/define.html

[120] The Eclipse Foundation: GEF Programmer's Guide. http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.gef.doc.isv%2Fguide%2Fguide.html (2014)

[121] The Eclipse Foundation: Graphical Modeling Framework. http://wiki.eclipse.org/Graphical_Modeling_Framework (2014)

[122] Vandenhouten, R.: Modellgetriebene Entwicklung mit Eclipse GMF oder Wie programmiert man einen graphischen Editor ohne eine Zeile Quellcode? Tutorial zum Graphical Modeling Framework. http://www.tm.tfh-wildau.de/vandenhouten/media/GMF-Step-By-Step.pdf (2008)

[123] Vouk, M.A., Singh, M.P.: Quality of Service and Scientific Workflows. In: Proceedings of the IFIP Conference on Quality of Numerical Software: Assessment and Enhancements, 8-12 July, Oxford, U.K. pp. 77–89 (1996)

[124] Wallis, J., Mayernik, M., Pepe, A., Borgman, C.: An Exploration of the Life Cycle of eScience Collaboratory Data (2008)

[125] Wang, S., Padmanabhan, A., Myers, J.D., Tang, W., Liu, Y.: Towards Provenance-aware Geographic Information Systems. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. GIS '08, ACM, New York, NY, USA (2008)

[126] Weidlich, M., Grosskopf, A., Barros, A.: Realising Dead Path Elimination in BPMN. In: Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC'09) (July 2009)

[127] Wickramaarachchi, C., Simmhan, Y.: Continuous Dataflow Update Strategies for Mission-Critical Applications. In: Proceedings of the 2013 IEEE 9th International Conference on e-Science, 22-25 October 2013, Beijing, China. pp. 155–163. ESCIENCE '13, IEEE Computer Society, Washington, DC, USA (2013)

[128] Wilson, P., Emmerich, W., Brodholt, J.: Leveraging HTC for UK eScience with Very Large Condor pools: Demand for transforming untapped power into results. In: Proceedings of the 2004 UK E-Science All Hands Meeting (2004)

[129] Yuan, D., Yang, Y., Liu, X., Chen, J.: A Local-Optimisation Based Strategy for Cost-Effective Datasets Storage of Scientific Applications in the Cloud. In: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD), 4-9 July 2011, Washington, DC, USA. pp. 179–186 (July 2011)

[130] Zeng, J., Plale, B.: Data Pipeline in MapReduce. In: Proceedings of the 2013 IEEE 9th International Conference on e-Science, 22-25 October 2013, Beijing, China. pp. 164–171. IEEE Computer Society, Los Alamitos, CA, USA (2013)

[131] Zimmer, W.: Relationships between Design Patterns. In: Pattern Languages of Program Design. pp. 345–364. Addison-Wesley (1994)

All links were last followed on July 31, 2014.

## Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature